A hybrid algorithm for a Vehicle Routing Problem with Realistic Constraints

Furong Ye[a], Defu Zhang[a,*], Yain-Whar Si[b], Xiangxiang Zeng[a] and Trung Thanh Nguyen[c]

[a]*School of Information Science and Engineering, Xiamen University, Xiamen, China.*
[b]*Department of Computer and Information Science, University of Macau, Macau, China*
[c]*Department of Maritime and Mechanical Engineering, Liverpool John Moores University, Liverpool, England*

Abstract

The growth in multi-national corporations and today's highly competitive market environment are fuelling an unprecedented demand for third-party logistics services, including the need for more realistic and efficient vehicle routing. Many recent works on the vehicle routing problem, however, only consider simple constraints and do not scale well to all types of real-world logistics problems. This paper introduces a new vehicle routing problem with time window and pallet loading constraints, taking into account the actual needs of businesses from the logistics industry in a specific scenario: delivery of consumer goods in pallets with time-windows. We propose a hybrid approach which is a combination of Tabu search and Artificial Bee Colony algorithm. Because the problem addressed in this paper is novel and has never been reported in the literature, we generate a new benchmark data set to verify the performance of the proposed algorithm. The computational experiments are reported for a data set of the Solomon's 56 vehicle routing problem with time windows. Computational results show the effectiveness of the proposed algorithm.

Keywords: Vehicle routing problem; Container loading; Tabu search; Artificial Bee Colony algorithm.

1. Introduction

The vehicle routing problem (VRP) is a classic problem occurring in the logistics and transportation field and is considered as one of the most important problems in operational research. The VRP is concerned with route planning for vehicles starting from a central depot to a set of customers. It was first proposed by Dantzig and Ramser [1], and further algorithm improvements and variants of this problem have been extensively studied in the recent years. Some problems consider different objectives, but most of the variants of the VRP are related to the addition of new constraints to the original problem. For instance, the pallet constraint (goods of different sizes must be transported in boxes of standard size and limited capacity) and time window constraint (goods must be delivered within a certain time window) are often used by different researchers. Leung et al. [2] proposed a meta-heuristic algorithm for heterogeneous fleet VRPs with two-dimensional loading constraints, Tarantilis et al. [3] presented a hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem, Bortfeldt and Homberger [4] developed a "packing first, routing second" heuristic for the vehicle routing and loading problem and Junqueira et al. [5] proposed an optimization model for the VRP with practical three-dimensional loading constraints. Tavakkoli-Moghaddam et al. [6] created a new mathematical model for a competitive VRP with time windows and solved by simulated annealing, Hifi and Wu [7] proposed a hybrid metaheuristic for the VRP with time windows, Gong et al. [8] developed a discrete particle swarm optimization approach to solve the VRP with time windows, and Cherkesly et al. [9] proposed a population-based metaheuristic for the pickup and delivery problem with time windows and last-in, first-out (LIFO) loading.

The two aforementioned constraints, however, are normally considered separately in the existing research. In reality, both constraints co-exist in transportation problems. Based on a managerial perspective of most logistic companies, dealing with daily dispatching problems effectively is always concerned. The simple VRP considered in many existing research neglects various constraints in certain practical situations and some variants of the simple VRP tackle constraints independently, which may not be realistic. On the one hand, the VRP with time windows constructs routes without taking into account properties of items to be shipped. Though there are approaches that consider the capacity of vehicles, they still ignore the actual physical dimensions of items. On the other hand, the VRPs with pallet constraint constructs routes with feasible loading condition, but they do not take into account the time factor, which may cause an inability of achieving items within the specific time period. Therefore, it is necessary to further investigate realistic situations for the VRP with both the time window and pallet constraints (the three-dimensional loading problem in particular) taken into account. Regarding these two constraints, recently, Wei et al. [10] proposed an adaptive variable neighbourhood search for a heterogeneous fleet VRP with three-dimensional loading constraints, and Silva et al. [11] reviewed solution methods and computational experiments for the pallet loading problem.

To the best of our knowledge, the VRP with both time window and three-dimensional loading constraints has only been addressed in the work of Zachariadis et al. [12]. The problem in [12] is not a simple VRP with specific constraints; rather, it includes a mix of different request types. This problem, however, does not meet the dispatching requirements from some logistics providers since it considers the pick-up and delivery. Pickup and delivery may be not realistic in some logistics industry, like electronic commerce, which focuses significantly more on the dispatching duty.

The problem introduced in this paper is also a VRP with both time window and three-dimensional loading constraints. However, in contrast to [12], the problem addressed here focuses on a realistic situation that involves both the actual needs of businesses (detailed below) and the unique type of depots and customers. Regarding the time window constraints of this problem, both the depot and customers are specified with a time window having an exact start time and end time. This means that vehicles must depart and return within the depot’s work time window. A vehicle must also visit a customer between the specified start time and end time. If a vehicle arrives before the start time of a customer, the unloading process will must be postponed until the customer is available. Amongst the three-dimensional loading constraints, in addition to the basic space constraint, we also consider fragility, supporting surface and order of unloading. All these constraints are beneficial for businesses like handling electrical appliances and distributing fresh agricultural products, where goods need to be packed and customers request items within specific time window. For the VRP with loading constrains, a container loading problem can be adopted to check the feasibility of loading. In the VRP with time window and pallet loading constraints (VRPTWP), the loading problem is the three-dimensional bin loading problem in which a fixed number of rectangular items are loaded into larger rectangular boxes [36].

Earlier works on the VRP were mainly concentrated on the exact approaches. Balinski and Quandt [13] reported an integer programming approach that can be viewed as a generalisation of the covering problem to solve the VRP. Eilon et al. [14] used dynamic planning to solve the VRP with a fixed number of vehicles. Christofides et al. [15] solved the primary VRP with K-means. Fisher and Jaikumar [16] and Laporte et al. [17] also researched exact approaches for the VRP.

The rapid development of the logistics industry and the dynamic nature of today's business environment have significant effects on the scale and complexity of the VRP. Obviously, traditional exact approaches may not be able to match the scale of real-world situations. This has led to the development of heuristic/meta-heuristic algorithms for the VRP. Tabu search is one of the most important methods in early research on meta-heuristic. One of the earliest reports is by Gendreau et al. [18] in which a Tabu search was used to solve the VRP. Later, many other researchers also used Tabu search to solve the VRP. Renaud et al. [19] solved a multi-depot VRP by Tabu search, Taillard et al. [20] presented a Tabu search heuristic for the VRP with soft time windows, Cordeau et al. [21] proposed a unified Tabu search heuristic for VRPs with time windows, Toth and Vogo [22] described a new variant granular Tabu search, Montané and Galvao [23] proposed a Tabu search algorithm for the VRP with simultaneous pick-up and delivery service, Zachariadis et al. [24] presented a guided Tabu search for the VRP with two-dimensional loading constraints and Leung et al. [25] applied an extended guided Tabu search and a new packing algorithm for the two-dimensional loading VRP.

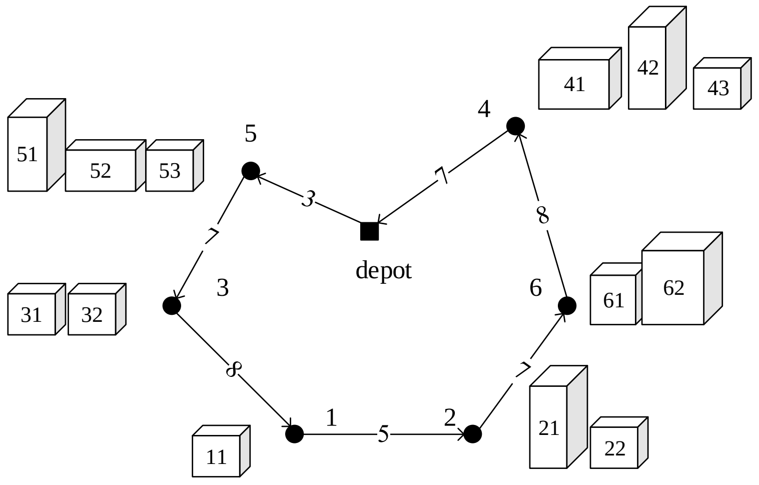
Different to Tabu search, the artificial bee colony has only come to the view of researchers in the last ten years. It was introduced by Karaboga [26]. Szeto et al. [27] firstly applied artificial bee colony to solving the Capacitated VRP. Later, Yao et al. [28] proposed an artificial bee colony algorithm with a scanning strategy for the periodic VRP. Meanwhile, some other meta-heuristics, like simulated annealing [29] and genetic algorithms [30], are also widely used to solve the VRPs. Breedam [31] solved the VRP by using a simulated annealing in 1995 and Baker and Ayechew [32] developed a genetic algorithm for the VRP in 2003. Beyond that, many researches presented some hybrid approaches. Osman [33] proposed a metastrategy of simulated annealing and Tabu search algorithms for the VRP, Vidal et al. [34] proposed a hybrid genetic algorithm for multi-depot and periodic VRPs, and Bortfeldt et al. [35] developed a hybrid algorithm for the VRP with clustered backhauls and three-dimensional loading constraints.

The main contribution of our work is two-fold. First, we consider a new and practical variant of the VRP with loading constraints and time windows. Second, we propose a hybrid algorithm with Tabu search and artificial bee colony (Tabu-ABC), and compare it with other heuristics on a set of Solomon's 56 VRPs with time windows (VRPTW). In addition, the set of benchmark data generated for VRPTWP can be adopted by other interested parties for further research in this area.

The rest of this paper is organised as follows: In Section 2, we introduce the VRPTWP in detail. In Section 3, we describe the new hybrid algorithm Tabu-ABC in detail. In Section 4, we present the computational results of the algorithm, and finally, we conclude our work in Section 5.

2. The Problem

Let *G (V, A)* be an undirected graph where *V = {0, 1, …, n}* is the set of vertices and *A* is the set of edges. Let C_{ij} be the transportation cost between vertices *i* and *j*. The vertex *0* in *V* is called the depot. The main objective of solving the VRP is to search for a solution offering the minimum consumption of distance for traversing every vertex starting from the depot



Time window and work time ([t_{si}, t_{ei}], t_i) of each vertex:
depot:[0,70],0
1:[15,23],2 **2:**[25,30],1 **3:**[10,14],1
4:[35,55],3 **5:**[2,8],2 **6:**[10,40],1

Fig. 1. An example for a route

and come back to the depot.

Recall that the problem to be addressed in this paper is a practical variant of the VRP, and it consists of time windows and loading constraints. We assume that there is a customer at each vertex, and each customer needs some items from the depot. The items are specified with pre-defined total weights, and they are a three-dimensional cuboid of length l_i, width w_i and height h_i. At the depot vertex, there are some vehicles with a fixed loading space (a container) of dimension $L \times W \times H$, where L, W and H are the length, width and height of the loading space, respectively. In addition, each vehicle is specified with a weight capacity D. To make this VRP realistic, time windows are also considered in this paper. Each vertex (the depot and customers) has a particular work time window $[\mathrm{t}_{\mathrm{si}}, \mathrm{t}_{\mathrm{ei}}]$$[t_{si}, t_{ei}]$ and a work time t_i. All work at a vertex must be started within that vertex's work time window. An example of time windows in a route is given in Fig. 1, and the loading constraint is discussed below.

In the VRPTWP, the following demands must be met:

(i) Each customer (vertex) is visited only once, i.e., one customer belongs to only one route.

(ii) All routes start from the depot and end at the depot.

(iii) All routes (vehicles) must depart and return within the time window of the depot.

(iv) All customers (vertices) are available during their work time window.

(v) All items needed by the customers in one route are loaded on the vehicle serving that route.

Before a vehicle can depart from the depot, it is necessary to ensure that all needed items should be loaded on the vehicle. This can be considered as a process of verifying the feasibility of the three-dimensional loading problem while taking into account the following conditions:

(i) All items must be completely loaded and fitted into the container of a vehicle, i.e., the edges of items and the container must not intersect each other.

(ii) Items are not allowed to overlap.

(iii) The bottom of each item must be sufficiently supported by the top of other items or by the bottom of the container.

(iv) The surface of each item must be in parallel with the surface of container.

In addition to the above basic conditions, we add the following conditions to better reflect real-world situations:

(v) Orientation: Items have fixed vertical orientation, which means they have fixed bottom surfaces.

(vi) Capacity: The sum of the items' weight is less than or equal to the vehicle's loading capacity.

(vii) Fragility: Nonfragile items cannot be loaded on the top of fragile items.

(viii) LIFO: If customer *i* is visited earlier than customer *j*, then the items of customer *j* should be packed earlier than those of customer *i*.

In mathematical formulations, our objective is to find the minimum travel distance, which can be expressed as follows:

$$\min \sum_{i=1}^{r} \left(\sum_{j=1}^{n_i - 1} C_{j(j+1)} \right) \tag{1}$$

where r is the number of routes, which is equal to the number of vehicles, n_i is the number of customers in route i and $C_{j(j+1)}$ denotes the cost of traveling from customer j to customer $j+1$ (in this paper the traveling cost is equal to the distance).

The time windows constraints can be described as follows

$$t_{ai} < t_{ei} \tag{2}$$

$$\text{where } \ t_{ai} = t_{l(i-1)} + tC_{i(i-1)}, \tag{3}$$

$$t_{li} = \begin{cases} t_{ai} + t_i, if\ t_{ai} > t_{si} \\ t_{si} + t_i, \text{otherwise} \end{cases} \tag{4}$$

where t_{si} is the start time of customer (vertex) i in the route, t_{ei} is the end time of customer i in the route, t_{ai} is the time the vehicle arrives in customer i, t_{li} is the time the vehicle leaves from customer i, t_i is the complete work time for customer i in the route, and $tC_{i(i-1)}$ is the time cost traveling from customer i to customer i-*1*.

Because the loading constraint is met by a construction heuristic algorithm, there is no need to provide the mathematical formulation for this constraint. Note that the VRPTWP can be considered as a combination of two NP-hard sub-problems. Thus, the VRPTWP as a whole is an NP-hard problem, and it is addressed in this work using a heuristic algorithm.

3. The Proposed Approach

According to the description in Section 2, the VRPTWP can be treated as a combination of a VRP with three-dimensional loading constraints and time window constraints. We propose a two-stage approach that considers multiple strategies to solve the three-dimensional loading problem and a hybrid algorithm with Tabu Search and artificial bee colony to solve the VRPTW.

3.1. Three-dimensional Loading Problem

One of the key aspects of the proposed approach is the method to judge whether boxes (items) needed by customers in a route may be packed on the vehicle. Such a feasibility test has a significant effect as it is repeatedly invoked by the master algorithm described in Section 3.2.

3.1.1. Loading Positions

We place the container (of a vehicle) in a three-dimensional system of coordinates (Fig. 2.), and the origin of coordinates is in the left bottom of the container. The values of L, W and H represent the length, width and height of the container. Based on loading good practices, the boxes to be loaded should be close to the container or boxes already loaded. Since boxes can be rotated in the horizontal direction, when a box i of length l_i, width w_i and height h_i is loaded in the container, its values along the x, y and z axes $< l_i', w_i', h_i' >$ may be considered as $< l_i, w_i, h_i >$ or $< w_i, l_i, h_i >$.

We denote $B = \{b_1, b_2, \ldots, b_n\}$ as a set of boxes. Firstly, there is an available loading position $(0, 0, 0)$ and b_1 is loaded on $(0, 0, 0)$. After that b_2 gets three available loading positions, they are $(l_1', 0, 0)$, $(0, w_1', 0)$ and $(0, 0, h_1')$. Supposed b_2 is loaded on $(l_1', 0, 0)$, then $(l_1', 0, 0)$ is deleted and another three available loading positions $(l_1' + l_2', 0, 0)$, $(l_1', w_2', 0)$ and

$(l_1', 0, h_2')$ are generated, which means b_3 has five available loading positions. Considering the i^{th} box, if it is loaded on *(x, y, z)*, *(x, y, z)* will be deleted from available loading positions list, *$(x+l_i', y, z)$*, *$(x, y+ w_i', z)$* and *$(x, y, z+ h_i')$* will be added to the available loading positions list (Fig. 1.). When a box is loaded, one available loading position is deleted and three new available loading positions are added, so there will be 2(i-1) +1 available loading positions considering b_i. If b_i cannot be loaded on at least one of the available loading positions, then it is assumed that boxes $\{b_1, b_2, \ldots, b_n\}$ cannot be packed in the container.

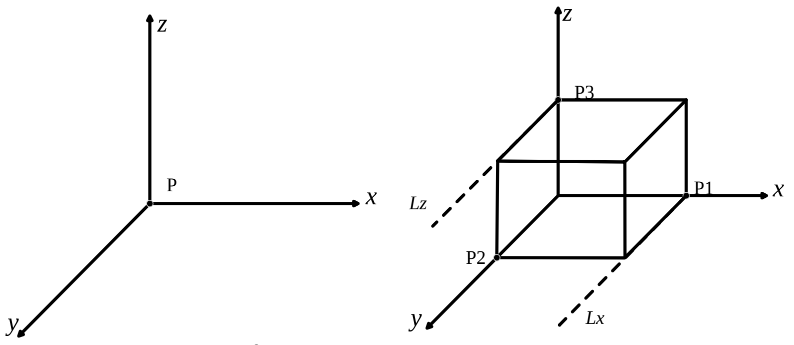


Fig. 2. Updating process of loading positions.

3.1.2. Reference Line

To control the space used and to pack efficiently, two reference lines are considered in this work. They are reference line *Lz* on the *z* axis and reference line *Lx* on the *x* axis. When we check whether b_i can be loaded on *(x, y, z)*, it must not only meet demands in Section 2, but also must satisfy $z+h_i < Lz$ and $x+l_i < Lx$. Once an available loading position is feasible for b_i, b_i will be loaded on it and the available loading positions list will be updated. If none of available positions is feasible, then, two situations are considered. (1) If $Lx < L$, *Lx* will increase to *L*; (2) if $Lz < H$, *Lz* will increase to *H*. If there still is not a feasible position for b_i after (1) and (2), then boxes cannot be packed into the container.

3.1.3. Translational operator.

Once an available loading position is chosen and the box is loaded, according to loading good practices, we will try to move box towards a lower *x*, then move it towards a lower *y*, finally move it towards a lower *z* until the container and other boxes blocks.

3.1.4. Loading algorithm.

In this paper, the loading algorithm (Algorithm 1) is from [37]. *I* is a list of available loading positions from small to large sorted by *x*, breaking ties by *y*, and breaking ties by *z*, and *I* will be kept in order as it is being updated. A flag variable is used to express whether a box can be loaded into the container or not. The inputs of the loading algorithm are set B of ordered boxes and the container of the vehicle. The algorithm will return whether all boxes can be loaded into the container.

Initially, the original available loading position is *(0, 0, 0)*, and *Lz* and *Lx* are equal to 0. Boxes are to be loaded in the container in order. For each box, the algorithm first tries to load the box in the container in a position that does not exceed the reference lines. If this fails, the algorithm changes the value of *Lx*. If *Lx* is equal to 0 or *L*, it means that *Lz* should be increased. When *Lz* is equal to *H* and the box still cannot be loaded, *Lx* should be increased. As *Lx* is increased, all positions whose $x = Lx$ and $y = 0$ have to be tested for loading feasibility. Once *Lx* is equal to *L* and the box cannot be loaded, the algorithm returns *false*. When a box is loaded successfully, the selected loading position is deleted from the list *I*. After the box is moved with translational operators, three new available loading positions are generated. The algorithm will return *true* if all boxes are loaded successfully, otherwise it will return *false*. If it returns *false*, we will consider some boxes cannot be loaded successfully.

3.2. Overall Structure of the VRPTWP Solution

Recall that the VRPTWP consists of two problems, namely, the VRP with three-dimensional loading constraints (3L-CVRP) and the VRPTW. As the three-dimensional loading problem has been addressed in Section 3.1, we are now ready to solve the VRPTWP. In our approach, we use a local search as the fundamental way to improve the solution. We define six neighbourhood structures, one of them is randomly selected and try to find a better solution with less cost at each iteration. However, this strategy may lead the algorithm to be stuck in local optima. To avoid this risk, we adopt the ideas of Tabu search and artificial bee colony.

Algorithm 1. Verifying the Feasibility of Three-Dimensional Loading

```
     PalletisationFeasibility(B, Vehicle)
1.   $I = \{(0, 0, 0)\}$, $Lz = Lx = 0$;
2.   for $i = 0$ to $n$
3.       $flag = false$;
4.       for $(x, y, z) \in I$
5.           if $x + l_i' \leq Lx, z + h_i' \leq Lz$ and $b_i$ can be loaded on $(x, y, z)$
6.               $flag = true$, go to line 19;
7.       if $Lx = 0$ or $Lx = L$
8.           if $b_i$ can be loaded on $(0, 0, Lz)$
9.               $x = 0, y = 0, flag = true, z = Lz, Lz = Lz + h_i', Lx = l_i'$;
10.          else
11.              $Lz = H, Lx = L, i = i - 1$;
12.      else
13.          for $(x, y, z) \in I$
14.              if $x = Lx$ and $y = 0$
15.                  if $z + h_i' \leq Lz$ and $b_i$ can be loaded on $(x, y, z)$
16.                      $Lx = Lx + l_i'$, $flag = true$, go to line 18;
17.                  else
18.                      $Lx = L, i = i - 1$;
19.      if $flag = true$
20.          load $b_i$ on $(x, y, z)$, $I = I / (x, y, z)$, move $b_i$ with the translational operator and mark $b_i$'s new position $(x', y', z')$, $I = I \cup$
     $(x' + l_i', y', z'), (x', y' + w_i', z'), (x', y', z' + h_i')$;
21.      else
22.          return $false$;
23.  return $true$;
```

3.2.1. Construction of the Initial VRPTWP Solution

Our algorithm begins by generating an initial feasible solution (explained below). Once this initialisation step is done, further improvement can be performed to achieve better results. Note that all constraints described in Section 2 must be satisfied by this initial solution.

Algorithm 2. Initialisation of *VRPTWP*

```
Initialisation (C)
1.  Route_Num = 0, $c = C, S = \phi$;
2.  while $c \neq \phi$
3.      Select customer $c_i$ from $C$ randomly.
4.      if Route_Num = 0
5.          Route_num = Route_num + 1;
6.          Generate a new route $s$, $S = S + s$;
7.          Insert $c_i$ to $s$, $c = c/ c_i$;
8.      else if $c_i$ can be inserted into a route in $S$
9.          Select $s$ that causes least increase of cost after insertion;
10.         Insert $c_i$ into $s$ in the position that causes least increase of cost after insertion;
11.         $c = c/ c_i$;
12.     else
13.         Route_num = Route_num + 1;
14.         if Route_num > number of vehicles
15.             Go to line 1;
16.         else
17.             Generate a new route $s$, $S = S + s$;
18.             Insert $c_i$ to $s$, $c = c/ c_i$;
```

C is the set of customer, S is the set of routes and Route_Num is the number of routes.

Customers are randomly inserted into the routes one by one. During the insertion process, the customer is to be inserted in the position which leads to the smallest increase of cost, and all of the constraints mentioned in Section 2 must be satisfied. When a new customer is being inserted and all existing routes cannot be assigned for that customer, a new route will be set up. If the total number of routes exceeds the number of available vehicles, the algorithm restarts; otherwise, the algorithm continues with the traversal of all customers until they are processed. The process of construction of initialization is presented in Algorithm 2.

3.2.2. Neighbourhood Solutions

When we try to find a better solution, we can reassign positions of vertices in different routes to construct a new solution. At each step, we choose a neighbourhood structure randomly. Six neighbourhood structures (Fig. 3.) are applied in this paper and are defined as follows:

(i) Swapping: In this strategy, the locations of two customers are exchanged. Two customers can be in the same route, and also they can be in different routes.

(ii) Relocation: With this strategy, a customer is moved to another position. The new position can be in its original route or another route.

(iii) Routes swapping: Each route is divided into two sub-routes by a vertex (customer). Two sub-routes (the part that is visited after the specific vertex) of the two routes are exchanged entirely.

(iv) Route reversal: A sub-route reverses the order of customers.

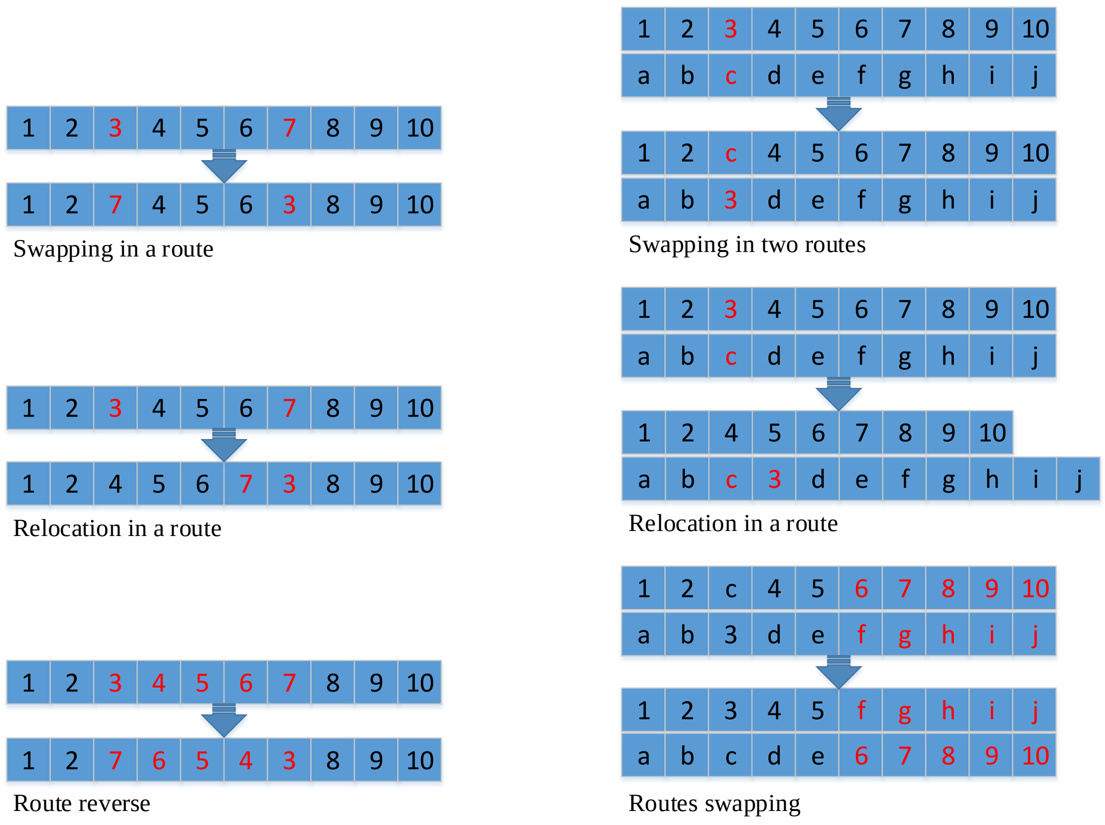


Fig. 3. Neighbourhood structures.

3.2.3. Tabu-ABC

Tabu-ABC is a combination of Tabu search and artificial bee colony algorithm. It uses Tabu search to rapidly generate ~~fast and~~ high quality solutions that are used by artificial bee colony. Meanwhile, artificial bee colony takes advantage of the Tabu to increase food source variety.

Tabu search is designed to search for the best solution in its neighbourhood, even if there is no better solution. This could pose a risk of getting stuck in local optima (Fig. 4). In the example in Fig. 4, A, B, C and D are four different solutions. In A’s neighbourhood, B is the best solution, C is the best in B’s neighbourhood, and A is the best in C’s neighbourhood. Using the proposed algorithm, the best solution found can only be B, C and A. This is the situation in which the local search can become stuck in a loop, and the solution cannot be improved any further. Although D is the true best solution, it is missed by the algorithm. This case highlights the problem of the search being trapped in the local optima.

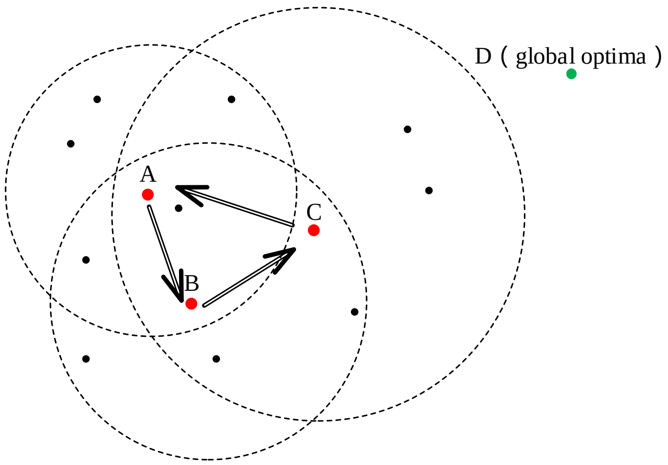


Fig. 4. Trap of Local optima

To avoid this trap, a Tabu list is introduced to prohibit previously visited customers from being revisited by the algorithm. Once we have found a better solution with the neighbourhood structure, the current exchanged customers are inserted into a Tabu list. In the next iterations of searching, these customers are not selected unless they can obtain a better solution. The maximum length of the Tabu list is called the Tabu tenure.

The artificial bee colony algorithm is a class of swarm intelligence techniques. Honey bees are classified into three types: employed bees, onlookers and scouts. The job of employed bees is to exploit the food sources. They gather and share information with onlookers. According to the information shared by employed bees, onlookers are going to choose a food source with higher equality. Hence, good sources are chosen by onlookers. Scout bees’s job is to randomly explorenew food sources. When onlookers and scout bees find a new food source, they become employed bees.

Since Tabu search starts with a random solution, the performance of Tabu search is easily influenced by the initial solution. Therefore, we combine it with the artificial bee colony algorithm to help alleviate this shortcoming. Artificial bee colony can generate a set of initial solutions for Tabu search, which eliminates the influence from single initial solutions. Also, the Tabu principle helps artificial bee colony generate better food sources: it encourages the colony to explore new food sources rather than coming back to previously explored sources. So a hybrid algorithm (Tabu-ABC) is developed in this paper.

Tabu-ABC (Algorithm 3) starts by generating random solutions as the food sources and associating each source with some employed bees. Before associating, Tabu search is applied to improve initial solutions, which are food sources. Then each employed bee determines a new food sources in the neighbourhood of its associated food source. If it finds a new better food sources, it will leave the old one and move to the new one. After all employed bees finish their works in a fixed iteration, they share information with onlookers. According to the traditional roulette wheel selection, onlookers select food sources. After onlookers have selected their food sources, they explore and evaluate new food sources around its chosen food source. For each old food source, if a new better food source is found, the old one is replaced by the new one. Also, a food source is abandoned if it has not been improved in a predetermined iteration times. In this case, the employed bee becomes a scout, and associates itself with a new food source. Here we introduce Tabu again. The similarity of the new food source and old food sources cannot exceed a predetermined limit. New food source are randomly generate until one satisfies similarity demand. For the VRPTWP in this paper, the similarity of two routes, say route a and route b, is determined by the length of their longest common subsequence divided by the length of route a. The similarity of two solutions, say solution A and solution B, is determined by the average of highest similarities of all A’s routes to B’s routes. The algorithm is terminated as it satisfies a termination condition, such as a predetermined iteration number.

Algorithm 3. The framework of Tabu-ABC

```
Tabu-ABC
1.  Find original food sources $x_i, i = 1,2,\dots,n$, with initialisation;
2.  Evaluate the fitness of each food source $f(x_i), i = 1,2,\dots,n$;
3.  Improve the fitness of food sources with Tabu search (Searching for new food sources in neighbourhood, if a new food source is better than old one
    and not in the Tabu tenure, replace the old one with it);
4.  No update times: $l_1 = l_2 = \cdots = l_n = 0$;
5.  **While** not satisfy termination condition **do**
6.          Employed bees stage: search $x_i'$ in neighbourhood of $x_i$, **if** $f(x_i')$ is greater than $f(x_i)$, replace $x_i$ with $x_i', l_i = 0$; **else** $l_i = l_i + 1$;
7.          Onlookers stage: Select a $x_i'$ according to the traditional roulette wheel selection; find the best $x_i'$ in neighbourhood of $x_i$, **if** $f(x_i')$ is
    greater than $f(x_i)$, replace $x_i$ with $x_i', l_i = 0$; **else** $l_i = l_i + 1$;
8.          Scout bees stage: for each food source $x_i$, if $l_i = limit$, find a new food sources that similarity with all existed food sources is lower than
    the predetermined limit, and replace $x_i$ with it;
9.  **End while**.
```

4. Computational Results

To obtain effective solutions for the VRPTWP and to provide a basis of comparison for further study, we construct a set of benchmarks for the VRPTWP and report their computational results. To generate the benchmarks, we combine two famous instances. One is the set of instances proposed in [38] , the 3L-CVRP, and the other is the well known set of the VRPTW introduced by Solomon [39]. The positions information in the 3L-CVRP instances are replaced by 27 instances (C1, C2, and R101 - R110) of Solomon's one by one. Meanwhile, the time window information is imported, too. To keep the feasibility of instances, double vehicles are supplied. We generated a set of benchmarks[1] and tested the proposed approach. The computational results will serve as baseline approaches for any future developments in this field.

In this section, the computational results on the VRPTWP and the VRPTW are presented. The proposed algorithm is coded in C++ and run on a machine with Intel Core i5 CPU, 2.6GHz/8G of RAM.

4.1. Sensitivity analysis of parameters

Considering the influence of parameters, after numerous experiments, we choose some parameters and do some experiments with different parameters setting. In order to know the relationship between the consumption of time and quality of solutions, we applied Tabu search on four instances with 800 iterations in 10 runs. Fig.5 presents the process of

[1] https://github.com/maomiT/VRPTWP

Tabu search on instances r104, r208, rc104 and rc208, where the horizontal axis shows the number of iterations, the vertical axis shows the total travel distance, which is also the cost of a solution. The figure shows the improvements are not significant after 300 iterations (finally reaching the lowest point at 1037.95, 758.38, 1224.74 and 946.74 respectively). In Fig. 6, the average results from different Tabu tenures are shown. The travel distance with Tabu tenure 30 is shorter generally. In order to test more parameters on Tabu-ABC, Fig. 7 presents the influence of update restriction (the maximum allowable number of iterations without an update) and Fig. 8 presents the influence of similarity restriction (the least similarity that allows scout bees to accept a new source). According to Fig.7 and Figl8, update restriction and similarity restriction show~~s~~ insignificant influence on final results. Although different values of update restriction may cause a large influence on specific instances (such as r104 and rc208), they still show similar results on average. Changing the similarity restriction basically do not have any clear impact on the results, althouth when similarity restriction is 0.7, the average result of four instances is slightly better. According to the figures, we can conclude that Tabu-ABC is robust against changes in the parameters.

Based on the sensitivity analysis, the chosen value for each parameter is as follow: For the experiments on the VRPTWP instances, the length of Tabu tenure is 30 and the Tabu search is conducted within 300 iterations, the group size of food sources is 10, the limit of no update times is 10n (n is the number of customers), the limit of sources' similarity is 0.7, and the ABC is conducted 50n iterations. For the experiments on the VRPTW, Tabu search is conducted within 500 iterations, the population size of food sources is 20, ABC is conducted 200n iterations, and others are the same with the VRPTWP instances.

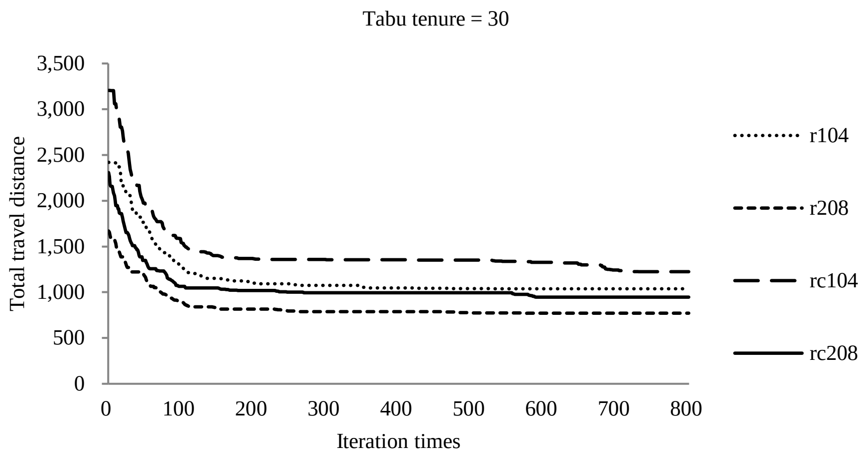


Fig. 5. Process of *Tabu search* on instances.

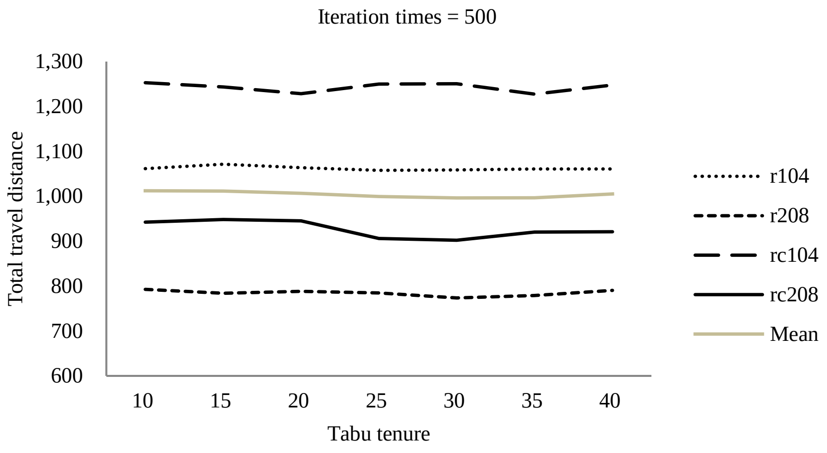


Fig. 6. Results on different *Tabu tenure.*

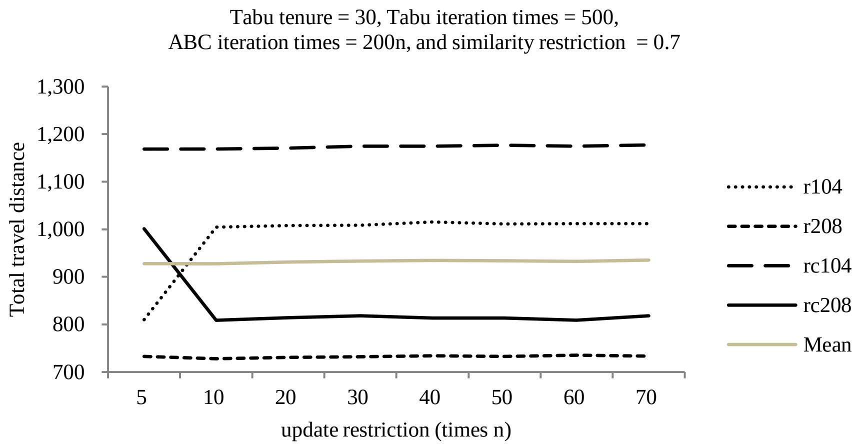


Fig. 7. Results on different update restriction.

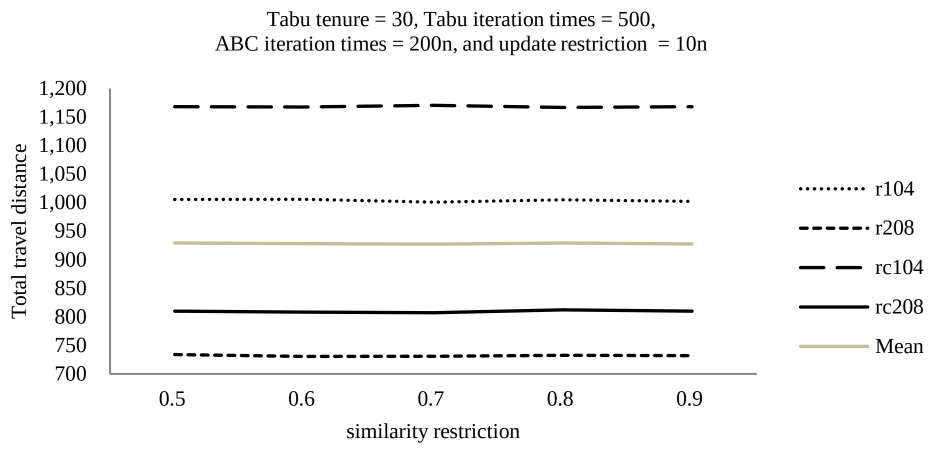


Fig. 8. Results on different similarity restriction.

4.2. Computational Results on the VRPTWP instances

Results on the VRPTWP are presented in table II, in which "NV" represents the number of vehicle, "TD" means the total travel distance (solution cost) and "CPU" denotes the computational times (in second). For each instance, the results were averaged over 10 runs. The proposed Tabu-ABC algorithm is computationally expensive, because it deals with the three-dimensional loading problem that is also computationally expensive (some computational results of three-dimensional loading problem are presented in [37]) According to the table II, a greater fleet size does not necessarily mean a greater cost. As can be seen in the VRPTWP15 and VRPTWP22, the lowest cost actually has a greater fleet size than the average fleet size over all runs. Same situations are shown in table V and table VI.

The parameters setting are shown in table I.

TABLE I

PARAMETERS SETTING

Tabu tenure	30
Tabu search iterations	300 in the VRPTWP and 500 in the VRPTW
group size of food sources	10 in the VRPTWP and 20 in the VRPTW
limit of no update times	10n, n = the number of customers
the limit of sources' similarity	0.7
ABC iteraiion times	50n in the VRPTWP and 200n in the VRPTW
	n = the number of customers

.

TABLE II

RESULTS OF THE VRPTWP

Data Set	Best		Mean		Gap	Sd	CPU
	NV	TD	NV	TD			
VRPTWP01	5	322.33	5	322.33	0	0	325.10
VRPTWP02	5	295.29	5	299.89	0.02	3.61	232.28
VRPTWP03	5	303.60	5	303.60	0	0	430.27
VRPTWP04	6	380.19	6	380.80	0	0.57	371.88
VRPTWP05	7	416.35	7	417.66	0	2.95	545.72
VRPTWP06	7	408.99	7	409.09	0	0.07	306.38
VRPTWP07	7	407.91	7	409.89	0	1.81	604.70
VRPTWP08	7	425.90	8	425.90	0	0	637.32
VRPTWP09	8	530.50	10	532.63	0	1.80	549.60
VRPTWP10	10	668.74	11	669.11	0	0.83	956.40
VRPTWP11	11	619.34	9	626.27	0.01	4.21	1032.98
VRPTWP12	9	688.60	10.40	690.10	0	1.56	671.37
VRPTWP13	11	626.72	8.40	630.43	0.01	4.40	1347.24
VRPTWP14	8	765.37	12	766.78	0	1.93	1221.68
VRPTWP15	12	713.23	10.60	715.72	0	3.41	1091.86
VRPTWP16	11	746.67	11.80	748.15	0	0.83	429.32
VRPTWP17	15	994.28	15.20	997.27	0	4.55	484.99
VRPTWP18	18	1206.51	18	1207.60	0	1.49	636.18
VRPTWP19	16	1211.99	15.60	1217.63	0	4.87	783.59
VRPTWP20	24	1644.56	23.20	1652.96	0.01	6.52	1512.11
VRPTWP21	23	1603.88	22	1612.42	0.01	9.29	2174.74
VRPTWP22	26	1811.19	26.20	1817.75	0	5.83	2170.98
VRPTWP23	24	1654.13	24	1675.25	0.01	15.22	1950.79
VRPTWP24	21	1644.55	21.40	1649.61	0	7.22	1745.54
VRPTWP25	27	1836.02	26.80	1856.52	0.01	16.43	2877.48
VRPTWP26	32	2144.47	32	2160.37	0.01	10.42	3250.49
VRPTWP27	28	2002.63	29.60	2030.71	0.01	21.35	3037.26

$\text{Gap} = (\text{Mean TD} - \text{Best TD})/\text{Best TD}$, and Sd is the standard deviation of 15 times TD.

4.3. Computational Results on the VRPTW instances

The contribution of our work in this paper is not only proposing the new VRPTWP, but also proposing a new strategy which is Tabu-ABC. To quantify the performance of the algorithm in this paper, in this section, the algorithm runs in the set of Solomon's 100 customers VRPTW~~P~~, and results are compared with other heuristic approaches. Solomon's VRPTW are divided into six sets C1, C2, R1, R2, RC1 and RC2. The customers in sets C1 and C2 are clustered in groups, in sets R1 and R2 they are uniformly distributed, and in sets RC1 and RC2, they are semi-clustered. The proposed algorithm has given 10 independent runs on each instance.

In order to prove the effectiveness of Tabu-ABC, results are compared with the results from nine other heuristic approaches. The comparison is presented in Table III, which presents the average best total distance and number of vehicles for each set. The proposed algorithm achieves the best result in C1, R2 and RC2, and the average result also better than some heuristics. Table IV compares the average mean total distance and number of vehicles for each set. The results show that Tabu-ABC achieves the best solution in four sets, which are R1, R2, RC1 and RC2, and it also achieves the best average value over all sets.

In table V we compare Tabu-ABC with heuristics from some recent published papers on the VRPTW. According to the table, in most instances, the proposed algorithm achieves the best solutions. We also compare our work with the best-known solutions from the literature in Table VI. Most of the best-known results are summarized in [44], and some values are updated according to some papers which we have known. According to the table, out of 56 instances our algorithm achieves better solutions in 15 instances. equals the best-known solutions in 4 other instances, and achieves near best solutions in all other instances. Also, Tabu-ABC achieves better results on the four specific instances on Fig.5, which demonstrates that Tabu-ABC improves the pure Tabu effectively. Table VI illustrates the fact that solving this type of problem is computational expensive (see the CPU time column), and Table VII demonstrates that Tabu-ABC can significantly decrease computational time by using the right set of parameters (the left side of Table VII) while still achieve similar high quality solutions.

TABLE III

COMPARISON AMONG DIFFERENT HEURISTICS ON THE VRPTW (AVERAGE OF BEST SOLUTIONS)

Date Set		Tan et al. (2006) [43]	Yu et al. (2011) [44]	Cordeau and Maischberger (2011) [49]	Gong et al. (2012) [45]
C1	NV	10	10	10	10
	TD	828.71	829.01	828.38	835.91
C2	NV	3	3.3	3	3
	TD	590.07	590.78	589.86	593.41
R1	NV	12.92	13.1	12	12.58
	TD	1187.35	1196.96	1209.19	1232.28

R2	NV	3.55	4.6	2.73	3
	TD	951.74	951.36	951.17	1016.66
RC1	NV	12.38	12.7	11.5	12.13
	TD	1355.37	1380.55	1385.9	1385.47
RC2	NV	4.25	5.6	3.25	3.38
	TD	1068.26	1095.84	1120.53	1169.07
Avg.	NV	6.59	7.04	6.07	6.30
	TD	996.92	1007.42	1014.17	1038.80

		Barbucha. (2014) [46]	Luo et al. (2015) [47]	Yassen et al. (2015) [48]	Tabu-ABC
C1	NV	10	10	-	10
	TD	828.38	828.38	838.47	828.38
C2	NV	3	3	-	3
	TD	589.86	589.86	605.41	590.39
R1	NV	11.92	11.92	-	13.75
	TD	1232.13	1210.34	1207.76	1187.90
R2	NV	3.09	2.73	-	4.64
	TD	922.48	951.03	977.19	891.24
RC1	NV	12	11.5	-	13.13
	TD	1355.36	1384.16	1381.96	1361.08
RC2	NV	3.38	3.25	-	5.5
	TD	1106	1119.24	1099.12	1017.47
Avg.	NV	7.23	7.07	-	8.34
	TD	1005.70	1013.84	1018.32	979.41

TABLE IV

COMPARISON AMONG DIFFERENT HEURISTICS ON THE VRPTW (AVERAGE OF MEAN SOLUTIONS)

Date Set		Chiang and Russell (1997) [41]	Lau et al. (2003) [42]	Tan et al. (2006) [43]	Yu et al. (2011) [44]
C1	NV	10	10	-	10
	TD	828.38	828.38	837.21	841.92
C2	NV	3	3	-	3.3
	TD	591.42	589.86	632.42	612.75
R1	NV	12.17	12	-	13.1
	TD	1204.19	1217.73	1240.31	1213.16
R2	NV	2.73	2.73	-	4.6
	TD	986.32	967.75	1068.57	952.3
RC1	NV	11.88	11.63	-	12.7
	TD	1397.44	1382.42	1381.23	1415.62
RC2	NV	3.25	3.25	-	5.6
	TD	1229.54	1129.19	1154.88	1120.37
Avg.	NV	7.17	7.10	-	8.22
	TD	1039.55	1019.22	1052.44	1026.02

		Cordeau and Maischberger (2011) [49]	Gong et al. (2012) [45]	Luo et al. (2015) [47]	Tabu-ABC
C1	NV	10	10	10	10
	TD	828.94	856.44	828.38	828.73
C2	NV	3	3.03	3	3
	TD	590.85	612.93	589.86	591.45
R1	NV	12.02	13.01	11.92	13.82
	TD	1213.57	1263.25	1210.75	1195.49
R2	NV	2.73	3.1	2.7	4.5

	TD	959.62	1073.72	951.51	902.88
RC1	NV	11.55	12.66	11.5	13.56
	TD	1386.39	1400.97	1384.62	1373.25
RC2	NV	3.25	3.59	3.25	5.47
	TD	1130.27	1228.95	1119.63	1028.92
Avg.	NV	7.09	7.57	7.06	8.39
	TD	1018.27	1072.71	1014.13	986.79

TABLE V

COMPARISON AMONG FOUR HEURISTICS

Algorithm	HSFLA (2015) [47]		CPLA (2014) [46]		PITSH (2012) [49]		Tabu-ABC	
Data Set	TD	NV	TD	NV	TD	NV	TD	NV
C101	828.94	10	828.94	10	828.94	10	828.94	10
C102	828.94	10	828.94	10	828.94	10	828.94	10
C103	828.06	10	828.06	10	828.07	10	828.07	10
C104	824.78	10	824.78	10	824.78	10	824.78	10
C105	828.94	10	828.94	10	828.94	10	828.94	10
C106	828.94	10	828.94	10	828.94	10	828.94	10
C107	828.94	10	828.94	10	828.94	10	828.94	10
C108	828.94	10	828.94	10	828.94	10	828.94	10
C109	828.94	10	828.94	10	828.94	10	828.94	10
C201	591.56	3	591.56	3	591.56	3	591.56	3
C202	591.56	3	591.56	3	591.56	3	591.56	3
C203	591.17	3	591.17	3	591.17	3	591.17	3
C204	590.6	3	590.6	3	590.6	3	594.89	3
C205	588.88	3	588.88	3	588.88	3	588.88	3
C206	588.49	3	588.49	3	588.49	3	588.49	3
C207	588.29	3	588.29	3	588.29	3	588.29	3
C208	588.32	3	588.32	3	588.32	3	588.32	3
R101	1650.8	10	1656.21	19	1650.8	19	1643.18	20
R102	1486.12	17	1501.97	17	1486.12	17	1460.26	18
R103	1292.67	13	1295.6	13	1294.23	13	1217.39	15
R104	1007.31	9	1017.38	9	981.2	10	987.61	11
R105	1377.11	14	1381.89	14	1377.11	14	1363.91	15
R106	1252.03	12	1258.76	12	1252.62	12	1247.90	13
R107	1104.66	10	1117.85	10	1104.66	10	1087.50	12
R108	960.88	9	976.06	9	963.99	9	961.85	11
R109	1194.73	11	1229.71	11	1194.73	11	1152.99	13
R110	1118.84	10	1196.49	10	1118.84	10	1091.50	12
R111	1096.73	10	1123.64	10	1096.73	10	1067.46	12
R112	982.14	9	1030.02	9	989.27	9	973.25	10
R201	1252.37	4	1253.02	4	1252.37	4	1174.69	6
R202	1191.7	3	1086.08	4	1191.7	3	1046.10	5
R203	939.5	3	945.8	3	941.08	3	884.02	5
R204	825.52	2	752.13	3	825.52	2	750.40	4
R205	994.43	3	1017.93	3	994.43	3	960.75	5
R206	906.14	3	920.37	3	906.14	3	900.97	4
R207	890.61	2	815.26	3	890.61	2	809.72	4
R208	726.82	2	729.42	2	726.82	2	723.14	5
R209	909.16	3	916.33	3	909.16	3	863.12	5
R210	939.37	3	943.1	3	939.37	3	927.54	5
R211	885.71	2	767.82	3	885.71	2	763.22	4
RC101	1696.95	14	1626.09	15	1696.95	14	1646.17	16

RC102	1554.75	12	1486.17	13	1554.75	12	1481.61	14
RC103	1261.67	11	1268.79	11	1261.67	11	1280.76	12
RC104	1135.48	10	1136.27	10	1135.48	10	1162.03	11
RC105	1629.44	13	1542.29	14	1633.72	13	1545.30	16
RC106	1424.73	11	1394.1	12	1424.73	11	1401.17	14
RC107	1230.48	11	1234.06	11	1232.2	11	1235.28	12
RC108	1139.82	10	1155.1	10	1147.69	10	1136.35	11
RC201	1406.94	4	1435.27	4	1406.94	4	1271.78	7
RC202	1365.64	3	1162.8	4	1367.09	3	1116.21	6
RC203	1049.62	3	1062.32	3	1050.64	3	941.81	5
RC204	798.46	3	799.08	3	798.46	3	801.87	4
RC205	1297.65	4	1303.68	4	1297.65	4	1165.82	7
RC206	1146.32	3	1155.33	3	1153.61	3	1072.85	5
RC207	1061.14	3	1095.37	3	1061.14	3	977.11	5
RC208	828.14	3	834.16	3	828.71	3	792.33	5

TABLE VI

DETAIL RESULTS OF OUR ALGORITHM AND COMPARISON WITH BEST-KNOWN SOLUTIONS

	Best-known			This work						
Data Set	NV	TD	Authors	Best TD	NV	Mean TD	NV	Gap	Sd	CPU
C101	10	827.3	Desrochers,Desrosiers, and Solomon (1992) [50]	828.94	10	828.94	10	0	0	3592.54
C102	10	827.3	Desrochers et al. (1992) [50]	828.94	10	828.94	10	0	0	668.07
C103	10	826.3	Tavares, Pereira, Machado, and Costa (2003) [51]	828.07	10	828.07	10	0.00	0.00	252.88
C104	10	822.9	Tavares et al. (2003) [51]	824.78	10	827.93	10	0.54	5.57	140.18
C105	10	827.3	Tavares et al. (2003) [51]	828.94	10	828.94	10	0	0	1204.79
C106	10	827.3	Desrochers et al. (1992) [50]	828.94	10	828.94	10	0	0	1184.74
C107	10	827.3	Tavares et al. (2003) [51]	828.94	10	828.94	10	0	0	782.00
C108	10	827.3	Tavares et al. (2003) [51]	828.94	10	828.94	10	0	0	498.37
C109	10	827.3	Tavares et al. (2003) [51]	828.94	10	828.94	10	0	0	252.69
C201	3	589.1	Cook and Rich (1999) [52]	591.56	3	591.56	3	0	0	4155.89
C202	3	589.1	Cook and Rich (1999) [52]	591.56	3	591.56	3	0	0	630.48
C203	3	591.17	Li and Lim (2003) [53]	591.17	3	591.31	3	0.35	3.66	278.99
C204	3	590.6	Potvin and Bengio (1996) [40]	594.89	3	603.16	3	2.95	10.12	152.45
C205	3	588.88	Potvin and Bengio (1996) [40]	588.88	3	588.88	3	0.00	0.00	1731.04
C206	3	588.49	Lau te al. (2003) [42]	588.49	3	588.49	3	0.00	0.00	1181.93
C207	3	588.29	Rochat and Tailard (1995) [54]	588.29	3	588.29	3	0	0	853.12
C208	3	588.03	Tan et al. (2006) [43]	588.32	3	588.32	3	0	0	476.75
R101	18	1607.7	Desrochers et al. (1992) [50]	1643.18	20	1645.82	20	0.39	5.48	1137.93
R102	17	1434	Desrochers et al. (1992) [50]	1460.26	18	1463.91	18.08	0.43	4.00	434.26
R103	13	1175.67	Lau, Lim, and Liu (2001) [55]	1217.39	15	1223.27	14.92	0.30	2.41	398.70
R104	10	974.2	Tan et al. (2006) [43]	987.61	11	1002.54	11.33	0.92	6.04	227.74
R105	15	1346.12	Kallehauge, Larsen, and Madsen (2006) [56]	1363.91	15	1372.01	15.75	0.68	5.45	641.26
R106	13	1234.6	Cook and Rich (1999) [52]	1247.90	13	1256.45	13.42	0.93	4.76	457.90
R107	11	1051.84	Kallehauge et al. (2006) [56]	1087.50	12	1097.41	12.00	1.03	4.46	336.94
R108	10	954.03	Tan et al. (2006) [43]	961.85	11	965.82	10.83	1.25	5.91	220.71
R109	12	1013.2	Chiang and Russell (1997) [41]	1152.99	13	1163.07	13.00	1.23	7.20	365.31
R110	12	1068	Cook and Rich (1999) [52]	1091.50	12	1100.83	12.17	1.49	8.51	340.76
R111	12	1048.7	Cook and Rich (1999) [52]	1067.46	12	1076.61	12	0.98	7.16	302.90
R112	10	953.63	Rochat and Tailard (1995) [54]	973.25	10	978.16	10.67	0.71	6.47	240.36
R201	8	1198.15	Tan et al. (2001) [57]	1174.69	6	1178.90	6.08	0.46	4.85	547.67
R202	6	1077.66	Tan et al. (2001) [57]	1046.10	5	1053.44	5.08	1.41	7.88	331.63
R203	5	933.286	Tan et al. (2001) [57]	884.02	5	896.05	4.92	1.43	7.77	273.62
R204	3	752.13	Barbucha (2014) [46]	750.40	4	758.13	4	1.52	6.50	230.77

R205	3	994.42	Rousseau, Gendreau, and Pesant (2002) [58]	960.75	5	975.83	4.92	1.36	7.61	290.46
R206	3	833	Thangiah, Osman, and Sun (1994) [59]	900.97	4	908.18	4.25	1.61	6.76	226.36
R207	3	814.78	Rochat and Tailard (1995) [54]	809.72	4	826.74	4	2.15	11.31	234.11
R208	2	729.42	Barbucha (2014) [46]	723.14	5	732.31	3.42	1.37	6.99	198.33
R209	3	855	Thangiah et al. (1994) [59]	863.12	5	882.22	4.92	1.35	6.71	226.84
R210	3	943.10	Barbucha (2014) [46]	927.54	5	938.63	4.92	2.09	9.31	232.11
R211	2	767.82	Barbucha (2014) [46]	763.22	4	781.23	4	2.18	7.61	189.63
RC101	15	1619.8	Kohl, Desrosiers, Madsen, Solomon, and Soumis (1999) [61]	1646.17	16	1656.01	16.33	0.66	6.93	663.29
RC102	13	1470.26	Tan et al. (2006) [43]	1481.61	14	1488.79	14.83	0.81	7.29	441.74
RC103	12	1196.12	Tan et al. (2006) [43]	1280.76	12	1298.32	12.08	1.94	18.50	308.63
RC104	10	1135.48	Cordeau, Laporte, and Mercier (2001) [62]	1162.03	11	1168.13	11.00	0.60	5.43	217.15
RC105	14	1542.29	Barbucha (2014) [46]	1545.30	16	1554.79	16.17	1.57	8.50	545.30
RC106	13	1371.69	Tan et al. (2006) [43]	1401.17	14	1413.38	14.00	0.82	7.41	301.56
RC107	11	1222.16	Tan et al. (2006) [43]	1235.28	12	1256.98	12.50	1.96	11.91	277.52
RC108	11	1133.82	Luo, Li, Chen and Liu (2015) [47]	1136.35	11	1149.65	11.17	1.67	9.76	235.20
RC201	6	1134.91	Tan et al. (2006) [43]	1271.78	7	1284.59	6.92	0.78	5.71	476.51
RC202	5	1130.53	Tan et al. (2006) [43]	1116.21	6	1122.97	5.75	0.71	5.25	350.15
RC203	4	1026.61	Tan et al. (2006) [43]	941.81	5	951.30	5.08	0.79	6.33	226.52
RC204	3	799.08	Barbucha (2014) [46]	801.87	4	809.09	4	2.22	8.13	162.89
RC205	5	1295.46	Tan et al. (2006) [43]	1165.82	7	1172.80	7	2.23	16.34	335.38
RC206	4	1112.2	Yu et al. (2011) [44]	1072.85	5	1082.93	5.50	1.15	8.40	272.82
RC207	4	1040.67	Tan et al. (2006) [43]	977.11	5	998.46	5.42	2.34	9.82	211.82
RC208	3	829.69	Rousseau et al. (2002) [58]	792.33	5	809.23	4.67	2.11	8.65	191.15

$\text{Gap} = (\text{Mean TD} - \text{Best TD})/\text{Best TD}$, and Sd is the standard deviation of 15 times TD.

TABLE VII

DETAIL COMPARISON BETWEEN TWO PARAMETERS SETTING

Data Set	Tabu tenure = 30, Tabu iteration times = 500, ABC iteration times = 200n, similarity restriction = 0.7 and update restriction = 10n					Tabu tenure = 30, Tabu iteration times = 500, ABC iteration times = 200n, similarity restriction = 0.6, and update restriction = 50n				
	Best TD	NV	Mean TD	NV	CPU	Best TD	NV	Mean TD	NV	CPU
C101	828.94	10	828.94	10	3592.54	828.94	10	828.94	10	1070.42
C102	828.94	10	828.94	10	668.07	828.94	10	828.94	10	280.49
C103	828.07	10	828.07	10	252.88	828.07	10	828.07	10	116.66
C104	824.78	10	827.93	10	140.18	824.78	10	829.26	10	60.94
C105	828.94	10	828.94	10	1204.79	828.94	10	828.94	10	478.01
C106	828.94	10	828.94	10	1184.74	828.94	10	828.94	10	389.41
C107	828.94	10	828.94	10	782.00	828.94	10	828.94	10	322.39
C108	828.94	10	828.94	10	498.37	828.94	10	828.94	10	185.52
C109	828.94	10	828.94	10	252.69	828.94	10	828.94	10	83.06
C201	591.56	3	591.56	3	4155.89	591.56	3	591.56	3	1337.71
C202	591.56	3	591.56	3	630.48	591.56	3	591.56	3	348.48
C203	591.17	3	591.31	3	278.99	591.17	3	593.21	3	132.09
C204	594.89	3	603.16	3	152.45	590.6	3	608.03	3	53.45
C205	588.88	3	588.88	3	1731.04	588.88	3	588.88	3	619.37
C206	588.49	3	588.49	3	1181.93	588.49	3	588.49	3	473.96
C207	588.29	3	588.29	3	853.12	588.29	3	588.29	3	322.58
C208	588.32	3	588.32	3	476.75	588.32	3	588.32	3	234.39
R101	1643.18	20	1645.82	20	1137.93	1644.5	20	1650.89	20.27	521.88
R102	1460.26	18	1463.91	18.08	434.26	1463.52	18	1469.81	18.07	187.46
R103	1217.39	15	1223.27	14.92	398.70	1224.44	15	1228.17	14.8	141.13
R104	987.61	11	1002.54	11.33	227.74	1001.89	12	1011.12	11.8	85.29

R105	1363.91	15	1372.01	15.75	641.26	1369.07	16	1378.44	15.93	235.14
R106	1247.90	13	1256.45	13.42	457.90	1251.71	13	1263.34	13.73	174.85
R107	1087.50	12	1097.41	12.00	336.94	1093.99	12	1105.3	11.93	130.82
R108	961.85	11	965.82	10.83	220.71	958.44	11	970.46	10.87	87.48
R109	1152.99	13	1163.07	13.00	365.31	1160.06	13	1174.37	13.27	162.67
R110	1091.50	12	1100.83	12.17	340.76	1095.34	12	1111.61	12.13	154.76
R111	1067.46	12	1076.61	12	302.90	1073.29	12	1083.78	12.2	131.07
R112	973.25	10	978.16	10.67	240.36	981.67	11	988.63	10.87	118.97
R201	1174.69	6	1178.90	6.08	547.67	1178.91	6	1184.33	6.07	196.05
R202	1046.10	5	1053.44	5.08	331.63	1041.48	5	1056.15	5.2	119.36
R203	884.02	5	896.05	4.92	273.62	889.4	5	902.09	4.53	85.53
R204	750.40	4	758.13	4	230.77	748.15	4	759.52	4	69.77
R205	960.75	5	975.83	4.92	290.46	968.89	5	982.09	4.8	116.61
R206	900.97	4	908.18	4.25	226.36	897.2	4	911.63	4	78.55
R207	809.72	4	826.74	4	234.11	812.61	4	830.07	4	73.13
R208	723.14	5	732.31	3.42	198.33	725.58	4	735.53	3.13	65.18
R209	863.12	5	882.22	4.92	226.84	872.65	5	884.44	4.93	90.78
R210	927.54	5	938.63	4.92	232.11	919.41	5	938.66	4.8	95.88
R211	763.22	4	781.23	4	189.63	769.28	4	786.04	4	77.49
RC101	1646.17	16	1656.01	16.33	663.29	1650.2	16	1661.04	16.4	271.72
RC102	1481.61	14	1488.79	14.83	441.74	1481.48	14	1493.5	14.8	193.72
RC103	1280.76	12	1298.32	12.08	308.63	1288.36	12	1313.42	12.13	143.69
RC104	1162.03	11	1168.13	11.00	217.15	1174.46	11	1181.55	11.27	92.34
RC105	1545.30	16	1554.79	16.17	545.30	1535.94	15	1560.12	16.53	195.66
RC106	1401.17	14	1413.38	14.00	301.56	1411.51	14	1423.01	13.87	163.77
RC107	1235.28	12	1256.98	12.50	277.52	1236.27	12	1260.54	12.07	137.43
RC108	1136.35	11	1149.65	11.17	235.20	1131.4	11	1150.3	11.4	132.3
RC201	1271.78	7	1284.59	6.92	476.51	1285.14	7	1295.15	6.73	174.15
RC202	1116.21	6	1122.97	5.75	350.15	1116.58	6	1124.5	5.87	122.95
RC203	941.81	5	951.30	5.08	226.52	944.87	5	952.32	5	82.79
RC204	801.87	4	809.09	4	162.89	791.76	4	809.33	4.2	56.04
RC205	1165.82	7	1172.80	7	335.38	1163.29	7	1189.25	6.6	115.08
RC206	1072.85	5	1082.93	5.50	272.82	1070.82	6	1083.09	5.47	98.9
RC207	977.11	5	998.46	5.42	211.82	980.87	5	1003.8	5.13	85.6
RC208	792.33	5	809.23	4.67	191.15	799.11	4	816	4.73	81.55

5. Conclusion

This paper introduces a vehicle routing problem with time windows and pallet loading constraints, and proposes a new algorithm named Tabu-ABC. The VRPTWP comprises two sub NP-hard problems, namely the three-dimensional loading problem and the VRPTW. In addition, the VRPTWP addressed in this paper closely reflects real-world situations, and time window constraints are considered. To the best of our knowledge, the VRPTWP is a new problem that has never been addressed before. We apply Tabu-ABC to solve the VRPTWP and create a set of benchmark for the new VRPTWP. Tabu-ABC is a hybrid algorithm with Tabu search and ABC. According to the comparison among some heuristics on

Solomon’s VRPTW instances, it is proved to be effective. As a future work, we would like to develop more efficient approach to solving VRPTWP, since the CPU times cost of Tabu-ABC to solve VRPTWP is not stable and high sometimes. Meanwhile, Tabu-ABC can be applied to other problems also, and we will consider using it in other problems.

Acknowledgements

The authors would like to thank the referee for their valuable comments that help improve this paper. This work is partially supported by the National Nature Science Foundation of China (Grant no. 61272003) and by the University of Macau under grant MYRG2015-00054-FST, and is also partially supported by two grants from the British Council, a UK-ASEAN Knowledge Partnership grant and a Newton Institutional Links grant, and a Newton Research Collaboration Programme (3) funded by the Royal Academy of Engineering.

References

[1] G. B. Dantzig and J. H. Ramser, The truck dispatching problem, Management Science 6 (1) (1959) 80-91.

[2] S. C. H. Leung, Z. Zhang, D. Zhang, X. Hua and M. K. Lim, A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints, European Journal of Operational Research 225 (2) (2013) 199-210.

[3] C.D. Tarantilis, E. E. Zachariadis and C. T. Kiranoudi, A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem, IEEE Transaction on Intelligent Transportation Systems 10 (2) (2009) 255-271.

[4] A. Bortfeldt and J. Homberger, Packing first, routing second-a heuristic for the vehicle routing and loading problem, Computers and Operations Research 40 (3) (2013) 873-885.

[5] L. Junqueira, J. F. Oliveira, M. A. Carravilla and R. Morabito. An optimization model for the vehicle routing problem with practical three-dimensional loading constraints, International Transactions in Operational Research 20 (5) (2013) 645-666.

[6] R. Tavakkoli-Moghaddam, M. Gazanfari, M. Alinaghian, A. Salamatbakhsh and N. Norouzi, A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing, Journal of Manufacturing Systems 30 (2) (2011) 83-92.

[7] M. Hifi and L. Wu. A hybrid metaheuristic for the vehicle routing problem with time windows. 2014 International Conference on Control, Decision and Information Technologies (CoDIT), Metz, 2014, pp. 188-194.

[8] Y. J. Gong, Jun Zhang, O. Liu, R.-Z. Huang, H. S.-H. Chung and Y.-H. Shi, Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach, IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (2) (2012) 254-267.

[9] M. Cherkesly, G. Desaulniers and G. Laporte, A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading, Computers and Operations Research 62 (2015) 23-35.

[10] L. Wei, Z. Zhang and L. Andrew, An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints, Computational Intelligence Magazine 9 (4) (2014) 18-30.

[11] E. Silva, J. F. Oliveira and G. Wäscher, The pallet loading problem: a review of solution methods and computational experiments, International Transactions in Operational Research (2014) DOI:10.1111/itor.12099.

[12] E. E. Zachariadis, C. D. Tarantilis and C. T. Kiranoudis, Designing vehicle routes for a mix of different request types, under time windows and loading constraints, European Journal of Operational Research 229 (2) (2013) 303-317.

[13] M. L. Balinski, and R. E. Quandt, On an integer program for a delivery problem, Operations Research 12 (2) (1964) 300-304.

[14] S. Eilon, C. D. T. Watson-Gandy, and N. Christofides, Distribution management, Griffin, London, 1971.

[15]N. Christofides, A. Mingozzi and P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxation,Mathematical Programming 20 (1) (1981) 255-282.

[16]M. L. Fisher and R. Jaikumar, A generalized assignment heuristic for vehicle routing, Networks 11 (2) (2006) 109-124.

[17]G. Laporte, Y. Nobert and M. Desrochers, Optimal routing under capacity and distance restrictions. Operations Research 33 (5) (1985) 1050-1073.

[18]M. Gendreau, A. Hertz and G. Laporte. A tabu search heuristic for the vehicle routing problem, Management Science 40 (10) (1994) 1276-1290.

[19]J. Renaud, G. Laporte and F. F. Boctor, A tabu search heuristic for the multi-depot vehicle routing problem, Computers and Operations Research 23 (3) (1996) 229-235.

[20]É. Taillard, P. Badeau, M. Gendreau, F. Guertin and J. Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, Transportation Science 31 (2) (1997) 170-186.

[21]J.-F. Cordeau, G. Laporte and A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows, Journal of the Operational Research Society 52 (8) (2001) 928-936.

[22]P. Toth and D. Vigo, The granular tabu search and its application to the vehicle-routing problem, INFORMS Journal on Computing 15 (5) (1998) 333-346.

[23]F. A. T. Montané and R. D. Galvao, A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, Computers and Operations Research 33 (3) (2004) 595-619.

[24]E. E. Zachariadis, C. D. Tarantilis and C. T. Kiranoudis, A guided tabu search for the vehicle routing problem with two-dimensional loading constraints, European Journal of Operational Research 195 (3) (2009) 729-743.

[25]S. C. H. Leung, X. Zhou, D. Zhang and J. Zheng, Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem, Computers and Operations Research 38 (1) (2011) 205-215.

[26]D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, 2005.

[27]W. Y. Szeto, Yongzhong Wu and Sin C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem, European Journal of Operational Research 215(1) (2011) 126-135.

[28]B. Z. Yao, P. Hu, M. H. Zhang and S. Wang, Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem, Simulation 89 (6) (2013) 762-770.

[29]S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, Journal of Statistical Physics 34 (1984) 975-986.

[30]J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT Press, Cambridge, MA, 1992.

[31]A. V. Breedam, Improvement heuristics for the vehicle routing problem based on simulated annealing, European Journal of Operational Research 6 (3) (1995) 480-490.

[32]B. M. Baker and M. A. Ayechew, A genetic algorithm for the vehicle routing problem, Computers and Operations Research 30 (5) (2003) 787-800.

[33]I. H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, Annals of Operations Research 41 (4) (1993) 421-451.

[34]T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi and W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research 60 (3) (2011) 611-624.

[35]A. Bortfeldt, T. Hahn, D. Männel and L. M. Hybrid, algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints, European Journal of Operational Research 243 (1) (2015) 82-96.

[36]M. Hifi, I. Kacem, S. Nègre and L. Wu, A linear programming approach for the three-dimensional bin-loading problem, Electronic Notes in Discrete Mathematics 36 (2010) 993-1000.

[37]D. F. Zhang, L. J. Wei, Q. S. Chen and H. W. Chen, A combinational heuristic Algorithm for the three-Dimensional packing problem, Journal of Software, 18 (9) (2007) 2083-2089.

[38]M. Gendreau, M Iori, G. Laporte and S. Martello. A tabu search algorithm for a routing and container loading problem, Transportation Science, 40 (3) (2006) 342-350.

[39]M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Operations Research 35 (2) (1987) 254-265.

[40]J. Y. Potvin and S. Bengio, The vehicle routing problem with time windows part II: genetic search. INFORMS journal on Computing 8 (2) (1996) 165-172.

[41]W. C. Chiang and R. A. Russell, A reactive tabu search metaheuristic for the vehicle routing problem with time windows, INFORMS Journal on computing 9 (4) (1997) 417-430.

[42]H. C. Lau, M. Sim and K. M Teo. Vehicle routing problem with time windows and a limited number of vehicles, European Journal of Operational Research 148 (3) (2003) 559-569.

[43]K. C. Tan, Y. H. Chew and L. H. Lee, A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows, Computational Optimization and Applications 34 (1) (2006) 115-151.

[44]B. Yu, Z. Z. Yang and B. Z. Yao, A hybrid algorithm for vehicle routing problem with time windows, Expert Systems with Applications 38 (1) (2011) 435-441.

[45]Y. J. Gong, et al, Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42 (2) (2012) 254-267.

[46]D. Barbucha, A cooperative population learning algorithm for vehicle routing problem with time windows, Neurocomputing 146 (2014) 210-229.

[47]J. P. Luo, X. Li, M. R. Chen and H. W. Liu, A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows, Information Sciences 316 (2015) 266-292.

[48]E. T. Yassen, M. Ayob, M. Z. A. Nazri and N. R. Sabar, Meta-harmony search algorithm for the vehicle routing problem with time windows, Information Sciences 325 (2015) 140–158.

[49]J-F. Cordeau and M. Maischberger, A parallel iterated tabu search heuristic for vehicle routing problems, Computer & Operations Research 39 (2012) 2033-2050.

[50]M. Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Operations research 40 (2) (1992) 342-354.

[51]J. Tavare, P. Machado, F. B. Pereira and E, Costa, On the influence of GVR in vehicle routing, Proceedings of the 2003 ACM symposium on Applied computing, New York, 2003, pp. 753-758.

[52]W. Cook and J. L. Rich, A parallel cutting-plane algorithm for the vehicle routing problem with time windows, Computational and Applied Mathematics Department, Rice University, Houston, TX, Technical Report, 1999.

[53]H. Li and A. Lim, Local search with annealing-like restarts to solve the VRPTW, European journal of operational research 150 (1) (2003) 115-127.

[54]Y. Rochat, É. D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of heuristics 1 (1) (1995) 147-167.

[55]H. C. Lau, Y. F. Lim and Q. Z. Liu, Diversification of search neighborhood via constraint-based local search and its applications to VRPTW, 3rd international workshop on integration of AL and OR techniques, Kent, United Kingdom, 2001.

[56]B. Kallehauge, J. Larsen and O. B. G. Madsen, Lagrangian duality applied on vehicle routing with time windows, Computer & Operations Research 33 (5) (2006) 1464-1487.

[57]K. C. Tan, L. H. Lee and K. Ou, Artificial intelligence heuristics in solving vehicle routing problems with time window constraints, Engineering Applications of Artificial Intelligence 14 (6) (2001) 825-837.

[58]L. M. Rousseau, M. Gendreau and G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, Journal of heuristics 8 (1) (2002) 43-58.

[59]S. R. Thangiah, I. H. Osman and T. Sun, Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows, Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27, 1994.

[60]J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows, Infor-Information Systems and Operational Research 37 (3) (1999) 297-318.

[61]N. Kohl, J. Desrosiers, O. B. G. Madsen, M. Solomon and F. Soumis, Two-path cuts for the vehicle routing problem with time windows, Transportation Science 33 (1) (1999) 101-116.

[62]J. F. Cordeau, G. Laporte and A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows, Journal of the Operational research society 52 (8) (2001) 928-936.