European Journal of Operational Research 225 (2013) 199–210

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints

Stephen C.H. Leung [a], Zhenzhen Zhang [b], Defu Zhang [b,*], Xian Hua [b], Ming K. Lim [c]

[a] Department of Management Sciences, City University of Hong Kong, Hong Kong
[b] Department of Computer Science, Xiamen University, Xiamen 361005, China
[c] Derby Business School, University of Derby, Derby, UK

ARTICLE INFO

Article history:
Received 18 October 2011
Accepted 16 September 2012
Available online 3 October 2012

Keywords:
Routing
Packing
Simulated annealing
Heterogeneous fleet

ABSTRACT

The two-dimensional loading heterogeneous fleet vehicle routing problem (2L-HFVRP) is a variant of the classical vehicle routing problem in which customers are served by a heterogeneous fleet of vehicles. These vehicles have different capacities, fixed and variable operating costs, length and width in dimension, and two-dimensional loading constraints. The objective of this problem is to minimize transportation cost of designed routes, according to which vehicles are used, to satisfy the customer demand. In this study, we proposed a simulated annealing with heuristic local search (SA_HLS) to solve the problem and the search was then extended with a collection of packing heuristics to solve the loading constraints in 2L-HFVRP. To speed up the search process, a data structure was used to record the information related to loading feasibility. The effectiveness of SA_HLS was tested on benchmark instances derived from the two-dimensional loading vehicle routing problem (2L-CVRP). In addition, the performance of SA_HLS was also compared with three other 2L-CVRP models and four HFVRP methods found in the literature.

1. Introduction

The vehicle routing problem (VRP) was firstly addressed by Dantzig and Ramser (1959), proposing the most cost-effective way to distribute items between customers and depots by a fleet of vehicles. Taking into account of the attribute of the fleet, the traditional VRP has evolved to different variants. Amongst them include CVRP (homogenous VRP) that only considers a constraint of vehicles having the same limited capacity (Rochat and Taillard, 1995), HVRP (heterogeneous VRP) that serves customers with different types of vehicles (Golden et al., 1984; Gendreau et al., 1999; Lima et al., 2004; Prins, 2009; Brandao, 2011), VRPTW (VRP with time windows) that requires the service of each customer to start within the time window subject to time windows constraints (Kolen et al., 1987); and SDVRP (split deliver VRP) that allows more than one vehicle serving a customer (Chen et al., 2007). Readers are to refer to Crainic and Laporte (1998) and Toth and Vigo (2002) for a detailed description of VRP and its variants. To solve the VPR variants above effectively, a number of metaheuristics have been applied, such as simulated annealing (Osman, 1993), Tabu search (Brandao, 2011; Gendreau et al., 1999), genetic algorithms (Lima et al., 2004), variable neighborhood search (Imran et al., 2009), and ant colony optimization (Rochat and Taillard, 1995; Li et al., 2009).

In the real world, logistics managers have to deal with routing and packing problems simultaneously. This results in another domain of VRP to be investigated. In the literature, there are a number of frameworks proposed to address these two problems simultaneously. Iori et al. (2007) addressed the VRP with two-dimensional packing constraints (2L-CVRP) with an algorithm based on branch-and-cut technique. Gendreau et al. (2008) proposed a Tabu search heuristic algorithm to solve large instances with up to 255 customers and more than 700 items in the 2L-CVRP. Zachariadis et al. (2009) developed a new meta-methodology guided Tabu search (GTS) which can obtain better results. In this work, a collection of packing heuristics was proposed to check the loading feasibility. Fuellerer et al. (2009) presented a new ant colony optimization algorithm deriving from saving-based ant colony optimization method and demonstrated its performance to successfully solve the 2L-CVRP. More recently, Leung et al. (2011) developed a new efficient method that consists of a series of algorithms for two-dimensional packing problems. The method has proven its capability to improve the results of most instances used by Zachariadis et al. (2009). Duhamel et al. (2011) proposed a GRASP × ELS algorithm for 2L-CVRP, whereby the loading constraints were transformed into resource constrained project scheduling problem (RCPSP) constraints before a packing problem can be solved. However, only basic CVRP and *Unrestricted* version

* Corresponding author. Tel.: +86 592 2582013; fax: +86 592 2580258.
E-mail address: dfzhang@xmu.edu.cn (D. Zhang).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.09.023

of 2L-CVRP were solved with their algorithm. Some researchers have extended their heuristics to three-dimensional problems. Gendreau et al. (2006) proposed a multi-layer Tabu search algorithm that iteratively invokes an inner Tabu search procedure to search the optimal solutions of a three-dimensional loading sub-problem. Tarantilis et al. (2009) used a guided Tabu search (GTS) approach with a combination of six packing heuristics to solve 3L-CVRP. In their work, a manual unloading problem was also tested. Furthermore, Fuellerer et al. (2010) also proposed their methods to deal with three-dimensional loading constraints. In addition, Iori and Martello (2010) provided a review in regard to vehicle routing problems with two and three-dimensional loading constraints.

Since most enterprises own a heterogeneous fleet of vehicles or hire different types of vehicles to serve their customers, it is therefore crucial to study VRP with a fleet of heterogeneous vehicles. The heterogeneous fleet VRP (HFVRP) addresses the VRP with a heterogeneous fleet of vehicles which have various capacities, fixed costs and variable costs (Choi and Tcha, 2007; Imran et al., 2009). In the literature, three versions of HFVRP have been studied. Golden et al. (1984) considered the variable costs to be uniformly spread across all vehicle types and the availability of each type of vehicle to be unlimited. Gendreau et al. (1999) considered the different variable costs for different types of vehicle. The third HFVRP was introduced by Taillard (1999) and Tarantilis et al. (2004), in which the number of available vehicles of each type is limited. Recently, Penna et al. (2011) introduced an Iterated Local Search, combined with a Variable Neighborhood Decent procedure and a random neighborhood ordering (ILS-RVND), to solve all variants of HFVRP.

In this paper, we combined the HFVRP with two-dimensional loading constraints, called the heterogeneous fleet vehicle routing problems with two-dimensional loading constraints (2L-HFVRP). However, to the best of our knowledge, no work has been conducted to address such VRP although it is a practical problem in real-world transportation and logistics industries. In 2L-HFVRP, there are different types of vehicles with different capacity, fixed cost, variable cost, length and width in vehicle dimension, and two-dimensional loading constraints. The demand of a customer is defined by a set of rectangular items with given width, length and weight. All the items belonging to one customer must be assigned to the same route. The objective is to describe the minimum transportation costs with a function of the distance travelled, fixed and variable costs associated with the vehicles.

This paper presents a simulated annealing (SA) algorithm for 2L-HFVRP. In the literature SA has been proven to be an effective method to solve combinatorial optimization problems and it has been successfully applied to 2L-CVRP (Leung et al., 2010). In this paper, a heuristic local search is used to further improve the solution of SA. In addition, six promising packing algorithms, whereby five were developed by Zachariadis et al. (2009) and one by Leung et al. (2010), are also used to solve the loading constraints in 2L-HFVRP. These algorithms are extensively tested on benchmark instances derived from the 2L-CVRP test problems with vehicles of different capacity, fixed and variable costs, length, and width. The comparison with several effective methods of the 2L-CVRP and pure HFVRP is also given.

2. Problem description

The 2L-HFVRP is defined on an undirected connected graph $G = (V,E)$, where $V = \{0,1,\ldots,n\}$ is a vertex set corresponding to the depot (vertex 0) and the customers (vertices 1, 2, ..., n) and $E = \{e_{ij}: i,j \in V\}$ is an edge set. For each $e_{ij} \in E$, a distance d_{ij} ($d_{ii} = 0$) is associated. A fleet of P different types of vehicles is located at the depot, and the number of vehicles of each type is

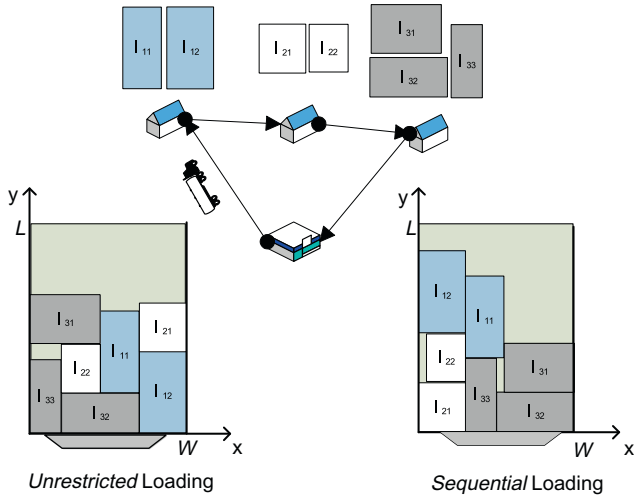

Fig. 1. The unrestricted loading and sequential loading processes.

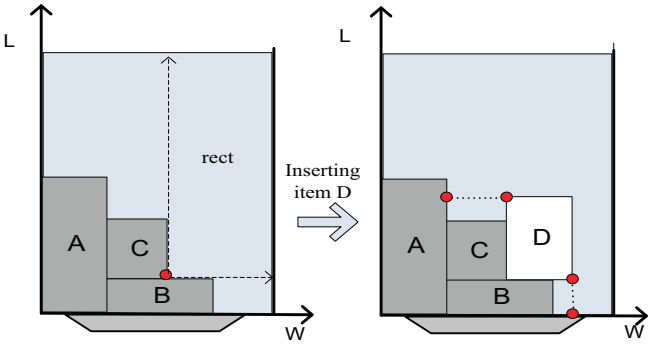

Fig. 2. The process of inserting an item.

unlimited. Capacity Q_t, fixed cost F_t, variable cost V_t, length L_t and width W_t are associated to each type of vehicle t ($t = 1, 2, \ldots, P$). The loading surface of vehicle of type t is $A_t = L_t * W_t$.On the basis that a vehicle with larger capacity usually has higher cost and greater fuel consumption, we assume that $Q_1 \leqslant Q_2 \leqslant \cdots \leqslant Q_P$, $F_1 \leqslant F_2 \leqslant \cdots \leqslant F_P$ and $V_1 \leqslant V_2 \leqslant \cdots \leqslant V_P$. The traveling cost of each edge $e_{ij} \in E$ by a vehicle type t is $C_{ij}^t = V_t * d_{ij}$. The transportation cost of a route for vehicle type t is $C_R = F_t + \sum_{i=1}^{i<|R|} V_t * d_{R(i),R(i+1)}$, where R is the route whose start point and end point are the depot. Each customer i ($i = 1, 2, \ldots, n$) demands a set of m_i rectangular items, denoted as IT_i, and the total weight of IT_i equals to D_i. Each item $I_{ir} \in IT_i$ ($r = 1, 2, \ldots, m_i$) has a specific length l_{ir} and width w_{ir}. We also denote $a_i = \sum_{r=1}^{m_i} w_{ir} * l_{ir}$ as the total area of the items of customer i. In 2L-HFVRP, a feasible loading must satisfy the following constraints:

(i) All items of a given customer must be loaded on the same vehicle and split deliveries are not allowed.
(ii) All items must have a fixed orientation and must be loaded with their sides parallel to the sides of the loading surface.
(iii) Each vehicle must start and finish at the depot.
(iv) Each customer can only be served once.
(v) The capacity, length and width of the vehicle cannot be exceeded.
(vi) No two items can overlap in the same route.

The objective of 2L-HFVRP is to assign customer i ($i = 1, 2, \ldots, n$) to one of the routes, so that the total transportation cost is minimized and all the routes fulfill the constraints. In this paper, we

Table 1
The pseudo-code for the packing heuristics.

```
Is_Feasible(Route r)
  if total weight of all items exceeds the capacity then
    return false
  end if
  sort the items to generate two orderings Ord1, Ord2
  for each ordering Ordi of the two orderings do
    if Heur1||Heur2||Heur3|| Heur4||Heur5||Heur6 then
      return true
    end if
  end for
  return false
```

consider two versions of 2L-HFVRP which is the same as 2L-CVRP: the *Unrestricted* only deals with feasible loading of the items unto the vehicles, and the *Sequential* considers both loading and unloading constraints (e.g. when visiting a customer, his/her items can be unloaded without the need to move items that belong to other customers in the same route). Fig. 1 gives an example of the two versions.

3. The optimization heuristics for two-dimensional loading problems

For a given route, it is necessary to determine whether all the items required by the customers can be feasibly loaded onto the vehicle. In this paper, we will first investigate if the total weight of items demanded by the customers exceeds the capacity of the vehicle. Otherwise, six packing heuristics are used to solve the two-dimensional loading problem. As mentioned earlier, the loading position of an inserted item must be feasible, i.e. it must not lead to any overlaps (for both *Unrestricted* and *Sequential* problems), or sequence constraint violations (for *Sequential* only). The first five heuristics Heur_i ($i = 1, 2, \ldots, 5$) are based on the work by Zachariadis et al. (2009). Each heuristic loads an item in the most suitable position selected from the feasible ones according to the individual criterion as follows:

- Heur_1: Bottom-left fill (W-axis).
  The selected position is the one with the minimum W-axis coordinate, breaking ties by minimum L-axis coordinate.
- Heur_2: Bottom-left fill (L-axis).
  The selected position is the one with the minimum L-axis coordinate, breaking ties by minimum W-axis coordinate.
- Heur_3: Max touching perimeter heuristic.
  The selected position is the one with the maximum sum of the common edges between the inserted item, the loaded items in the vehicle, and the loading surface of the vehicle.
- Heur_4: Max touching perimeter no walls heuristic.
  The selected position is the one with the maximum sum of the common edges between the inserted item and the loaded items in the vehicle.
- Heur_5: Min area heuristic.
  The selected position is the one with the minimum rectangular surface. The rectangular surface corresponding to the position at the circle point is shown on the left in Fig. 2.
  More details of these five heuristics can be found in Zachariadis et al. (2009). In order to handle a more complex system Heur_6 was also used, which was proposed in Leung et al. (2010).
- Heur_6: Max fitness value heuristic.
  This heuristic gives a priority to a loading point if it can decrease the number of corner positions when we place an item on the point. As a result, every time an item is loaded, we will select the best loading point for the item, which would increase the probability to obtain a better loading position.

Table 2
The pseudo-code of SA_HLS for the 2L-HFVRP.

```
SA_HLS_2L-HFVRP(customer demands, vehicle information)
  Generate initial Order through sorting the customers by decreasing total
   weight
  Assign_Vehicle (Order) to construct the initial solution
  Tk = T0, Iter = 0//Iter is the number of iteration
  while stopping criteria not met do
    for i = 1 to Len do
      if Iter < 10 then
        generate a new Order based on the old one
        Assign_Vehicle (new Order)
        if the new solution is packing-feasible and better than the current
   one then
            accept the new solution as current solution
        end if
        accept the new Order based on the acceptance rule of SA
      end if
      stochastically select NS from {NS1,NS2, NS3}, then get a feasible solution
      if new solution is better than the current one then
        accept the new solution as the current solution
      else
        accept the new solution through the acceptance probability function
      end if
      Local_Search (), and get a new feasible solution
      if new solution is better than the best one then
        replace the current solution with this new one
      end if
      update the best solution when the solution is better than it
    end for
    Tk = 0.9 * Tk, Iter = Iter + 1
  end while
  return the best solution
```

These six heuristics are called in sequence, which means if Heur_1 fails to produce a feasible loading solution, the more complex Heur_i ($i = 2, \ldots, 6$) will be called one at a time to find the solution. If a feasible solution is found, the loading process stops and the solution is stored.

During the loading process, feasible loading positions are recorded. At first, only the front left corner (0,0) is available. When an item is successfully inserted, four new positions are added onto the list, and the occupied and duplicated positions are removed. As shown in Fig. 2, item D is inserted in the position shown by a circle and four new positions are created.

The items are loaded one at a time according to a given sequence. Here, two orders (Ord_1,Ord_2) are generated. In a given route, each customer has a unique visit order. Ord_1 is produced by sorting all items by reverse customer visit order, and breaking ties by decreasing area. In Ord_2, all items are simply sorted by decreasing area. Both orders will be evaluated by the six heuristics to search for feasible loading solutions. The pseudo-code for the packing heuristics is given in Table 1.

4. The simulated annealing meta-heuristics for 2L-HFVRP

Simulated annealing (SA) is a point-based stochastic optimization method, which explores iteratively from an initial solution to a better result (Cerny, 1985; Kirkpatrick et al., 1983). The search mechanism of SA has a very good convergence, and it has been widely applied in solving various NP-hard problems. Each iteration

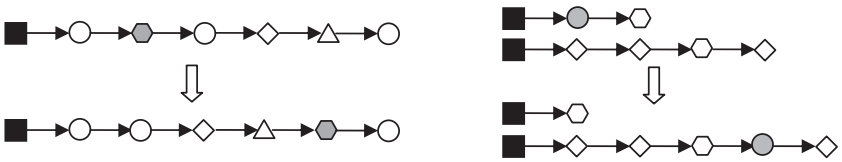

Fig. 3. Customer relocation move.

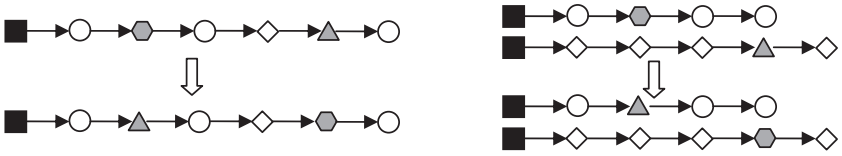

Fig. 4. Route exchange move.

Table 3
The pseudo-code for assigning customers into vehicles.

```
Assign_Vehicle (Order)
  iused[1,2,...,P] = {0}
  for each customer i in Order do
    while true do
      select the vehicle k which is not tabu for i and has minimum
   (freeD_k − D_i) * F_k
      insert the customer i at the last position of route for vehicle k
      if !Is_Feasible (route) then
        iused[P] = iused[P] + 1;//add a new vehicle with largest capacity
        Tabu this vehicle k for customer i
      else
        if freeD_k < MinD then//vehicle k cannot service any customer
          iused[t] = iused[t] + 1//assuming the type of vehicle k is t, add one
   new vehicle
        end if
        accept the new route, and break the loop//start to assign successive
   customer
      end if
    end while
  end for
  return generated solution
```

in SA generates a candidate solution using a neighborhood function. This is a vital step to develop an efficient SA. However, in many cases, the neighborhood function alone is inadequate when seeking for a global optimum solution. In addition to the proposed SA, we also use heuristic local search algorithms to improve the solutions. Therefore, our algorithm is denoted as SA_HLS. Some mechanisms are adopted to adjust the search trajectory.

One important characteristic of SA is that it can accept a worse solution on a probabilistic manner, aiming to search for a better result. With the initial temperature T_0, the temperature cooling schedule is $T_k = 0.9 * T_{k-1}$. For a specific temperature T_k, a sequence of moves are carried out, which is a Markov chain whose length is denoted as *Len*. In every iteration, after applying the neighborhood function, if the new solution is better than the current solution (i.e. the cost is lower), then it is accepted. However, if the cost is higher, the new solution may be accepted subject to the acceptance probability function $p(T_k, S_{new}, S_{cur})$, which depends on the difference between the corresponding cost values and the global parameter T_k:

$$p(T_k, S_{new}, S_{cur}) = \exp\left(\frac{\cos t(S_{cur}) - \cos t(S_{new})}{T_k}\right) \quad (1)$$

where S_{cur} and S_{new} represent the current solution and the new solution respectively. Table 2 provides a framework for the proposed SA_HLS methodology.

4.1. Initial solution

Good initial solutions are often a key to the overall efficiency of the metaheuristic. We construct the initial solution focusing on the demand of the customers, so that the use of different types

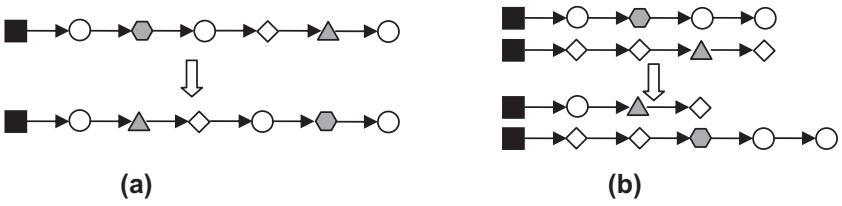

Fig. 5. Route interchanging move.

Table 4
The characteristics of items of Classes 2–5 instances.

Class	m_i	Vertical		Homogeneous		Horizontal	
		Length	Width	Length	Width	Length	Width
2	[1,2]	[0.4*L*,0.9*L*]	[0.1*W*,0.2*W*]	[0.2*L*,0.5*L*]	[0.2*W*,0.5*W*]	[0.1*L*,0.2*L*]	[0.4*W*,0.9*W*]
3	[1,3]	[0.3*L*,0.8*L*]	[0.1*W*,0.2*W*]	[0.2*L*,0.4*L*]	[0.2*W*,0.4*W*]	[0.1*L*,0.2*L*]	[0.3*W*,0.8*W*]
4	[1,4]	[0.2*L*,0.7*L*]	[0.1*W*,0.2*W*]	[0.1*L*,0.4*L*]	[0.1*W*,0.4*W*]	[0.1*L*,0.2*L*]	[0.2*W*,0.7*W*]
5	[1,5]	[0.1*L*,0.6*L*]	[0.1*W*,0.2*W*]	[0.1*L*,0.3*L*]	[0.1*W*,0.3*W*]	[0.1*L*,0.2*L*]	[0.1*W*,0.6*W*]

Table 5
Dataset for different types of vehicle.

Inst	A					B					C					D				
	Q_A	L_A	W_A	F_A	V_A	Q_B	L_C	W_C	F_C	V_C	Q_C	L_C	W_C	F_C	V_C	Q_D	L_D	W_D	F_D	V_D
1	20	10	10	10	1.0	25	15	15	20	1.1	40	25	25	30	1.2	60	40	20	40	1.3
2	20	10	10	10	1.0	25	15	15	20	1.1	40	25	25	30	1.3	55	40	20	40	1.5
3	20	10	10	10	1.0	30	15	15	20	1.1	60	40	20	40	1.2					
4	20	10	10	10	1.0	40	20	20	20	1.1	60	40	20	30	1.2					
5	1000	10	10	10	1.0	2500	15	15	20	1.1	4000	25	25	30	1.3	6000	40	20	50	1.5
6	2000	10	10	10	1.0	2500	15	15	20	1.1	4000	40	20	30	1.3					
7	200	10	10	10	1.0	500	15	15	20	1.1	2000	25	25	120	6.0	4500	40	20	250	8
8	200	10	10	10	1.0	500	15	15	20	1.1	2000	25	25	120	5.0	4500	40	20	250	10
9	20	10	10	10	1.0	25	15	15	20	1.1	48	40	20	30	1.3					
10	200	10	10	10	1.0	500	15	15	20	1.1	2000	25	25	120	8.0	4500	40	20	250	10
11	200	10	10	10	1.0	500	15	15	20	1.1	2000	25	25	120	8.0	4500	40	20	250	10
12	20	10	10	10	1.0	25	15	15	20	1.1	40	40	20	30	1.3					
13	2000	10	10	10	1.0	5000	15	15	50	2.0	30,000	40	20	200	10					
14	500	10	10	10	1.0	1500	15	15	50	2.1	3000	20	20	400	3.2	5000	40	20	800	5
15	500	10	10	10	1.0	1500	15	15	50	2.1	3000	20	20	400	3.2	5000	40	20	800	5
16	20	10	10	10	1.0	40	20	20	20	1.1	60	40	20	30	1.2					
17	20	10	10	10	1.0	25	15	15	20	1.1	40	25	25	30	1.3	60	40	30	40	1.4
18	200	10	10	10	1.0	500	20	20	30	2.0	2000	40	20	120	5.0					
19	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	150	40	20	90	5.0
20	2000	10	10	10	1.0	4000	20	10	20	1.1	10,000	30	15	60	4.0	30,000	40	20	150	8
21	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
22	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
23	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
24	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	100	40	20	60	3.2
25	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
26	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
27	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	100	40	20	60	3.2
28	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
29	200	10	10	10	1.0	500	20	10	30	2.0	2000	40	20	120	8.0					
30	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
31	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
32	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
33	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
34	20	10	10	10	1.0	40	20	10	20	1.1	60	20	15	30	1.2	200	40	20	120	8
35	200	10	10	10	1.0	400	20	10	20	1.5	1000	40	20	60	4.0					
36	100	10	10	10	1.0	200	15	15	20	1.1	300	30	20	30	1.2	400	40	20	40	1.3

Table 6
Calibration experiment result for T_0 and *Len*.

	5		15		**25**		35	
	S	Sec_{tot}	S	Sec_{tot}	S	Sec_{tot}	S	Sec_{tot}
T_0								
Unrest	5266.65	327.94	5159.32	288.99	**5094.74**	402.30	5111.83	500.08
Seq	5427.43	461.79	5274.77	510.39	**5198.10**	580.95	5238.46	680.85
	3000		**5000**		7000		9000	
Len								
Unrest	5144.37	383.37	5094.74	402.30	**5078.97**	779.14	5083.18	968.25
Seq	5328.87	456.85	5198.10	580.95	5206.40	1025.19	**5191.07**	1242.93

of vehicles can be maximized. Firstly, all of the n customers are sorted on decreasing value of D_i ($i = 1, 2, \ldots, n$), where D_i is the total demand of customer i ($i = 1, 2, \ldots, n$) and the sequence is recorded as *Order*. Subsequently we assign the customers one at a time from the *Order* list to a vehicle. The decision of which vehicle is assigned to a given customer is based on the least value of $(freeD_k - D_i) * F_k$, where $freeD_k$ is the unused capacity and F_k is the fixed cost of the current vehicle k (procedure Assign_Vehicle ()). Because the number of each type of vehicle is unlimited, the procedure always finds a feasible solution. Table 3 provides a pseudo-code for the proposed Assign_Vehicle () algorithm. *iused* is an array presenting the number of used vehicles of different types. *MinD* is the minimal demand in all the customers. When assigning one customer i to vehicle k, the feasibility is examined to ensure the loading for the modified route is feasible. Otherwise, the assignment of customer i to vehicle k is forbidden and the procedure tries to assign the customer i to another vehicle.

As shown in Table 2, this procedure is used in SA. In each loop, a partial segment of *Order* is reversed to get a new *Order*. Then, we reassign the customers using this method. If a new solution is better than the current one, it becomes the new current solution in order to adjust the search trajectory. This is assumed that previous solution does not have a good characteristic that can be improved easily. In order to obtain a better solution, the new *Order* is adopted based on the SA acceptance rule. After several steps of improvement by SA, the solution constructed is usually not comparable to the current one. So this method is only applied during the first ten iterations.

Table 7
Result comparison of SA_HLS and SA on Class 1.

Inst	SA			SA_HLS			%Gap
	S	Sec_h	Sec_{tot}	S	Sec_h	Sec_{tot}	
1	665.64	6.41	162.40	596.07	29.78	33.59	10.45
2	732.85	39.30	159.30	679.18	5.05	32.01	7.32
3	813.87	162.92	182.44	745.51	12.39	54.24	8.40
4	745.50	142.72	184.50	694.33	6.15	18.36	6.86
5	916.40	214.52	226.97	761.19	5.62	26.99	16.94
6	814.85	20.02	192.53	809.56	4.77	25.65	0.65
7	3387.06	159.89	215.36	3211.53	3.67	29.76	5.18
8	3359.77	212.59	215.03	3184.45	3.21	24.30	5.22
9	1144.65	120.11	219.14	1029.95	7.39	40.28	10.02
10	5400.74	108.70	274.94	5149.51	6.97	25.88	4.65
11	5465.11	217.80	279.00	5119.40	6.97	26.47	6.33
12	1699.59	17.67	261.11	1658.56	35.80	92.80	2.41
13	19390.00	56.72	275.67	14655.40	1.93	32.18	24.42
14	10447.10	39.97	305.77	10019.00	13.51	73.21	4.10
15	10546.90	42.11	304.92	10151.70	5.49	55.29	3.75
16	1391.84	108.34	297.56	1292.58	17.94	76.92	7.13
17	1963.58	53.33	411.94	1770.83	38.88	225.59	9.82
18	4055.02	343.92	363.92	3140.55	13.54	35.35	22.55
19	1980.40	205.31	445.78	1553.11	32.56	107.81	21.58
20	3244.47	277.20	719.06	1956.97	56.00	71.57	39.68
21	5330.48	428.97	703.95	2567.18	75.00	195.66	51.84
22	5934.32	160.30	699.75	2605.90	76.39	174.72	56.09
23	5811.71	404.05	711.09	2643.84	93.99	239.29	54.51
24	4257.31	289.19	696.34	2555.41	63.98	156.41	39.98
25	5960.19	248.30	908.83	2972.59	129.04	253.09	50.13
26	5515.48	43.45	804.55	4049.64	88.09	180.23	26.58
27	5093.65	426.08	905.03	3561.58	159.20	230.49	30.08
28	7944.73	82.84	1103.06	6858.35	125.60	161.05	13.67
29	15643.10	822.16	1222.33	9695.00	139.73	142.63	38.02
30	11320.50	992.75	1500.93	5663.33	242.15	259.83	49.97
31	19297.80	1754.36	1946.53	8054.90	325.51	483.44	58.26
32	17767.80	732.97	1965.67	8408.61	379.53	410.86	52.68
33	18325.90	696.91	1962.11	8555.58	368.50	486.25	53.31
34	5713.85	1406.47	1998.50	5536.63	323.50	425.80	3.10
35	4875.69	0.98	1793.80	4444.59	324.64	401.58	8.84
36	4961.22	1723.8	2322.95	3669.89	555.13	605.31	26.03
Avg.							23.07

Table 8
Average computational results of Classes 2–5 for *Sequential* 2L-HFVRP.

Inst	SA			SA_HLS			%Gap
	S	Sec_h	Sec_{tot}	S	Sec_h	Sec_{tot}	
1	678.49	7.73	23.43	603.15	5.73	31.04	11.10
2	753.70	6.43	21.53	705.03	6.09	31.28	6.46
3	866.56	20.89	41.05	771.81	10.35	36.61	10.93
4	796.28	22.93	40.76	704.87	8.71	35.19	11.48
5	944.87	34.68	49.07	802.56	9.35	27.62	15.06
6	901.44	25.33	45.42	834.76	9.92	43.90	7.40
7	6634.13	57.86	94.40	5770.83	1.95	31.40	13.01
8	7064.92	27.52	74.11	5633.04	4.77	37.42	20.27
9	1181.86	48.39	56.42	1047.60	15.74	68.84	11.36
10	8695.22	114.15	194.21	7730.73	5.87	51.78	11.09
11	9789.89	171.07	225.85	8491.94	10.07	58.88	13.26
12	1707.99	15.11	23.73	1681.61	32.90	158.45	1.54
13	35464.18	183.90	287.93	26761.40	6.71	65.34	24.54
14	12027.55	46.68	112.05	11120.30	11.29	55.62	7.54
15	12871.23	118.99	190.72	11916.30	24.53	148.50	7.42
16	1437.74	11.15	45.81	1291.24	25.73	113.02	10.19
17	2037.01	24.87	35.34	1775.49	43.41	198.83	12.84
18	8364.54	344.74	600.49	5790.62	30.50	138.03	30.77
19	6186.88	369.25	781.22	4303.26	51.44	233.98	30.45
20	9586.34	941.00	1534.62	6215.48	98.14	217.73	35.16
21	14457.73	811.10	1613.92	8494.36	124.15	443.85	41.25
22	15677.58	1090.28	1798.92	8867.10	110.40	353.40	43.44
23	15533.28	1078.92	1615.03	8544.30	130.44	386.82	44.99
24	6756.02	595.33	1092.09	4714.08	104.98	287.02	30.22
25	22864.75	2937.97	3155.74	11602.30	186.28	605.53	49.26
26	20622.13	875.11	2351.06	12380.30	153.21	392.26	39.97
27	9652.10	2343.76	2607.82	5882.75	240.04	502.40	39.05
28	41547.50	2484.48	4178.81	23585.60	393.72	737.35	43.23
29	42142.58	3444.27	5609.81	22938.80	486.55	1045.69	45.57
30	36243.78	3609.86	7783.73	16489.70	536.78	1033.79	54.50
31	49143.85	3493.46	11927.09	22033.00	1110.80	1531.28	55.17
32	49142.65	4755.45	12121.48	20982.70	1040.06	1487.74	57.30
33	50660.65	3911.46	12173.27	21906.60	1087.93	1326.98	56.76
34	25388.93	780.55	14154.56	15005.00	1680.67	1887.41	40.90
35	12902.48	160.32	17513.84	9313.54	1722.96	1877.56	27.82
36	6279.65	256.37	16302.74	4567.29	2035.08	2052.99	27.27
Avg.							27.46

4.2. Neighborhood functions

In our work, three types of move are used to step from the current solution to the subsequent solutions. They are noted as NS_i ($i = 1, 2, 3$). In each loop, one of them is selected randomly with equal probability. To explore a larger search space, a dummy empty vehicle is added for each type of vehicle. NS_1 is a type of customer relocation (Or-opt) (Waters, 1987), which reassigns a customer from one route to another position on the same or different route (Fig. 3). It is worth noting that relocation between two different routes can reduce the number of vehicles required.

Waters (1987) introduced a "swap" type of route exchange which is represented by NS_2 (Fig. 4). It is only applied to vehicles of the same type as swapping loads of heterogeneous vehicles could lead to an unfeasible route from a loading perspective. Therefore, customers' positions can only be exchanged in the current solution if they belong to vehicles of the same type.

NS_3 is a variant of route interchange (2-opt) (Croes, 1958; Lin, 1965). As for NS_2, NS_3 only considers vehicles of the same type (Fig. 5). If the selected customers are in the same route as depicted in Fig. 5a, the positions of other customers between them (and including themselves) will be reversed. If they belong to different routes as illustrated in Fig. 5b, in each route from the selected customer to the last customer will be grouped as a block. Between the routes the blocks will be swapped.

4.3. Heuristic local search mechanism

In order to improve the quality of the solution, we also apply a heuristic local search mechanism, which consists of three methods, to the proposed SA algorithm. It is worth noting that we only apply the mechanism to the best solution with a probability of 5% and this is aimed to obtain more efficient solutions within a shorter period of time. The local search methods adopt the first improvement criterion using the neighborhood functions mentioned in the previous section. Because this neighborhood is not operated on two randomly selected customers, we define this mechanism as heuristic local search. We denote the local search methods as LS_i ($i = 1, 2, 3$) according to the neighborhoods NS_i ($i = 1, 2, 3$). These three methods are randomly executed.

Let us consider an instance with n customers and k vehicles. In LS_1, the relocation move of one customer involves the reassignment of $(n + k)$ positions. Hence, the complexity of examining NS_1 neighborhood of a solution is O ($n^*(n + k)$). For LS_2, in the worst case whereby all customers are assigned to one type of vehicle, n^2 pairs of customers can be exchanged and therefore the complexity of NS_2 is O (n^2). For LS_3, as for LS_2, the number of interchange points is $(n + k)$, so the cardinality of pairs for interchange in NS_3 is $(n + k)^2$. As a result, examining NS_3 neighborhood requires O ($(n + k)^2$) computational effort. In practice, the worst case hardly happened because the customers are usually spread out across different types of vehicle.

Table 9
Average computational results of Classes 2–5 for *Unrestricted* 2L-HFVRP.

Inst	SA			SA_HLS			%Gap
	S	Sec_h	Sec_{tot}	S	Sec_h	Sec_{tot}	
1	668.35	10.28	26.48	600.77	4.52	29.89	10.11
2	755.40	13.13	27.56	699.21	5.20	32.88	7.44
3	856.70	21.17	49.55	770.12	10.37	33.84	10.11
4	765.75	30.05	54.02	698.19	7.73	30.04	8.82
5	902.11	39.33	68.51	786.84	7.82	27.68	12.78
6	905.30	24.38	59.54	831.32	8.90	42.78	8.17
7	6779.89	64.65	132.14	5630.02	1.15	31.08	16.96
8	7036.90	91.22	114.14	5602.60	1.92	30.22	20.38
9	1183.96	45.78	64.45	1035.62	7.88	58.75	12.53
10	8397.92	156.13	264.91	7625.05	6.15	43.27	9.20
11	9178.77	116.22	277.96	8329.69	4.91	52.61	9.25
12	1709.22	16.00	37.74	1681.07	34.91	167.21	1.65
13	31077.83	216.41	403.16	25978.70	9.35	69.95	16.41
14	11864.33	85.23	138.80	10869.10	16.31	78.10	8.39
15	12355.43	144.88	230.36	11490.10	8.88	91.53	7.00
16	1433.28	27.82	59.66	1291.87	23.64	110.89	9.87
17	1984.71	23.49	57.94	1776.54	38.43	191.01	10.49
18	7126.99	348.14	674.90	5676.16	28.20	88.80	20.36
19	5822.84	582.01	937.74	4242.48	49.06	180.33	27.14
20	8735.97	1117.17	1722.42	6153.31	84.01	175.47	29.56
21	13922.45	1149.98	2005.18	8220.77	91.67	292.14	40.95
22	14508.43	537.29	1972.57	8574.44	96.32	317.36	40.90
23	14578.50	1087.49	1884.20	8316.57	84.47	357.79	42.95
24	6399.23	705.29	1156.35	4547.84	85.11	288.63	28.93
25	20788.20	2024.61	3324.21	11367.90	178.85	473.87	45.32
26	19654.18	569.69	2679.46	11781.50	165.73	363.10	40.06
27	9009.55	1522.14	2622.41	5695.23	194.58	282.47	36.79
28	39332.33	2386.78	4547.38	22611.00	450.17	614.35	42.51
29	34553.58	4564.89	6134.22	21876.20	325.51	494.77	36.69
30	34010.78	2179.55	6133.38	15793.10	458.44	811.79	53.56
31	47084.88	4193.16	10051.52	21125.50	750.53	1114.64	55.13
32	47402.75	2989.81	11131.47	20110.70	677.04	968.59	57.57
33	50534.23	3815.82	12505.96	21419.60	651.56	857.77	57.61
34	24771.70	90.41	12470.55	14484.50	1309.39	1596.04	41.53
35	12052.65	219.89	15793.42	8962.14	1181.83	1213.94	25.64
36	6120.32	630.43	15381.20	4385.78	1104.32	1119.91	28.34
Avg.							25.86

4.4. Stopping criterion

In order to control the trade-off between the quality of the solution and the computation time, two stopping criteria are used, which is either the current temperature lower than 0.01 or the solution unchanged for $2 * Len$ times.

4.5. Speed up the search process

For each new solution, the proposed packing heuristics are used to check the feasibility of loading. These heuristics are very computationally consuming. Therefore, we avoid duplicate examination with the use of a data structure called Trie in order to speed up the algorithm. Trie is employed to store the loading information examined. When a tentative route is examined, it is easy to retrieve the feasibility information stored in a Trie. If this route does not appear in the Trie, the packing heuristics are applied and the information is recorded. In practice, as more and more routes are examined, the required physical memory may exceed the computer memory available. Therefore, before storing the information of a new route, the capacity of computer memory is checked. If it exceeds a threshold, the Trie is destructed and rebuilt.

5. Parameter setting and computational results

This section presents the computational results based on a set of benchmark instances used by Gendreau et al. (2008), but with different types of vehicles. Firstly, a detailed discussion on the benchmark instances will be provided. Then, the simulation results of benchmark instances will be presented.

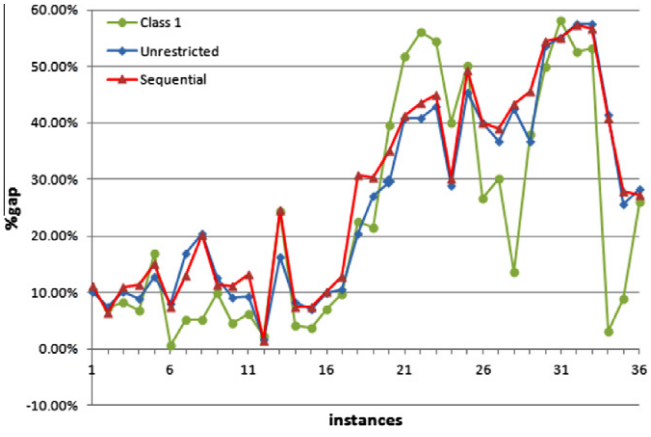

Fig. 6. The improvement percentage of SA_HLS over the SA for 2L-HFVRP.

5.1. Benchmark instances and parameter setting

The benchmark instances used for 2L-HFVRP are similar to the instances of 2L-CVRP, except in 2L-HFVRP different types of vehicles are considered. There are 36 instances in total and each has five classes. We refer the reader to Iori (2005) for further details on how the original instances were created.

Table 10
Comparison results for the *pure* CVRP of Class 1.

Inst	S				Sec_h				*%gap*		
	SA_HLS	GTS	ACO	G × E	SA_HLS	GTS	ACO	G × E	GTS	ACO	G × E
1	278.73	278.73	278.73	278.73	1.4	2.9	0.1	0.0	0.00	0.00	0.00
2	334.96	334.96	334.96	334.96	0.2	1.4	0.1	0.0	0.00	0.00	0.00
3	358.40	358.40	358.40	358.40	2.3	3.8	0.2	0.0	0.00	0.00	0.00
4	430.89	430.88	430.88	430.88	1.7	1.0	0.3	0.0	0.00	0.00	0.00
5	375.28	375.28	375.28	375.28	1.9	1.3	0.3	0.0	0.00	0.00	0.00
6	495.85	495.85	495.85	495.85	3.0	1.9	0.3	0.0	0.00	0.00	0.00
7	568.56	568.56	568.56	568.56	2.9	0.8	0.2	0.0	0.00	0.00	0.00
8	568.56	568.56	568.56	568.56	3.3	0.4	0.2	0.0	0.00	0.00	0.00
9	607.65	607.65	607.65	607.65	2.3	1.2	0.6	0.0	0.00	0.00	0.00
10	535.80	535.80	535.80	535.80	3.9	5.9	2.3	0.0	0.00	0.00	0.00
11	505.01	505.01	505.01	505.01	3.4	3.8	0.8	0.0	0.00	0.00	0.00
12	610.00	610.00	610.00	610.00	8.0	6.3	1.5	0.2	0.00	0.00	0.00
13	2006.34	2006.34	2006.34	2006.34	3.2	5.8	1.3	0.0	0.00	0.00	0.00
14	837.67	837.67	837.67	837.67	6.0	17.1	4.1	0.2	0.00	0.00	0.00
15	837.67	837.67	837.67	837.67	6.6	7.9	2.8	0.0	0.00	0.00	0.00
16	698.61	698.61	698.61	698.61	5.6	13.0	2.0	0.0	0.00	0.00	0.00
17	861.79	863.27	861.79	861.79	15.8	32.9	3.3	0.0	0.17	0.00	0.00
18	727.74	730.85	723.54	723.54	7.0	47.1	9.5	8.3	0.43	−0.58	−0.58
19	524.61	524.61	524.61	524.61	12.0	100.2	7.9	0.3	0.00	0.00	0.00
20	249.66	244.54	241.97	241.97	27.0	198.3	56.1	4.5	−2.09	−3.18	−3.18
21	696.02	687.60	690.20	687.60	15.6	221.5	26.5	1.4	−1.22	−0.84	−1.22
22	750.62	740.66	742.91	740.66	20.4	662.9	57.4	2.1	−1.34	−1.04	−1.34
23	855.51	839.07	845.34	835.26	20.1	1531.4	55.9	3391.3	−1.96	−1.20	−2.42
24	1032.65	1035.33	1030.25	1026.60	15.9	1012.7	49.8	53.3	0.26	−0.23	−0.59
25	836.90	829.45	830.82	827.39	21.9	953.8	167.1	2.4	−0.90	−0.73	−1.15
26	839.78	819.56	819.56	819.56	19.7	1031.7	175.7	0.4	−2.47	−2.47	−2.47
27	1103.16	1097.63	1100.22	1082.65	27.2	871.2	190.5	486.5	−0.50	−0.27	−1.89
28	1168.00	1042.12	1062.23	1042.12	42.5	781.4	252.5	129.8	−12.08	−9.96	−12.08
29	1217.27	1188.15	1168.13	1162.96	42.7	1641.9	769.1	549.6	−2.45	−4.21	−4.67
30	1062.27	1037.05	1041.05	1033.42	42.3	873.3	310.3	2165.9	−2.43	−2.04	−2.79
31	1464.87	1421.20	1341.89	1306.07	188.0	631.4	521.8	5096.1	−3.07	−9.16	−12.16
32	1356.23	1328.68	1334.26	1303.52	84.7	905.5	517.7	4492.4	−2.07	−1.65	−4.04
33	1344.02	1328.19	1331.69	1301.06	72.8	1708.6	476.6	4842.1	−1.19	−0.93	−3.30
34	745.00	719.91	712.32	713.51	124.8	834.1	614.5	3007.4	−3.48	−4.59	−4.41
35	911.48	877.04	868.12	870.63	124.8	907.2	1452.6	2616.5	−3.93	−4.99	−4.69
36	619.50	594.10	616.69	592.87	77.9	1492.6	1588.3	5264.7	−4.28	−0.46	−4.49
Avg.									−1.24	−1.35	−1.87

Class 1: Each customer demands one item with unit width and length. 36 instances of Class 1 are not related with packing algorithm and it is a pure HFVRP problem.

Classes 2–5: For each customer i, a set of m_i items with uniform distribution on a certain range is created (Table 4). Each item is randomly classified into one of three possible shapes with an equal probability: vertical, homogeneous and horizontal. The dimensions of the items are uniformly created in a given range (see Table 4).

For 2L-HFVRP, each benchmark data is generated by eliminating the limit of number of vehicles and adding information about capacity, loading surface, fixed and variable cost for each type of vehicle based on the corresponding instance of 2L-CVRP. For the five classes in same instance, types of vehicle are the same. The fleet is composed of four types of vehicle: A, B, C and D. For vehicles of different types, the following properties are given: capacity Q_i, length L_i, width W_i, fixed cost F_i, variable cost V_i, i = A, B, C and D. Table 5 gives details about the fleets which are created randomly.

5.2. Parameter setting

In the proposed algorithmic framework, two parameters are used: the initial temperature T_0 and the length of Markov chain *Len* which are determined by extensive experiments involving both *Unrestricted* and *Sequential* versions. 30 instances (6, 12, 18, 24, 30, 36, Classes 1–5), which cover cases with different customer set sizes and item characteristics, were used to analyze the sensitivity. The average results obtained are reported in Table 6. Where, S is the average value of solutions obtained, and Sec_{tot} is the average CPU time to run the algorithm. The best results are marked in bold.

According to Table 6, when T_0 was 25, the best results were obtained for both *Unrestricted* and *Sequential* versions. For *Len*, the results were very close when the length was more than 5000, but the computational time required was rather high. Therefore, the *Len* was set equal to 5000.

5.3. Computational results

The SA and SA_HLS for 2L-HFVRP were both coded in C++. All experiments were executed on a Core 2 Duo 2.2 GHz with 2 GB RAM PC under windows 7. Since no algorithm is reported in the published literature, we compare our proposed algorithm with the classical SA, which is executed under the same conditions as SA_HLS. The results are shown in Tables 7–9. S is the best solution we obtained, *%gap* is the percentage of improvement over the classical SA, Sec_h is the CPU time in seconds elapsed when this solution was obtained and Sec_{tot} is the CPU time in seconds to run the algorithm.

As seen in Tables 7–9, the cost was improved by 23.07%, 27.46% and 25.86% on average for Class 1, Sequential and Unrestricted Classes 2–5, respectively. The computational time by SA_HLS was also less than classical SA, especially for larger cases. The results obtained are reported in Table A1 in the Appendix.

Fig. 6 illustrates the improvement of SA_HLS over SA for all instances based on Tables 7–9. Our algorithm outperforms SA for all

Table 11
Average comparison results for the *Unrestricted* 2L-CVRP on Classes 2–5.

Inst	*S*				Sec_h				*%gap*		
	SA_HLS	GTS	ACO	G × E	SA_HLS	GTS	ACO	G × E	GTS	ACO	G × E
1	290.37	295.74	285.77	282.65	2.5	2.2	2.9	0.9	1.81	−1.61	−2.73
2	341.35	341.89	341.02	339.26	0.6	1.3	0.2	0.1	0.16	−0.10	−0.62
3	377.37	384.49	376.32	376.32	6.6	0.7	1.2	0.5	1.85	−0.28	−0.28
4	435.01	441.45	438.65	435.01	3.1	2.2	1.4	0.2	1.46	0.83	0.00
5	379.49	382.22	379.03	379.03	4.7	4.7	6.4	0.1	0.71	−0.12	−0.12
6	501.02	499.47	497.27	497.04	2.0	4.4	2.0	0.4	−0.31	−0.75	−0.80
7	698.65	703.49	696.91	691.11	10.1	4.5	4.8	1.4	0.69	−0.25	−1.09
8	703.13	705.60	691.14	678.84	15.2	6.4	10.0	0.8	0.35	−1.73	−3.58
9	612.02	615.65	612.02	612.01	4.3	5.1	2.1	0.6	0.59	0.00	0.00
10	701.61	713.00	682.53	675.79	15.9	9.5	30.1	15.1	1.60	−2.79	−3.82
11	736.30	740.04	721.82	705.95	13.5	18.1	23.6	11.3	0.50	−2.01	−4.30
12	613.94	616.83	611.26	611.26	7.8	61.9	3.8	16.9	0.47	−0.44	−0.44
13	2561.40	2599.40	2520.73	2490.62	7.4	44.4	39.1	78.0	1.46	−1.61	−2.84
14	1015.81	1036.77	991.26	984.42	34.6	167.4	96.5	79.9	2.02	−2.48	−3.19
15	1193.97	1197.83	1167.28	1144.69	44.7	86.1	68.9	257.7	0.32	−2.29	−4.31
16	701.27	702.30	699.80	699.79	8.4	78.3	6.2	6.0	0.15	−0.21	−0.21
17	865.94	864.26	864.06	864.05	15.8	26.4	4.0	21.6	−0.19	−0.22	−0.22
18	1065.76	1076.81	1043.31	1029.71	23.4	250.7	162.7	413.5	1.03	−2.15	−3.50
19	765.55	776.91	752.05	739.19	73.3	376.5	60.5	268.5	1.46	−1.80	−3.57
20	541.48	551.71	528.17	522.68	212.5	518.7	448.3	1658.3	1.85	−2.52	−3.60
21	1044.46	1050.43	1015.05	994.58	95.6	129.0	405.1	1450.8	0.57	−2.90	−5.02
22	1079.34	1076.11	1044.49	1021.45	105.8	941.1	331.2	965.0	−0.30	−3.34	−5.67
23	1086.42	1091.17	1047.23	1038.16	102.4	1000.8	518.1	1373.6	0.44	−3.74	−4.65
24	1150.13	1149.12	1122.20	1107.93	69.9	553.5	164.2	480.3	−0.09	−2.49	−3.81
25	1410.90	1418.87	1365.77	1345.08	93.6	635.9	864.2	2967.7	0.56	−3.30	−4.89
26	1392.35	1395.63	1346.22	1317.41	154.8	875.3	1033.6	2299.2	0.24	−3.43	−5.69
27	1400.70	1396.60	1342.28	1323.54	87.3	492.5	713.1	2716.6	−0.29	−4.35	−5.83
28	2727.64	2737.01	2630.12	2560.06	169.1	1079.1	3600.0	5065.5	0.34	−3.71	−6.55
29	2311.84	2315.20	2246.78	2191.46	180.4	1059.0	3600.0	4128.6	0.15	−2.90	−5.49
30	1891.90	1889.84	1802.04	1775.44	219.9	1711.2	3600.0	4753.7	−0.11	−4.99	−6.56
31	2418.54	2413.45	2317.10	2282.28	320.1	2500.7	3600.0	4988.2	−0.21	−4.38	−5.97
32	2365.28	2351.69	2261.98	2233.27	342.4	2240.1	3600.0	4900.6	−0.58	−4.57	−5.91
33	2436.08	2455.79	2337.89	2284.82	420.2	2074.1	3600.0	4988.9	0.80	−4.20	−6.62
34	1248.17	1233.46	1200.64	1191.13	990.3	2549.7	3600.0	5244.5	−1.19	−3.96	−4.79
35	1527.65	1498.78	1455.70	1435.22	1051.2	2964.5	3600.0	5015.5	−1.93	−4.94	−6.44
36	1788.36	1801.41	1755.26	1729.79	1192.3	2680.3	3600.0	4874.0	0.72	−1.89	−3.39
Avg.									0.48	−2.27	−3.51

instances. We can see that all instances have promising improvement. It is worth noting that large instances have resulted in greater improvement.

5.4. Applied to the 2L-CVRP

The proposed SA_HLS algorithm was also applied to 2L-CVRP, so as to compare the results with that obtained from the existing algorithms in the literature, namely GTS (Zachariadis et al., 2009), ACO (Fuellerer et al., 2009), GRASP × ELS (Duhamel et al., 2011). It is worth noting that the benchmark for 2L-CVRP has a fixed number of homogeneous vehicles and there are no fixed and variable costs for the vehicles. When applying the SA_HLS to the 2L-CVRP, only one type of vehicle is available, the fixed cost is equal to 0 and the variable cost is equal to 1. To guarantee to obtain a solution subject to the limit of the number of vehicles, we adjusted the objective function by giving a large penalty value to each extra vehicle. Here, we keep all the parameters unchanged for SA_HLS.

We have implemented the SA_HLS algorithm for 2L-CVRP in C++ and run it on a 2.2 GHz Core Duo notebook with 2G RAM under Windows 7 over five runs for each instance. The GTS algorithm was coded in C#, executed on a Pentium IV 2.4 GHz with 1G RAM under Windows XP. The ACO procedure was coded using C++ and compiled using g++ compiler, and ran on a Pentium 4 with 3.2 GHz under Linux operating system. The GRASP × ELS algorithm was also coded in C++ and compiled using the g++ compiler, but tested on Opteron 2.1 GHz under Linux operating system.

Tables 10–12 give the comparison results for the pure CVRP, *Unrestricted* and *Sequential* 2L-CVRP, respectively. Where, G × E indicates the GRASP × ELS. During the simulation, we found it easy to obtain the solution with a given number of vehicles for pure CVRP and *Unrestricted* 2L-CVRP, but for several large cases of *Sequential* version, it was hard to obtain the solution coping with the limit of vehicles for each run. In particular, for five instances (2302, 2604, 3003, 3304, 3504; here, 2302 means Class 2 of instance 23) no solution could be obtained with the given number of vehicles. These instances are marked with an asterisk '∗' in Table 12. The detailed results are also reported in Table A2 in the Appendix.

Fig. 7 depicts the gap between the proposed SA_HLS algorithm and the other three methods for Class 1 and the result shows that the performance of the proposed algorithm is promising. The proposed SA_HLS has the same performance as the others for small instances 1–17 and 19. For other instances, the gaps are less than 5%, except instances 28 and 31. According to Table 10, on average, the biggest percentage of gap is 1.87% over the GRASP × ELS method.

Fig. 8a illustrates the improvement percentage of our algorithm over other the three methods for *Unrestricted* 2L-CVRP. In comparison to GTS, our algorithm is able to produce better solutions for 26 instances out of 36, and the average performance is marginally better than GTS. The biggest gap over ACO for all instances is less than 5%. Among all these meta-heuristics tested, GRASP × ELS performs the best, especially for large cases. The gap achieves up to 3.51% on average according to Table 11.

Table 12
Average comparison results for the *Sequential* 2L-CVRP on Classes 2–5.

Inst	S			Sec_h			$\%gap$	
	SA_HLS	GTS	ACO	SA_HLS	GTS	ACO	GTS	ACO
1	298.34	304.22	294.48	4.2	2.9	6.9	1.93	−1.31
2	345.23	346.71	345.24	0.4	1.6	0.4	0.43	0.00
3	390.78	393.35	381.40	9.2	2.3	2.8	0.65	−2.46
4	444.21	444.62	441.11	4.5	2.7	2.8	0.09	−0.70
5	387.59	396.36	382.39	11.4	6.7	12.0	2.21	−1.36
6	503.66	505.04	499.49	4.9	3.1	4.1	0.27	−0.84
7	719.95	723.83	702.27	13.8	18.1	11.4	0.54	−2.52
8	716.62	715.72	711.65	18.3	21.7	11.9	−0.13	−0.70
9	621.23	622.2	614.54	5.2	10.4	4.9	0.16	−1.09
10	727.73	727.86	697.20	16.1	27.9	59.2	0.02	−4.38
11	761.25	768.15	728.61	14.1	54	68.7	0.90	−4.48
12	618.98	628.62	615.76	10.1	81.4	7.7	1.53	−0.52
13	2641.36	2679.34	2591.77	9.7	49.2	73.3	1.42	−1.91
14	1079.47	1092.78	1042.33	46.8	104.1	192.6	1.22	−3.56
15	1230.28	1234.21	1212.93	50.8	41.3	166.7	0.32	−1.43
16	703.27	707.56	701.27	9.3	38.5	7.8	0.61	−0.29
17	864.47	865.2	864.05	16.5	54.2	4.3	0.08	−0.05
18	1094.01	1102.25	1068.14	28.4	255.7	354.1	0.75	−2.42
19	796.36	800.94	774.09	114.5	340.9	146.3	0.57	−2.88
20	558.63	576.58	541.66	255.8	501.4	1307.7	3.11	−3.13
21	1081.02	1106.33	1049.25	153.3	893.7	1277.5	2.29	−3.03
22	1117.98	1128.61	1076.59	165.4	1314.7	894.1	0.94	−3.85
23	1117.05*	1151.24	1088.04	113.0	766.9	1482.6	2.97	−2.67
24	1182.54	1206.62	1150.80	87.5	1376.5	378.3	2.00	−2.76
25	1488.56	1479.03	1425.30	160.4	2247.4	2988.8	−0.64	−4.44
26	1424.31*	1475.96	1407.01	229.8	1136.5	2919.1	3.50	−1.23
27	1455.29	1463.34	1389.46	130.5	805.5	1873.4	0.55	−4.74
28	2851.23	2835.3	2738.79	231.4	1731.4	10373.6	−0.56	−4.11
29	2435.22	2460.74	2367.28	295.8	1161.7	10033.2	1.04	−2.87
30	1999.90*	2031.94	1932.00	471.7	1318.5	10256.2	1.58	−3.51
31	2569.45	2554.37	2463.82	811.2	1944.2	10510.3	−0.59	−4.29
32	2527.01	2462.45	2429.10	779.6	2933.7	10730.5	−2.62	−4.03
33	2556.88*	2565.41	2502.02	749.5	2531.6	10670.7	0.33	−2.19
34	1356.29	1303.49	1296.95	1716.4	4384.1	10653.1	−4.05	−4.58
35	1629.66*	1675.35	1644.52	1792.2	3887.9	10664.4	2.73	0.90
36	1875.27	1864.73	1864.68	1843.9	3158.3	10826.2	−0.57	−0.57
Avg.							0.71	−2.33

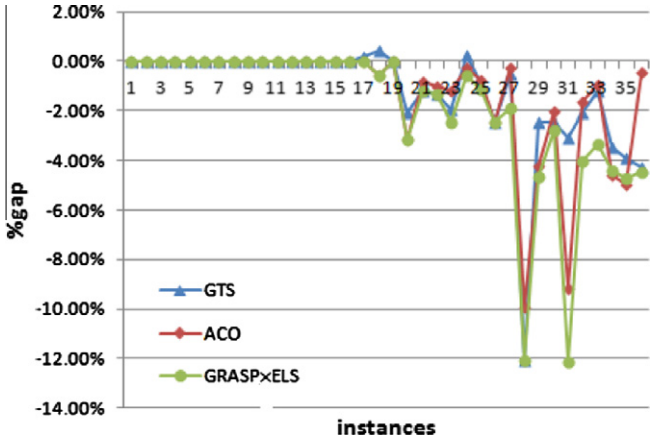

Fig. 7. The improvement percentage of SA_HLS over other methods for pure CVRP on Class 1.

Since GRASP × ELS did not solve the *Sequential* 2L-CVRP, it is not listed in Table 12. The improvement percentage is also expressed in Fig. 8b. We can see that, out of 36 instances, 29 instances obtain better results than GTS, and perform better on average. Although ACO can obtain better solutions than our method, especially for large cases, the gaps between them do not exceed 5% and the average gap is 2.33%.

Although SA_HLS is not designed for pure CVRP, as seen from Tables 10–12, SA_HLS performs similarly to GTS. In spite of the SA_HLS method producing slightly less favorable results than ACO and GRASP × ELS, the difference is minimal. More importantly, this study shows that SA_HLS requires much shorter computational time to achieve good solutions, especially for large instances.

5.5. Applied to the HFVRP

With a view to further exploring the performance of SA_HLS it was tested on the pure heterogeneous fleet VRP (HFVRP or HVRP) instances and the results were compared with the existing efficient algorithms in the literature, namely CG (Choi and Tcha, 2007), SMA_D1 (Prins, 2009), VNS1 (Imran et al., 2009) and ILS_RVND (Penna et al., 2011). The benchmark instances were widely used in the literature and have three different versions: with fixed cost only (HFVRP-F), with variable costs only (HFVRP-V) and with both costs (HFVRP-FV). And each version contains 17 instances with customer size varying from 20 to 100. Table 13 shows the average percentage gap (*%gap*) of SA_HLS over other algorithms.

According to Table 13, SA_HLS performs relatively well for HFVRP-V, but poorly for HFVRP-F, especially for HFVRP-FV. In our tests, we found that the algorithm performed poorly on the instances with more than 50 customers. The reason for the big gap might be that SA_HLS does not employ complex operations as other methods do, as those methods are tailored for pure HFVRP but not suitable for the problems with two-dimensional loading constraints. These operations can easily lead to loading-infeasible routes, such as the relocation of two customers.

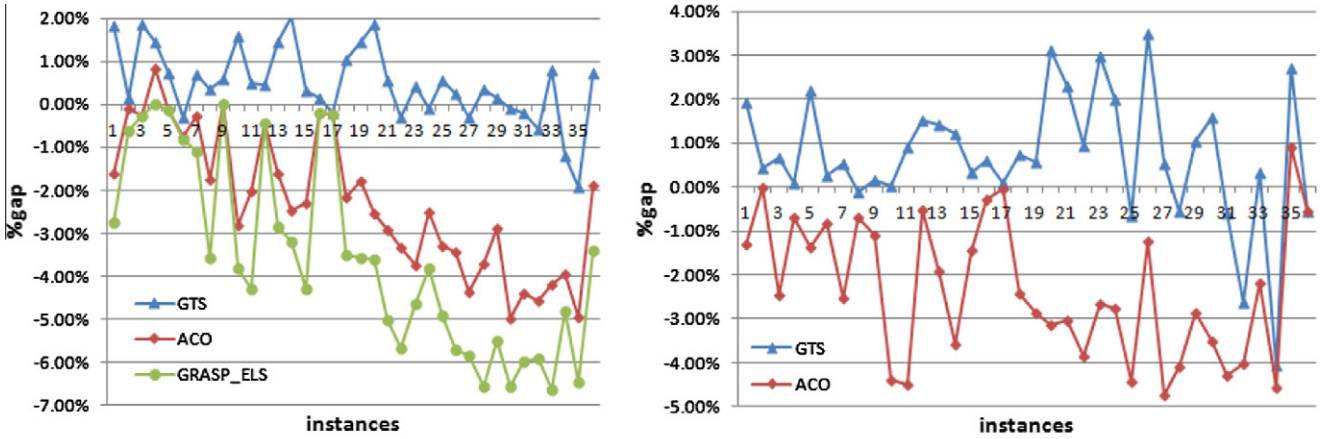

Fig. 8. The improvement percentage of SA_HLS over other methods.

Table 13
Average percentage gap of SA_HLS over other algorithms for the HFVRP instances.

	CG (%)	SMA_D1 (%)	VNS1 (%)	ILS_RVND (%)
HFVRP-V	1.89	2.80	2.81	1.89
HFVRP-F	4.18	4.14	4.19	4.23
HFVRP-FV	7.25	7.32	7.29	7.34

6. Conclusions

In this paper, the 2L-HFVRP, a new variant of vehicle routing problem was presented. In this problem, the fleet consists of vehicles with different attributes and the number of available vehicles is unlimited. The problem is interesting because it is NP-hard and has many real world applications. To the best of our knowledge, the 2L-HFVRP has not been previously addressed.

To solve the problem, we proposed a SA with heuristic local search. A series of experiments were carried out to evaluate the performance of the proposed algorithm. By testing on 360 benchmark instances, we have verified the effectiveness of our proposed SA_HLS algorithm. Besides, we also applied the SA_HLS algorithm to solve 2L-CVRP, and compared the results with the algorithms proposed in Zachariadis et al. (2009), Fuellerer et al. (2009), and Duhamel et al. (2011). Although the SA_HLS algorithm was designed for solving 2L-HFVRP, we can also see that it is capable of solving 2L-CVRP with a good set of results. The three heuristic local search mechanisms were also proved to be effective in producing better solutions. The future work could further assess the SA_HLS algorithms to solve different variants of 2L-CVRP with time window constraints (2L-CVRPTW).

Acknowledgments

The authors thank the anonymous referees for their valuable comments. This work has been supported by the National Nature Science Foundation of China (Grant No. 61272003) and the Open Fund of Chongqing Key Laboratory of Logistics (Project No. CQKLL12002).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejor.2012.09.023.

References

Brandao, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Computers & Operations Research 38 (1), 140–151.
Cerny, V., 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 45 (1), 41–51.
Chen, S., Golden, B., Wasil, E., 2007. The split delivery vehicle routing problem: applications, algorithms, test problems, and computational results. Networks 49 (4), 318–329.
Choi, E., Tcha, D.W., 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Computers & Operation Research 34 (7), 2080–2095.
Crainic, T.G., Laporte, G., 1998. Fleet Management and Logistics. Kluwer Academic, London, Dordrecht, Boston.
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.
Duhamel, C., Lacomme, P., Quilliot, A., Toussaint, H., 2011. A multi-starts evolutionary local search for the two dimensional loading capacitated vehicle routing problem. Computers & Operations Research 38 (3), 617–640.
Fuellerer, G., Doerner, K.F., Hartl, R.F., Iori, M., 2009. Ant colony optimization of the two-dimensional loading vehicle routing problem. Computers & Operations Research 36 (3), 655–673.
Fuellerer, G., Doerner, K.F., Hartl, R.F., Iori, M., 2010. Metaheuristics for vehicle routing problems with three-dimensional loading constraints. European Journal of Operational Research 201 (3), 751–759.
Gendreau, M., Iori, M., Laporte, G., Martello, S., 2006. A tabu search algorithm for a routing and container loading problem. Transportation Science 40 (3), 342–350.
Gendreau, M., Iori, M., Laporte, G., Martello, S., 2008. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. Networks 51 (1), 4–18.
Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, E.D., 1999. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers & Operation Research 26, 1153–1173.
Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing. Computers & Operations Research 11 (1), 49–66.
Imran, A., Salhi, S., Wassan, N.A., 2009. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 197 (2), 509–518.
Iori, M., 2005. Metaheuristic algorithms for combinatorial optimization problems. 4OR: A Quarterly Journal of Operations Research 3 (2), 163–166.
Iori, M., Martello, S., 2010. Routing problems with loading constraints. TOP 18 (1), 4–27.
Iori, M., Salazar Gonzalez, J.J., Vigo, D., 2007. An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transportation Science 41 (2), 253–264.
Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.
Kolen, A.W.J., Rinnooy Kan, A.H.G., Trienkens, H.W.J.M., 1987. Vehicle routing with time windows. Operations Research 35 (2), 266–273.
Leung, S.C.H., Zheng, J.M., Zhang, D.F., Zhou, X.Y., 2010. Metaheuristics for the vehicle routing problem with two-dimensional loading constraints. Flexible Services and Manufacturing Journal 22 (1–2), 61–82.
Leung, S.C.H., Zhou, X.Y., Zhang, D.F., Zheng, J.M., 2011. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. Computers & Operations Research 38 (1), 617–640.

Li, X.Y., Tian, P., Leung, S.C.H., 2009. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. Journal of the Operational Research Society 60 (7), 1012–1025.
Lin, S., 1965. Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269.
Lima, C.M.R.R., Goldbarg, M.C., Goldbarg, E.F.G., 2004. A memetic algorithm for the Heterogeneous fleet vehicle routing problem. Electronic Notes in Discrete Mathematic 18, 171–176.
Osman, I.H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 41 (4), 421–451.
Penna, P., Subramanian, A., Ochi, L., 2011. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. Journal of Heuristics, 1–32.
Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. Engineering Applications of Artificial Intelligence 22, 916–928.
Rochat, Y., Taillard, E.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1 (1), 147–167.
Taillard, E.D., 1999. A heuristic column generation method for heterogeneous fleet VRP. RAIRO 33 (1), 1–14.
Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 152 (1), 148–158.
Tarantilis, C.D., Zachariadis, E.E., Kiranoudis, C.T., 2009. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. IEEE Transactions on Intelligent Transportation Systems 10 (2), 255–271.
Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM, Philadelphia.
Waters, C.D.J., 1987. A solution procedure for the vehicle scheduling problem based on iterative route improvement. Journal of the Operational Research Society 38 (9), 833–839.
Zachariadis, E.E., Tarantilis, C.D., Kiranousdis, C.T., 2009. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. European Journal of Operational Research 195 (3), 729–743.