An improved particle swarm algorithm for heterogeneous fleet vehicle routing problem with two-dimensional loading constraints

HaiFeng Sun[1, a,*,#], KaiTai Dong[2,b,#], Qun Zhang[1,c], Rui Yan[3.d]

[1] School of Economics and Management, University of Science and Technology Beijing, Beijing, China

[2]School of Mathematics, University of Manchester, Manchester, UK.

[3]School of Economics and Management, Beijing Information Science and Technology University, Beijing, China

[a]hafeson1991@163.com, [b]kaitaidong@outlook.com, [c] zq@ustb.edu.cn, [d] yr1900@163.com

* Corresponding author. # These authors contributed to this work equally

Keywords: Particle swarm optimization, Two-dimensional Packing, VRP, 2L-HFVRP, Local Search

Abstract. This paper concerns the heterogeneous fleet vehicle routing problem with two-dimensional loading constraints (2L-HFVRP), which is a more complicated version of the vehicle routing problem as it combines heterogeneous vehicles and two-dimensional loading constraints. We give a detailed description of the problem and propose an improved Particle Swarm Optimization with Heuristic Local Search (PSO_HLS) to solve the problem. The PSO_HLS, based on the classical PSO algorithm, introduces the idea of heuristic local search to increase the diversity of the solutions. To effectively improve the loading efficiency, a new packing heuristic algorithm, called the fitness sorting depth-first heuristic (FSDFH), is designed. Additionally, the area contraction coefficient is introduced to optimize the routing and loading algorithm. The robustness and effectiveness of the proposed approach is verified by computational tests performed on widely used benchmark instances. The numerical results obtained show that the proposed approach outperforms the existing method and improves the best known solutions in some testing instances.

1. Introduction

The vehicle routing problem (VRP) was initially brought up by Dantzig and Ramser [1] in 1959, and it soon attracted considerable attention from the industry and academia. Since then, VRP has been well discussed and gradually derived into various forms of vehicle routing problems given by the practical distribution situations. The capacitated vehicle routing problem (CVRP), normally considering the properties of vehicles, has then been raised. It allows us to take practical variants into account and might lead to a more effective solution. CVRP is, however, only capable of solving one-type vehicle problems and in real distribution tasks, mixed types of vehicles are often required for different conditions. At the same time, various types of vehicles have various capacities and take diverse running costs, thus it increases the complexity of the routing optimization. The heterogeneous fleet of vehicle routing problem (HFVRP) has been discussed and fast developed from there. Gorden et al.[2] first studied the HFVRP and built a general model for an extension of the HFVRP, fleet size and mix vehicle routing problem (FSMVRP). Based on its model, a few algorithms, including the improved Clark-Wright (C-W) saving algorithm[3], have been attempted to solve this type of problem. Subsequently, Gendreau et al. [4] utilized tabu search to solve the HFVRP and achieved ideal results.

The CVRP only takes into account the capacity restriction of the vehicles, however, in the real-life loading process it also should be required that all the freights are loaded on the base surface of the vehicles and none of the freights can be stacked on other items due to their fragility or other special properties. It is thereby categorized as the two-dimensional bin packing problem (2BPP). In 2006, Iori et al. [5] first brought up the idea of the 2L-CVRP and proposed an effective approach for solving the routing issues by applying the branch-and-cut algorithm and loading issues by employing the branch-and-bound algorithm. Later on, a tabu search heuristic was raised by Gendreau et al. [6] to

solve the 2L-CVRP, in which the loading part was resolved by lower bounds and a truncated branch-and-bound algorithm. Furthermore, a guided tabu search algorithm was proposed by Zachariadis et al. [7] to solve the 2L-CVRP. Alongside the guided tabu search algorithm, they also designed five heuristic algorithms for loading plans and implemented them successively while loading the freights. Fuellerer et al. [8] then successfully solved the 2L-CVRP by employing various heuristics for the loading part and by introducing an ant colony optimization (ACO) algorithm for overall optimization. In addition, Duhamel et al. [9] proposed a newly combined approach, namely greedy randomized adaptive search procedure with evolutionary local search (GRASP-ELS), for the 2L-CVRP, which achieved an excellent computational result.

To summarize, remarkable progresses have been made on both the HFVRP and 2L-CVRP studies. Nonetheless, there is a lack of scientific achievements for the combined problem: two-dimensional loading heterogeneous fleet vehicle routing problems (2L-HFVRP) due to their high degrees of complexity. Leung et al. [10] first attempted to solve the 2L-HFVRP by introducing a simulated annealing with heuristic local search (SA_HLS).

The objective of this paper is to provide an effective and efficient tool for the 2L-HFVRP called Improved Particle Swarm Optimization with Heuristic Local Search method (PSO-HLS). This paper also proposes a new loading heuristic algorithm which is referred to as fitness sorting depth-first heuristic (FSDFH). Finally we takes into consideration both routing and loading optimization and adjust it to attain the best solution by using the area contraction coefficient.

2. Problem Description

The 2L-HFVRP, short for Heterogeneous Fleet Vehicle Routing Problem with Two-dimensional Loading constraints, is defined on a complete undirected graph $G = \{V, E\}$, where $V = \{0,1, \cdots, n\}$ is the vertex set that contains a central depot (node 0) and n customers (node 1,2,…,n); $E = \{e_{ij}: i, j \in V\}$ is the undirected edge set. Each edge $e_{ij} \in E$ is associated with a travel distance d_{ij} from vertex i to j. There are T different types of vehicles locating at the depot and the quantity of each type has no upper limit. Each type of vehicle $t(t = 1,2, \ldots, T)$ has a loading capacity Q_t, fixed cost F_t, variable cost V_t, a two-dimensional rectangular loading surface $A_t = L_t \times W_t$, where L_t and W_t are the length and width respectively. Generally speaking, A vehicle with larger capacity has higher fixed costs and variable costs, therefore we assume that $Q_1 \leq Q_2 \leq \cdots \leq Q_t$, $F_1 \leq F_2 \leq \cdots \leq F_t$ and $V_1 \leq V_2 \leq \cdots \leq V_t$. The travel cost of each edge $e_{ij} \in E$ by a vehicle type t is $C_{ij}^t = V_t \times d_{ij}$. As a consequence, the total cost of a route R for a vehicle type t then can be formulated as $C_t^R = F_t + \sum_{i=1}^{i<|R|} V_t \times d_{R(i),R(i+1)}$. Each customer $i(i = 1,2, \ldots, n)$ requires a set of m_i rectangular items, denoted as IT_i, in which the total weight of all items is D_i. In addition, each item $I_{ir} \in IT_i (r = 1,2,.., m_i)$ has specific length l_{ir} and width w_{ir}. Hence the total area of the items required by customer i is denoted as $a_i = \sum_{r=1}^{m_i} w_{ir} * l_{ir}$. Besides, all the items are required to load on the surface of the vehicles, no piling-up allowed.

In the 2L-HFVRP, the objective is to optimize the routes and loading plans for vehicles that are served to various customers and thereby minimize the total cost. The total cost consists of the fixed cost of the selected vehicles and also the travel cost on each route. A feasible loading plan in the 2L-HFVRP must satisfy the following constraints:

1) Each vehicle must start and end at the central depot;
2) Each customer can only be serviced once by one vehicle;
3) All the items of each customer must be loaded on the same vehicle.
4) The total capacities of items on one vehicle cannot exceed the maximum capacity of the vehicle;
5) Items are loaded with their edges parallel to the sides of the vehicle;
6) There are no overlapping areas between any items.

3. The Particle swarm optimization with heuristics local search algorithm

Particle swarm optimization (PSO) was initially developed by Dr. Kennedy and Dr. Eberhart [11] in 1995. It was designed to simulate social behaviors, such as bird flocking and fish schooling, to discover an optimal solution by moving the swarm towards the best position. In PSO, every candidate solution is referred to as 'a particle' and each particle moves around in the search-space based on given mathematical formulae over the particle's position and velocity. Furthermore, the movements of the particles are led by their previous local best positions and also the best position of the entirety. When a better position is found, it will then lead the swarm move toward it, and repeatedly, an optimal position will be ultimately discovered. The algorithm can be mathematically interpreted as follows:

Assume the search-space is D-dimensional and the total number of the particles is n. Therefore, the position of the ith particle can be expressed by the vector $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ and the best previous position of ith particle in its moving history is $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ and then let the position P_g of particle g be the best previous position among all the particles in the whole swarm. Moreover, the speed of the ith particle is $V_i = (v_{i1}, v, \cdots, v_{iD})$ and each particle moves along the path defined by the following formulae:

$$x_{id}^{(t+1)} = x_{id}^{t} + v_{id,}^{t+1}, \qquad 1 \le i \le n, 1 \le d \le D \tag{1}$$

$$v_{id}^{(t+1)} = w \cdot v_{id}^{t} + C_1 \cdot rand() \cdot (p_{id}^{t} - x_{id}^{t}) + C_2 \cdot rand() \cdot (p_{gd}^{t} - x_{gd}^{t}) v_{id}^{t+1}. \tag{2}$$

Where C_1 and C_2, referred to as acceleration coefficients, are positive constants; $rand()$ denotes randomly generated numbers within [0,1]; w is termed the 'inertia weight', and normally larger w is more suitable for extensive exploration and smaller w performs better in exploitation of local optima. Additionally, the ranges of the positions and speed of the particles are $[-X_{max}, X_{max}]$ and $[-V_{max}, V_{max}]$ respectively. If any of these exceeds the given range, then it will be set to the boundary value. To solve the large scale problems, our algorithm explores the solution space by employing particle swarm optimization with heuristic local search mechanism (PSO_HLS). Table1 provides a framework for the proposed PSO_HLS algorithm.

Table1. The pseudo-code for the PSO-HLS

```
PSO_HLS(customer demand, vehicle information)
1: Generate initial solution randomly for all particles
2: Adjust _Vehicle(new order)
3: while Number of iterations is not finished do
4:    Calculate fitness values for all particles
5:    Calculate P_i and P_g.
6:    for i=1:n_particle do // n_particle is the total number of particles
7:       Update speed of particle i using formula(2)
8:       Update position of particle i using formula(1)
9:       Adjust _Vehicle(particle i);
10:      Calculate fitness for particle i
11:      Local_Search(particle i)
12:      If new solution is better than the best existed one then
13:         Replace the current solution with this new one
14:      end if
15:   end for
16: end while
```

3.1 The encoding and decoding methods of the particles. The particle swarms in this paper are encoded in 2 × L dimensional vectors and the corresponding 2 × L dimensional vector X of each

particle can be decomposed into two L-dimensional vectors. The first vector X_v indicates the delivery vehicles for customers and the second vector X_r represents the delivery orders of the corresponding vehicles. For instance, assuming there are 8 customers and 3 vehicles with two types in total, then mark the two vehicles with type A as 1 and 2 and similarly the one vehicle with type B as 3. Then the position vector of a particle can be described and encoded as Table 2:

Table 2. Position vector of a particle

Customer	1	2	3	4	5	6	7	8
X_v	1	2	1	2	3	3	1	2
X_r	2	2	3	1	2	1	1	3

The corresponding route of this particle can be encoded as follows:

Vehicle 1 with type A： $0 \rightarrow 7 \rightarrow 1 \rightarrow 3 \rightarrow 0$

Vehicle 2 with type A： $0 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0$

Vehicle 3 with type B： $0 \rightarrow 6 \rightarrow 5 \rightarrow 0$

3.2 Initial solution. The initial solutions are generated at random in our algorithm. First of all, regardless of the loading and capacity constraints, a total of k routes are randomly generated and a total of n customers are randomly selected to be put into each route. As a result, the initial position of each particle is also randomly generated. For position vector X_v, an integer between 1 and k will be selected at random in each dimension. Similarly, for position vector X_r, an integer between 1 and n will be obtained randomly.

Not all the randomly generated initial solutions are always feasible; therefore, it is necessary to examine all the initial solutions and make adjustments to the non-feasible ones. First off, we need to check whether there are any vehicles overloaded or oversized. If the total weight of the items exceeds the capacity of the vehicle, then we reallocate customer i's items to the vehicle with the smallest free$Q_t - D_i$ value, where freeQ_t is the remaining capacity of vehicle t and D_i is the total weight of customer i's items. Similarly, if the total occupation area of the items exceeds the base area of the vehicle, then we reallocate customer i's items to the vehicle with the smallest free$A_t - a_i$ value, where freeA_t is the remaining base area of vehicle t and a_i is the total base area of customer i's items. However, if the allocated vehicle does not have enough remaining capacity or space for the items or all vehicles are overloaded or oversized, then add a new vehicle with biggest capacity. The pseudo-code of the process of vehicle selection and adjustment is given in Table 3.

Table 3. The pseudo-code for adjust and select vehicle

Adjust_Vehicle(Order)
1: **For** each vehicle k in order **do**
2: calculate the total weight W_k and total surface area S_k of customers' items served by vehicle k
3: **while** Q_k<W_k **do // vehicle k is overweight**
4: randomly select customer i from vehicle k
5: **If** max freeQ_t< D_i **then //** customer i cannot be served by any used vehicle
6: V[t]=V[t]+1//add a new vehicle t with largest capacity
7: Insert customer i into the new vehicle
8: **Else**
9: insert customer into vehicle t which has minimum free$Q_t - D_i$
10: **end if**
11 **end while**
12: **while**$A_k < S_k$ **do // vehicle k is oversized**
13: randomly select customer i from vehicle k
14: **If** max freeA_t< a_i **then**
15: V[t]=V[t]+1//add a new vehicle t with largest surface space
16: Insert customer i into the new vehicle
17: **Else**
18: insert customer into vehicle t which has minimum free$A_t - a_i$
19: **end if**
20: **end while**
21: **end for**

3.3 Heuristic local search. To increase the diversity and improve the quality of the solutions, two effective moves of heuristic local search are introduced in the following paragraphs.

a) *Route-exchange*: Randomly selecting a pair of customers from a single route or two various routes and swapping the positions of the selected pair of customers, we then compute the fitness value after the swapping process is performed. We will keep this move if we obtain a better fitness value, otherwise, cancel the exchange operation, as shown in Fig. 1.

b) *Route-interchange*: The route-interchange can be implemented within any single route and between any randomly selected route pair. When carrying out the route-interchange with in a single route, two original arcs are deleted and the middle section will be reversed. Then two new arcs are created to connect the route as a whole, as illustrated in Fig. 2.

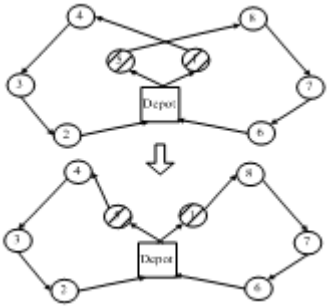

Fig.1. Route-exchange

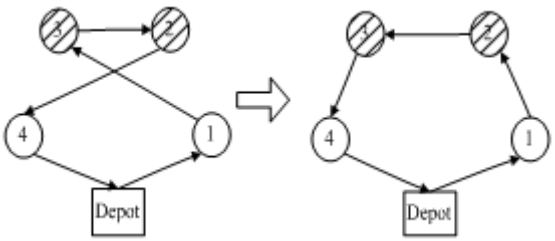

Fig. 2. Route-interchange

4. The heuristic algorithm for two-dimensional loading problems

To determine whether all the items demanded by the customers in a given route can be feasibly loaded onto the vehicle, a new packing heuristic algorithm, which is called Fitness Sorting Depth-First Heuristic (FSDFH), is developed.

At the beginning of loading, the left bottom corner is hypothetically set as the optimal loading point and other feasible loading points will keep changing as the items are added in. Assuming the left bottom corner is the origin of the coordinate plane,

In FSDFH, according the minimal size granularity of items, divided the rectangular loading space into a m × n grid. According to the FSDFH, the rectangular loading space can be divided into an m × n grid based on the minimal size of items. The loading point, which is denoted by (Abscissa, Surplus Width, Surplus Length), will keep changing as the items are added in. As shown in Fig. 3 , after loading item A in the left bottom corner, feasible loading points contain $\mathrm{a}(0, \mathrm{W}, \mathrm{L}-l_A)$, $\mathrm{b}(1, \mathrm{W}-1, \mathrm{L}-l_A)$, $\mathrm{c}(2, \mathrm{W}-w_A, \mathrm{L})$, …, $(\mathrm{m}-2, 1, \mathrm{L})$, $(\mathrm{m}-1, 0, \mathrm{L})$ and thereby form a $\mathrm{m} \times 3$ matrix.

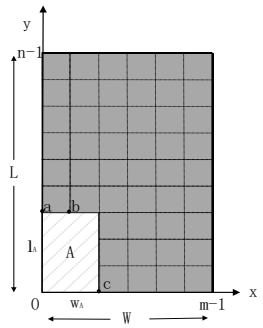

Fig. 3. The illustration of the way to insert an item

In addition, it is necessary to calculate the fitness values of all possible loading points using the following rules every time we load an item.

(1) If the length and width of item A are both equal to the available loading space left, then the fitness value is set to 1.

(2) If the width of item A is equal to the remaining loading space, but the length is smaller than the remaining loading space so then the fitness value is set to 2.

(3) If the length of item A is equal to the remaining loading space, but the width is smaller than the remaining loading space so then the fitness value is set to 3.

(4) If the length and width of item A are both smaller than the available loading space left, then the fitness value is set to 4.

(5) If the length and width of item A are both equal to the available loading space left, then the fitness value is set to infinity

More intuitively, the rules can be illustrated in Fig. 4.

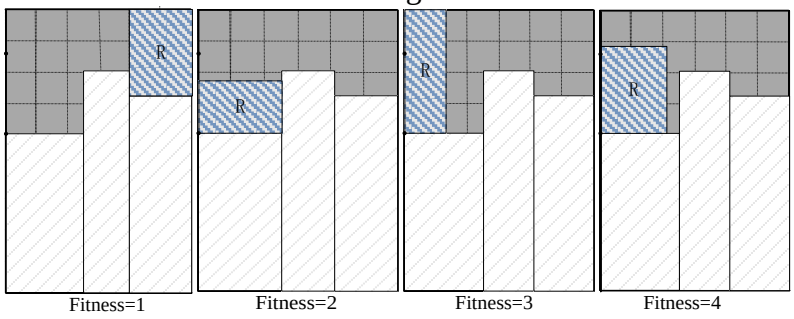

Fig. 4 The rules with various fitness values

In order to improve the efficiency of loading, a few adjustments of the rules need to be made as follows:

(a) The selected position with the minimum W-axis coordinate goes first rule: the fitness value of the position with bigger W-axis coordinate will be added A, where A is a constant.

(b) The selected position with the minimum L-axis coordinate goes first rule: the fitness value of the position with bigger L-axis coordinate will be added B, where B is a constant.

(c) Remaining area rule: in order to keep the remaining gap area as small as possible, for each loaded item, the fitness value of the loading point will be added a constant C if the remaining gap area is large.

In the process of loading, the fitness value of every possible loading point for each item will be calculated and sorted to obtain the feasible loading solutions. By first sorting the loading point for each item according to the fitness value of loading point, it forms the feasible loading solution.

Subsequently, the loading process can be described as a tree structure and the loading solution to item i is represented as the $(\mathrm{i}+1)th$ layer of branches in the tree. The initial state of nodes indicates the empty container and there are m loading points to choose. Load the first item at the loading point with the smallest fitness value and then update information of the remaining loading points, continuing to load items by this rule. This loading procedure forms a tree structure, which can be referred to as the Packing Tree. Aside from the root node, each layer of the Packing Tree has m child nodes and each of them represents a loading solution to the loading item. Given that the root node of the Packing Tree has a depth of 0, then the depth will increase by 1 every time an item is loaded, therefore the depth of the Packing Tree stands for the total number of items needed for shipment on each route. Sometimes the loading point with the smallest fitness value, however, is not a feasible solution in the real process. In order to solve this problem, we adopted a new searching method that attempts all the other possible solutions when the current loading point fails. Making an assumption that the loading point with the smallest fitness value for the $\mathrm{i}th$ item fails to work, we then attempt the loading point with the second smallest fitness value, and so on, until a feasible loading point is found. If it fails to find such a feasible loading point, and then return to the $(\mathrm{i}-1)th$ item, select the point with the second smallest fitness value and repeat the previous procedure. Analogously, we keep tracking back to the previous item and making adjustments if it fails until we come up with the most efficient loading plan for all the items. Table 4 shows the pseudo-code for the packing heuristics.

Table 4. Pseudocode for the packing heuristics(FSDFH)

```
Is_Feasible( Route r)
1: Initial loading points' information
2: for each sorting rule do
3:    obtain the loading sequence of all items
4:    i=1;P[1,…,i]=1;// P[1,…,i] is the loading points array of all items in route r
5:    while i<= n_item do// n_item is total number of items
6:        while P[i]<= m do// m is number of loading points;
7:            if loading point P[i] is feasible for item i then
8:                i=i+1 //begin to load next item
9:                update loading points' information
10:               if  i> n_item then //all items are packing-feasible
11:                   return true
12:               end if
13:           else
14:               if P[i]=m then // all the loading points are tried out
15:                    if i >1 then //item i isn't the first item
16:                        i=i-1// return to the loaded item
17:                       P[i]=P[i]+1 //loading item (i-1) on the next loading point
18:                       Break
19:                   else
20:                       return false
21:                   end if
22:           end if
23:           P[i]=P[i]+1// loading item i on the next loading point
24:           end if
25:       end while
26:   end while
27: end for
```

5. Collaborative optimization for loading and route problems

In order to conduct the collaborative optimization for effectively solving the loading and route problems, area contraction coefficient PS is introduced to help seek the optimal solution. The basic

idea is to load the items every time we discover an optimal route, if the loading is successful, that is to say that all the items can be loaded onto the vehicle without exceeding the maximum capacity and maximum area, then return the routing and loading information; otherwise, reduce the vehicle feasible loading area A_t by multiplying the area contraction coefficient PS. The pseudo-code of collaborative optimization between packing and routing problems is demonstrated in Table 5.

Table 5. The pseudo-code for the Collaborative optimization

PSO_HLS(customer demand, vehicle information)
1: Set a series of vectors
2: **while** Number of iterations T_{max} is not finished **do**
3: PSO_HLS(customer demand, vehicle information)
4: Obtain best path solution r
5: **if** Is_feasible(route r) is true **then**
6: Output the best solution
7: break of loop
8: **else**
9: Area contraction: $A_t = PS \cdot A_t$
10: **end if**
11: **end while**

6. Experimental results and analysis

The PSO-HLS algorithm used in this paper is coded in Matlab, and all of the experiments and computations are executed on a PC with Core 2 Duo 2.67GHz , 2GB RAM and Windows 7 operating system.

6.1 Parameter settings. After our preliminary tests on the data, we decide to use the following parameters for the PSO-HLS algorithm: the total number of particles n_particle=200, the maximum number of iterations IterMax =100, the maximum number of consecutive non-optimizing results K= 20, the total number of loops T_{max} = 10, the area contraction coefficient PS= 0.95 and the penalty coefficient $\emptyset$ = 20. In the results, the cost listed is the best cost achieved over 10 runs, BKS is the best known solution among previous approaches (ACO[8], SA[12], EGTS_LBFH[13], GRASP*ELS[9], and PRMP[14]) and Time(s) is the CPU time in seconds to run the algorithm. The improvement, as shown in the %Gap column, can be calculated in terms of cost and BKS（Gap% = 100*(BKS-Cost)/BKS）

6.2 2L-CVRP testing result. To validate the effectiveness of our algorithm, we test the algorithm using 2L-CVRP benchmark instances in the first place. A total number of 180 instances, based on the 36 classic CVRP instances introduced by Toth and Vigo [15], are developed and expanded through five various classes by Iori[16]. As stated above, these instances comprise of 5 classes in total. In the benchmark instances used in this test, the length and width of the vehicle are preset to 40 and 20 respectively. Additionally, the distance between each customer is Euclidean distance. In the end, we compare the obtained results with the known solutions of Duhamel's GRASP*ELS algorithm and Zachariadis's PRMP algorithm.

Table 6 shows the comparison of the computational results between the PSO_HLS algorithm and other known method on the pure CVRP instances of Class 1, where the operating time and cost are recorded.. From Table 6, it can be observed that the PSO_HLS algorithm enables to obtain the majority of the optimal solutions to these instances, in which the obtained results are even better than the best known solutions (BKS) in 8 of them. The biggest percentage gap between the average and best solution is up to 5.72%.

Table 6. Comparison results on the pure CVRP instances of Class1

Inst.	BKS	GRASP*ELS		PRMP		PSO_HLS		%Gap
		Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	
1	278.73	278.73	0.0	278.73	0.0	**273.20**	0.0	1.98
2	334.96	334.96	0.0	334.96	0.0	**325.60**	0.0	2.79
3	358.40	358.40	0.0	358.40	0.0	361.38	0.0	-0.83
4	430.88	430.88	0.0	430.88	0.0	430.88	0.0	0.00
5	375.28	375.28	0.0	375.28	0.0	376.20	0.0	-0.25
6	495.85	495.85	0.0	495.85	0.0	**467.47**	0.0	5.72
7	568.56	568.56	0.0	568.56	0.0	**563.41**	0.0	0.91
8	568.56	568.56	0.0	568.56	0.0	**561.11**	0.0	1.31
9	607.65	607.65	0.0	607.65	0.0	**582.92**	0.0	4.07
10	535.74	535.80	0.0	535.80	0.1	535.80	0.4	-0.01
11	505.01	505.01	0.0	505.01	0.0	505.01	0.6	0.00
12	610.00	610.00	0.2	610.00	0.2	**592.00**	0.5	2.95
13	2006.34	2006.34	0.0	2006.34	0.3	2006.34	0.3	0.00
14	837.67	837.67	0.2	837.67	0.1	837.67	0.8	0.00
15	837.67	837.67	0.0	837.67	0.4	837.67	0.3	0.00
16	698.61	698.61	0.0	698.61	0.3	**697.50**	1.1	0.16
17	861.79	861.79	0.0	861.79	1.6	861.79	1.5	0.00
18	723.54	723.54	8.3	723.54	3.6	723.54	10.1	0.00
19	524.61	524.61	0.3	524.61	2.1	524.61	0.9	0.00
20	241.97	241.97	4.5	241.97	7.2	251.32	11.7	-3.86
21	687.60	687.60	1.4	687.60	3.8	687.60	8.2	0.00
22	740.66	740.66	2.1	740.66	2.8	740.66	4.1	0.00
23	835.26	835.26	3391.3	835.26	48.7	835.26	61.2	0.00
24	1024.69	1026.60	53.3	1024.69	38.1	1024.69	51.9	0.00
25	826.14	827.39	2.4	826.14	8.6	826.14	10.4	0.00
26	819.56	819.56	0.4	819.56	11.2	821.76	20.1	-0.27
27	1082.65	1082.65	486.5	1082.65	172.3	1082.65	183.6	0.00
28	1040.70	1042.12	129.8	1042.12	71.2	1043.70	201.4	-0.29
29	1162.96	1162.96	549.6	1162.96	121.9	1162.96	194.5	0.00
30	1028.42	1033.42	2165.9	1028.42	267.5	1028.42	1903.2	0.00
31	1299.56	1306.07	5096.1	1299.56	353.8	1302.63	3014.5	-0.24
32	1296.91	1303.52	4492.4	1296.91	312.0	1296.91	3822.6	0.00
33	1299.55	1301.06	4842.1	1299.55	434.1	1299.55	4018.1	0.00
34	709.82	713.51	3007.4	709.82	328.2	709.82	2176.3	0.00
35	866.06	870.63	2616.5	866.06	396.3	866.06	3011.2	0.00
36	585.46	592.87	5264.7	585.46	228.9	590.65	4019.4	-0.89
Avg	769.66	770.77	892.1	769.70	78.2	767.63	631.4	0.37

Table 7 represents the average comparison results for the unrestricted and non-oriented 2L-CVRP of Classes 2-5. According to the computational results on Table 7, the proposed PSO_HLS algorithm is able to acquire the BKS in most testing instances and some results are better than the BKS, and the rest of the results are very close to the BKS. Overall the outcome gained from the PSO_HLS algorithm reaches to an average 0.18 percent improvement than the BKS. Despite the fact that the average operating time of the PSO_HLS is relatively longer than other existing algorithms as it belongs to swarm intelligence algorithm, it still has high comparability.

Table 7. Average comparison results for the Unrestricted and Non-Oriented 2L-CVRP(Classes2-5)

Inst.	BKS	GRASP*ELS		PRMP		PSO_HLS		%Gap
		Cost	Time(s)	Cost	Time(s)	Cost	Time(s)	
1	281.23	282.65	0.9	281.23	0.4	281.23	0.6	0.00
2	339.26	339.26	0.1	339.26	0.3	**327.75**	0.5	3.39
3	376.32	376.32	0.5	376.32	0.4	376.32	0.7	0.00
4	435.00	435.01	0.2	435.01	0.3	435.01	0.9	0.00
5	379.03	379.03	0.1	379.03	1.1	379.03	2.4	0.00
6	496.77	497.04	0.4	497.04	0.3	**494.21**	0.3	0.52
7	690.67	691.11	1.4	690.67	1.6	690.67	2.1	0.00
8	678.84	678.84	0.8	678.84	2.6	678.84	3.6	0.00
9	611.05	612.01	0.6	612.01	1.6	**597.57**	4.1	2.21
10	675.59	675.79	15.1	676.75	26.9	676.75	41.1	-0.17
11	702.96	705.95	11.3	703.22	27.2	703.22	32.5	-0.04
12	611.26	611.26	16.9	611.26	1.4	**607.5**	3.6	0.62
13	2486.17	2490.62	78.0	2491.18	52.7	2490.62	45.2	-0.18
14	974.04	984.42	79.9	975.88	164.3	975.88	154.1	-0.19
15	1128.86	1144.69	257.7	1132.91	20.1	1132.91	29.2	-0.36
16	699.79	699.79	6.0	699.79	4.1	**692.56**	6.3	1.03
17	862.26	864.05	21.6	864.05	2.4	**848.75**	4.5	1.57
18	1028.61	1029.71	413.5	1031.95	33.0	1031.95	46.2	-0.32
19	739.15	739.19	268.5	741.78	24.3	741.78	37.3	-0.36
20	515.44	522.68	1658.3	515.44	552.2	515.44	603.8	0.00
21	991.50	994.58	1450.8	992.78	241.5	992.78	309.1	-0.13
22	1017.33	1021.45	965.0	1023.01	166.6	1023.01	201.2	-0.56
23	1032.36	1038.16	1373.6	1032.36	336.8	1032.36	451.1	0.00
24	1099.57	1107.93	480.3	1104.64	319.6	1104.64	343.6	-0.46
25	1340.18	1345.08	2967.7	1341.26	921.7	1341.26	1021.5	-0.08
26	1311.79	1317.41	2299.2	1311.79	403.5	1311.79	511.2	0.00
27	1318.04	1323.54	2716.6	1318.04	438.2	1318.04	457.3	0.00
28	2530.46	2560.06	5065.5	2530.46	3701.9	2530.46	3952.4	0.00
29	2173.02	2191.46	4128.6	2173.02	1835.7	2173.02	2014.2	0.00
30	1760.16	1775.44	4753.7	1760.59	2151.8	1760.59	2559.3	-0.02
31	2244.13	2282.28	4988.2	2244.13	2927.4	2244.13	3089.7	0.00
32	2196.85	2233.27	4900.6	2196.85	3713.8	2196.85	4538.2	0.00
33	2261.68	2284.82	4988.9	2261.68	1964.8	2261.68	2597.1	0.00
34	1157.22	1191.13	5244.5	1157.22	3551.7	1157.22	4103.1	0.00
35	1401.17	1435.22	5015.5	1401.17	2756.5	1401.17	3011.4	0.00
36	1669.44	1729.79	4874.0	1669.44	4245.6	1669.44	4322.6	0.00
Avg	1117.14	1127.53	1640.1	1118.11	849.8	1116.57	958.4	0.18

6.3 Instance tests on 2L-HFVRP. Leung et al. expanded 180 instances of the 2L-CVRP and created more general instances for the 2L-HFVRP. Similarly, there are also a total number of 180 instances, however, unlike the 2L-CVRP, it offers multiple choices of vehicle information to select in each instance. For the instances of the 2L-HFVRP, the testing data gets rid of the limitation on the number of vehicles and is replaced by the vehicle information instead. That is to say, for a given type of fleet, we can employ as many vehicles for delivery as we require. Nonetheless, in order to reduce the total cost, obviously the number of used vehicles should be controlled in a relatively small range.

Table 8 illustrates the comparison of the computational results between the PSO_HLS and SA_HLS algorithm on the 2L-HFVRP instances of Class 1, where the running time and average cost are recorded as before. Based on the obtained result, it can be concluded that the PSO_HLS outperforms the SA_HLS in many testing instances. Moreover, the average improvement percentage is 2.06%, which indicates that the average performance of the PSO_HLS is better than the best known algorithm. Again, the running time of the PSO_HLS, however, is longer than other algorithms as it is a swarm intelligence algorithm.

Table 8. Comparison results for the 2L-HFVRP of Class1

No.	SA_HLS		PSO_HLS		%Gap
	Cost	Time(s)	Cost	Time(s)	
1	596.07	33.59	**589.21**	50.24	1.15
2	679.18	32.01	**677.46**	61.52	0.25
3	745.51	54.24	749.41	73.54	-0.52
4	694.33	18.36	696.36	45.2	-0.29
5	761.19	26.99	761.19	70.44	0.00
6	809.56	25.65	**806.54**	54.81	0.37
7	3211.53	29.76	**3201.76**	61.21	0.30
8	3184.45	24.3	**3182.03**	72.42	0.08
9	1029.95	40.28	1030.09	54.61	-0.01
10	5149.51	25.88	**5012.65**	52.59	2.66
11	5119.4	26.47	**4932.54**	39.71	3.65
12	1658.56	92.8	**1655.27**	124.21	0.20
13	14655.4	32.18	**14596.16**	55.42	0.40
14	10019	73.21	**9929.08**	97.74	0.90
15	10151.7	55.29	**10048.57**	77.6	1.02
16	1292.58	76.92	**1283.79**	96.21	0.68
17	1770.83	225.59	**1766.48**	256.33	0.25
18	3140.55	35.35	**3131.38**	65.43	0.29
19	1553.11	107.81	1555.21	142.15	-0.14
20	1956.97	71.57	**1948.52**	92.67	0.43
21	2567.18	195.66	2568.22	226.38	-0.04
22	2605.9	174.72	**2507.45**	203.42	3.78
23	2643.84	239.29	**2641.75**	273.41	0.08
24	2555.41	156.41	**2551.13**	174.53	0.17
25	2972.59	253.09	**2874.27**	288.49	3.31
26	4049.64	180.23	4049.64	208.21	0.00
27	3561.58	230.49	**3556.79**	246.69	0.13
28	6858.35	161.05	6858.35	177.36	0.00
29	9695	142.63	**9689.06**	165.71	0.06
30	5663.33	259.83	5663.56	271.01	0.00
31	8054.9	483.44	8054.9	519.24	0.00
32	8408.61	410.86	**7405.79**	441.94	11.93
33	8555.58	486.25	**7556.83**	535.55	11.67
34	5536.63	425.8	**4945.34**	474.42	10.68
35	4444.59	401.58	**4253.26**	424.73	4.30
36	3669.89	605.31	**3064.76**	619.48	16.49
Avg	4167.29	164.30	4049.86	191.52	2.06

Table 9 provides the average comparison results for the unrestricted and non-oriented 2L-HFVRP of Classes 2-5. By observing the data in Table 4, we realize that the average improvement percentage for all the instances is up to 1.6% and the running time is also reasonable and acceptable.

Table 9. Average Comparison results for the Unrestricted and Non-Oriented 2L-HFVRP (Class2-5)

No.	SA_HLS		PSO_HLS		%Gap
	Cost	Time(s)	Cost	Time(s)	
1	600.77	29.89	**591.46**	40.91	1.55
2	699.21	32.88	**690.52**	56.22	1.24
3	770.12	33.84	**768.43**	81.31	0.22
4	698.19	30.04	**689.71**	45.02	1.21
5	786.84	27.68	786.84	51.97	0.00
6	831.32	42.78	831.39	60.24	-0.01
7	5630.02	31.08	5630.02	49.17	0.00
8	5602.60	30.22	**5502.27**	44.23	1.79
9	1035.62	58.75	**1021.80**	79.02	1.33
10	7625.05	43.27	**7518.65**	71.83	1.40
11	8329.69	52.61	**8120.83**	82.45	2.51
12	1681.07	167.21	**1673.58**	201.44	0.45
13	25978.70	69.95	**25576.49**	79.28	1.55
14	10869.10	78.10	10876.37	101.40	-0.07
15	11490.10	91.53	**11486.31**	152.50	0.03
16	1291.87	110.89	1291.87	142.56	0.00
17	1776.54	191.01	1783.59	301.31	-0.40
18	5676.16	88.80	5676.77	107.80	-0.01
19	4242.48	180.33	**4232.54**	244.74	0.23
20	6153.31	175.47	6154.37	205.46	-0.02
21	8220.77	292.14	8224.77	401.93	-0.05
22	8574.44	317.36	8584.59	409.20	-0.12
23	8316.57	357.79	8316.57	414.56	0.00
24	4547.84	288.63	**4440.19**	396.02	2.37
25	11367.90	473.87	11367.90	503.49	0.00
26	11781.50	363.10	12772.63	405.22	-8.41
27	5695.23	282.47	5697.76	203.51	-0.04
28	22611.00	614.35	**20604.36**	582.43	8.87
29	21876.20	494.77	21881.79	489.34	-0.03
30	15793.10	811.79	**14788.31**	905.78	6.36
31	21125.50	1114.64	**20126.65**	1503.82	4.73
32	20110.70	968.59	**19100.58**	893.50	5.02
33	21419.60	857.77	**19421.82**	1043.43	9.33
34	14484.50	1596.04	**12486.43**	1805.67	13.79
35	8962.14	1213.94	**8709.22**	1711.24	2.82
36	4385.78	1119.91	4385.65	1904.73	0.00
Avg	8640.04	353.71	8383.70	438.13	1.60

Fig. 5 reflects the improvement percentages of the PSO_HLS algorithm over the best known algorithm in all the experiments for the 2L-HFVRP, consisting of both Class 1 and Class 2-5. It can be concluded from Fig. 5 that compared to the given best algorithm, the PSO_HLS has shown significant improvement. The PSO_HLS presents an excellent ability to solve the 2L-HFVRP. The biggest improvement percentage is over 16%, which signifies that our proposed approach is an effective solution to the 2L-HFVRP.

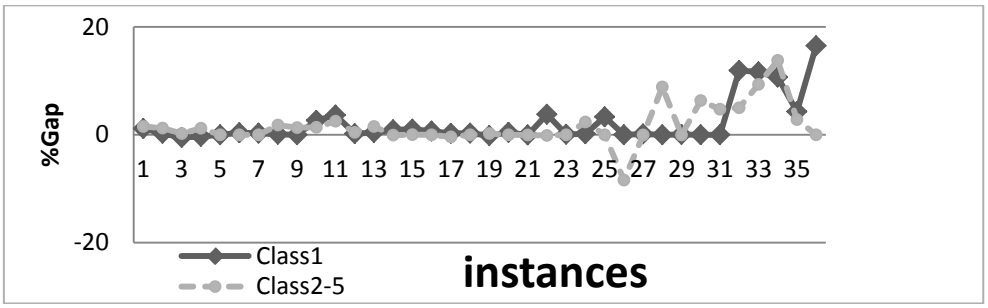

Fig. 5. 2L-HFVRP Results

7. Summary

This paper focuses on the heterogeneous fleet vehicle routing problems with two-dimensional loading constraints (2L-HFVRP) and its objective is to generate the minimum cost routes with feasible loading plans. To solve this problem, this paper proposed an improved particle swarm optimization algorithm with the heuristic local search (PSO_HLS) to enhance the diversity of the solutions. Additionally, in order to improve the loading efficiency, we designed a new heuristic loading algorithm named the Fitness Sorting Depth-First Heuristic (FSDFH). The FSDFH algorithm adopts the tree structure to record the routes' feasible loading information. To optimize the loading and route algorithms, the area contraction coefficient is introduced to help lead to a feasible solution. In the end, to test the robustness and effectiveness of the proposed approach, we initially solve a wide array of 2L-CVRP benchmark instances, compare the results with the known solutions and validate its excellent performance in solving 2L-CVRP. Moreover, testing the proposed algorithm on the 2L-HFVRP instances also returns an ideal outcome.

Acknowledgements

This research was funded by the National Nature Science Foundation of China (Grant No. 71172168). I would also like to thank Talisa Hernandez for her great support and help throughout the research.

References

[1] G.B. Dantzig, J.H. Ramser: The Truck Dispatching Problem. Management Science. Vol. 6(1) (1959), p. 80-91.

[2] B. Golden, A. Assad and L. Levy: The Fleet Size and Mix Vehicle Routing Problem. Computers & Operations Research. Vol. 11(1), (1984), p. 49-66.

[3] G.U. Clarke, J.W. Wright: Scheduling of Vehicle From A Central Depot To A Number of Delivery Points. Operations Research. Vol. 12(4), (1964), p. 568-581.

[4] M. Gendreau, A. Hertz and G. Laporte: A Tabu Search Heuristics for the Vehicle Routing Problem. Management Science. Vol. 40 (1994), p. 1276-1290.

[5] M. Iori, J.J. Salazar Gonzalez and D. Vigo: An Exact Approach for the Vehicle Routing Problem with Two-dimensional Loading Constraints. Transportation Science. Vol. 41(2) (2007), p. 253-264.

[6] M. Gendreau, M. Iori, and G. Laporte: A Tabu Search Heuristic for the Vehicle Routing Problem with Two-dimensional Loading Constraints. Networks. Vol. 51(1) (2008), p. 4-18.

[7] 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. Vol. 195(3) (2009), p. 729-743.

[8] G. Fuellerer, K.F. Doerner and R.F. Hartl: Ant Colony Optimization for the Two-dimensional Loading Vehicle Routing Problem. Computers & Operations Research. Vol. 36(3) (2009), p. 655-673

[9] C. Duhamel, P. Lacomme and A. Quilliot: A Multi-start Evolutionary Local Search for the Two-dimensional Loading Capacitated Vehicle Routing Problem. Computers & Operations Research. Vol. 38(3) (2011), p. 617-640.

[10] S.C. Leung, Z. Zhang and D. Zhang: A Meta-heuristic Algorithm for Heterogeneous Fleet Vehicle Routing Problems with Two-dimensional Loading Constraints. European Journal of Operational Research. Vol. 225(2) (2013), p. 199-210.

[11] J. Kennedy, R.C. Eberhart: Particle Swarm Optimization. Proc. IEEE International Conference on Neural Networks , IV. Piscataway , NJ : IEEE Service Center (1995), p. 1942 -1948.

[12]S. Leung, J. Zheng, D. Zhang and X. Zhou: Simulated Annealing for the Vehicle Routing Problem with Two-dimensional Loading Constraints. Flexible Services and Manufacturing Journal. Vol. 22 (2010), p. 61–82.

[13]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 & Operations Research. Vol. 38 (2011), p. 205–215.

[14] E.E. Zachariadis, C.D. Tarantilis, and C.T. Kiranoudis: Integrated Distribution and Loading Planning via a Compact Metaheuristic Algorithm. European Journal of Operational Research. Vol. 228 (2013), p. 56–71.

[15] P. Toth, D. Vigo: The Vehicle Routing Problems. Philadelphia, PA, SIAM Monographs on Discrete Mathematics and Applications (2002).

[16] M. Iori: Metaheuristic Algorithms for Combinatorial Optimization Problems. 4OR: A Quarterly Journal of Operations Research. Vol. 3(2) (2015), p.163-166.