An Enhanced Evolutionary Algorithm for the Two-Dimensional Loading Capacitated Vehicle Routing Problem with Mixed Backhauls and Linehauls

Yangguang Wang(✉), Chuang Liu, and Huaping Chen

University of Science and Technology of China,
Hefei 230026, Anhui, People's Republic of China
ygw@mail.ustc.edu.cn

Abstract. In this paper, we consider a new combinatorial optimization problem, namely the two-dimensional loading capacitated vehicle routing problem with mixed backhauls and linehauls (2L-VRPMBL). This is a more complicated but practical extension of the well-known vehicle routing problem with two-dimensional loading constraints (2L-CVRP). Compared with capacitated vehicle routing problem (CVRP) as the routing part in the 2L-CVRP, vehicle routing problem with mixed backhauls and linehauls are considered in the 2L-VRPMBL. Besides, two-dimensional loading constraints are assumed. 2L-VRPMBL aims at finding the minimum cost route set without violating all the related constraints. To solve this problem, a combinatorial packing method utilizing two basic bin packing heuristics is designed to make a specific packing scheme. In the meanwhile, in order to further increase the possibility of finding a feasible packing scheme, the packing method is integrated into a random local search process and is repeatedly called. Furthermore, an evolutionary algorithm with enhanced local search ability and broader search space is presented for routing. Finally, the packing heuristic method and evolutionary algorithm are integrated into a large evolutionary neighbourhood search framework. The effectiveness of the proposed algorithm is verified by conducting experiments on the new 2L-VRPMBL benchmark instances.

Keywords: Vehicle routing problem · Backhauls · Evolutionary algorithm

1 Introduction

The vehicle routing problems (VRPs) represent a series of realistic problems confronted in the fields of logistics and transportation, as well as in many other fields. The most representative prototype of the VRPs is the capacitated vehicle routing problem (CVRP) [12]. Given a set of customers with different delivery requirements and a fleet of homogeneous vehicles with a fixed load capacity, the CVRP aims to find the minimum cost (i.e., total travelled distance). The integration of the constraints of CVRP and two-dimensional loading constraints causes the 2L-CVRP, which was first proposed by [6]. More specifically, in addition to the constraints of the CVRP, it is assumed that the shape of each item is cuboid and that the items cannot be stacked because of their fragility or for the requirement of stabilization. In the related literature, four variants are involved depending on the loading method adopted and whether rotation is allowed [3,16]. The loading methods include the sequential loading and unrestricted loading. In addition, items can be loaded with rotation, that is, each item can be rotated by 90°, or without rotation. This paper focuses on a new realistic extension of the 2L-CVRP, known as 2L-VRPMBL, where the VRP with mixed backhauls and linehauls and two-dimensional bin packing are both taken into consideration. In addition, an unrestricted loading method is adopted, and two cases, i.e., the items can be rotated and cannot be rotated, are considered. To the best of our knowledge, this is the first time the problem has been studied in the literature. Real-life examples of this problem can be frequently found in furniture and industrial machinery transportation [3] and in the grocery industry [7]. To solve the problem proposed in this paper, an enhanced evolutionary algorithm is presented for the routing, and a combinatorial packing method combined with a random local search process is proposed for the loading. Finally, the packing heuristic method and the evolutionary algorithm are integrated into a large evolutionary neighbourhood search framework to find feasible solutions.

2 Literature Review

The combination of vehicle routing and two-dimensional bin packing, called 2L-CVRP, was first introduced in [6], and the authors proposed an exact approach to obtain the solutions for small-scale instances. Then, [18] proposed a guided tabu search which employs a collection of basic packing heuristics to check the loading feasibility. Later, [9] supplemented the collection of packing heuristics presented in [18] with an additional basic packing heuristic, which leads to a significant improvement in the quality of the solutions. [17] proposed a variable neighbourhood search for the routing and a skyline packing heuristics for the loading. [16] presented a heuristic packing method which runs on the basis of repeated calls to a basic packing method; in the meanwhile, a simulated annealing algorithm was presented for the routing component. In addition to the research on the original 2L-VRPB problem, some other extensions have also been studied. [8] introduced a new variant, called the two-dimensional loading heterogeneous fleet vehicle routing problem (2L-HFVRP), where it's assumed that the fleet consists of heterogeneous vehicles and there is no maximum limit for the number of vehicles of each type. Besides, a simulated annealing algorithm is presented to solve the problem. Later, by utilizing the biased randomization technique on basic CWS [2] heuristic and two basic packing heuristics, i.e. best-fit [1] and maximum touching perimeter (MTP) [10] heuristics, [4] proposed a multi-start algorithm to solve the problem. [3] introduced another new extension, called 2L-VRPB, which is a combination of the vehicle routing problem with backhauls

(VRPB) and the two-dimensional bin packing problem (2BPP). In the scenario of 2L-VRPB, for each route, all the deliveries for linehaul customers must be made before any pickups from backhaul customers. Similarly, the multi-start algorithm [4] can effectively solve this problem and give high-quality solutions.

3 Problem Description

The 2L-VRPMBL can be defined as follows. Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a complete graph, where $V = \{0, 1, \ldots, N\}$ is the vertex set and $E = \{(i, j)|i, j \in V, i \neq j\}$ is the edge set. The vertex set V is composed of three subsets $V = V^0 \cup V^l \cup V^b$: subset V^0 has only one vertex v_0, which represents the central depot, while disjoint subsets V^l and V^b represent linehaul and backhaul customers, respectively. For concise representation, let $V^c = V^l \cup V^b$. For each edge $(i, j) \in E$, the associated travel cost c_{ij} is defined as the direct travel distance from v_i to v_j or from v_j to v_i. In the central depot, a fleet of K homogenous vehicles is available. Each vehicle has a maximum loading capacity D and a rectangular loading surface $S = W \times L$. Each customer $i \in V^c$ has m_i rectangular items, denoted as I_i, which need to be delivered (linehaul) or collected (backhaul) and whose total weight is d_i. The width and length of each item $I_{ir} \in I_i$ are $w_{ir}\ (r = 1, \ldots, m_i)$ and $l_{ir}\ (r = 1, \ldots, m_i)$, respectively. Thus, the total area of I_i is denoted as $s_i = \sum_{r=1}^{m_i} w_{ir} l_{ir}$.

The goal is to find a set of routes servicing all customers, such that the total travel cost is minimized. In addition, the constraints can be classified into two types: the first type is about the routing, and the second is about the packing. Both types of constraints are described as follows.

1. Each vehicle starts and finishes its task at the central depot.
2. No more than K vehicles are used to satisfy the customers' demands.
3. Each customer, linehaul or backhaul, is visited exactly once. To satisfy this requirement, all the items belonging to the same customer should be loaded into one single vehicle.
4. In each route, the vehicle's capacity and loading surface area is not allowed to be exceeded.
5. The positions of the linehaul and backhaul customers are mixed arrayed in one route.
6. All items assigned to the same vehicle must be loaded without overlapping, and the edges of the items must be parallel to the edges of the vehicle.
7. According to the loading configuration, items can be loaded without rotation (unrestricted oriented loading), or with a 90-degree rotation (unrestricted rotation-allowed loading).

4 The Enhanced Evolutionary Algorithm

In this paper, an enhanced evolutionary algorithm is employed to design route set for the fleet. The algorithm is based on the Hybrid Genetic Search with Adaptive

Diversity Control (HGSADC) presented in [13,15]. HGSADC is proven to have strong search capabilities for routing and can effectively handle a wide variety of vehicle routing problems. To deal with the more complicated solution structure in this paper, tailor-made strategies and mechanism, involving the acceptance criteria in the local search, individual evaluations and the diversity population management mechanism, are redesigned. In the remainder of this section, we first introduce the search space of the solutions and individual evaluations, and then clarify the parent selection and crossover operation. Next, we further describe the generation process of new offspring, including the mutation operators, i.e. local search procedure. Finally, the initial population construction and population diversity maintenance mechanism is presented in detail.

4.1 Chromosome Representation and Evaluation

Chromosomes in traditional evolutionary algorithm refers to the individual solution with the corresponding encoding and decoding method. In the enhanced evolutionary algorithm, a complete chromosome is composed of three components: a giant tour, a route set and the fitness value. The chromosome is encoded by the representation of giant tour, which is first introduced by [11]. The giant tour is a permutation of all the linehaul and backhaul customers without trip delimiters. It can be understood as the whole visiting sequence of one vehicle in the context of the TSP problem. The representation makes it possible to apply some sequence based operators originally used in the TSP problem, like OX crossover etc.

Corresponding to the encoding method, the decoding method of the chromosome is through the application of a polynomial split algorithm [11]. In other words, the specific visiting routes for the vehicles is obtained by applying the split algorithm based on a shortest path Bellman–Ford algorithm to the giant tour. More specifically, in our context, we relax the restrictions for generating a single visiting route in the split algorithm. That is, instead of generating only feasible routes, one infeasible route can also be generated as long as the total surface area of its items do not exceed the total surface area of the vehicle. Hence, both the feasible and infeasible routes are generated in the auxiliary graph. However, the infeasible routes are penalized through its fitness value. That is, the infeasible route is penalized by adding penalty values to its fitness value according to its degree of deviation from the related constraints including the maximum load capacity and the packing feasibility.

4.2 Parent Selection

In each iteration, two parents are selected by the binary tournament method. That is, each time two individuals are randomly selected from the union of the feasible population and infeasible population, then the one with the better fitness value is chosen as a parent. Another parent is selected in the same way. The distinctive encoding and decoding method of chromosome makes it possible to apply recycling crossovers initially designed for TSP. In each iteration, two giant tours from the parents undergo the OX crossover, leading to the generation of two new giant tours. Then, one giant tour is randomly selected and the complete chromosome is generated by the application of the split algorithm.

4.3 Local Search Procedure

Similar to the effects of mutation operations on the traditional evolutionary algorithms, local search procedure enables the algorithm presented in this paper to have strong local search capability. When the algorithm has approached the neighbourhood of the optimal solution through the OX crossover operator, the local search procedure can speed up the convergence. The local search procedure takes a complete chromosome as the input. In the search process, four kind of operators, i.e., swap, relocate, intra-2opt and inter-2opt, are alternatively used to change and improve the current route set. The complete local search process involves two phases: the initial optimization phase and the repair phase.

In the initial optimization phase, for each iteration, all the possible promising pairs of customers [14] are examined, which means four kind of operators make the related changes on each pair of customers in random order, and in the meanwhile, calculate the corresponding fitness value of the involved routes. If one operator can lead to a smaller fitness value and the involved routes are feasible after the changes, the modification made by the operator is accepted. Furthermore, to speed up the convergence of the search process, the changes made by one operator is also accepted as long as the involved routes have changed from unfeasible to feasible. One iteration of the local search terminates when the modification yielding an improvement first occurs. Then, new iteration restarts. The initial optimization phase stops when all possible pairs of customers have been successively examined without making any improvements.

The repair phase can be regarded as two repeats of the initial optimization phase, but with different penalty parameter settings. More specifically, an individual generated by the OX crossover first undergoes the initial optimization phase, after which the modified one will be added to the related population according to its feasibility. If the modified one is infeasible, a repair phase is performed with a probability of 0.5. In the process of the repair phase, the penalty parameters are first multiplied by 10, and the first round of optimization work will be done. If the individual becomes feasible, it will be inserted to the feasible population. Otherwise, another round of optimization is performed with the penalty weights multiplied by 100, and the related insertion will be made if the individual changed into a feasible one. It is noteworthy that after each repair phase, all the penalty parameters are reset to 1.

4.4 Population Management Mechanism

Unlike the general evolutionary algorithm retaining only one population consisting of feasible individuals, two populations, one consisting of feasible individuals and the other consisting of infeasible individuals, keep evolving throughout the search process. In the initial phase, φ permutations of customers are randomly generated. Then these permutations go through the decoding and local search processes in sequence, and finally become the individuals with complete solution structures. After the initial phase, the algorithm enters a relatively stable stage, called the evolutionary phase. In this phase, a new offspring is generated by performing the parent selection, OX crossover and mutation (local search procedure) in sequence. It is noteworthy that the insertion of an individual to the corresponding population should respect a strict dispersal rule, that is, the interval of the fitness values between any two individuals in one population must be greater than 0.5. This rule aims at maintaining the diversity of the populations and prevent immature convergence. A new individual is discarded if it violates the dispersal rule when added to the related population.

The size of each population is maintained between λ and $\lambda + \delta$, and a survivor selection process is triggered once the size of one population has reached the maximum limit. In the survivor selection process, δ individuals with the worst fitness value are discarded. To further increase the overall diversity, a diversification procedure is executed when it^{div} iterations are performed without finding a better feasible solution. In the diversification phase, only the best $\lambda/3$ individuals in each population are retained. Then, the initialization phase restart and the populations are complemented with the newly generated individuals.

5 The HeuristicPack Algorithm for the Loading Subproblem

5.1 The HeuristicPack Program

For a given route visited by a vehicle, to check the loading feasibility, it would be necessary to carry out loading checks for multiple times on its different subroutes. First, it is necessary to carry out the loading check on the subroute composed of all its linehaul customers, so that all the items belonging to the linehaul customers can be successfully packed into the vehicle. Then, at each backhaul customer's location, the loading check is also required.

In general, the heuristicPack program aims at generating different item sequences and attempts to call the subprogram combinatorialPack as many as possible. The heuristicPack is improved on the basis of random local search RandomLS proposed by [16]. The combinatorialPack makes the specific packing scheme on the given input sequence, and the result is stored in two lists: the occupied list indicates those items already loaded and their corresponding position on the surface of the vehicle; the unoccupied list indicates the items which cannot be loaded into the vehicle. For a given customer sequence visited by a vehicle, let R_{set} represent the corresponding item set, and R_{seq} denote the initial item sequence. R_{seq} is generated by sorting all the items in descending order according to their surface area. Then, the subprogram combinatorialPack is first called on the initial sequence. If the result indicates that all the items have been loaded, the HeuristicPack terminates and a constant TRUE is returned. Otherwise, a heuristic improving process is triggered. The improving process attempts to optimize the input sequence by consistently swapping two different items in the item sequence. In addition, a parameter *noImp* is used to control the iterations of the improving process. That is, if *noImp* reaches the total number of items in the R_{seq}, the process stops. Otherwise, the search process continues. For a new item sequence generated by one swap operation, the procedure terminates once combinatorialPack succeed in making a feasible packing scheme. In other cases, the new sequence is accepted if the area utilization of vehicle is improved. If that occurs, *noImp* is reset to one; otherwise, *noImp* increases by one. To avoid making repeated loading checks for the same item sequence and in the meanwhile improve the running efficiency, each item sequence already checked by the combinatorialPack, together with its loading result, is stored in a hash table. Next time before evaluating a newly generated sequence, the program first attempts to retrieve it from the hash table. If the sequence is stored in the hash table, the corresponding packing scheme is directly used; otherwise, the combinatorialPack is called.

5.2 The CombinatorialPack Subprogram

Basically, the CombinatorialPack method works by employing two basic on-line packing algorithms, i.e. the best area utilization (BAU) heuristic [18] and the MTP heuristic [10]. Therefore, CombinatorialPack is essentially an on-line packing algorithm. In addition, combinatorialPack employs two packing patterns: item priority and free area priority, and BAU and MTP heuristics are called in turn in each pattern.

Let $freeList$ denote a list of free rectangles that represent the free rectangular area of the surface of the vehicle. At first, there is only one rectangle in $freeList$, i.e., the rectangular surface of the vehicle. Then, in each packing pattern, items are loaded one by one. Each time a free rectangle is selected from the $freeList$ to load the current item, the selected free rectangle will be erased from $freeList$, and the corresponding new free rectangles will be generated and added directly to $freeList$. Then, we have to check all the free rectangles in $freeList$ and remove some unnecessary rectangles.

The combinatorialPack subprogram is described in Algorithm 1. Considering one basic packing heuristic, i.e., BAU or MTP, in the item priority pattern, the packing heuristic method loads the items one by one according to their original positions in the input sequence. That is, the first item in the sequence will be first loaded, and then will be the second, the third, etc. More specifically, for a selected item, BAU attempts to find a free feasible rectangle, which means the item can be successfully packed into the rectangle, and in the meanwhile, the surface area of the rectangle is as small as possible; MTP attempts to find a feasible free rectangle resulting in the maximal total touching perimeter. While for the free area priority pattern, the next item to be loaded by the packing algorithm is the one best fitted to the most bottom-left free rectangle in the $freeList$. In other words, considering one step in the packing process, the most bottom-left free rectangle is first selected from the current $freeList$, and then the unloaded item with the highest evaluation score will be chosen and placed on that free

Algorithm 1. The CombinatorialPack Algorithm.

```
1: //Item priority pattern
2: Initialize $sourList$, $destList$
3: **for** $item_i \in sourList(i = 1, 2, \ldots, |sourList|)$ **do**
4:     $Execute\ BAU(sourList,\ destList, item_i)$
5:     $Update\ sourList,\ destList$
6: **end for**
7: **if** $sourListisempty$ **then**
8:     **return** $TRUE$;
9: **end if**
10: Initialize $sourList$, $destList$
11: **for** $item_i \in sourList(i = 1, 2, \ldots, |sourList|)$ **do**
12:     $Execute\ MTP(sourList,\ destList, item_i)$
13:     $Update\ sourList,\ destList$
14: **end for**
15: **if** $sourListisempty$ **then**
16:     **return** $true$;
17: **end if**
18: // Free area priority pattern
19: Initialize$freeList$, $sourList$, $destList$
20: **while** $freeList \neq \varnothing$ **and** $sourList \neq \varnothing$ **do**
21:     Select the most bottom-left rectangle $rect_\alpha$
22:     Find the $item_\beta$ leading to the highest area utilization of $rect_\alpha$
23:     **if** such item is found **then**
24:         Pack $rect_\beta$ into $rect_\alpha$;
25:         Update $freeList$, $sourList$, $destList$;
26:     **else**
27:         Remove $rect_\alpha$ from the $freeList$;
28:     **end if**
29: **end while**
30: **if** $sourList$ is empty **then**
31:     **return** $true$
32: **end if**
33: **while** $freeList \neq \varnothing$ **and** $sourList \neq \varnothing$ **do**
34:     Select the most bottom-left rectangle $rect_\alpha$
35:     Find the $item_\beta$ leading to the highest area utilization of $rect_\alpha$
36:     **if** such item is found **then**
37:         Pack $rect_\beta$ into $rect_\alpha$;
38:         Update $freeList$, $sourList$, $destList$;
39:     **else**
40:         Remove $rect_\alpha$ from the $freeList$;
41:     **end if**
42: **end while**
43: **if** $sourList$ is empty **then**
44:     **return** $true$
45: **end if**
46: //fail to generate a feasible packing scheme
47: **return** $false$
```

rectangle. Specifically, the evaluation score of BAU is the area utilization, and for MTP, it's the total touching perimeter. If no unloaded items can be loaded into the current free rectangle, the rectangle is removed and another new bottom-left rectangle is selected.

6 Computational Results

The algorithm proposed in this paper is implemented in C++ language. The hardware environment for running the program is a desktop computer equipped with a 3.4 GHz Intel i7 6700 processor and 16 GB of RAM; the software environment is Visual Studio 2019 and Windows 10 operating system. To verify the effectiveness and robustness of the proposed algorithm to 2L-VRPMBL, we generate new benchmark instances from the classical 2L-CVRP data set introduced in [5,6]. Like the method of generating benchmark for the 2L-VRPB (i.e. vehicle routing problem with backhauls and two-dimensional loading constraints) proposed in [3], we select a certain percentage of customers from the customer set as the linehaul customers and the others as the backhaul customers. More specifically, the first one of every four customers in the 2L-CVRP instance is selected as the backhaul customer, and the remaining customers belongs to the linehaul customers. The new data set has 180 instances and can be classified into five categories, from class one to class five. Each category includes 36 instances, and the scale of instance increases from 15 customers to 255 customers. For class one, the width and length of each item are equal to one unit, so the 36 instances in this class can be regarded as the data set for the vehicle routing problem with mixed backhauls and linehauls, which is purely a routing problem. For class two to class five, the items have different sizes, which is described in detail in [6].

For each benchmark instance, We conduct five experiments with different random seed settings and present the best solution result. The related parameters are set as follows: $\varphi = 50, \lambda = 12, \delta = 20, it^{div} = 600$. Furthermore, the algorithm terminates when the total number of crossovers in the evolutionary phase reaches 5000. It is noteworthy that this is the first time to study the two-dimensional loading capacitated vehicle routing problem with mixed backhauls and linehauls, so there are no other results for comparison. We present the detailed computational results for future comparisons. As mentioned before, we consider two scenarios: the oriented loading and the rotated loading. Since there is no difference between the oriented loading and rotated loading for the 36 instances in class one, the results are shown only once. As shown in Table 1, for each instance, the rotated loading leads to a lower cost than the oriented loading, which is easy to understand because allowing rotation brings greater flexibility when making a packing scheme.

Table 1. Experimental results of CPLEX VS GA.

#	Oriented loading					Rotated loading			
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 2	Class 3	Class 4	Class 5
1	257.86	263.73	263.73	259.97	266.72	263.73	263.73	259.97	259.97
2	295.02	295.02	308.01	295.02	295.02	295.02	305.81	295.02	295.02
3	311.25	330.03	340.91	326.77	311.25	317.92	340.91	325.05	311.25
4	363.21	363.21	363.21	363.21	363.21	363.21	363.21	363.21	363.21
5	341.86	341.86	345.61	341.86	341.86	341.86	345.61	341.86	341.86
6	421.10	421.10	424.17	421.10	421.10	421.10	424.17	421.10	421.10
7	527.12	666.97	653.44	633.09	600.24	660.27	647.21	633.09	600.24
8	527.12	653.96	682.53	653.44	588.43	653.09	676.87	653.44	588.43
9	533.64	533.64	533.64	533.64	533.64	533.64	533.64	533.64	533.64
10	477.66	590.18	581.73	655.63	569.45	557.69	519.56	627.03	567.75
11	477.66	596.69	579.21	704.47	550.30	578.48	571.89	668.53	550.30
12	489.19	510.62	493.95	507.14	489.19	496.37	489.19	498.07	489.19
13	1938.66	2332.80	2268.46	2258.40	2190.67	2287.82	2204.62	2255.51	2187.84
14	704.71	849.66	851.91	825.77	732.89	840.75	841.63	805.95	727.25
15	704.71	843.45	921.95	978.06	975.45	839.34	861.12	965.02	974.37
16	580.79	580.79	589.64	592.60	580.79	580.79	580.79	589.60	580.79
17	685.31	696.19	685.31	688.76	685.31	696.19	685.31	685.31	685.31
18	654.74	934.49	958.87	986.37	847.67	890.92	909.58	976.64	842.25
19	489.74	665.41	689.97	693.08	576.90	640.69	656.03	673.62	569.21
20	231.75	467.37	423.11	461.56	380.21	431.99	416.93	447.19	378.00
21	627.94	874.68	963.13	806.19	780.14	840.36	928.84	789.31	775.27
22	664.71	896.21	912.29	974.51	792.52	859.86	879.25	951.65	782.24
23	730.25	925.49	946.92	972.84	833.80	865.47	911.19	944.19	825.12
24	853.92	990.44	918.77	970.63	880.05	958.38	895.04	950.55	876.05
25	747.15	1209.08	1177.30	1233.81	999.98	1165.57	1126.30	1201.28	987.12
26	729.97	1127.39	1171.70	1145.19	1041.47	1089.32	1154.98	1134.02	1029.61
27	933.66	1108.33	1186.63	1121.94	1065.23	1054.84	1150.96	1106.10	1044.82
28	939.06	2152.49	2124.03	2180.57	1966.49	2055.43	2058.93	2113.62	1906.24
29	936.36	1792.61	1798.78	1835.05	1736.08	1746.05	1749.82	1799.73	1708.15
30	913.50	1511.69	1571.25	1534.18	1359.81	1449.58	1527.97	1484.42	1325.96
31	1129.31	1929.50	2011.33	2028.49	1692.12	1857.01	1935.62	1962.97	1679.31
32	1131.69	1941.26	1891.78	1921.28	1664.72	1856.57	1828.56	1874.38	1626.20
33	1103.15	1922.16	1991.44	2000.34	1675.62	1855.14	1927.33	1936.27	1643.53
34	589.92	987.17	1009.40	1024.76	868.89	941.34	975.75	995.65	858.59
35	736.08	1166.46	1209.89	1242.47	1036.05	1127.00	1168.61	1207.93	1017.36
36	502.42	1373.01	1420.65	1388.64	1235.01	1322.35	1369.37	1345.24	1209.94
Avg	674.51	967.92	979.57	987.80	886.90	937.09	950.73	967.30	876.74

7 Conclusion

In this paper, we introduce and study the two-dimensional loading capacitated vehicle routing problem with mixed backhauls and linehauls. This is a new combinatorial optimization problem consisting of routing and packing, which can be frequently found in logistics industry but has not been studied yet. Due to the introduction of the mixed backhauls, more packing feasibility checks need to be performed for one route. As a result, the complexity is greatly increased and the capability of packing becomes the main bottleneck of obtaining high-quality solutions. To solve this more complicated problem, a combinatorial packing method, together with a random local search process, is incorporated into an evolutionary algorithm with enhanced local search ability. The effectiveness of the proposed algorithm is verified through experiments conducted on new 2L-VRPMBL benchmark instances.

References

1. Burke, E.K., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Oper. Res. **52**(4), 655–671 (2004)
2. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**(4), 568–581 (1964)
3. Dominguez, O., Guimarans, D., Juan, A.A., de la Nuez, I.: A biased-randomised large neighbourhood search for the two-dimensional vehicle routing problem with backhauls. Eur. J. Oper. Res. **255**(2), 442–462 (2016)
4. Dominguez, O., Juan, A.A., Barrios, B., Faulin, J., Agustin, A.: Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. Ann. Oper. Res. **236**(2), 383–404 (2016)
5. Gendreau, M., Iori, M., Laporte, G., Martello, S.: A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. Netw. Int. J. **51**(1), 4–18 (2008)
6. Iori, M., Salazar-González, J.J., Vigo, D.: An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transp. Sci. **41**(2), 253–264 (2007)
7. Koç, Ç., Laporte, G.: Vehicle routing with backhauls: Review and research perspectives. Comput. Oper. Res. **91**, 79–91 (2018)
8. Leung, S.C., Zhang, Z., Zhang, D., Hua, X., Lim, M.K.: A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. Eur. J. Oper. Res. **225**(2), 199–210 (2013)
9. Leung, S.C., Zheng, J., Zhang, D., Zhou, X.: Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. Flex. Serv. Manuf. J. **22**(1–2), 61–82 (2010)
10. Lodi, A., Martello, S., Vigo, D.: Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS J. Comput. **11**(4), 345–357 (1999)
11. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**(12), 1985–2002 (2004)
12. Toth, P., Vigo, D.: The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Pennsylvania (2002)

13. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)
14. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. **40**(1), 475–489 (2013)
15. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. Eur. J. Oper. Res. **234**(3), 658–673 (2014)
16. Wei, L., Zhang, Z., Zhang, D., Leung, S.C.: A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. Eur. J. Oper. Res. **265**(3), 843–859 (2018)
17. Wei, L., Zhang, Z., Zhang, D., Lim, A.: A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. Eur. J. Oper. Res. **243**(3), 798–814 (2015)
18. Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T.: A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. Eur. J. Oper. Res. **195**(3), 729–743 (2009)