Computers & Industrial Engineering 156 (2021) 107252

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem

Wei Qin *, Zilong Zhuang, Zizhao Huang, Haozhe Huang

School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Keywords:
Vehicle routing problem
Heterogeneous fleet
Hyper-heuristic
Reinforcement learning

ABSTRACT

This study investigates a practical heterogeneous vehicle routing problem that involves routing a predefined fleet with different vehicle capacities to serve a series of customers to minimize the maximum routing time of vehicles. The comprehensive utilization of different types of vehicles brings great challenges for problem modeling and solving. In this study, a mixed-integer linear programming (MILP) model is formulated to obtain optimal solutions for small-scale problems. To further improve the quality of solutions for large-scale problems, this study develops a reinforcement learning-based hyper-heuristic, which introduces several meta-heuristics with different characteristics as low-level heuristics and policy-based reinforcement learning as a high-level selection strategy. Moreover, deep learning is used to extract hidden patterns within the collected data to combine the advantages of low-level heuristics better. Numerical experiments have been conducted and results indicate that the proposed algorithm exceeds the MILP solution on large-scale problems and outperforms the existing *meta*-heuristic algorithms.

1. Introduction

The vehicle routing problem (VRP) is a famous and well-studied field of research that was first introduced by Dantzig and Ramser (1959). This problem aims to find an optimal set of routing paths for a fleet with identical vehicles to serve a series of customers with the minimum transportation costs. For each routing path, a vehicle starts and ends at the depot and meets several customers' requirements (Braekers, Ramaekers, & Van Nieuwenhuyse, 2016). One variant of VRP is the heterogeneous vehicle routing problem (HVRP) which was first introduced by Golden, Assad, Levy, and Gheysens (1984). This problem extends VRP by adding different types of vehicles with various capacities and a fixed requirement of customers (Laporte, 2009). In reality, to serve all customers with minimum cost, heterogeneous fleets are preferred in road-based transportation due to unbalanced demand and cost constraints (Liu, Smith, & Qian, 2016). Besides, to meet the environmental restrictions in the routing area, a heterogeneous fleet with various carbon emissions and capacities has been widely employed (Hoff, Andersson, Christiansen, Hasle, & Lokketangen, 2010). Especially for some logistics companies, to cope with seasonal changes or door-to-door delivery service in transportation demand, it is necessary to rent vehicles to deal with the growing demand, thus finding a balance between reducing vehicle maintenance costs and expanding the market. However, there are many types of vehicles in the market, resulting in vehicles with various specifications and models in the same fleet (Huang, Huang, & Guo, 2019). HVRP is extremely common in most practical problems and has therefore attracted extensive attention.

Similar to VRP, HVRP usually considers a pre-defined fleet of capacitated vehicles with different capacities to fulfill a series of public demands from customers. Among the previous research, there are two major categories of this problem, which are the fleet size and mix vehicle routing problem (FSMVRP) and the heterogeneous fixed fleet vehicle routing problem (HFFVRP) (Koc, Bektas, Jabali, & Laporte, 2016). In the former problem, each type of vehicle has an unlimited number and specific cost. The type and route of each vehicle used should be determined to minimize the total cost. The latter problem is raised by Taillard (1999), which determines the fleet size in advance and reduces the total travel time and cost by making full use of all vehicles. Since the transportation fleet of logistics companies is usually decided before route planning, the latter problem may be closer to reality. Therefore, this study focuses on the heterogeneous fixed fleet vehicle routing problem and introduces minimizing the longest time cost as the objective.

Although HVRP is well-known, it remains a challenge to find the optimal solution to the problem. As a classic problem in the field of

* Corresponding author.
E-mail address: wqin@sjtu.edu.cn (W. Qin).

https://doi.org/10.1016/j.cie.2021.107252

Available online 22 March 2021

combinatorial optimization, VRP has been proved to be an NP-hard problem. Due to the existence of various vehicles with different capacities, HVRP needs to determine the route of each vehicle which is the same as VRP, as well as the number of each type of vehicle. Therefore, HVRP is more difficult to solve, as it includes VRP as a particular case (Baradaran, Shafaei, & Hosseinian, 2019). Since the first introduction of the problem, many researchers have been trying to approach the optimal solution, however, efforts are still required to make full use of various kinds of vehicles, further improve the performance and solve practical problems.

To narrow the above research gap, this study develops a novel reinforcement learning-based hyper-heuristic (RLHH) to improve the quality of solutions. The main novelties and contributions of this study are threefold. Firstly, a mixed-integer linear programming model (MILP) is established for HVRP which can accurately solve small-scale HVRP and provide accurate performance evaluation for *meta*-heuristics. Secondly, to offer a high-quality solution to large-scale HVRP, a hyper-heuristic framework is developed to make full use of the advantages of various *meta*-heuristics. Thirdly, in the proposed RLHH framework, reinforcement learning and deep learning are used to improve the combination quality.

The remainder of this study is organized as follows. Section 2 provides a brief literature review. Section 3 describes the problem and formulates the MILP model. Section 4 presents the proposed reinforcement learning-based hyper-heuristic. In Section 5, computational experiments are conducted to investigate the performance of the proposed algorithm. Finally, conclusions and future directions are summarized in Section 6.

2. Literature review

2.1. Heterogeneous vehicle routing problem

In HVRP, researchers have developed many different solutions, and significant progress has been made in the past few decades. Similar to VRP and many other NP-hard problems, typical approaches include exact methods and heuristic algorithms. In the field of exact algorithms, four exact algorithms have been developed to solve the two branches of HVRP, including mixed-integer linear programming (Li, Wang, & Zhang, 2018), column generation algorithm (Choi & Tcha, 2007), set partitioning formulation (Baldacci & Mingozzi, 2009), and branch-cut-and-price algorithm (Pessoa, Sadykov, & Uchoa, 2018). These methods can successfully solve small-scale HVRP within an acceptable time. However, as the scale of the problem increases, the computational space and time required by exact methods increase exponentially, making it impractical to find the optimal solution for large-scale problems. In comparison, heuristic algorithms are more practical to find near-optimal solutions while ensuring high efficiency.

Leggieri and Haouari (2017) adopted the reformulation–linearization technique to lift the formulation of the HVRP, which is a polynomial-size ordered path-based formulation. The proposed polynomial-size formulation could achieve better solutions and improve the quality of approximate solutions. Mohamed, Klibi, Labarthe, Deschamps, and Babai (2017) introduced multiple heuristic rules to HVRP in city logistics, using the greedy method to build initial solutions and many other insert rules to improve them. Fu, Aloulou, and Triki (2017) developed a two-phase iterative heuristic to solve large-scale problems, and the algorithm was proved beneficial. Besides, *meta*-heuristics have been widely used in HVRP and its different variants, such as evolutionary algorithm (Ghannadpour & Zarrabi, 2019), tabu search (Yousefikhoshbakht, Didehvar, & Rahmati, 2014), ant colony optimization (Huang, Blazquez, Huang, Paredes-Belmar, & Latorre-Nuñez, 2019) and simulated annealing (Pinto, Vitorugo, Rosa, Arpini, & Caprini, 2018).

Hybrid *meta*-heuristic algorithms with various strategies added to *meta*-heuristic have also been widely used in HVRP and its variants. Li et al. (2018) considered the fuel and carbon emissions during fleet routing and developed a split-based adaptive tabu search that combines an optimal split scheme and an adaptive tabu search algorithm. Penna, Subramanian, Ochi, Vidal, and Prins (2019) introduced a hybrid *meta*-heuristic that combines an iterated local search with variable neighborhood descent to produce high-quality solutions for HVRP. Derbel, Jarboui, and Bhiri (2019) implemented a variable neighborhood search method with a threshold accepting mechanism to solve HVRP as well. For variants of HVRP, Liu (2013) developed a hybrid population heuristic with a local search for two variants of HVRP. Vidal, Crainic, Gendreau, and Prins (2014) introduced a unified hybrid genetic search *meta*-heuristic with problem-independent local search and advanced diversity management methods for multi-attribute vehicle routing problems. Avci and Topaloglu (2016) presented an adaptive local search integrated with tabu search for HVRP with simultaneous pickup and delivery. Coelho et al. (2016) added multi-trips and docking constraints to HVRP and designed a trajectory search heuristic to solve the problem, which combines iterated local search, greedy randomized adaptive search procedure, and variable neighborhood descent. Hiermann, Puchinger, Ropke, and Hartl (2016) introduced time windows and recharging stations to HVRP and presented an adaptive large neighborhood search that combines an embedded local search with a labeling procedure to improve the solution quality, which was proved to be efficient. To make the literature overview clearer, Table 1 summarizes the above works. To conclude, although there are many solution methods for HVRP, there is still much room for improvement in algorithm performance.

Table 1
Literature overview of HVRP.

Method	Article	Specific method
Exact methods	Choi and Tcha (2007)	Column generation algorithm
	Baldacci and Mingozzi (2009)	Set partitioning formulation
	Li et al., 2018	Mixed-integer linear programming
	Pessoa et al., 2018	Branch-cut-and-price algorithm
Heuristic methods	Leggieri and Haouari (2017)	Reformulation–linearization technique
	Mohamed et al. (2017)	Insert heuristic rules
	Fu et al. (2017)	Two-phase iterative heuristic
Meta-heuristic methods	Yousefikhoshbakht et al. (2014)	Tabu search
	Hiermann et al. (2016)	Adaptive large neighborhood search
	Huang et al. (2019)	Ant colony optimization
	Derbel et al. (2019)	Variable neighborhood Search
	Pinto et al. (2018)	Simulated annealing
	Ghannadpour & Zarrabi, 2019	Evolutionary algorithm
Hybrid *meta*-heuristic methods	Liu, 2013	Hybrid population heuristic
	Vidal et al. (2014)	Hybrid genetic search *meta*-heuristic
	Coelho et al. (2016)	Trajectory search heuristic
	Avci and Topaloglu (2016)	Adaptive local search with tabu search
	Li et al. (2018)	Split-based adaptive tabu search
	Penna et al. (2019)	Hybrid variable neighborhood descent
	This study	RLHH

2.2. Hyper-heuristic method

Since the concept of hyper-heuristic was raised, it has been widely accepted as a promising technique for synthesizing multiple algorithms. Hyper-heuristic is a search method that selects or generates new heuristics from a series of potential heuristics instead of solving the combinatorial problems directly (Pillay & Qu, 2018). As an attempt to provide higher quality solutions to combinatorial optimization problems, hyper-heuristic operates on the space of heuristics rather than the

space of direct solutions and therefore can be considered as a kind of high-level methodology that automatically produces an appropriate combination of low-level heuristics (LLHs) to solve problems effectively (Epitropakis & Burke, 2018). Therefore, in recent years, many studies have focused on this field and applied hyper-heuristic methods to various problems. Specifically, hyper-heuristic has been introduced to urban transit route design (Ahmed, Mumford, & Kheiri, 2019), vehicle routing (Olgun, Koç, & Altıparmak, 2020), assembly line sequencing (Mosadegh, Ghomi, & Süer, 2020), multi-objective optimization (Zhang, Ren, Li, & Xuan, 2020), etc. Therefore, it can be safely concluded that hyper-heuristic is an effective tool to improve the performance of heuristics. However, how to apply the hyper-heuristic method in HVRP and how to select or generate new heuristics from a series of LLHs for HVRP remains to be studied.

2.3. Reinforcement learning

As a powerful decision-making tool, reinforcement learning (RL) has attracted extensive attention. It learns which action to take in a given situation to achieve the biggest reward through interaction with an environment (Sutton & Barto, 2018). Many successful applications of reinforcement learning have been reported such as games (Silver et al., 2016; Silver et al., 2017), vehicle routing problem (James, Yu, & Gu, 2019), critical nodes identification (Fan, Zeng, Sun, & Liu, 2020), semiconductor fabs (Hwang & Jang, 2020), etc. In the field of hyper-heuristic, reinforcement learning can be naturally used as a high-level strategy to perform the selection of LLHs. Li, Özcan, and John (2017) develops two learning automata-based selection hyper-heuristics for multiobjective optimization with three well-known multiobjective evolutionary algorithms as LLHs. Choong, Wong, and Lim (2018) considered Q-learning as the high-level strategy and heuristic rules as LLHs, which was competitive in six different problem domains. Therefore, it can be seen that reinforcement learning has been considered during the design of hyper-heuristic and much progress has been made in recent years.

However, to our knowledge, most of the applications are limited to value-based reinforcement learning, only few studies focus on policy-based methods. Since value-based reinforcement learning may have the disadvantages of limited problem description and random policy which are common in the search space of LLHs, their performance can be further improved by using policy-based learning methods. Besides, existing reinforcement learning-based hyper-heuristics have not taken deep learning into account, which can classify the environment and enhance the performance of reinforcement learning. Therefore, this study takes a policy-based reinforcement learning named distributed proximal policy optimization (DPPO) as a high-level strategy. where deep learning is used to classify the search space, and several *meta*-heuristics are used as LLHs to seek high-quality solutions for HVRP.

In summary, although various *meta*-heuristics have been developed to solve HVRP due to its practical applications, there is much room for improvement in the quality of the solution. Meanwhile, hyper-heuristics and reinforcement learning have attracted widespread attention in recent years, and have achieved fruitful results in many fields. Therefore, this study aims to investigate the reinforcement learning-based hyper-heuristic for HVRP, and develops a novel hyper-heuristic with policy-based reinforcement learning.

3. Preliminaries

3.1. Problem formulation

In the HVRP discussed in this study, we focus on the slowest vehicle and try to minimize the largest distance of each vehicle instead of considering the total distance of all vehicles in standard HVRP. Since the total time cost is closely related to the maximum distance, this objective is essential for fleet owners. However, due to the particularity of the goal, the modeling of the problem becomes more difficult. So graph theory is used to describe the problem in this study (Avci & Topaloglu, 2016): Let $G=(N, A)$ be a complete graph, where $N=\{n_0,n_1,\ldots,n_n\}$ is the set of vertexes and $A = \{(n_i, n_j) : n_i, n_j \in N, i \neq j\}$ is the set of arcs. Vertex n_0 denotes the depot while the other vertexes correspond to customers. Each customer n_i requires a nonnegative demand r_i while each arc between any two vertexes i and j is associated with a distance $d_{i,j}$ (also named cost or travel time). Distances are known, symmetric and satisfy the triangle inequality: $d_{i,j} \leq d_{i,l} + d_{l,j}$, with l an additional vertex (Liu, 2013). The fleet of vehicles involves h different vehicle types, and the kth type of vehicles has a vehicle capacity Q_k. The decision-maker has to determine a set of routes to minimize the largest total distance of each vehicle, on condition that each vehicle starts and ends its route at the depot n_0, and the demand of every customer is satisfied by exactly one vehicle (Tarantilis, Zachariadis, & Kiranoudis, 2008). Before presenting the mathematical model, the notations including sets, parameters, and variables used in the model are listed in Table 2.

Table 2
Notations used in the mathematical model.

Notation	Description
N	set of all customers, {1,2,…,n}
N_0	set of all customers and the depot,$N_0 = N \cup \{0\}$
H	set of all types of vehicles, {1,2,..,h}
Q_k	capacity of the kth type of vehicles$k \in H$
$d_{i,j}$	distance between nodes$i, j \in N_0$
r_i	demand of the ith customer,$i \in N$
x_{ijk}	takes value 1 if arc(i,j) is traversed by the kth type of vehicle and 0 otherwise
y_{ijk}	total load picked up by the kth type of vehicle while traversing arc(i,j)

Then the MILP model for HVRP is as follows:

min z (1)

subject to

$$\sum_{k\in H}\sum_{i\in N_0} x_{ijk} = 1 \ \forall j \in N \tag{2}$$

$$\sum_{i\in N_0} x_{ilk} - \sum_{j\in N_0} x_{ljk} = 0 \ \forall l \in N, \forall k \in H \tag{3}$$

$$y_{0jk} = Q_k * x_{0jk} \ \forall j \in N, \forall k \in H \tag{4}$$

$$\sum_{i\in N_0} y_{ilk} - \sum_{j\in N_0} y_{ljk} = r_l * \sum_{i\in N_0} x_{ilk} \ \forall l \in N, \forall k \in H \tag{5}$$

$$y_{ijk} \leqslant Q_k * x_{ijk} \ \forall i, j \in N_0, \forall k \in H \tag{6}$$

$$z \geqslant \sum_{i\in N_0}\sum_{j\in N_0} x_{ijk} * d_{ij} \ \forall k \in H \tag{7}$$

$$x_{ijk} \in \{0, 1\} \ \forall i, j \in N_0, \forall k \in H \tag{8}$$

$$y_{ijk} \geqslant 0 \ \forall i, j \in N_0, \forall k \in H \tag{9}$$

The objective function of the model is given in (1) which minimizes the completion time of the last vehicle to leave the system. Constraint set (2) ensures that each customer is visited exactly once. Constraint set (3) indicates that each customer is visited and left by the same vehicle. Constraint set (4) implies that the initial load of one route is determined by the capacity of the vehicle used. Constraint set (5) are flow equations for covering the demand of each customer by only one vehicle. Constraint set (6) provides the upper limit for each arc traversed. Constraint set (7) indicates that the objective value is not less than the complete time of any vehicle. Constraint sets (8) and (9) define the decision variables.

3.2. Reinforcement learning formulation

Reinforcement learning is a framework for learning from interaction to achieve goals. The learner is called an agent, while the thing that it interacts with is called an environment. In reinforcement learning, an agent interacts continually with an environment, where the agent chooses actions, and the environment responds to these actions and evolves to new situations for the agent. Besides, the environment gives rise to rewards, which the agent attempts to maximize over time. Specifically, the agent and environment interact in a sequence of discrete-time steps. At each time step t, the agent receives the representation of the environment state $S_t \in S$, where S is the set of possible states. Then, the agent needs to select an action $A_t \in A(S_t)$, where $A(S_t)$ is the set of actions available under state S_t. After one time step, the agent receives a numerical reward $R_{t+1} \in R$, and enters a new state. A complete specification of the environment defines a task, which is one instance of the reinforcement learning problem (Sutton & Barto, 2018).

A reinforcement learning task that meets the Markov property is called a Markov decision process (MDP). If the state and action spaces are finite, it is a finite MDP, which can be defined by its state and action sets and one-step dynamics of the environment. The dynamics of a finite MDP can be formulated as follows: given any state s and action a, the probability of each possible pair of next state s' and reward r is denoted $p(s', r|s, a) \doteq Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\}$. The agent's policy refers to the mapping from states to probabilities of selecting each possible action at any time step, denoted π_t. The probability that $A_t = a$ when $S_t = s$ is denoted $\pi_t(a|s)$. Reinforcement learning methods specify how the agent changes its policy based on its experience, where the agent's goal is to maximize the total amount of reward it receives for executing these actions in the long term. This framework is abstract and flexible, and can be applied to many different problems in many different ways.

4. Methodology

In HVRP, many *meta*-heuristics have been applied but how to develop a more efficient algorithm to achieve a better solution is still under research. To deal with the challenge, reinforcement learning-based hyper-heuristic (RLHH) is developed to make full use of the advantages of various *meta*-heuristics. RLHH can be divided into two levels, namely LLHs and high-level control strategy. Since LLHs have different advantages, a high-level strategy is required to select the appropriate low-level heuristic at each decision moment. Moreover, reinforcement learning is used as the high-level strategy to fulfill the demand for choosing the proper low-level heuristic according to the environment. Specifically, in LLHs, traditional and newly developed *meta*-heuristics are used to expand the algorithm pool as large as possible. The detailed information of these *meta*-heuristics is shown in later parts. The high-level strategy adopts a state-of-art policy-based reinforcement learning algorithm, namely distributed proximal policy optimization (DPPO), which takes the asynchronous advantage actor-critic (A3C) based multiple threads method to accelerate the entire process and proximal policy optimization to obtain an adaptive learning rate to converge policy gradient quicker (Mnih et al., 2016; Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). Besides, deep learning is applied to compute policy gradient due to its advantage of larger state space and better performance. The whole procedure of the proposed RLHH and reinforcement learning strategy are explained in detail as follows.

4.1. Hyper-heuristic framework

In HVRP, due to the challenge of problem mapping and solving, the hyper-heuristic framework is used to form the whole procedure of the proposed RLHH. To apply the hyper-heuristic method to the HVRP, the overall procedure, problem mapping, and LLHs should be developed and these parts will be explained below.

4.1.1. Overall procedure

During the operation process of the proposed RLHH, the population of *meta*-heuristics is generated at first. Then, the state is calculated, and DPPO is used to compute the policy gradient to find out the action which shows the next algorithm to be taken. After that, the chosen *meta*-heuristic runs at a fixed generation, and then a new population is generated. Finally, the process is repeated until the terminal condition is met. The whole framework of RLHH is shown in Algorithm 1.

Algorithm 1: RLHH()

input: location of depot l_d, location of customers l_c, demand of customers d_c and capacity of vehicles c_v
output: algorithm model or routing result
begin
1:Initialize actor's new policy network $actor_{new}$, *a_para*; actor's old policy network $actor_{old}$, a_para_{old}; critic network critic, *c_para*
2:Initialize *meta*-heuristics LLH[*i*], $i = 1, 2, \ldots,$ m
3:**if** training process **then**
4:Initialize a central agent thread t_c, multiple worker threads t_{wi}, $i = 1, 2 \ldots w_n$ and thread event *mutex*
5:state buffer, action buffer, reward buffer s_b, a_b, $r_b \leftarrow$ empty
6:**for** $num \leftarrow 1$ **to** $episode_{max}$ **do**
7:**for** $k \leftarrow 1$ **to** w_n **do**
8://the following program uses t_{wk}
9:new training problem environment *depot, customers* and *vehicles* $\leftarrow l_d, l_c, d_c, c_v$
10:population $x[i] \leftarrow$ Rand (), $i = 1, 2, \ldots, P$
11:individual best result $ibest[i] \leftarrow x[i]$, $i = 1, 2, \ldots, P$
12:state $s \leftarrow$ scores and differences of x
13:**for** $i \leftarrow 1$ **to** dm_{max} **do**
14:policy gradient $w \leftarrow actor_{new}(s)$
15:a, s_{new}, $r \leftarrow$ ApplyAction(w, i)
16:s_b, a_b, $r_b \leftarrow s_b$, a_b, $r_b + s$, a, r
17:$s \leftarrow s_{new}$
18:$t_{max} \leftarrow$ length of r_b
19:**if** $t_{max} > batch_{min}$ **then**
20://the following line uses t_c
21:TrainNetwork(*mutex*)
22:**end**
23:**end**
24:**end**
25:algorithm model $\leftarrow$ *a_para*, *c_para*
26:**return** algorithm model
27:**else**
28:problem environment *depot, customers* and *vehicles* $\leftarrow l_d, l_c, d_c, c_v$
29:population $x[i] \leftarrow$ Rand (), $i = 1, 2, \ldots, P$
30:individual best result $ibest[i] \leftarrow x[i]$, $i = 1, 2, \ldots, P$
31:state $s \leftarrow$ scores and differences of x
32:**for** $i \leftarrow 1$ **to** dm_{max} **do**
33:policy gradient $w \leftarrow actor_{new}(s)$
34:a, s_{new}, $r \leftarrow$ ApplyAction(w, i)
35:$s \leftarrow s_{new}$
36:**end**
37:best result index *best* $\leftarrow$ index of $\times$ with biggest score
38:routing result $\leftarrow x[best]$
39:**return** routing result
end

4.1.2. Problem mapping

In the hyper-heuristic method, the problem mapping schema includes two parts, namely the environment between high-level strategy and LLHs, and the coding schema of LLHs for HVRP. On one hand, the environment is the bridge between the high-level strategy and the problem. High-level strategy can learn the problem and its change history only from the environment, so the environment should contain all characteristics of the problem. In RLHH, the high-level strategy learns from the process of the *meta*-heuristics chosen. Therefore, the feature of the problem is the population of *meta*-heuristics and its score. Also, for the convenience of calling the specific low-level heuristic, the best history result of every individual and scores of them are included in the environment.

On the other hand, the coding schema is the bridge between the

Table 3
Location of depot and customers.

Index	Location	Demand
1	(0.530,0.419)	8
2	(0.958,0.403)	4
3	(0.518,0.579)	8
4	(0.623,0.438)	2
5	(0.042,0.108)	5
6	(0.893,0.332)	3
7	(0.990,0.820)	8
8	(0.719,0.802)	7
9	(0.709,0.541)	9
10	(0.876,0.824)	7

Table 4
Capacity of the heterogeneous fleet.

Index	Capacity
1	15
2	20
3	25

meta-heuristic and the specific problem. A proper schema can formula the problem well and make the search space as small as possible to maximize the algorithm efficiency. So the coding schema should be explained before introducing LLHs. After considering the length of code and decoding difficulty, we found that the following coding schema may be suitable. For a HVRP with h vehicles and n customers, the solution code is a sequence with length $n+h$-1 and integer number from 1 to $n+h$-1, which is shown in (10).

$$Sol = \{1, 2, \cdots, n+h-1\} \tag{10}$$

In the sequence, numbers from 1 to n represent customers and numbers from $n+1$ to $n+h$-1 represent separators of different vehicles. The separators separate the solution into h parts, and each part is linked to a specific vehicle. The customers in each part are to be served by this specific vehicle. When serving customers, vehicles are loaded and move to customers one by one until the vehicle is full. After that, the vehicle moves to the depot and load more goods to serve the next customer until all customers are served. To make this process clear, one simple example is shown below. In the example, the problem has 10 customers and 3 vehicles. The location of depot and customers are shown in Table 3, while the capacity of the heterogeneous fleet is shown in Table 4. Therefore, the solution is a sequence with a length of 12, and one available solution is shown in (11).

$$Sol = \{3, 5, 2, 11, 4, 1, 9, 8, 12, 6, 10, 7\} \tag{11}$$

In the solution, the first number is 3, which means vehicle 1 goes to customer 3 at first. The second number is 5, which implies the second destination of vehicle 1 is customer 5. Since customer 3 needs 8 goods and customer 5 needs 5 goods and vehicle 1 has a capacity of 15 goods, vehicle 1 does not need to go back to the depot. The third number is customer 2, and customer 2 needs 4 goods. Therefore, vehicle 1 will go to the depot and then go to customer 2. Then the number 11 appears, which is larger than 10. Therefore, vehicle 1 has finished its trip, and then we consider vehicle 2 in the following route. The process is done repeatedly, and the final route can be found out.

Although some solutions cannot be represented by the solution described above because vehicles will load as much as possible in this coding schema, the best solution of HVRP should have been covered by this schema. Besides, one sequence has only one route result. Therefore, a sequence representing the order of all customers is suitable for *meta*-heuristics.

4.1.3. Low-level heuristics

Several *meta*-heuristics that have been reported that have been reported to be effective for HVRP are taken as LLHs, including artificial bee colony algorithm (Baradaran et al., 2019), ant colony optimization (Huang et al., 2019), cuckoo search (Teymourian, Kayvanfar, Komaki, & Zandieh, 2016), genetic algorithm (Liu, Huang, & Ma, 2009), particle swarm optimization (Yao, Yu, Hu, Gao, & Zhang, 2016), simulated anneal (Vincent, Redi, Hidayat, & Wibowo, 2017). Besides, ε-greedy method is used to jump out of local optima. Therefore, it can balance exploration and exploitation to provide a reliable solution with limited computing power and time. Moreover, due to the difference in the operating mechanism, different *meta*-heuristics have different focuses and advantages. Therefore, we can use several different algorithms to combine their advantage and get a better solution. These *meta*-heuristics used in this study are described below.

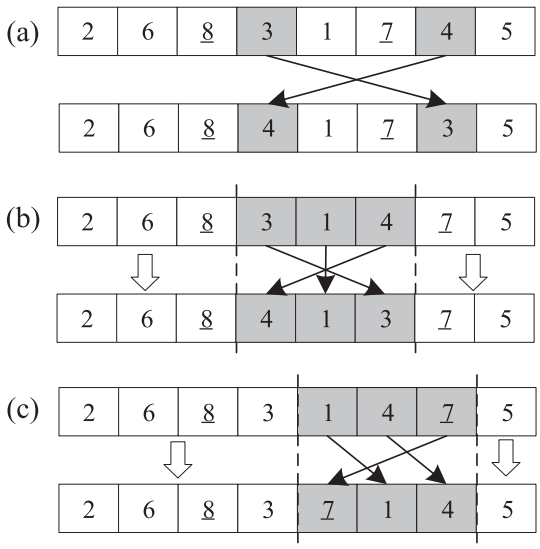


Fig.1. Examples of local search.

- **Artificial bee colony algorithm (ABC).** For each generation, employed bee, onlooker bee, and scout bee perform searching action one by one. The first two perform a local search to exchange customers, flip customers of a vehicle, or change the location of vehicle separators. Examples of local search are shown in Fig. 1, where the sequence represents a solution of 6 customers and 3 vehicles. In each case, numbers 7 and 8 with underline represent separators of different vehicles. The third kind of bee performs a global search, which generates a random solution. This algorithm loads food location directly from the population in the environment and updates the individual best during the process.
- **Ant colony optimization (ACO).** For each generation, all ants leave pheromone according to the score of paths, and pheromone evaporates at a fixed rate. In our system, after loading the population and individual best from the environment, we assume that ants move through these paths and leaves pheromone behind. After that, the algorithm begins its next iteration and finally outputs the ants' path as the population.
- **Cuckoo search (CS).** It employs levy flight in (12) to search for a good result and abandon strategy to perform better exploration. Therefore, it can balance global and local search and achieve a good result. During the process, the arrangement of the nest location is the population to be inputted and outputted. The location of nests is loaded from the environment, and the best result is kept when outputting.

$$x(t+1) = x(t) + \alpha Levy(\lambda) \tag{12}$$

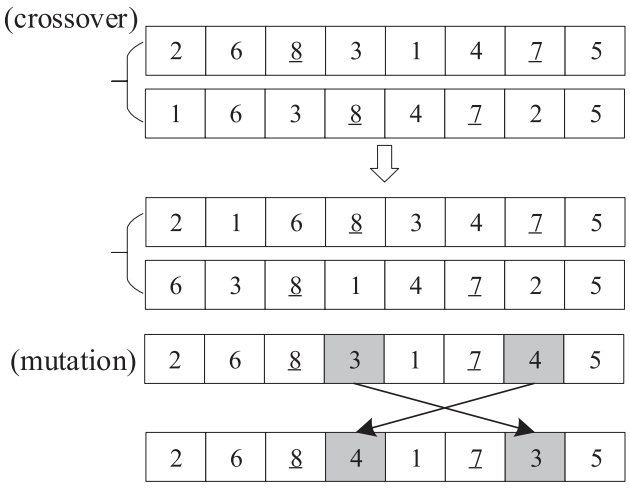


Fig. 2. Crossover and mutation example.

- **Genetic algorithm (GA)**. It uses selection, crossover, and mutation to get a better solution. In the proposed algorithm, the selection process chooses the better chromosome from randomly picked two chromosomes in the father generation. In the crossover process, the first appeared number is put in the first son chromosome while the second appeared number is in the second son chromosome. In the mutation process, randomly chosen numbers will exchange to explore search space. Examples of crossover and mutation processes are shown in Fig. 2. Besides, when using the algorithm, the population will be loaded directly from the environment, individual best will be updated, and the best result will be kept in population by using the elite strategy.
- **Particle swarm optimization (PSO)**. It uses the individual best and global best of the particle to guide all particles to move. The velocity and position of the particle are calculated in each generation by using (13) and (14), and the arrangement of position is the population used in the proposed algorithm. Besides, in our system, initial position and individual best are loaded from the environment, and they are updated through the computing process.

$$v(t) = wv(t-1) + r_1c_1(x_{gb} - x(t-1)) + r_2c_2(x_{ib} - x(t-1)) \tag{13}$$

$$x(t) = x(t-1) + v(t) \tag{14}$$

- **Simulated anneal (SA)**. It uses probability and allows a small amount of regression to improve the searching effect. In the proposed algorithm, the search method is the same as the local search of the artificial bee colony. Besides, the search times increases while the temperature becomes smaller. Finally, the population is loaded from the environment and is updated through the search process.

4.2. Policy-based reinforcement learning strategy

To improve the effectiveness of the hyper-heuristic framework, a policy-based reinforcement learning strategy called DPPO is used as the high-level strategy. To design the RL strategy, many attributes like state, action, reward, and training policy need to be determined. Besides, deep learning is introduced in the network structure to better integrate LLHs. The following parts will explain such features in detail.

4.2.1. State

After each update of the population by *meta*-heuristic, the RL strategy calculates the state of the environment, which is $s = \{f, diff\}$. Among the state, f is the score of each individual with the size of 1xp, where p is the population scale. Since the score is the makespan of the individual solution, and the output is the smallest makespan among all solutions, f also determines the output of the algorithm, which is the best score among all populations. Furthermore, to expand the scope of application, the score is normalized by dividing by the largest score during the entire process so that all scores are between zero and one. Meanwhile, $diff$ is a matrix with the size of pxp indicating the difference between all individuals, which is used to show the level of similarity of the population in the *meta*-heuristic and guide the decision of algorithm selection. The $diff_{ij}$ represents the difference between individual i and individual j of the population, which can be calculated using (15), where individual i and individual j are marked as ind_i and ind_j, and q represents the encoding length of each individual. The difference value ranges from zero to one and $diff$ is a symmetric matrix with diagonal elements of all zeros. An example of $diff$ is shown in Fig. 3. As shown in the figure, the first two individuals have 7 different elements. Since the encoding length is 8, the $diff_{12}$ is 0.875. Similarly, the $diff_{13}$ and $diff_{23}$ are 0.5 and 0.75, respectively. Therefore, the diff matrix can be calculated as follow.

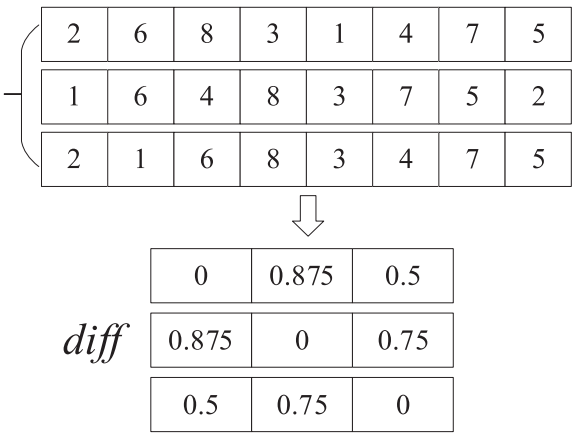


Fig. 3. Example of differences of individuals.

$$diff_{ij} = \frac{\left|\{t \in \{1, \cdots, q\} | ind_i(t) \neq ind_j(t)\ \}\right|}{q} \tag{15}$$

4.2.2. Action

During the learning and running process of RL strategy, actions should be taken to interact with the environment. In the proposed algorithm, the RL strategy will select the next *meta*-heuristic at each decision moment. So action $a = \{0, 1, 2, \ldots, m\}$ represents specific *meta*-heuristic and m is the number of *meta*-heuristic. When the RL strategy chooses the specific *meta*-heuristic that algorithm will be used to run several generations until the next decision moment reaches.

Besides, the key is to choose the correct action in different situations through exploration and exploitation. Exploration means searching in space that has not been visited while exploitation means doing the action which has been performed to get the reward. In the proposed algorithm, a common approach to deal with this conflict called ε-greedy method is used. At each decision moment, a random number is generated, and if the number is greater than a fixed probability ε ($0 \leq \varepsilon \leq 1$), the action is chosen greedily. Otherwise, the action is randomly selected. In this system, the actor-network will output policy gradient to evaluate the action's value which is converted to possibility through the softmax function. Then the action is chosen according to this possibility. Therefore, exploration and exploitation are balanced in the choosing process. After the action is chosen, the specific low-level heuristic is applied to update the population. The algorithm will run for several generations to eliminate the random error introduced by using a random number in it. Finally, the new state and reward will be calculated to provide data for network training. The pseudo-code of this part is shown

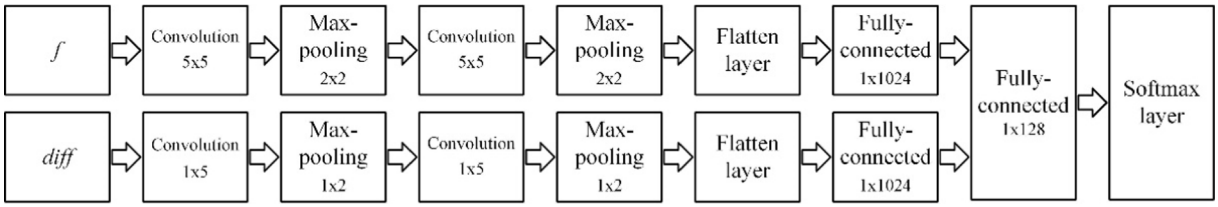


Fig. 4. Neural network structure.

in Algorithm 2.

Algorithm 2: ApplyAction()

input: policy gradient w and current decision moment index i
output: action a, new state s_{new} and reward r
begin
1:$temp \leftarrow$ Rand (range = [0,1])
2:**if** *temp* greater than 0.8 **then**
3:a $\leftarrow$ Randint (range = [0, m − 1], weight = w)
4:**else**
5:$a \leftarrow$ Randint (range = [0, m − 1])
6:chosen *meta*-heuristic $MH \leftarrow$ LLH[a]
7:new population $x_{new} \leftarrow x$
8:new individual best result $ibest_{new} \leftarrow ibest$
9:**for** $j \leftarrow 0$ **to** gen_{max} **do**
10:x_{new}, $ibest_{new} \leftarrow MH$ (x_{new}, $ibest_{new}$)
11:**end**
12:$x \leftarrow x_{new}$
13:$ibest \leftarrow ibest_{new}$
14:$s_{new} \leftarrow$ scores and differences of x_{new}
15:**if** $i < dm_{max}$ **then**
16:$r \leftarrow 0$
17:**else**
18:$r \leftarrow$ score improvement during the process
19:**return** a, s_{new} and r
end

4.2.3. Reward

The reward defines the goal of the RL strategy and provides immediate and afterward result of the action according to state changes. During the process, the RL strategy chooses an action to maximize the total reward and attempts to reach the final goal with the guide of the reward. In the proposed algorithm, the overall goal is to find out the optimal solution with the minimum completion time of all vehicles. Therefore, the reward given by the round is designed to find out the most effective algorithm selection policy. During the process, the reward is the score difference between the final state and the initial state to judge whether the selection sequence of LLHs is correct or not. This reward will be given to the RL strategy in the end, and the reward will be set to zero during the whole process.

4.2.4. Training policy

In the proposed RLHH, the actor-critic framework is introduced and the corresponding training method to achieve an acceptable result. Since the traditional actor-critic structure takes too much time to train, the A3C method is introduced in recent years (Mnih et al., 2016). A3C method is another modern technique in reinforcement learning which can make good use of multiple threads to accelerate the training process. In the proposed algorithm, the reinforcement learning system includes a central agent and multiple workers. The central agent is responsible for updating and training the network and provide workers with network parameters. The workers use network parameters from the central agent and run in parallel to collect state, action, and reward in different situations. This parallel training process can speed up the training process a lot and significantly reduce running time.

After workers collect enough running data for a training episode, the RL strategy should be trained to learn from the experience of all workers. The experience is composed of state, action, and reward of each decision moment. The training process is running at the central agent, and the first step is to suspend all workers using a standard thread synchronization method called the event. Then the central agent uses all collected data to train the actor and critic network. For actor-network, we use clipped surrogate objective to update, and it is shown in (16) (Schulman et al., 2017).

$$L^{CLIP}(\theta) = \widehat{E}_t\left[min\left(r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}_t\right)\right] \tag{16}$$

In this equation, $r_t(\theta)$ is the ratio of new policy over old policy and can be computed using the new policy gradient network and old policy gradient network. A_t is the advantage of the prediction, which is the difference between reward from the collected data and value of the corresponding state computed by the critic network. For the critic network, the training process is relatively easy. The loss is the above advantage A_t which can show the performance of state value's prediction. Finally, after both actor and critic network finish training, the central agent uses the event to tell all workers to use new network parameters to continue collecting data. The pseudo-code of the training process is shown in Algorithm 3.

Algorithm 3: TrainNetwork()

input: event *mutex*
output: event *mutex*
begin
1:*mutex* $\leftarrow$ 'wait' state
2:$a\_para_{old} \leftarrow a\_para$
3:$t_{max} \leftarrow$ length of r_b
4:value of state $v_s \leftarrow$ critic (s_b)
4:**for** $j \leftarrow 1$ **to** t_{max} **do**
5:$s \leftarrow s_b[j]$, $a \leftarrow a_b[j]$, $r \leftarrow r_b[j]$, $v \leftarrow v_s[j]$
6:advantage $adv \leftarrow r - v$
7:new policy gradient $newp \leftarrow \text{actor}_{\text{new}}(a)$
8:old policy gradient $oldp \leftarrow \text{actor}_{\text{old}}(a)$
9:$ratio \leftarrow newp \,/\, oldp$
10:actor's loss *aloss* is calculated by using (16)
11:Train actor network and update *a_para*
12:critic's loss *closs* $\leftarrow adv$
13:Train critic network and update *c_para*
14:**end**
15:s_b, a_b, $r_b \leftarrow$ empty
16:*mutex* $\leftarrow$ 'set' state
17:**return** *mutex*
end

4.2.5. Network structure

The proposed algorithm uses the actor-critic framework to extract hidden patterns within the environment. This framework is a mature policy gradient method in which the actor inherits the calculation of policy gradient, and the critic adds other learning methods based on value like Q-learning to fasten the training speed and improve effectiveness. The actor-network computes policy gradient at every decision moment while the critic network predicts state value to teach the actor-network to have a better policy gradient. Both networks share the same input, which is the state, but actor-network outputs policy gradients while critic network outputs state values. Therefore, similar neural network structures are introduced to both actor and critic network. The input state is divided into two parts which are score and difference at first. Then the two pieces go through the convolution layer respectively

Table 5
Parameters of HVRP instances.

Property	Value
Space	(0, 0) to (1, 1)
Depot	(0.5, 0.5)
Number of customers	10 to 100
Customer location	(0, 0) to (1, 1)
Customer demand quantity	1 to 10
Number of vehicles	2 to 6
Vehicle capacity	25 to 50

Table 6
Parameters chosen for the proposed RLHH.

Object	Property	Value
RL strategy	Episode decision moment number	1.5**customer number*
	Max episode	60
	Number of workers	6
	Reward discount factor	0.9
	Learning rate	0.0001
	Minimum batch size for updating	40
	Clipping surrogate objective range	0.2
	Network neural numbers	128
Meta-heuristics	Artificial bee colony	$scale = 48$, $generation = 60$, $limit = 59$
	Ant colony optimization	$scale = 48$, $generation = 40$, $dR = 0.9$, $iR = 3$
	Cuckoo search	$scale = 48$, $generation = 75$, $coeff = 1.5$, $step = 1$, $aP = 0.25$
	Genetic algorithm	$scale = 48$, $generation = 75$, $cP = 0.6$, $mP = 0.4$
	Particle swarm optimization	$scale = 48$, $generation = 130$, $c1 = 1.5$, $c2 = 2$, $w = 0.9$
	Simulated annealing	$scale = 48$, $generation = 40$, $Tstart = 0.5$, $Tend = 0.01$, $times = 2$

to discover hidden information among them. After that, the results are merged into the complete data to put all information together. Finally, the actor-network uses the softmax technique to output policy gradients, and the critic network uses fully connected layers to get the state value number. The network structure is shown in Fig. 4.

5. Computational experiments

To investigate the effectiveness and performance of the proposed RLHH algorithm, three groups of experiments are conducted. The first group of experiments is conducted to train the RLHH model. For the lack of standard datasets of HVRP, the second group of experiments takes traditional VRP with the standard benchmark to verify the effectiveness of the proposed algorithm. Finally, randomly generated HVRP instances varying from small-scale to large-scale are tested to further prove the performance of the algorithm. For comparison, some state-of-art *meta*-heuristics and hyper-heuristics reported in the literature and MILP method are reported. All the algorithms are coded with Python 3.5 and TensorFlow 1.4, and implemented on a personal computer with an i7-8700 central processing unit at 3.2 GHz and with 16 GB RAM.

Since the HVRP is a relatively new problem and does not have a standard dataset, randomly generated instances are used in the first and third experiments. Similar to standard VRP instances, in all problems, the routing space is a square area from (0, 0) to (1, 1). In this area, the depot is in the middle, which is located at (0.5, 0.5). Meanwhile, customers' locations are randomly generated in the routing space, and customers' demands are randomly generated as well. Finally, vehicles with different capacities will start at the depot and transfer goods from the depot to all customers. After testing multiple sets of parameters, the problem parameters used are shown in Table 5.

Table 7
Location of customers and depot of the test problem.

Index	Location	Demand
1	(0.873,0.969)	8
2	(0.531,0.233)	6
3	(0.430,0.402)	5
4	(0.478,0.555)	2
5	(0.761,0.712)	3
6	(0.426,0.289)	3
7	(0.334,0.219)	8
8	(0.983,0.128)	7
9	(0.071,0.225)	9
10	(0.896,0.345)	7
11	(0.029,0.352)	3
12	(0.764,0.939)	8
13	(0.432,0.270)	1
14	(0.638,0.069)	7
15	(0.796,0.032)	3
16	(0.790,0.989)	1
17	(0.039,0.446)	9
18	(0.627,0.215)	1
19	(0.536,0.100)	1
20	(0.940,0.943)	2

Table 8
Capacity of the heterogeneous fleet of the test problem.

Index	Capacity
1	25
2	30
3	35
4	40

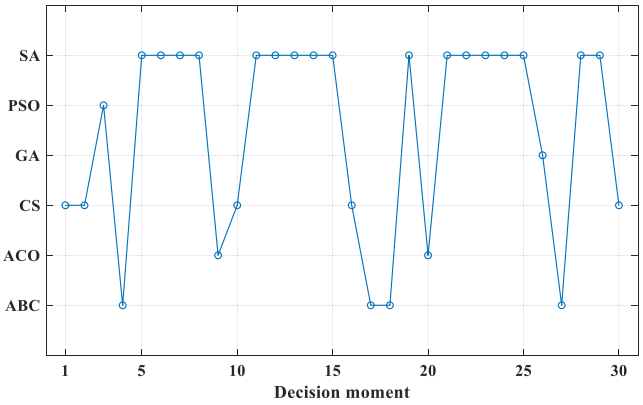


Fig. 5. Algorithm chosen for the test instance.

In the proposed algorithm, many parameters in the algorithm should be determined to achieve the best result. These parameters include two parts, namely high-level RL strategy and low-level *meta*-heuristics. They are chosen according to several previous studies and preliminary tests. Besides, many articles about reinforcement learning or *meta*-heuristics in other fields are considered. The chosen parameters are shown in Table 6.

5.1. Training process of RLHH

The first group of experiments focuses on generating effective hyper-heuristic. Since the high-level strategy of the proposed RLHH is reinforcement learning, the training process is required to maximize effectiveness. During the procedure, parallel workers are introduced to accelerate and randomly generated problems are used to provide training instances.

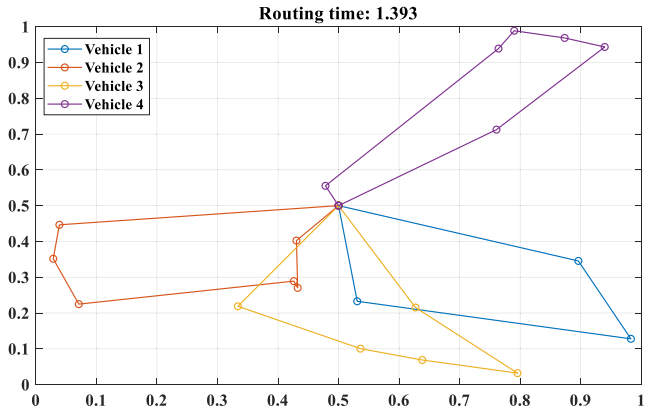


Fig. 6. Result of the test instance.

After the training process, the RL strategy model is saved and tested to check the performance. In the test phase, an instance with 20 customers and 4 vehicles is raised, and the algorithm is used to solve it. The problem is shown in Table 7 and Table 8, and the algorithm result is shown in Fig. 5 and Fig. 6.

From the result, it can be concluded that the proposed algorithm can solve the problem well by using reinforcement learning to dynamically select appropriate *meta*-heuristic algorithms. The routing time of the problem is decreasing during generations of computing, and different algorithms are chosen to increase this decreasing rate as much as possible. Specifically, in the early generation, the algorithm is likely to choose CS due to its strong exploration ability. Then, since the algorithm has found several good areas, SA is most likely to be used to find out local optima, which is good at exploitation. After that, the high-level strategy does not change policy until the local optimal value is found, and the optimal solution is unchanged. At this time, the algorithm will prefer to choose ABC or CS to focus on exploration again, trying to find other good search areas. Meanwhile, SA is used to exploit those areas and the process is repeated until the final solution is outputted. Therefore, the proposed algorithm can achieve excellent results within limited generations. Finally, the result is shown in Fig. 6, and it seems acceptable and proves the effectiveness of the proposed algorithm.

5.2. Experimental results on VRP

To verify the performance of the proposed algorithm, vehicle routing problems with the same vehicles are introduced. This type of problem is relatively traditional, and a standard benchmark can be found in Augerat, Belenguer, Benavent, Corberan, and Naddef (1995). In this part, the algorithm is modified to fit this kind of problem, and the hyper-heuristic (HH) raised in Ahmed et al. (2019) is taken for comparison to examine the superiority of the proposed algorithm over other hyper-heuristic technique. Besides, they are compared with *meta*-heuristics used as low-level heuristics to verify the necessity of using the hyper-heuristic technique. Furthermore, the optimal solution from the benchmark is used to show the overall level of the algorithm. During the experiment, all instances are used to test the performance of all algorithms. For each instance, the computing time is controlled to make sure all algorithms use the same computing power. Therefore, parameter CPU time is the maximum computing time of all algorithms, which is determined by summarizing the effectiveness change during algorithm running and balancing between the effectiveness and time cost. Besides, the instance named '*P-nX-kY*' means the problem with *X* customers and the optimal result has *Y* routes. The result of algorithms is the makespan of the solution. The result is shown in Table 9.

As shown in Table 9, the following observations can be obtained. Generally, these two hyper-heuristics are significantly superior to other *meta*-heuristics on average, which fully demonstrates the necessity and effectiveness of introducing the hyper-heuristic over the *meta*-heuristics. Besides, the proposed RLHH algorithm is performed significantly better than the HH reported in the literature. In particular, the proposed algorithm performs extremely well on the benchmark, and improves or achieves the best solution of other *meta*-heuristics to 21 of 24 instances. CS obtains the best solution on instance' P-n21-k2′ and 'P-n23-k8′ while SA performs best on instance 'P-n51-*k*10′. Even in these instances, the proposed algorithm is not useless, since it is worse than the best solution of less than 1% in the latter two cases. Moreover, the average result of the proposed algorithm on all instances is best among all algorithms under research as well. It is about 6.73% better than the best solution from other *meta*-heuristics. Therefore, we can safely conclude that it is necessary to introduce the hyper-heuristic technique and the proposed algorithm can significantly improve the *meta*-heuristic.

To show the overall performance of these algorithms, the gap be-

Table 9
Result of the experiment for vehicle routing problems.

No.	Instance	CPU time(s)	OPT	RLHH	ABC	ACO	CS	GA	PSO	SA	HH
1	P-n16-k8	30	450	**451.34**	451.95	451.95	**451.34**	**451.34**	457.95	451.95	451.95
2	P-n19-k2	40	212	**212.66**	234.06	242.26	**212.66**	222.39	268.11	234.98	232.25
3	P-n20-k2	40	216	**217.42**	236.68	243.27	229.29	219.13	276.96	249.20	249.36
4	P-n21-k2	40	211	219.98	270.84	227.56	**212.71**	216.09	262.12	300.58	235.83
5	P-n22-k2	40	216	**217.85**	230.77	262.69	234.17	234.39	282.21	221.43	245.98
6	P-n22-k8	40	588	**588.79**	615.99	648.51	590.62	**588.79**	608.71	600.84	595.27
7	P-n23-k8	50	529	537.94	543.18	595.27	**536.34**	546.64	559.21	549.25	538.35
8	P-n40-k5	140	458	**487.07**	566.38	938.87	582.60	490.44	706.48	512.80	508.74
9	P-n45-k5	170	510	**542.35**	639.08	1128.09	635.32	626.07	870.20	633.52	588.69
10	P-n50-k7	200	554	**587.90**	645.05	1082.94	787.23	624.93	877.31	686.06	626.04
11	P-n50-k8	200	631	**684.60**	709.64	1225.31	875.55	687.36	889.80	726.08	736.61
12	P-n50-*k*10	200	696	**722.36**	782.82	1275.86	902.26	796.43	995.54	791.35	737.69
13	P-n51-*k*10	200	741	791.76	801.90	1300.23	971.90	838.99	1080.02	**786.85**	802.05
14	P-n55-k7	250	568	**621.46**	682.58	1305.28	844.30	648.99	979.26	688.76	645.87
15	P-n55-k8	250	588	**623.58**	657.87	1269.91	866.28	739.27	923.25	751.85	649.89
16	P-n55-*k*10	250	694	**756.70**	837.88	1344.84	896.59	766.48	1026.38	923.92	806.35
17	P-n55-k15	250	989	**1007.70**	1017.64	1539.22	1224.33	1133.97	1288.91	1028.08	1016.51
18	P-n60-*k*10	300	744	**794.44**	815.59	1566.32	1037.43	967.71	1082.95	876.63	808.07
19	P-n60-k15	300	968	**1015.71**	1052.41	1611.22	1236.09	1108.49	1380.54	1056.90	1025.88
20	P-n65-*k*10	350	792	**863.68**	903.80	1714.69	1231.50	942.04	1274.04	876.46	890.24
21	P-n70-*k*10	400	827	**909.95**	1024.57	1901.53	1233.17	1026.48	1358.05	995.34	952.07
22	P-n76-k4	460	593	**659.30**	752.08	1697.20	1021.89	777.84	1084.22	771.95	764.20
23	P-n76-k5	460	627	**695.04**	748.68	1616.42	969.12	712.28	1118.67	735.46	753.33
24	P-n101-k4	800	681	**778.79**	847.62	2093.67	1404.82	1016.14	1519.37	929.28	820.99
	Average		587	**624.51**	669.54	1136.80	799.48	682.6	882.10	682.48	653.43

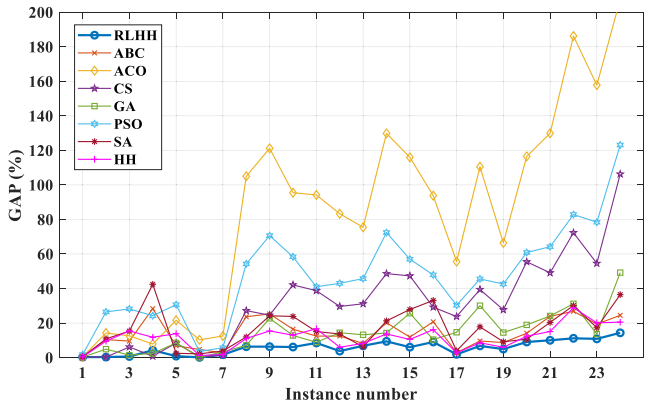


Fig. 7. Results of all algorithms compared with the optimal solution.

tween the heuristics and the optimal solution can be calculated by (17), where X means one of these algorithms, $T(X)$ means the average routing time of X and OPT means the optimal solution.

$$GAP = \frac{T(X) - T(OPT)}{T(OPT)} \tag{17}$$

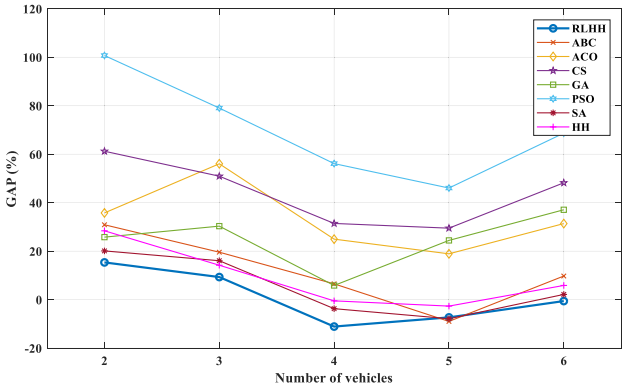
Fig. 7 shows the GAP curve for these VRP instances. For these instances, the gap between the optimal solution and the best solution to the proposed algorithm varies from 0.1% to 14.4% while the average gap is 6.4%. Besides, as the problem scale increases, the gap generally increases as well. From these results, although there are still improvements available, the solution provided by the proposed algorithm is acceptable because the proposed algorithm is specialized to the HVRP and the performance will decrease when solving the traditional VRP. To conclude, the proposed algorithm can achieve an acceptable solution in a limited time and the effectiveness is proved through VRP instances.

5.3. Experimental results on HVRP

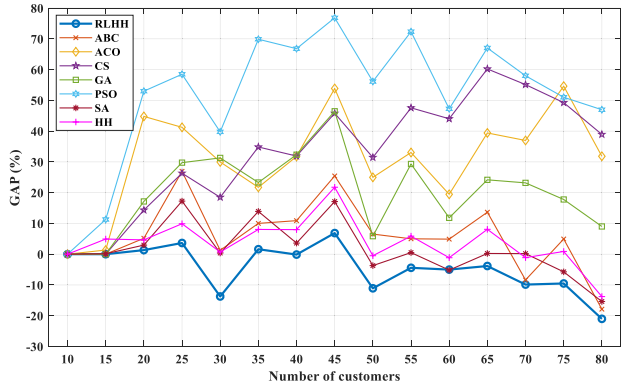
To further investigate the effectiveness of the proposed RLHH for HVRP, a series of HVRP instances with a different number of customers and vehicles are randomly generated. Similarly, *meta*-heuristics used as low-level heuristics are introduced as the control object. Meanwhile, the exact method named MILP using CPLEX is also presented. As shown in Table 10, MILP cannot solve some large-scale problems due to the memory limitation of the computer, and therefore the results of some instances remain empty. In the table, the instance named '*h*-*n*' means the problem with *h* vehicles and *n* customers. All scales of problems from the problem dataset are used to test the performance of all algorithms. Since the HVRP's scale is composed of two parts, the number of customers and vehicles, the experiment uses a control variable method to evaluate the

Table 10
Results of the experiment of heterogeneous vehicle routing problems.

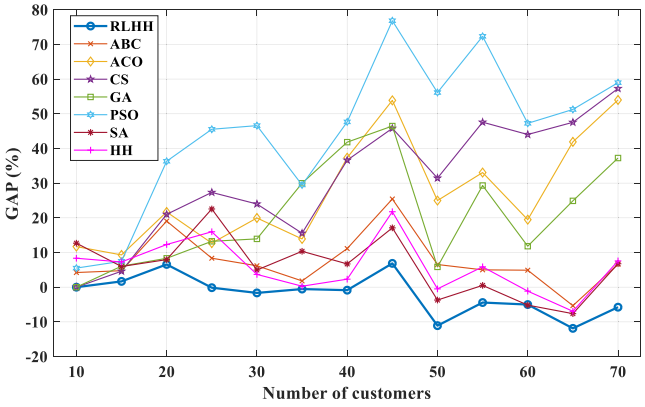
Instance	CPU time(s)	MILP	RLHH	ABC	ACO	CS	GA	PSO	SA	HH
50–2	200	5.178	**5.974**	6.778	7.031	8.349	6.514	10.393	6.220	6.649
50–3	200	3.441	**3.762**	4.115	5.370	5.194	4.485	6.161	3.994	3.930
50–4	200	3.075	**2.734**	3.276	3.843	4.042	3.255	4.801	2.961	3.060
50–5	200	2.682	2.514	2.469	3.225	3.512	3.375	3.961	**2.499**	2.611
50–6	200	2.265	**2.252**	2.486	2.976	3.357	3.105	3.821	2.314	2.398
10–4	10	1.207	**1.207**	**1.207**	**1.207**	**1.207**	**1.207**	**1.207**	**1.207**	**1.207**
15–4	20	1.401	**1.401**	**1.401**	1.420	**1.401**	**1.401**	1.559	1.403	1.470
20–4	40	1.353	**1.371**	1.423	1.959	1.547	1.585	2.070	1.393	1.416
25–4	60	1.455	**1.508**	1.847	2.055	1.838	1.888	2.306	1.706	1.599
30–4	80	2.125	**1.833**	2.152	2.762	2.518	2.790	2.970	2.133	2.140
35–4	100	2.287	**2.324**	2.516	2.784	3.084	2.820	3.885	2.605	2.471
40–4	140	2.467	**2.464**	2.736	3.251	3.254	3.265	4.115	2.556	2.664
45–4	170	2.753	**2.942**	3.453	4.236	4.014	4.032	4.868	3.224	3.352
50–4	200	3.075	**2.734**	3.276	3.843	4.042	3.255	4.801	2.961	3.060
55–4	250	3.458	**3.305**	3.631	4.601	5.103	4.472	5.959	3.476	3.661
60–4	300	3.596	3.416	3.772	4.296	5.179	4.021	5.295	**3.410**	3.555
65–4	350	3.773	**3.628**	4.288	5.260	6.045	4.685	6.302	3.782	4.078
70–4	400	4.214	**3.798**	3.861	5.772	6.537	5.191	6.658	4.222	4.167
75–4	450	4.739	**4.288**	4.973	7.328	7.074	5.582	7.156	4.467	4.781
80–4	500	5.715	**4.514**	4.694	7.534	7.940	6.230	8.399	4.836	4.928
85–4	575	–	**4.312**	4.738	7.089	7.897	6.570	7.175	4.562	4.633
90–4	650	–	**4.915**	5.110	8.030	9.101	7.612	8.742	5.006	4.980
95–4	725	–	**4.979**	5.821	8.780	9.485	8.041	9.913	5.333	5.277
100–4	800	–	**5.243**	6.012	9.219	9.726	7.322	9.306	5.561	5.595
10–2	10	1.726	**1.726**	1.799	1.928	**1.726**	**1.726**	1.821	1.945	1.870
15–2	20	1.913	**1.945**	2.004	2.091	2.001	2.029	2.056	2.027	2.052
20–2	40	2.666	**2.841**	3.174	3.244	3.226	2.888	3.634	2.878	2.994
25–3	60	2.071	**2.068**	2.244	2.334	2.637	2.345	3.014	2.538	2.402
30–3	80	2.606	**2.563**	2.767	3.126	3.231	2.969	3.820	2.735	2.702
35–3	100	3.149	**3.132**	3.206	3.587	3.639	4.093	4.077	3.475	3.158
40–3	140	3.250	**3.222**	3.612	4.463	4.440	4.609	4.799	3.468	3.325
45–4	170	2.753	**2.942**	3.453	4.236	4.014	4.032	4.868	3.224	3.352
50–4	200	3.075	**2.734**	3.276	3.843	4.042	3.255	4.801	2.961	3.060
55–4	250	3.458	**3.305**	3.631	4.601	5.103	4.472	5.959	3.476	3.661
60–4	300	3.596	3.416	3.772	4.296	5.179	4.021	5.295	**3.410**	3.555
65–5	350	3.210	**2.829**	3.039	4.555	4.736	4.009	4.855	2.965	2.987
70–5	400	3.088	**2.910**	3.292	4.755	4.857	4.238	4.909	3.297	3.322
75–5	450	–	**3.053**	3.698	5.489	5.501	4.562	6.343	3.302	3.331
80–5	500	–	**3.132**	3.560	6.401	6.142	4.900	6.722	3.944	3.438
85–6	575	–	**3.241**	3.370	5.688	5.649	4.782	6.124	3.351	3.352
90–6	650	–	**2.914**	3.193	5.738	6.121	4.904	5.713	3.109	3.102
95–6	725	–	**3.490**	3.703	6.090	6.871	5.512	5.725	3.492	3.578
100–6	800	–	**3.671**	3.814	7.030	6.641	5.692	6.474	3.741	3.737
Average			**3.083**	3.410	4.590	4.819	4.134	5.182	3.283	3.315



(a) Instances with the same number of customers

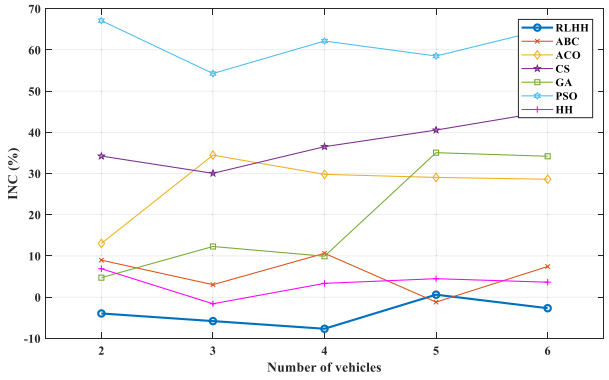


(b) Instances with the same number of vehicles

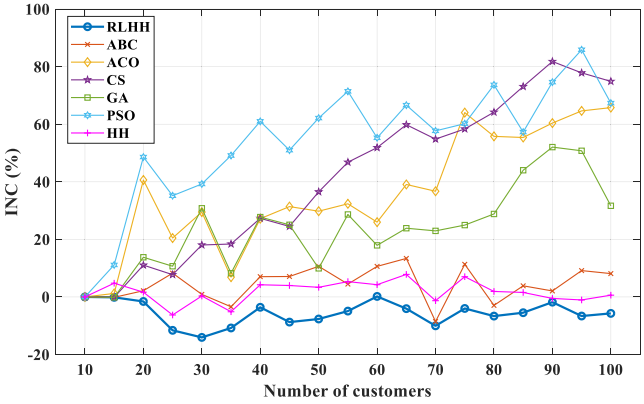


(c) Instances with an increasing number of customers and vehicles

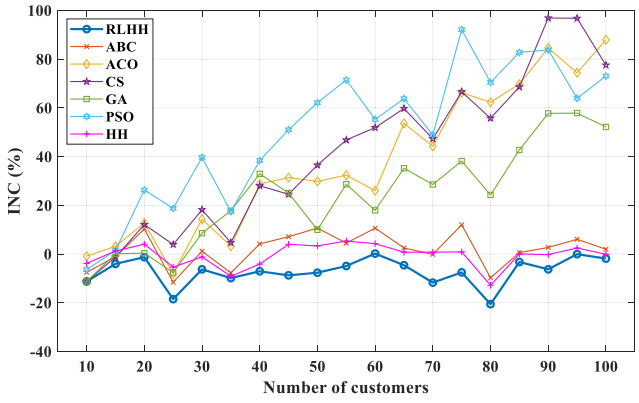
Fig. 8. Result of all algorithms compared with MILP.



(a) Instances with the same number of customers



(b) Instances with the same number of vehicles



(c) Instances with an increasing number of customers and vehicles

Fig. 9. Result of all algorithms compared with SA.

performance. It is divided into three parts which are two single variable experiments and one overall experiment. Firstly, the number of customers is controlled to medium scale to find out results under different numbers of vehicles. Secondly, the number of vehicles is fixed to four, and the customers vary from 10 to 100 to figure out performance comparison under different customers. Finally, an increasing number of vehicles and customers are used to show the overall level of results.

From table 10, it can be seen that the proposed algorithm has better results than other algorithms in 36 of 38 instances after the duplicate cases are removed. Besides, the average result of the proposed algorithm is about 6%∼40% smaller than single *meta*-heuristics. Therefore, the proposed algorithm has better performance than state-of-art *meta*-heuristics. Similarly, to show the overall performance more clearly, the MILP method is introduced to find out the best solution. The gap (*GAP*) between the heuristics and the exact MILP solution can be calculated by (18), and Fig. 8 shows the *GAP* curve for the third part of the experiment.

$$GAP = \frac{T(X) - T(MILP)}{T(MILP)} \tag{18}$$

From Fig. 8, it is evident that all heuristic algorithms are becoming more and more significant compared to MILP when the problem scale increases. Among them, the proposed algorithm has the best performance in these instances. Especially when the size of the problem exceeds 50 customers and 4 vehicles, the results of our system are superior to that of MILP which shows that the overall level of the algorithm acceptable. Furthermore, for three groups of experiments, there are more results worth discussing. As the customer number is fixed, and the vehicle number increases, the *GAP* curve of the proposed algorithm has a downtrend. It means that the performance of the proposed algorithm is becoming better with an increasing number of vehicles. Since there are

more vehicles in practice, it's worth using our algorithm in real problems. Besides, when the number of vehicles is fixed and the number of customers increases, the proposed algorithm's performance is also increasing, and the *GAP* curve is roughly moving downward. Therefore, the proposed algorithm can achieve an even better result for large-scale problems in a limited time and has application value in real-world problems. Finally, for the overall experiment, since the number of vehicles and customers are both increasing, the descent trend of the GAP curve can also be figured out except for some special cases which support the above conclusion.

Besides, to more intuitively demonstrate the effectiveness of the comparison between hyper-heuristic and *meta*-heuristics, the increased percentage (*INC*) of results by these algorithms is calculated by (19), where *X* means one of these algorithms, *T(X)* means the average routing time of *X*. Since the result shows that SA performs best among *meta*-heuristics, it is used as the base algorithm. Besides, given that the lesser routing time means a better solution, the lower *INC* means better algorithm performance. Fig. 9 shows the *INC* of all algorithms other than SA.

$$INC = \frac{T(X) - T(SA)}{T(SA)} \tag{19}$$

From three parts of the experiment in Fig. 9, it is clear that the proposed algorithm has the smallest *INC* value at most times, and even if it is not the best algorithm, its result is only about 2% larger than the smallest *INC* value of all algorithms. Therefore, it can be concluded that the hyper-heuristic algorithm performs best.

From the above experiments, it's easy to conclude that the proposed algorithm can achieve 15% less than the exact MILP solution for small-scale instances and better than it when problems getting larger, which is acceptable in reality and shows the high overall level of the proposed algorithm. Besides, it performs better than any single *meta*-heuristic algorithm in large-scale problems, achieving about 6% improvement in different instances because of the hyper-heuristic framework. Therefore, it can be concluded that the proposed algorithm performs best among all heuristics in this study, and provides high-quality solutions to the HVRP.

6. Conclusion

This study deals with a practical heterogeneous vehicle routing problem to minimize the maximum routing time of vehicles, and establishes a mixed-integer linear programming formulation to provide the exact solution for small-scale instances. However, the exact method will fail in large-scale instances. Given that *meta*-heuristic algorithms may be more promising, this study develops a novel reinforcement learning-based hyper-heuristic, which takes reinforcement learning as the high-level selection strategy and *meta*-heuristic as low-level heuristics. Through comparative experiments, it can be seen that the proposed algorithm can combine the advantage of different *meta*-heuristic algorithms after appropriate training to achieve significant improvements compared with the single *meta*-heuristic algorithm. Besides, when the problem size increases, the proposed algorithm can beat the exact MILP solution at the same time, which further verifies the high level of solutions provided by the proposed algorithm.

Although the research has solved several academically challenging issues, more works can be done to improve the system further to achieve more effectiveness and performance. The first extension is to develop a more efficient network structure and framework in the reinforcement learning strategy. Another extension is to introduce more and better low-level heuristics, including but not limited to meta-heuristics, simple heuristics, and hybrid algorithms. Finally, more complicated scenes that are more similar to reality can be considered in the future, leading the research to overcome more challenges.

CRediT authorship contribution statement

Wei Qin: Conceptualization, Data curation, Formal analysis, Resources, Funding acquisition, Methodology, Writing - original draft. **Zilong Zhuang:** Methodology, Validation, Visualization, Investigation, Software, Writing - review & editing. **Zizhao Huang:** Project administration, Resources, Supervision, Validation, Funding acquisition, Writing - review & editing. **Haozhe Huang:** Investigation, Software, Writing - review & editing.

References

Ahmed, L. N., Mumford, C. L., & Kheiri, A. (2019). Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research, 274*(2), 545–559.

Augerat, P., Belenguer, J., Benavent, E., Corberan, A., Naddef, D., & Rinaldi., G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. IMAG.

Avci, M., & Topaloglu, S. (2016). A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications, 53*, 160–171.

Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming, 120*(2), 347–380.

Baradaran, V., Shafaei, A., & Hosseinian, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering, 131*, 187–199.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering, 99*, 300–313.

Choi, E., & Tcha, D. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research, 34*(7), 2080–2095.

Choong, S. S., Wong, L. P., & Lim, C. P. (2018). Automatic design of hyper-heuristic based on reinforcement learning. *Information Sciences, 436*, 89–107.

Coelho, V. N., Grasas, A., Ramalhinho, H., Coelho, I. M., Souza, M. J., & Cruz, R. C. (2016). An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints. *European Journal of Operational Research, 250*(2), 367–376.

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

Derbel, H., Jarboui, B., & Bhiri, R. (2019). A skewed general variable neighborhood search algorithm with fixed threshold for the heterogeneous fleet vehicle routing problem. *Annals of Operations Research, 272*(1–2), 243–272.

Epitropakis, M. G., & Burke, E. K. (2018). Hyper-heuristics. *Handbook of Heuristics*, 489–545.

Fan, C., Zeng, L., Sun, Y., & Liu, Y. Y. (2020). Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence, 2*, 317–324.

Fu, L. L., Aloulou, M. A., & Triki, C. (2017). Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows. *International Journal of Production Research, 55*(20), 5942–5957.

Ghannadpour, S. F., & Zarrabi, A. (2019). Multi-objective heterogeneous vehicle routing and scheduling problem with energy minimizing. *Swarm and Evolutionary Computation, 44*, 728–747.

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research, 11*(1), 49–66.

Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. *European Journal of Operational Research, 252*(3), 995–1018.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., & Lokketangen, A. (2010). Review: Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research, 37*(12), 2041–2061.

Huang, Y. H., Blazquez, C. A., Huang, S. H., Paredes-Belmar, G., & Latorre-Nuñez, G. (2019). Solving the feeder vehicle routing problem using ant colony optimization. *Computers & Industrial Engineering, 127*, 520–535.

Huang, Z., Huang, W., & Guo, F. (2019). Integrated sustainable planning of self-pickup and door-to-door delivery service with multi-type stations. *Computers & Industrial Engineering, 135*, 412–425.

Hwang, I., & Jang, Y. J. (2020). Q (λ) learning-based dynamic route guidance algorithm for overhead hoist transport systems in semiconductor fabs. *International Journal of Production Research, 58*(4), 1199–1221.

James, J. Q., Yu, W., & Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems, 20*(10), 3806–3817.

Koc, C., Bektas, T., Jabali, O., & Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research, 249*(1), 1–21.

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science, 43*(4), 408–416.

Leggieri, V., & Haouari, M. (2017). Lifted polynomial size formulations for the homogeneous and heterogeneous vehicle routing problems. *European Journal of Operational Research, 263*(3), 755–767.

Li, J., Wang, D., & Zhang, J. (2018). Heterogeneous fixed fleet vehicle routing problem based on fuel and carbon emissions. *Journal of Cleaner Production, 201*, 896–908.

Li, W., Özcan, E., & John, R. (2017). A learning automata-based multiobjective hyper-heuristic. *IEEE Transactions on Evolutionary Computation, 23*(1), 59–73.

Liu, J., Smith, A. E., & Qian, D. (2016). The vehicle loading problem with a heterogeneous transport fleet. *Computers & Industrial Engineering, 97*, 137–145.

Liu, S. (2013). A hybrid population heuristic for the heterogeneous vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review, 54*, 67–78.

Liu, S., Huang, W., & Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review, 45*(3), 434–445.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, 1928–1937.

Mohamed, I. B., Klibi, W., Labarthe, O., Deschamps, J., & Babai, M. Z. (2017). Modelling and solution approaches for the interconnected city logistics. *International Journal of Production Research, 55*(9), 2664–2684.

Mosadegh, H., Ghomi, S. F., & Süer, G. A. (2020). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and Q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research, 282*(2), 530–544.

Olgun, B., Koç, Ç., & Altıparmak, F. (2020). A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering, 107010*.

Penna, P. H., Subramanian, A., Ochi, L. S., Vidal, T., & Prins, C. (2019). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research, 273*(1), 5–74.

Pessoa, A., Sadykov, R., & Uchoa, E. (2018). Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. *European Journal of Operational Research, 270*(2), 530–543.

Pillay, N., & Qu, R. (2018). *Hyper-Heuristics: Theory and Applications*. Springer International Publishing.

Pinto, G. D., Vitorugo, L. R., Rosa, R. D., Arpini, B. P., & Caprini, L. A. (2018). Planning the transport of loads to oil platforms considering the arrangement of the loads on the ship's deck. *Computers & Industrial Engineering, 119*, 289–300.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv: Learning.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature, 550*(7676), 354–359.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO-Operations Research-Recherche Opérationnelle, 33*(1), 1–14.

Tarantilis, C. D., Zachariadis, E. E., & Kiranoudis, C. T. (2008). A guided tabu search for the heterogeneous vehicle routeing problem. *Journal of the Operational Research Society, 59*(12), 1659–1673.

Teymourian, E., Kayvanfar, V., Komaki, G. M., & Zandieh, M. (2016). Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. *Information Sciences, 334*, 354–378.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research, 234*(3), 658–673.

Vincent, F. Y., Redi, A. P., Hidayat, Y. A., & Wibowo, O. J. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing, 53*, 119–132.

Yao, B., Yu, B., Hu, P., Gao, J., & Zhang, M. (2016). An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. *Annals of Operations Research, 242*(2), 303–320.

Yousefikhoshbakht, M., Didehvar, F., & Rahmati, F. (2014). Solving the heterogeneous fixed fleet open vehicle routing problem by a combined metaheuristic algorithm. *International Journal of Production Research, 52*(9), 2565–2575.

Zhang, S., Ren, Z., Li, C., & Xuan, J. (2020). A perturbation adaptive pursuit strategy based hyper-heuristic for multi-objective optimization problems. *Swarm and Evolutionary Computation, 54*, Article 100647.