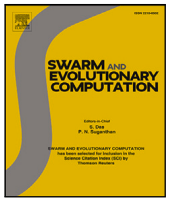
Swarm and Evolutionary Computation 91 (2024) 101719

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



Deep reinforcement learning assisted memetic scheduling of drones for railway catenary deicing

Yu-Jun Zheng [a,*], Xi-Cheng Xie [a], Zhi-Yuan Zhang [a], Jin-Tang Shi [b]

[a] School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China
[b] School of Traffic and Transportation, Beijing Jiaotong University, China

ARTICLE INFO

Keywords:
Drone scheduling
Catenary deicing
Memetic optimization
Evolutionary algorithms
Neighborhood search
Deep reinforcement learning

ABSTRACT

Icy rainfall and snowfall in 2024 Spring Festival struck the high-speed railway catenary systems and caused serious traffic disruptions in central and eastern China. Deicing drones are an effective method in response to these freezing events due to their fast speed and high environmental tolerance. However, the large disaster-affected area and the large scale and complexity of catenary networks make deicing drone scheduling a very difficult problem. In this paper, we formulate two versions of deicing drone scheduling problem, one for single drone scheduling and the other for multiple drone scheduling. Unlike most existing vehicle/drone routing problems, our problem aims to minimize the total negative effect caused by the freezing events on train operations, which reflects the prime concern of the decision-maker and is highly complex. To efficiently solve the problem, we propose a reinforcement learning assisted memetic optimization algorithm, which integrates global mutation and a set of neighborhood search operators adaptively selected by deep reinforcement learning. Computational results on real-world problem instances demonstrate its significant performance advantages over selected popular optimization algorithms in the literature.

1. Introduction

January and February 2024 were the famous Chinese Spring Festival travel rush, during which the railway transportation was subject to heavy traffic. Unfortunately, from January 31 to February 5, a wave of severe blizzards and freezing rain weather swept across 18 provincial-level areas in central and eastern China. Icy rainfall and snowfall froze on wires of the catenary systems over the railway, obstructing power supply of the high-speed trains, forcing the trains to out of action or run at lower speeds, and leaving millions of passengers stranded at railway stations. Fig. 1(a) presents a picture of high-speed trains suspended in Hankou Station due to the icing disaster.

Deicing drones (i.e., unmanned aerial vehicles, UAVs) are an effective method in response to power grid icing [1]. An illustrative example is shown in Fig. 1(b), where a deicer hanged on a fixed-wing drone slides along a wire to break the snow and ice on it. Compared to human workers carrying deicing equipment, deicing drones have many advantages including fast speed, high work efficiency, accessibility to human-inaccessible or dangerous sites, tolerance under harsh environments, and preventing humans from the danger of electric shock [2].

As a matter of course, the aim of deicing is to restore the catenary network operations as soon as possible and, consequently, minimize the negative effect on the train operations. However, the affected area is large and the deicing workload is high. Moreover, the catenary network structure and its cascading effects on the train operations are complex: when we complete the deicing work on a catenary line, the corresponding railway section can be opened; however, given a train passing through the railway section, if any preceding railway section is not opened, the work does not take effect on the restoration of the train operation. Given the large number of lines (i.e., edges/segments) of the catenary network to be deiced, the problem of deicing drone scheduling is very difficult, especially when there are multiple drones to be scheduled.

To address the above issues, in this study, we formulate two versions of deicing drone scheduling problem, one for single drone scheduling and the other for multiple drone scheduling. The objective of the problem is to minimize the total negative effect caused by the freezing events on train operations, which is evaluated based on the delay or cancellation of trains under the scheduling solution. We propose a memetic optimization algorithm, which integrates global mutation search and a set of neighborhood search operators selected by deep reinforcement learning based on historical search knowledge to efficiently solve the problem. We conduct extensive computational tests on real-world problem instances, the results of which demonstrate the

* Corresponding author.
E-mail address: yujun.zheng@computer.org (Y.-J. Zheng).

https://doi.org/10.1016/j.swevo.2024.101719
Received 15 June 2024; Received in revised form 17 August 2024; Accepted 26 August 2024
Available online 29 August 2024

(a) (b)

Fig. 1. (a) High-speed trains suspended due to the icing disaster. (b) A fixed-wing drone equipped with a deicer (from Hunan Daily).

performance advantages of our method over a number of selected popular optimization algorithms in the literature. The main contributions of this paper can be summarized as follows:

- We formulate a practical problem of drone scheduling for railway catenary deicing to minimize the total negative effect on train operations. The objective function of the problem is significantly more complex that those of most existing vehicle routing/scheduling problems, as it not only calculates drone routes and deicing times, but also evaluates the cascading effects on train timetables, which is the prime concern of the decision-maker.
- We propose a memetic algorithm integrating global mutation search and neighborhood search to efficiently solve the problem, where the neighborhood search is controlled by deep reinforcement learning that predicts conditional selection probabilities based on the states considering both solution fitness and diversity.
- We validate the performance of the proposed algorithm through extensive computational tests on real-world instances.

The remainder of this paper is organized as follows. Section 2 discusses related work, Section 3 presents the deicing drone scheduling problem, Section 4 proposes the memetic optimization algorithm, Section 5 presents the computational results, and Section 6 concludes with a discussion.

2. Related work

2.1. Deicing scheduling

Deicing (written as de-icing in some literature) is one of the most important techniques for restoring power networks stricken by icing disasters. Current ice-melting and mechanical deicing are two most widely used techniques. Current ice melting uses the thermal effect of overcurrent/short-current on the wire to melt the ice. Although having the advantages of high efficiency, current ice melting is power consuming and, more importantly, requires to cut the power transmission during the deicing work [3]. Mechanical deicing, by contrast, uses deicing devices attached to the conductors or wires to induce sustained vibrations to shed ice, which is easier to use but more time-consuming than current ice melting [4] when deicers are carried by human or ground vehicles. The emerging drone technologies significantly improve the applicability and efficiency of mechanical deicing by using drones to carry deicers in high speed and to sites that are difficult to access by human or ground vehicles.

For current ice melting scheduling, Huneault et al. [5] proposed a dynamic programming method to determine deicing strategies by minimizing ice buildup on power lines during ice storms over the set of scenarios and over the time horizon spanning the anticipated duration of the ice storm. Hou et al. [6] presented a multi-objective deicing outage scheduling with the aim to minimize the ice thickness peak value of icing lines and the expected energy not supplied of the system; they used the non-dominated sorting genetic algorithm (NSGA-II) to find the Pareto optimal solutions, from which the final deicing schedule is chosen using a decision making method.

Nevertheless, few research works have been devoted to mechanical deicing scheduling considering the use of drones, although there are some studies on scheduling drones for power network failure detection restoration. Deng et al. [7] proposed a multi-platform drone system, in which different types of drones perform different functionalities including long-distance and short-distance imaging and communication relay in power line inspection. Considering the use of industrial Internet of drones for power line inspection, Zhou et al. [8] formulated the energy consumption minimization as a joint optimization problem involving trajectory scheduling, velocity control, relay selection, and power allocation; they transformed the problem into a two-stage suboptimal problem and solved it by combining dynamic programming, auction theory, and matching theory. Zheng et al. [2] studied a cooperative human-drone scheduling problem, where drones are used to inspect faults, which are subsequently repaired by human technicians; the authors proposed a cooperative evolutionary algorithm that simultaneously evolves drone scheduling sub-solutions and human-team scheduling sub-solutions to obtain an optimal or near-optimal integrated solution. Atat et al. [9] proposed an optimization framework for post-disaster drone-based damage assessment of overhead power lines, which optimizes the drone flight paths to inspect the critical loads in an efficient order to minimize the total inspection time while considering drone battery recharging.

2.2. Vehicle and drone routing

Scheduling drones to perform deicing work on icing lines, from an abstract perspective, can be regarded as a special version of vehicle routing problem (VRP) by regarding drones as vehicles and icing lines as customers. The special characteristics include that each customer (line) has a service (deicing) time, the battery capacity of drones should be explicitly considered, and the objective value should be evaluated based on the train delay and cancellation under the deicing solution rather than simply based on makespan. Dorling et al. [10] presented two multi-trip VRPs for drone delivery integrating an energy consumption model, one minimizing costs and the other minimizing the overall delivery time, which were solved by a simulated annealing heuristic for finding suboptimal solutions. Hong et al. [11] developed a method for optimizing delivery routes of drones together with a network of recharging station locations by integrating planar-space routing, range-restricted flow-refueling location, and maximal coverage location. Zheng et al. [12] proposed an evolutionary algorithm integrating comprehensive learning, variable mutation, and local search to solve a problem of collaborative human-drone search for escaped criminals, the aim of which is to minimize the expected time of capture rather than detection. Pachayappan and Sudhakar [13] presented a problem of planning optimal pickup and delivery routes of a drone,

which is synchronized with a docking station for recharging during the services; they proposed a heuristic solution method for the problem. Gómez-Lagos et al. [14] studied a pickup-to-delivery drone routing problem for finding a drone scheduling such that a drone serves the customer's order from a set of available facilities, with the objective to minimize the makespan associated with the drone fleet; they designed a greedy randomized adaptive search procedure to find near-optimal solutions. Wen and Wu [15] proposed a two-echelon heterogeneous multi-drone routing problem for parcel delivery, which was solved by a three-stage iterative optimization algorithm based on the divide and conquer. Guo et al. [16] formulated a routing model of electric trackless rubber-tyred vehicles to minimize the total energy consumption under the constraint of vehicle avoidance, allowable load, and endurance power; they proposed an improved artificial bee colony algorithm integrating adaptive neighborhood search to solve the problem. Considering coverage path planning of heterogeneous drones to visit and search multiple separated regions, Chen et al. [17] presented an ant colony system based algorithm to seek approximately optimal solutions and minimize the time consumption of tasks. Wang et al. [18] presented a problem of delivery route planning for a fleet of drones with different capacities, speeds, and maximum flight ranges; they proposed a modified, rescheduling-based genetic algorithm to search optimal or near-optimal solutions. To coordinate ground vehicles and drones to monitor urban road networks, Xu et al. [19] proposed a hybrid variable neighborhood descent search and simulated annealing algorithm for vehicle-drone arc routing. In [20], Zheng et al. studied a problem of scheduling multiple vehicles to replace and recycle batteries to keep a high operational efficiency of the shared e-bicycle system, which was efficiently solved by an evolutionary algorithm based on variable local-search-based mutation. To solve multiobjective VRPs, Fan et al. [21] proposed a conditional neural heuristic trained by a reinforcement algorithm that leverages stochastic preferences to further enhance the training performance. The problems studied by the above work have some characteristics similar to the problem considered in this paper, but none of them shares most key characteristics with our problem.

2.3. Reinforcement learning assisted memetic algorithms

Reinforcement learning is a model-free, self-learning method that obtains feedback information from the environment to select the optimal action through trial and error. Recently, researchers introduce reinforcement learning into memetic algorithms for proper strategy selection. To solve a real-time multirobot path-planning problem, Rakshit [22] proposed a memetic algorithm consisting of differential evolution (DE) for global search and Q-learning for local refinement by adaptively selecting memes from a meme pool of the scale factor values for individual solutions, whose rewards/penalties depend on the improvement/deterioration in fitness measures. Ozcan et al. [23] proposed a steady-state memetic algorithm, which maintains a utility score for each meme option within each individual in a population and updates the score based on reinforcement Learning. Peng et al. [24] presented a memetic algorithm for a green vehicle routing problem, where an adaptive procedure was utilized to manage multiple neighborhood moves based on a reward and punishment mechanism inspired by reinforcement learning. For distributed assembly flow shop scheduling, Cai et al. [25] proposed a shuffled frog-learning algorithm, which embeds Q-learning composed of four actions based on the combination of global search, neighborhood search and solution acceptance rule to dynamically select search strategies for memeplex search. The work of Zhao et al. [26] presented a knowledge-driven cooperative scatter search method for distributed blocking flow shop scheduling, where the neighborhood perturbation operator and Q-learning are combined to select the appropriate perturbation operator during the search. In [27], the authors proposed a cooperative metaheuristic algorithm based on Q-learning for energy-efficient distributed no-wait flow shop scheduling, where Q-learning is utilized in two neighborhood search mechanisms to determine the most appropriate search operator to generate neighborhood solutions for two cooperative sub-populations. In [28], Zhao et al. equipped a hyperheuristic with Q-learning to solve an energy-efficient distributed blocking flow shop scheduling problem, where Q-learning is employed to select an appropriate low-level heuristic from a pre-designed heuristic set according to their historical performance. For flexible job shop scheduling with finite transportation resources, Pan et al. [29] studied a learning-based multi-population evolutionary algorithm, where reinforcement learning is utilized for mating selection to realize the cooperation of different subpopulations. In [30], the authors studied an extended multi-objective green flexible job-shop scheduling problem with fuzzy processing time, which is solved by a learning-based reference vector memetic algorithm using reinforcement learning-based parameter selection strategy to improve the diversity of nondominated solutions. Considering a similar job shop scheduling problem with dual resource constraints, Chang et al. [31] presented a multi-objective memetic algorithm, which incorporates Q-learning to perform variable neighborhood search and further strengthen the exploitation capability. Recently, Zhao et al. [32] proposed an iterative greedy algorithm with variable neighborhood descents, which employs Q-learning to adapt the weighting coefficients of neighborhood structures.

Classical Q-learning algorithm requires describing the state of solution distribution and the transition between states makes the Q-table design challenging. Facing large solution spaces, deep reinforcement learning adopts deep neural networks to better fit the state space distribution. Zhang et al. [33] proposed a deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling, where each solution from the elite archive is fed into the network to output the conditional probabilities of the actions for performing local search. In [34], Hu et al. presented a co-evolutionary DE to solve constrained optimization problems, which uses reinforcement learning to evaluate the population state so as to select suitable parent populations and corresponding archives for mutation. Qu et al. [35] proposed a heuristic method for satellite range scheduling, which integrates deep reinforcement learning to update the gradient information based on the reward obtained by the scheduling method. For drone routing in the presence of multiple charging stations, Fan et al. [36] presented a deep reinforcement learning method, where a multi-head heterogeneous attention mechanism is designed to facilitate learning a policy for constructing the drone route. The deep reinforcement learning method proposed by Wang et al. [37] to solve traditional and improved VRP with backhauls trains an encoder–decoder structured policy network to sequentially construct vehicle routes, where a two-stage attention-based encoder is designed to yield more informative representations of the nodes so as to deliver high-quality solutions. Recently, Xiao et al. [38] proposed a double-level deep reinforcement learning approach, which uses an encoder–decoder policy network in the upper level to allocate tasks to UAVs and exploits another attention-based policy network in the lower level to construct UAV routes. Motivated by its power in discerning and utilizing the correlation between solutions and algorithm actions, we employ deep reinforcement learning for operator selection in our algorithm.

3. Problem formulation

The problem is schedule one or multiple deicing drones to remove ices on wires of a railway catenary network. A catenary network hit by ices can be represented by a graph $G = \langle V, E \rangle$, where V is the set of vertices and E is the set of edges. Here, V contains not only joints connecting power grid lines, but also points dividing lines into segments such that different segments in a line have different freezing conditions. Typically, two segments can be separated according to obvious differences in altitude, temperature, humidity, wind force, etc. The set E can be divided into two subsets: $E^{\dagger}$ where the edges (segments) need to be deiced (otherwise the corresponding rail segments are impassable), and

$E \backslash E^{\dagger}$ where the edges (such as segments in stations and tunnels) do not need to be deiced. For each $e \in E$, $A(e)$ denotes the amount of deicing work on the segment e ($A(e) = 0$ if $e \notin E^{\dagger}$). For convenience, we also use $V^{\dagger}$ to denote the set of vertices of the edges in $E^{\dagger}$. For each pair of vertices $u_i, u_j \in V$, the distance of the shortest drone path from u_i to u_j is known as $d(u_i, u_j)$. In the following two subsections, we present the two versions of the problem of deicing drone scheduling for removing the ices on the catenary network G, one using a single drone and the other using multiple drones.

3.1. Single-drone scheduling for catenary deicing

First we consider scheduling a single drone for catenary deicing. A scheduling solution can be represented by a vector $X = \{x_1, x_2, \dots, x_n\}$, where $n = |E^{\dagger}|$ is the number of segments that need to be deiced, each component x_i is a tuple (x_i^u, x_i^{δ}), where $x_i^u \in V^{\dagger}$ is the starting vertex of the ith segment to be deiced, and $x_i^{\delta} \in \{0, 1\}$ is the direction (left or right) towards which the drone will deice from x_i^u ($1 \le i \le n$). From each component $x_i = (x_i^u, x_i^{\delta})$, we can obtain the corresponding segment $e(x_i)$ to be deiced. Let u_0 be the initial location of the drone, B_0 be the full battery power of the drone, v be the drone velocity, a be the amount of deicing work the drone can complete per unit time, b be the amount of battery power per unit mileage consumed by the drone in flying, and $b^{\dagger}$ be the amount of battery power per unit work consumed by the drone in deicing. It is assumed that the full battery power is sufficient for the drone to fly to any segment and complete the deicing work on it. The time at which the deicing work on the first segment $e(x_1)$ is completed can be computed as

$$t_c(x_1) = \frac{d(u_0, x_1^u)}{v} + \frac{A\big(e(x_1)\big)}{a} \tag{1}$$

And the remaining battery power of the drone after the completion of the work on $e(x_1)$ is

$$B(x_1) = B_0 - b \cdot d(u_0, x_1^u) - b^{\dagger} \cdot A\big(e(x_1)\big) \tag{2}$$

After the completion of the work on each segment $e(x_i)$, the drone is located at the other endpoint of $e(x_i)$, which is denoted by $u(x_i)$ here. We also use $D(u(x_i))$ to denote the battery depot closest to $u(x_i)$, and the power required for the drone to fly from $u(x_i)$ to $D(u(x_i))$ is

$$\underline{B}(x_i) = b \cdot d\big(u(x_i), D(u(x_i))\big) \tag{3}$$

While the power required by the drone to directly fly to and complete the work on the next segment is

$$\widetilde{B}(x_{i+1}) = b \cdot d\big(u(x_i), x_{i+1}^u\big) + b^{\dagger} \cdot A\big(e(x_{i+1})\big) \tag{4}$$

At each $u(x_i)$, the drone can directly go to the next segment if the remaining battery power is not below the safe level $\widetilde{B}(x_{i+1}) + \underline{B}(x_{i+1})$, otherwise it should go to $D(u(x_i))$ to replace the battery and then continue the remaining tasks. Therefore, the time at which the deicing task on each segment is completed can be iteratively computed as follows ($1 < i \le n$):

$$t_c(x_{i+1}) = \begin{cases} t_c(x_i) + \dfrac{d\big(u(x_i), x_{i+1}^u\big)}{v} + \dfrac{A(e(x_{i+1}))}{a}, & B(x_i) \ge \widetilde{B}(x_{i+1}) + \underline{B}(x_{i+1}) \\ t_c(x_i) + \dfrac{d\big(u(x_i), D(u(x_i))\big) + d\big(D(u(x_i)), x_{i+1}^u\big)}{v} & \\ \quad + t^{\mathrm{R}} + \dfrac{A(e(x_{i+1}))}{a}, & \text{otherwise} \end{cases} \tag{5}$$

where t^{R} denotes the time duration for replacing the battery, including the time for landing and takeoff (we assume that the number of fully-charged batteries is sufficient at each depot).

And the remaining battery power $B(x_i)$ of the drone after the completion of the work on $e(x_i)$ can be iteratively computed as follows ($1 < i \le n$):

$$B(x_{i+1}) = \begin{cases} B(x_i) - b \cdot d\big(u(x_i), x_{i+1}^u\big) - b^{\dagger} \cdot A(e(x_{i+1})), & B(x_i) \ge \widetilde{B}(x_{i+1}) + \underline{B}(x_{i+1}) \\ B_0 - b \cdot d\big(D(u(x_i)), x_{i+1}^u\big) - b^{\dagger} \cdot A(e(x_{i+1})), & \text{otherwise} \end{cases} \tag{6}$$

Based on the completion time of deicing work on each segment, we can obtain the set of segments that have been deiced at time t, denoted by $E^{\ddagger}(t)$, as follows:

$$\begin{aligned} E^{\ddagger}\big(t_c(x_1)\big) &= \big\{e(x_1)\big\} \\ E^{\ddagger}\big(t_c(x_2)\big) &= E^{\ddagger}\big(t_c(x_1)\big) \cup \big\{e(x_2)\big\} \\ &\vdots \\ E^{\ddagger}\big(t_c(x_{i+1})\big) &= E^{\ddagger}\big(t_c(x_i)\big) \cup \big\{e(x_{i+1})\big\}, \quad 1 < i \le n \end{aligned} \tag{7}$$

Then, for each block section s of the railway network affected by ices, let $\Phi(s)$ be the set of power grid line segments along the section s, that is, the section could not be opened unless all segments in $\Phi(s)$ have been deiced; the earliest open time of s under the scheduling solution X can be calculated as

$$t_o(s, X) = \min_{1 \le i \le n} \Big\{ t_c(x_i) \big| \Phi(s) \subseteq E^{\ddagger}\big(t_c(x_i)\big) \Big\} \tag{8}$$

According to the train timetable, we have the set $\Gamma(s)$ of trains that will enter the railway section s, the route of each train r represented as a sequence of consecutive railway sections $\{s_1^r, s_2^r, \dots, s_{K_r}^r\}$, and the time $t_e(r, s_k^r)$ at which the train r plans to enter the section s ($\forall r \in \Gamma(s), k = 1, 2, \dots, K_r$), where K_r is the number of sections in the route of train r. Let $\Delta t(s)$ denote the time duration for the train to pass through section s; based on the timetable and deicing schedule, we can obtain the actual time $t_e^{\ddagger}(r, s, X)$ at which the train r can enter each section s in its route as

$$t_e^{\ddagger}(r, s_k^r, X) = \begin{cases} \max\big(t_e(r, s_k^r), t_o(s_k^r, X)\big), & k = 1 \\ \max\big(t_e(r, s_k^r), t_o(s_k^r, X), t_e^{\ddagger}(s_{k-1}^r, X) + \Delta t(s_{k-1}^r)\big), & k > 1 \end{cases} \tag{9}$$

Remark 1. Eq. (9) is a simplified equation that does not involve the contradictions among the trains. In case that multiple trains need to enter one section in a short period, we first determine the priority of each train according to the grade of the train, the time duration that the train has already delayed, and the cascading effect of the further delay on other trains; then we arrange the trains to enter the section in the order of priority, while keeping a minimum safe distance between any two consecutive trains (see [39] for more details).

We use the following function to evaluate the negative effect of the freezing events on each train r for traveling each railway section s in its route under the scheduling solution X:

$$\psi(r, s, X) = \begin{cases} 0, & t_e^{\ddagger}(r, s, X) \le t_e(r, s) \\ w_{r,s}\big(t_e^{\ddagger}(r, s, X) - t_e(r, s)\big), & t_e(r, s) < t_e^{\ddagger}(r, s, X) < T_{\mathrm{close}} \\ \rho_{r,s} w_{r,s}, & t_e^{\ddagger}(r, s, X) \ge T_{\mathrm{close}} \end{cases} \tag{10}$$

where $w_{r,s}$ is the importance weight of the train r in the railway section s, which is determined according to the grade of the train, the number of the passengers on the train, and the number and importance of the stations in the section, T_{close} is the closing time of the current operational duration (which is typically 24:00:00 of the current day, but can be prolonged by the decision-maker under special conditions), and $\rho_{r,s}$ is the penalty coefficient regarding the cancellation of the train r on the section s.

Remark 2. If a train is canceled on a section s, it is automatically canceled on the subsequent sections along its route, but it can still complete its operations on sections before s.

The aim of the problem is to minimize the total negative effect evaluated in terms of the delay or cancellation of trains caused by the freezing catenary network, which can be represented as follows:

$$\min f(X) = \sum_{s\in S}\sum_{r\in\Gamma(s)} \psi(r,s,X) \tag{11}$$

$$\text{s.t.} \quad \left(\bigcup_{1\le i\le n}\{e(x_i)\}\right) = E^{\dagger} \tag{12}$$

$$e(x_i)\bigcap e(x_{i'}) = \emptyset, \quad \forall 1\le i<i'\le n \tag{13}$$

Eqs. (1)–(10)

where S is the set of railway sections affected by freezing segments, and the constraints (12) and (13) indicate that each segment in $E^{\dagger}$ should be deiced once and only once.

3.2. Multi-drone scheduling for catenary deicing

Given m drones, a scheduling solution $X = \{x_{1,1},\dots,x_{1,n_1},x_{2,1},\dots,x_{2,n_2},\ \dots,x_{m,1},\dots,x_{m,n_m}\}$ consists of m parts, where n_j is the number of segments assigned to the jth drone, satisfying $\left(\sum_{j=1}^{m} n_j\right) = n$. For each jth part $\{x_{j,1},\dots,x_{j,n_j}\}$, we can use the procedure similar to Eqs. (1)–(6) to iteratively calculate completion time of the deicing work on each segment $e(x_{j,i})$, where $1\le i\le n_j$. Then, by combining the results of the m drone sub-schedules, we can calculate the open time for each freezing segment and then evaluate the objective function for minimizing the effect of delay or cancellation of trains using the procedure shown in Algorithm 1.

Algorithm 1: Procedure of calculating the objective function value of any scheduling solution X based on the m drone sub-schedules.

```
Sort all completion times t_c(x_{j,i}) in non-decreasing order;
foreach completion time t_c(x_{j,i}) in the order do
    Calculate E^‡(t_c(x_{j,i})) as the union of the set E^‡ at the previous
      time and {e(x_{j,i})};
    foreach railway section s ∈ S do
        if Φ(s) ⊆ E^‡(t_c(x_{j,i})) then
            Set t_o(s, X) = t_c(x_{j,i});
            Remove s from S;
Sort all trains in the timetable in non-increasing order of their
  priorities;
foreach train r in the timetable do
    Set t_e^‡(r, s_1, X) = t_o(s_1, X) for the first segment s_1 in the route of r;
    foreach subsequent railway section s_k in the route of r do
        Set t_e^‡(r, s_k, X) = max(t_e(r, s_k), t_o(s_k, X), t_e^‡(s_{k-1}, X)+Δt(s_{k-1}));
        if t_e^‡(r, s_k, X) = t_o(s_k, X) and there is already at least one train r'
          prior to r enters s_k at time t_o(s_k, X) then
            set t_e^‡(r, s_k, X) to the time at which r' leaves the block
              section s_k;
Calculate the objective function value f(X) according to Eqs. (10)
  and (11).
```

4. Memetic optimization to solve the problem

The above single/multiple drone scheduling problems are essentially single/multiple permutation optimization problems, which are known as NP-hard. In addition, the objective function of minimizing the effect of train delay or cancellation caused by the freezing catenary network is expensive. Traditional exact optimization algorithms cannot solve large or even medium problem instances within a reasonable time. Population-based evolutionary algorithms are capable of obtaining satisfactory solutions within an acceptable time, and memetic algorithms can further improve solution accuracies by integrating with adaptive local search [40]. Moreover, because catenary network deicing requires fast response, we introduce learning-based optimization to utilize historical problem-solving knowledge to further improve the problem-solving performance [41,42]. To this end, we propose a memetic optimization algorithm integrating global search and variable neighborhood search to solve the problem, where reinforcement learning is introduced for selecting the most appropriate operator for each neighborhood search operation. In the following two subsections, we describe the algorithm versions for single drone scheduling and multiple drone scheduling, respectively.

4.1. Memetic optimization for single drone scheduling

For single drone scheduling, each initial solution is first generated as a random permutation of the set $E^{\dagger}$ of freezing segments to be deiced. Then, each solution has a probability of p_I (a control parameter) of being tentatively improved using the following procedure:

1. For the first segment e_1 in the permutation, let x_1^u be the endpoint closer to the initial location u_0 and let x_1^{δ} towards the other endpoint of e_1.
2. For $i = 2$ to n do:
   (a) Calculate the remaining battery power of the drone at the other endpoint $u(x_{i-1})$;
   (b) If the remaining battery power is sufficient for the drone to complete the work on e_i, let x_i^u be the endpoint closer to $u(x_{i-1})$; otherwise, let x_i^u be the endpoint closer to the battery depot $D\left(u(x_{i-1})\right)$;
   (c) let x_i^{δ} towards the other endpoint of e_i.

Note that the above tentative improvement procedure does not guarantee improving the solution fitness, because making x_i^u closer to $u(x_{i-1})$ may cause $u(x_i)$ farther from the next (i+1)th segment or farther from the next depot. Nevertheless, it have a large probability of success: according to our test, approximately 80% random solutions can be improved by this procedure. Consequently, we set the initial value of p_I to 0.8. If the procedure successfully improves the solution, the updated solution replaces the initial solution; otherwise, the initial solution is kept in the population.

The global search is based on the water wave optimization metaheuristic [43] that assigns each solution a wavelength proportional to the objective function value, i.e., inversely proportional to the fitness of the solution, and mutates the solution with a degree proportional to the wavelength, such that low-fitness solutions tend to explore in large spaces while high-fitness solutions facilitate exploitation in small regions to achieve a good dynamic balance between diversification and intensification. The wavelength of each solution X is initialize to 0.5 and then updated at each generation as

$$\lambda(X) = \lambda(X)\cdot\beta^{-(f_{\max}-f(X)+\epsilon)/(f_{\max}-f_{\min}+\epsilon)} \tag{14}$$

where $f_{\max}$ and $f_{\min}$ are the maximum and minimum objective function values among the population, respectively, β is the wavelength reduction coefficient which is suggested to be 1.0026, and ϵ is a very small number to avoid division by zero.

The mutation on a solution X is performed as follows: for $i = 1$ to n, with a probability of $\lambda(X)$, reverse a random subsequence of X; afterwards, with a probability of p_I, use the tentative improvement procedure described above to adjust the start point and direction of each component. The value of p_I decreases with generation number g as follows (where $g_{\max}$ is the maximum number of generations of the algorithm):

$$p_I(g) = \frac{g_{\max}-0.5g}{g_{\max}}p_I(0) \tag{15}$$

For a solution X whose fitness is *not* improved from the previous generation or whose fitness is *not* in the first half of the population, the mutation operator is deterministically applied. Otherwise, the solution X is newly updated and relatively good, for which we select the

mutation operator or one of the following three neighborhood search operators:

NS1. Continuously exchange two randomly selected components of X until the fitness is improved or the number of exchange reaches an upper limit $n^2/4$.
NS2. Continuously select a randomly component and reinsert it into another random position of X until the fitness is improved or the number of reinsertion reaches an upper limit $n^2/4$.
NS3. Continuously reverse a randomly selected subsequence of X until the fitness is improved or the number of reversion reaches an upper limit n.

Remark 3. NS3 differs from the global search in that, for neighborhood search the solution is evaluated after each reversion, while for global search the solution is evaluated once after all reversions.

The selection among the four operators is based on deep reinforcement learning, that is, we utilize a deep Q-network (DQN) that combines deep neural network and Q-learning [44] to estimate the Q-value $Q(S_{t+1}, A_i)$ of each action (operator) A_i $(1 \le i \le 4)$ as the conditional selection probability in each state S. Here, a state of each solution X is defined by a tuple of seven variables $\{f/f_{\text{best}}, \Delta f/f_{\text{best}}, rank, d_{\text{closest}}, d_{\text{avg}}, d_{\text{best}}, id_{\text{op}}\}$, where f is the objective function value of the solution, f_{best} is the best objective function value found so far, Δf is the decrement of the objective function value of X from the previous generation, $rank$ is the rank of X in the population, d_{closest} is the distance between X and its closest solution, d_{avg} is the average distance between X and the other solutions in the population, d_{best} is the distance between X and the best known solution, and id_{op} is the index of the operator last applied to X. Taking the state variables as inputs, the DQN uses multiple hidden layers (two layers in this study) with the ReLU activation function to perform forward calculation and finally outputs the conditional probabilities based on softmax activation.

Then, we adopt the ϵ-greedy strategy to determine the operator to be applied to X, i.e., with probability $1-\epsilon$ we select the operator with the maximum conditional probability, and with probability ϵ we randomly select one of the four operators.

After applying the selected operator A to X, it is transferred to a new state, and the reward r is calculated as $\Delta f/f_{\text{best}}$ is $f(X)$ is decreased (i.e., the fitness is improved) and is 0 otherwise.

The DQN parameters θ is trained using the Adam optimizer [45] to minimize the loss function:

$$L_\theta = \mathbb{E}_\theta\left(r + \gamma \max_i Q(S_{t+1}, A_i) - Q(S_t, A)\right)^2 \tag{16}$$

where γ is the discount factor.

If a solution X has not been improved for consecutive K_G generations (where K_G is a control parameter typically set to 6), it will be replaced by a new solution generated as follows:

1. Randomly select an exemplar solution X^* from the first half of the population.
2. For $i = 1$ to n do: with a probability of $\lambda(X)$, reverse a random subsequence of X^*.
3. With a probability of p_I, use the tentative improvement procedure to adjust the start point and direction of each component.

That is, the new solution is generated by mutating the exemplar solution X^*, but the mutation intensity is proportional to $\lambda(X)$ rather of $\lambda(X^*)$.

Algorithm 2 presents the pseudo-code of the memetic optimization algorithm for single drone scheduling. In global mutation (Lines 10–13), the operation of n subsequence reversion is significantly more complex than tune the start point and direction tuning, and therefore the time complexity of global mutation is $O(n^2/2) + O(f)$, where $O(f)$ denotes the complexity of the objective function (11), which is also significantly higher than $O(n^2)$. The time complexities of NS1, NS2, and NS3 are $O(n^2/4)$, $O(n^2/4)$, and $O(n^2/2)$, respectively, and therefore the time complexity of neighborhood search is $O(n^2/2) + O(f)$. Let N_P be the population size, the complexity of each generation is $N_P[O(n^2)+O(f)]$, and the complexity of Algorithm 2 with G generations is $G \cdot N_P[O(n^2/2) + O(f)]$. The overall framework of the algorithm is illustrated in Fig. 2.

Algorithm 2: Proposed memetic algorithm for single drone scheduling.

```
1  Randomly initialize a population of solutions;
2  while the stopping condition is not met do
3      Calculate the wavelengths of the solutions;
4      foreach solution X in the population do
5          if X is not improved or is in the second half of the population then
6              Choose the global mutation operator;
7          else
8              Use the DQN to select the operator;
9          if the selected operator is global mutation then
10             for i = 1 to n do
11                 if rand() < λ(X) then
12                     Reverse a random subsequence of X;
13             Try to tune the start points and directions of the components of X;
14         else
15             Apply the selected neighborhood search operator to X;
16         if the operator is selected by the DQN then
17             Calculate the reward and add the transaction to the training set;
18         if the fitness of X is improved then
19             Update the solution in the population;
20         else if X has not been improved for K_G generations then
21             Randomly select an exemplar solution X* from the first half of the population;
22             for i = 1 to n do
23                 if rand() < λ(X) then
24                     Reverse a random subsequence of X*;
25             Try to tune the start points and directions of the components of X;
26             Replace X with the solution mutated from X*;
27     if the generation number reaches a predefined number then
28         Train the DQN using the samples accumulated during this period;
29 return the best-known solution.
```

4.2. Memetic optimization for multiple drone scheduling

In this subsection, we extend the above algorithm for multiple drone scheduling. To randomly initialize a solution, the algorithm first randomly divides the set $E^\dagger$ of freezing segments into m parts, each being assigned to one drone; then, for each jth part, the algorithm randomly generates a sub-schedule of the jth drone using the procedure of solution initialization for single drone scheduling described in the above subsection.

The global search (mutation) on a multiple drone scheduling solution X consists of two steps, one for moving components among sub-schedules and one for perform permutation within sub-schedules:

1. For $j = 1$ to $m^2/4$ do:
   (a) Randomly select two drone sub-schedules, the indices of which denoted by $j1$ and $j2$;
   (b) Randomly select a subset C of segments from the $j1$th sub-schedule and move it into the $j2$th sub-schedule, where the cardinality of subset is at most $n_{j1}/2$;

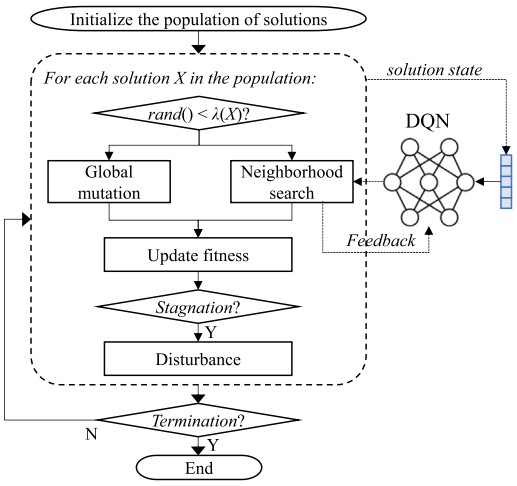


Fig. 2. Framework of the proposed memetic algorithm assisted by deep reinforcement learning.

2. For $j = 1$ to m do:

   (a) For $i = 1$ to $n_j/2$, with a probability of $\lambda(X)$, reverse a random subsequence of the jth sub-schedule;
   (b) With a probability of p_I, use the tentative improvement procedure to the components of the jth sub-schedule.

Neighborhood search are extended to the following four operators for multiple drone scheduling:

NS1. Randomly select $m/2$ sub-schedules and for each jth sub-schedule, continuously exchange two randomly selected components of the sub-schedule until the fitness is improved or the number of exchange reaches an upper limit $n_j^2/4$.

NS2. Randomly select $m/2$ sub-schedules and for each jth sub-schedule, continuously select a randomly component and reinsert it into another random position of the sub-schedule until the fitness is improved or the number of reinsertion reaches an upper limit $n_j^2/4$.

NS3. Randomly select $m/2$ sub-schedules and for each jth sub-schedule, continuously reverse a randomly selected subsequence of the sub-schedule until the fitness is improved or the number of reversion reaches an upper limit n_j.

NS4. For $i = 1$ to $m^2/4$, randomly select two sub-schedules, and continuously move a component from one sub-schedule to another until the fitness is improved or the number of movement reaches an upper limit $\min(n_{j1}, n_{j2})$.

Similar to that for single drone scheduling, we use a DQN to adaptively select among the four operators and the global mutation operator for each newly updated solution in the first half of the population, and replace any solution X that has not been improved for consecutive K_G generations by a new solution generated by mutating an exemplar solution randomly select from the first half of the population with an intensity proportional to $\lambda(X)$.

Algorithm 3 presents the framework of the memetic optimization algorithm for multiple drone scheduling. The time complexity of global mutation is $O(m^2n/8) + O(f)$; the time complexities of NS1, NS2, NS3, and NS4 are $O(n^2/4)$, $O(n^2/4)$, $O(n^2/2)$, and $O(mn/4)$, respectively, and therefore the time complexity of neighborhood search is $O(n^2/2)+O(f)$. Consequently, and the complexity of Algorithm 3 with G generations is $G \cdot N_P[O(m^2n/8) + O(n^2/2) + O(f)]$.

Algorithm 3: Proposed memetic algorithm for multiple drone scheduling.

```
Randomly initialize a population of solutions;
while the stopping condition is not met do
    Calculate the wavelengths of the solutions;
    foreach solution X in the population do
        if X is not improved or is in the second half of the population then
            Choose the global mutation operator;
        else
            Use the DQN to select the operator;
        if the selected operator is global mutation then
            Perform a global mutation on X with an intensity of λ(X);
        else
            Apply the selected neighborhood search operator to X;
        if the operator is selected by the DQN then
            Calculate the reward and add the transaction to the training set;
        if the fitness of X is improved then
            Update the solution in the population;
        else if X has not been improved for K_G generations then
            Randomly select an exemplar solution X* from the first half of the population;
            Replace X with a new solution mutated from X* with an intensity of λ(X);
    if the generation number reaches a predefined number then
        Train the DQN using the samples accumulated during this period;
    Update the selection probabilities of the neighborhood search operators;
return the best-known solution.
```

Table 1
Summary of the test instances of single deicing drone scheduling.

#	$\|V\|$	$\|E\|$	$\|E^\dagger\|$	$\sum_{e\in E} d(e)$	$\sum_{e\in E^\dagger} d(e)$	$\|\Gamma\|$
1	16	35	32	37.1	28.6	39
2	18	50	42	44.9	35.6	37
3	21	77	65	72.3	66.1	45
4	27	124	103	115.8	98.5	42
5	26	158	149	142.0	122.7	49
6	36	211	188	185.4	156.5	51
7	48	406	356	340.6	314.0	54
8	60	673	512	590.8	486.6	73

5. Computational results

To test the performance of the proposed algorithm, we use eight problem instances of single-drone scheduling and ten problem instances of multiple deicing drone scheduling, all of which are constructed based on real-world high-speed railway catenary networks in China. Tables 1 and 2 describe the basic information of the two sets of the instances, respectively, where $\sum_{e\in E} d(e)$ denotes the total length (in km) of the whole catenary work, $\sum_{e\in E^\dagger} d(e)$ denotes the total length of segments to be deiced (which represents the total workload of deicing), and $|\Gamma|$ denotes the number of trains in the timetable of the considered railway network.

5.1. Parameter tuning

We tune key parameters of the proposed algorithm on the first six single-drone test instances and the first six multi-drone test instances.

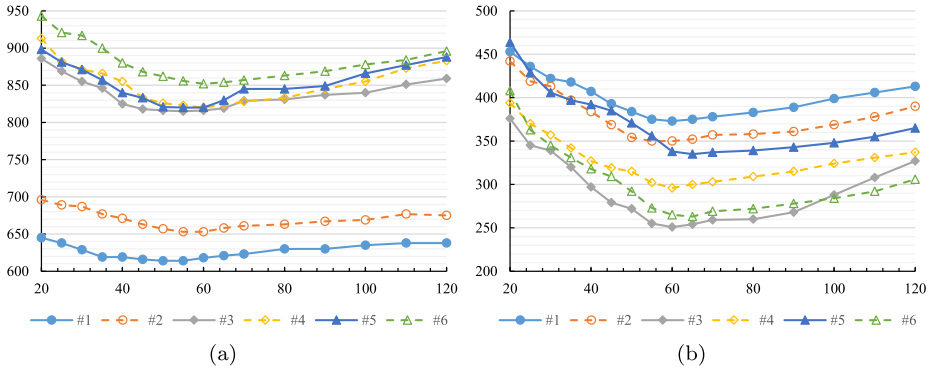


Fig. 3. Algorithm performance with different population sizes N_P on (a) single-drone test instances; (b) multi-drone test instances.

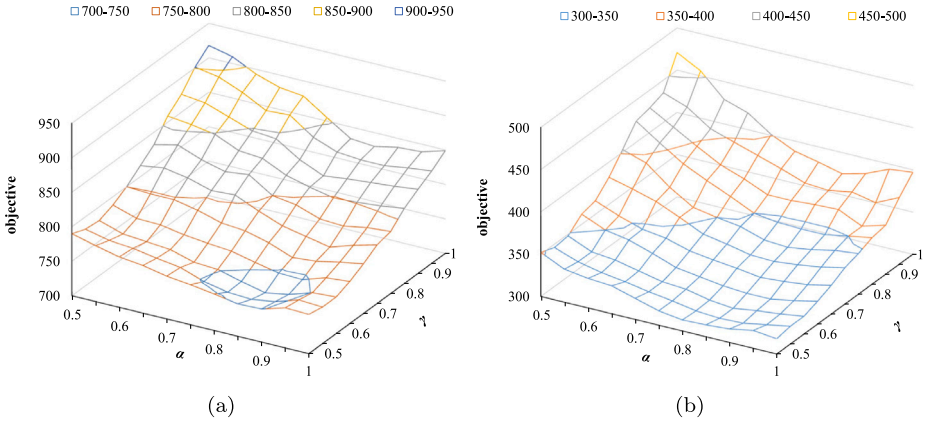


Fig. 4. Algorithm performance with different value combinations of learning rate α and discount factor γ on (a) single-drone test instances; (b) multi-drone test instances.

Table 2
Summary of the test instances of multiple deicing drone scheduling.

#	m	$\|V\|$	$\|E\|$	$\|E^\dagger\|$	$\sum_{e\in E} d(e)$	$\sum_{e\in E^\dagger} d(e)$	$\|\Gamma\|$
1	2	16	35	32	37.1	28.6	39
2	3	21	77	65	72.3	66.1	45
3	4	26	158	149	142.0	122.7	49
4	4	36	211	188	185.4	156.5	51
5	4	48	406	356	340.6	314.0	54
6	6	48	406	356	340.6	314.0	54
7	6	60	673	512	590.8	486.6	73
8	8	60	673	512	590.8	486.6	73
9	9	77	1453	1216	1289.5	997.7	90
10	12	77	1453	1216	1289.5	997.7	90

First, we tune the initial population size N_P of the memetic optimization algorithm without using reinforcement learning, where the selection probability of each neighborhood search operator is initially the same and then updated to be proportional to the number of success (i.e., improvement of solution fitness) of the operator during the previous ten generations. Fig. 3(a) and (b) present the objective function values obtained by the algorithm with 16 different N_P values in the range {20, 25, ..., 65, 70, 80, ..., 120} on the single-drone and multi-drone test instances, respectively. As we can observe, for single drone scheduling, the algorithm obtains the best results on when N_P is in the range of $[50, 60]$; for multiple drone scheduling, the algorithm obtains the best results on four instances when $N_P = 60$ and on two instances when $N_P = 65$. Thereby, we set $N_P = 60$ in the experiments.

Next, we tune the parameters of the DQN by testing 11 values of the learning rate α and 11 values of discount factor γ, both in {0.5, 0.55, ..., 1}. The results of single-drone and multi-drone test instances are presented in Fig. 4(a) and (b), respectively. On the single-drone test instances, the algorithm exhibits the best performance when $\alpha = 0.6$ and $\gamma = 0.85$; on the multi-drone test instances, the algorithm exhibits the best performance when $\alpha = 0.55$ and $\gamma = 0.85$. Therefore, we use these two settings for the two test sets, respectively.

5.2. Computational results on single drone scheduling

On the test instance, we test our deep reinforcement learning assisted memetic algorithm, denoted by Mem-DRL, with a set of popular metaheuristic and memetic optimization algorithms. The solution structure of single deicing drone scheduling is similar to that of the permutation flow-shop scheduling problem (FSP), except that our problem needs to determine an additional direction at each component and arrange the drone to visit depots for battery replacement if its power is insufficient. Therefore, we select the following 11 efficient metaheuristic/memetic algorithms for FSP from the literature and adapt them to our problem by adding the additional direction part to solution representation as well as performing the tentative improvement procedure for each new solution as our method:

- An adaptive order-based GA (AGA) with multiple operators [46]. The population size N_P is tuned to 40, crossover rate is 1.0, mutation rate is 0.8, and basic utilizing ratio is 0.04.
- A discrete PSO (DPSO) algorithm [47]. N_P is 20, learning coefficient is 1.5, and maximum velocity is 10.
- A discrete cuckoo search algorithm (DCSA) [48]. N_P is 25, and the minimum and maximum velocity are -4 and 4, respectively.
- A teaching-learning-based optimization (TLBO) algorithm [49]. N_P is 12, superior population size is 3, crossover rate is 0.2, local search rate is 0.2, initial temperature is 1, iteration of reconstruction is 10, and learning rate is 0.1.

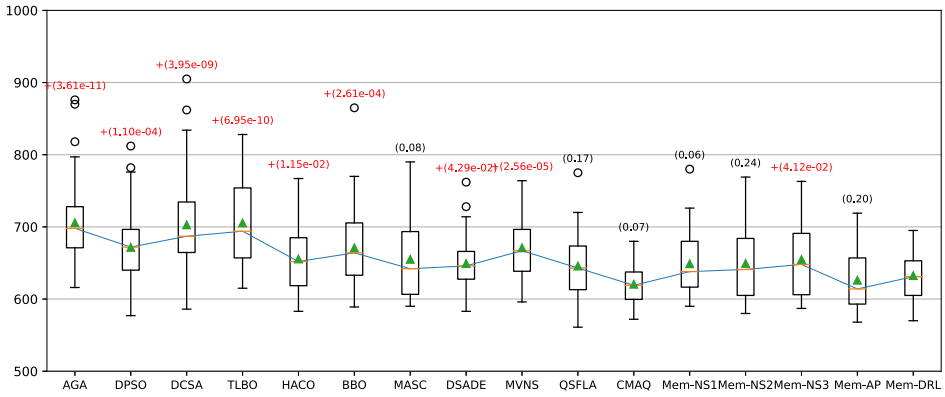


Fig. 5. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #1.

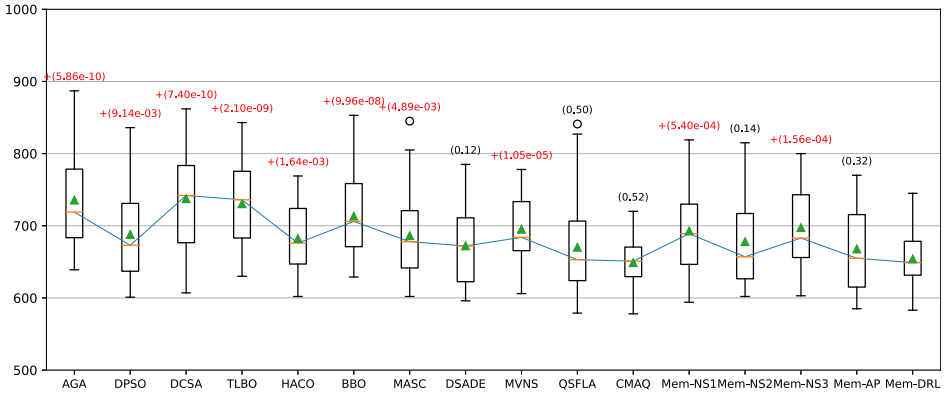


Fig. 6. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #2.

- A hybrid ant colony optimization (HACO) algorithm [50]. N_P is 25, the importance coefficients of pheromone trail and visibility are both 1, and the evaporation rate is 0.5.
- A biogeography-based optimization (BBO) algorithm using ecogeography-based migration [51] which outperforms other BBO variants [52,53]. N_P is 45, the maximum emigration and immigration rates are both 1, and the mutation rate is 0.02.
- A memetic algorithm combining GA using semi-constructive crossover (MASC), simulated annealing, and Nawaz–Enscore–Ham (NEH) [54]. N_P is 100, crossover rate is 0.8, and mutation/annealing rate is 0.05.
- A discrete self-adaptive DE (DSADE) algorithm [55]. N_P is 50, crossover rate generation parameter is 0.5, and mutation probability is 0.25.
- A multi-local search-based variable neighborhood search (MVNS) algorithm [56]. The strength of shaking is set to 1.
- A shuffled frog-learning algorithm with Q-learning (QSFLA) [25]. N_P is 60, number of memeplexes is 6, neighborhood size is 50, learning rate is 0.1, discount factor is 0.9, and epsilon-greedy is 0.9.
- A cooperative metaheuristic algorithm based on Q-learning (CMAQ) [27]. N_P is 100, neighborhood size is 20, number of shared solutions is 3, learning rate is 0.9, discount factor is 0.9, and epsilon-greedy is 0.7.

Moreover, to validate the effectiveness its adaptive neighborhood search components, we implement four versions of our algorithm: the first three versions use the wavelength-based global mutation, but each uses a single neighborhood search operator of NS1, NS2, and NS3 and is denoted by Mem-NS1, Mem-NS2, and Mem-NS3, respectively; the fourth version, denoted by Mem-AP, uses the success-rate-based selection probability.

For a fair comparison, we set the same stopping condition that the number of objective function evaluations reaches $300n$ for all algorithms. Each algorithm is independently run 51 times with random seeds on each instance. Figs. 5–12 present the box plots, including the mean (shown in green triangle) median, maximum, minimum, the first quartile (Q1) and the third quartile (Q3) objective function values obtained by the comparative algorithms on the eight instances, respectively. Any objective value below the lower limit $Q1 - 1.5(Q3 - Q1)$ or above the upper limit $Q3 + 1.5(Q3 - Q1)$ is considered as an outlier. We conduct nonparametric Wilcoxon rank sum test on the result obtained by our memetic algorithm and the result obtained by each other algorithm on each instance, and present the resulting p-value above the maximum value of the corresponding box and, if the value is smaller than 0.05, mark a red '+' before the p-value to indicate there is a statistically significant difference at a confidence level of 95%. We also summarize the ranking of the result obtained by each algorithm on each test instance as well as the sum of ranking on the whole test set in Table 3.

On the first two instances #1 and #2, the median values of AGA, DCSA and TLBO is obviously larger, and the median values of the other 13 algorithms are relatively close. According to the rank sum test results, on instance #1, there are no significant differences between the result of Mem-DRL and those of MASC, QSFLA, CMAQ, Mem-NS1,

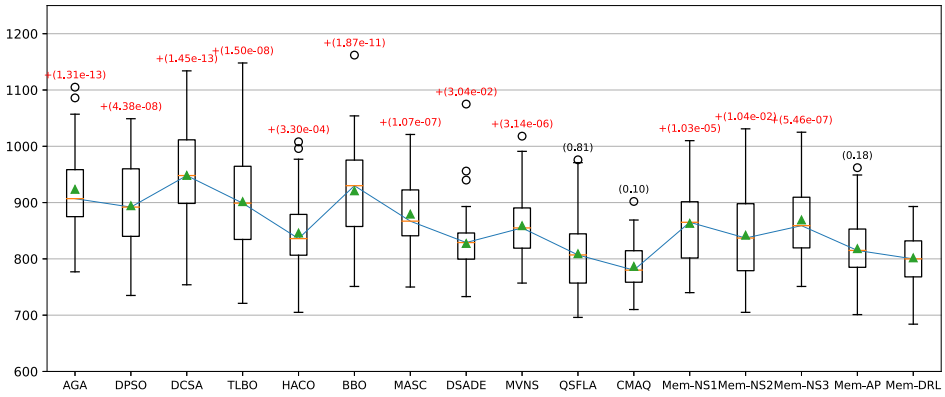


Fig. 7. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #3.

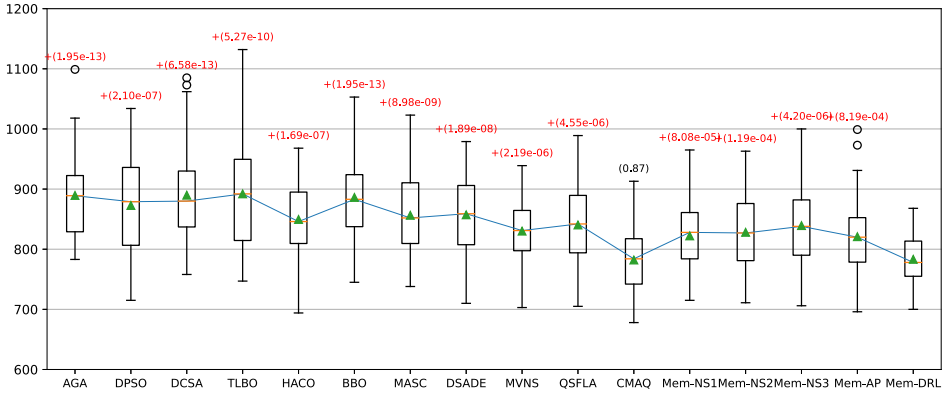


Fig. 8. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #4.

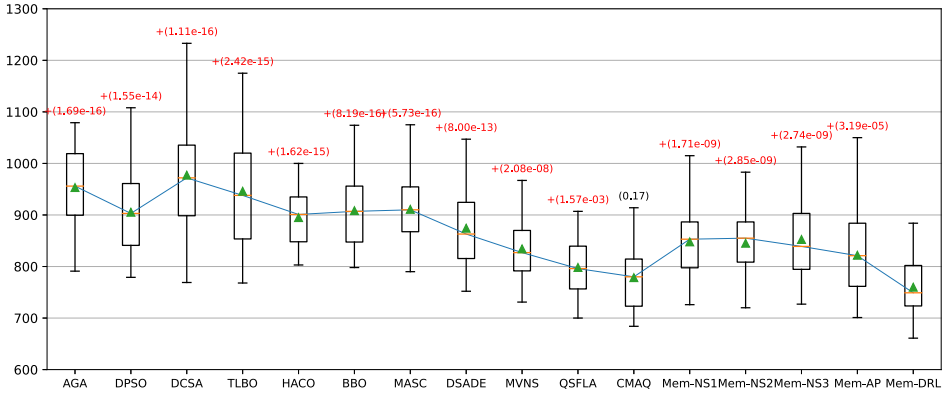


Fig. 9. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #5.

Mem-NS2 and Mem-AP, while the result of Mem-DRL is significantly better than those of the other nine algorithms; on instance #2, there are no significant differences between Mem-DRL and those of DSADE, QSFLA, CMAQ, Mem-NS2, and Mem-AP, while the result of Mem-DRL is significantly better than those of the other ten algorithms. The remaining instances become larger and more difficult, and the performance advantages of Mem-DRL become more obvious. On instance #3, Mem-DRL significantly outperforms 12 algorithms except QSFLA, CMAQ, and Mem-AP; on instances #4, #5, and #6, Mem-DRL significantly outperforms 14 algorithms except CMAQ; on the last large-size instances #7 and #8, the result of Mem-DRL is significantly better than those of all other 15 algorithms.

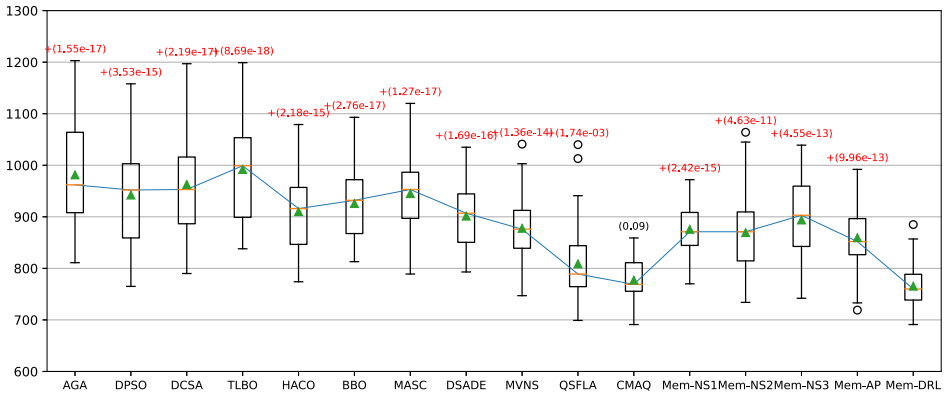


Fig. 10. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #6.

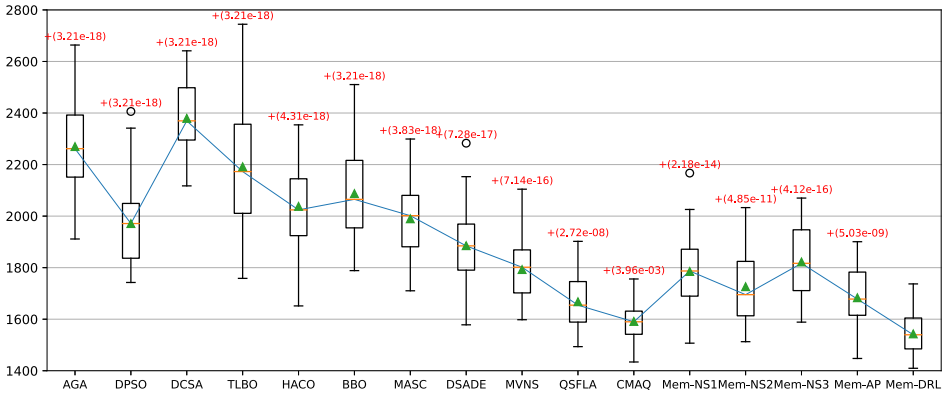


Fig. 11. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #7.

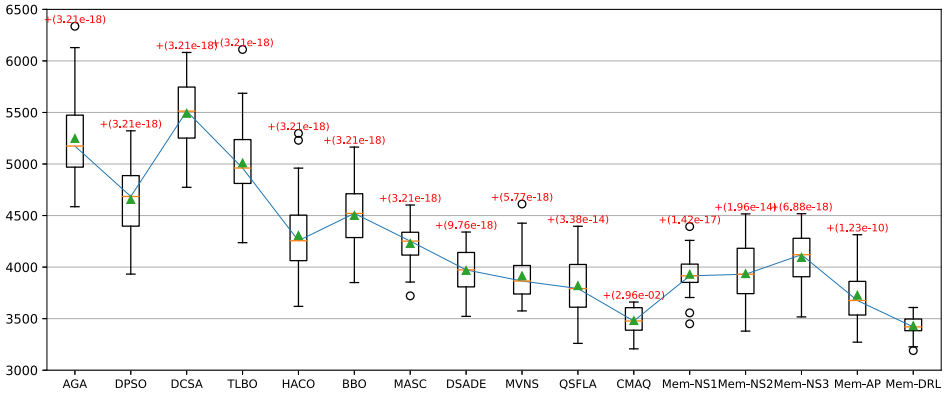


Fig. 12. Box plots of the results obtained by the 16 comparative algorithms on single drone scheduling instance #8.

In general, on relatively small-size instances, metaheuristic algorithms such as DPSO, HACO, and DSADE exhibit relatively good performance because their designs aim to achieve balance between global search and local search. In particular, the self-adaptive differential mutation mechanism of DSADE shows relatively strong search ability. However, for more difficult instances, they lack strong local search abilities to improve solution accuracies; in comparison, those algorithms combining metaheuristic and neighborhood search exhibit higher performance. In particular, MVNS and our Mem-AP algorithm using variable neighborhood search exhibit significantly better performance than the algorithms using a fixed neighborhood search operator.

Among the three versions of our algorithm with single neighborhood search operators, the performances of Mem-NS1 and Mem-NS2 are generally better than that of Mem-NS3, which indicates that the component swap and reinsertion operators are more effective than the subsequence reversion operator for the considered problem. Nevertheless, by integrating the three neighborhood search operators in an adaptive selection schema, Mem-AP and Mem-DRL exhibit significantly better performance than all three versions with single neighborhood search operators.

Except the first two relatively small-size instances, on the remaining six middle- or large-size instances, QSFLA, CMAQ, Mem-AP, and

Table 3
Ranking of the result (median objective function value) obtained by each algorithm on the test instances of single drone scheduling.

#	AGA	DPSO	DCSA	TLBO	HACO	BBO	MASC	DSADE	MVNS	QSFLA	CMAQ	Mem-NS1	Mem-NS2	Mem-NS3	Mem-AP	Mem-DRL
1	16	13	14	15	10	11	6	8	12	7	2	4	5	9	1	3
2	14	7	16	15	8	13	9	6	11	3	2	12	5	10	4	1
3	14	12	16	13	6	15	11	5	8	3	1	10	7	9	4	2
4	15	12	13	16	9	14	10	11	6	8	2	5	4	7	3	1
5	15	11	16	14	10	12	13	9	5	3	2	7	8	6	4	1
6	15	12	13.5	16	10	11	13.5	9	7	3	2	5.5	5.5	8	4	1
7	15	10	16	14	12	13	11	9	7	3	2	6	5	8	4	1
8	15	13	16	14	11	12	10	8	5	4	2	6	7	9	3	1
Sum	119	90	120.5	117	76	101	83.5	65	61	34	15	55.5	46.5	66	27	11

Table 4
Median CPU time (in seconds) consumed by each comparative algorithm on the test instances of single drone scheduling.

#	AGA	DPSO	DCSA	TLBO	HACO	BBO	MASC	DSADE	MVNS	QSFLA	CMAQ	Mem-NS1	Mem-NS2	Mem-NS3	Mem-AP	Mem-DRL
1	0.32	0.28	0.31	0.35	0.31	0.37	0.36	0.29	0.33	0.39	0.40	0.30	0.29	0.29	0.31	0.32
2	0.40	0.35	0.40	0.46	0.38	0.48	0.46	0.37	0.42	0.48	0.50	0.38	0.38	0.37	0.40	0.42
3	0.66	0.59	0.68	0.75	0.64	0.78	0.76	0.62	0.72	0.79	0.81	0.64	0.62	0.61	0.67	0.69
4	1.34	1.21	1.40	1.54	1.32	1.60	1.55	1.26	1.48	1.60	1.65	1.30	1.28	1.26	1.35	1.39
5	1.90	1.76	1.98	2.12	1.88	2.18	2.14	1.81	2.13	2.15	2.22	1.89	1.87	1.83	1.93	1.97
6	2.99	2.84	3.13	3.32	2.96	3.39	3.36	2.91	3.30	3.33	3.45	3.01	2.98	2.96	3.06	3.12
7	7.49	7.21	7.83	8.26	7.42	8.42	8.38	7.31	8.20	8.24	8.55	7.48	7.45	7.43	7.60	7.78
8	16.34	15.73	16.95	17.75	16.16	17.86	17.88	15.96	17.67	17.73	18.29	16.30	16.20	16.17	16.48	16.84

Mem-DRL obtain significantly better results than all of the first nine algorithms from the literature. The performance advantages of Mem-AP demonstrate the efficiency of the search operators of our algorithm. QSFLA, CMAQ, and Mem-DRL are all reinforcement-learning-based algorithms, and their performance advantages validate the effectiveness of the learning mechanisms in selecting the most appropriate operators for improving search efficiency and solution accuracy. The results of Mem-DRL is significantly better than those of QSFLA on five instances #4–#8 and better than CMAQ on instances #7 and #8, which indicates that the DQN is more effective than the basic Q-learning method for adaptive neighborhood search control, especially on large-size instances.

Although we use the same number of objective function evaluations as the stopping condition for all algorithms for the sake of fairness, there are still slight differences among the running times of the different algorithms, which are presented in Table 4. The maximum CPU time for solving the largest instance does not exceed 19 s, which meets the requirement of emergency response. The time costs of DPSO and DSADE are the least among the 16 algorithms, because their frameworks are comparatively simple and their search operators are time-efficient. The CPU times consumed by Mem-NS1, Mem-NS2, and Mem-NS3, three versions using single neighborhood search operators, are longer than those of DPSO and DSADE, but are shorter than those of the remaining algorithms in most cases. Compared to these three versions, Mem-AP takes approximately 4% time overhead to calculate success rates for neighborhood search operator selection; compared to Mem-AP, Mem-DRL takes approximately 3% time overhead for DQN calculation. Considering the significant improvement of the final solutions, the additional time cost is trivial.

5.3. Computational results on multiple drone scheduling

The solution structure of multiple deicing drone scheduling is similar to that of the electric VRP where vehicles need to visit stations for battery recharging or replacement, except that our problem needs to determine an additional direction at each segment (similar to a customer in VRP). For comparison, we adapt the following six electric vehicle or drone routing algorithms from the recent literature to our problem by adding the additional direction part to solution representation as well as performing the tentative improvement procedure as our algorithm:

- An iterative local search (ILS) algorithm [57].
- An adaptive large neighborhood search (ALNS) algorithm [58], control parameters of which are set as suggested in the original literature.
- A hybrid GA and column generation (GA-CG) algorithm [59]. N_P is 75, crossover rate is 0.3, mutation rate is 0.01, elitism size is 5, and the number of column generation iterations is 10.
- A bilevel ACO Algorithm [60]. N_P is 35, the evaporation rate is 0.98, and the coefficients of pheromone trail and visibility are 1 and 2, respectively.
- A rescheduling-based GA (RBGA) [18] that integrates first-in-first-out, task assignments, drone sub-route optimization. N_P is 40, crossover rate is 1, and mutation rate is 0.02.
- A diversity-enhanced memetic algorithm (DEMA) integrating new genetic operators and tabu search [61]. N_P is 10, the increased scores of three crossover operators are 10, 5, and 1, respectively, adaptive selection sensitive factor is 0.1, and the tabu search neighborhood size is $3|V|$.

We also implement five versions of our algorithms, including Mem-NS1, Mem-NS2, Mem-NS3, and Mem-NS4 that use a single neighborhood search operator of NS1, NS2, NS3, and NS4, respectively, and Mem-AP that adaptively selects operators based on success-rate-based probability.

For a fair comparison, we set the same stopping condition that number of objective function evaluations reaches 100 mn for all algorithms. Similarly, each algorithm is independently run 51 times with random seeds on each instance, and nonparametric Wilcoxon rank sum test is conducted on the result obtained by Mem-DRL and the result obtained by each other algorithm. The computational results are presented in Figs. 13–22, and the rankings of the comparative algorithms are presented in Table 5.

On each of the ten instances, the proposed Mem-DRL obtains the best median objective value among the ten comparative algorithms. Compared to the first six popular algorithms from the literature, according to the statistical tests, the result of Mem-DRL is not significantly different from the results of RBGA and DEMA on instance #1 and the result of ACO on instance #2; in all other cases, the result of Mem-DRL is significantly better than the result of each other algorithm. On these multiple drone test instances that are more difficult, the proposed algorithm has more obvious performance advantages over the existing algorithms. Comparatively, the performance of ISL is the worst because of its global search ability is relatively weak; the performance of GA-CG

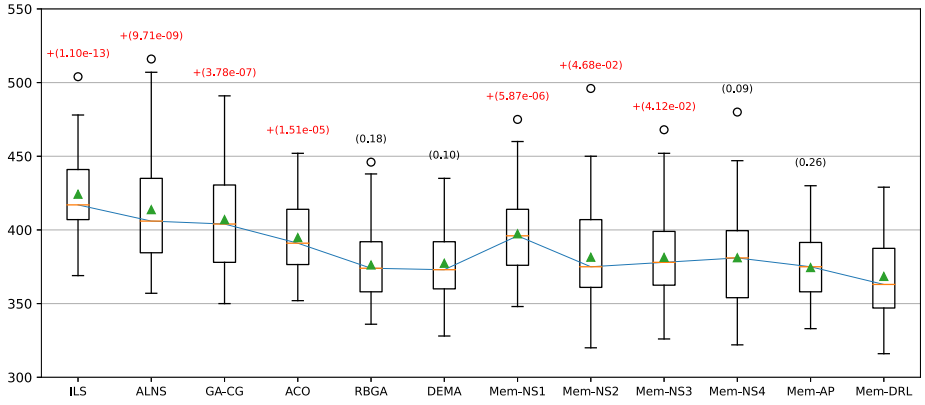


Fig. 13. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #1.

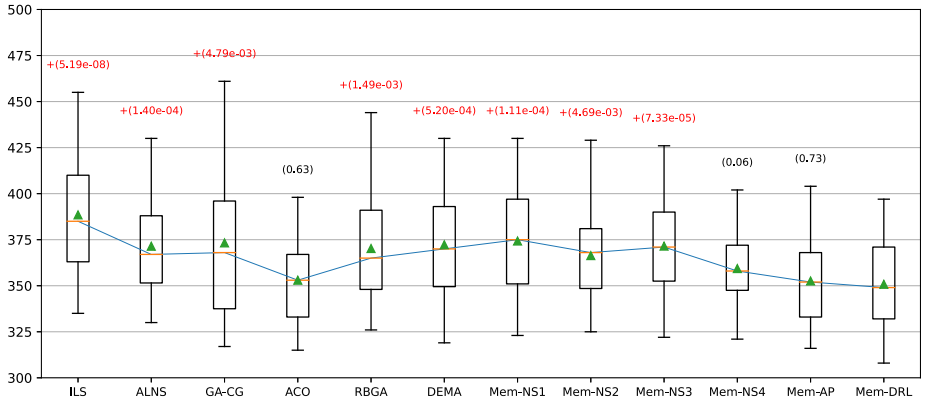


Fig. 14. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #2.

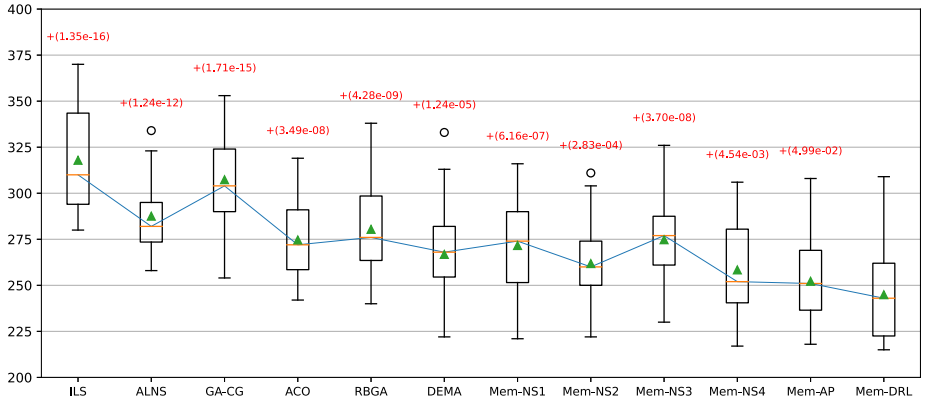


Fig. 15. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #3.

is the second worst mainly due to its poor local search ability. Similar to the case of single drone scheduling instances, the algorithms emphasizing local search exhibits better performance on more difficult instances. Particularly, another memetic algorithm DEMA performs best among these six existing algorithm, but still significantly worse than Mem-DRL which is elaborately designed for the considered problem.

Compared to the four versions of our algorithm using single neighborhood search and Mem-AP using success-rate-based adaptive neighborhood search, the result of Mem-DRL is not significantly different from the results of Mem-NS4 and Mem-AP on instances #1 and #2; in all other cases, the result of Mem-DRL is significantly better than the result of each other version. Using neighborhood search operators that share information among different sub-schedules of different drones, Mem-NS2 and Mem-NS4 perform generally better Mem-NS1

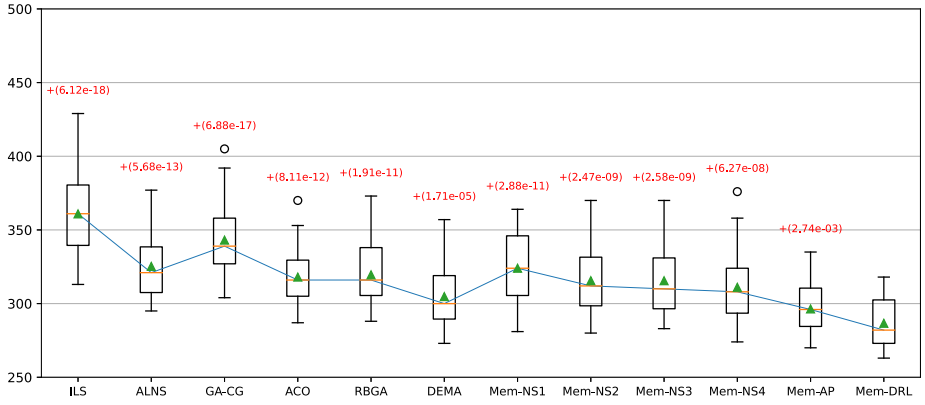


Fig. 16. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #4.

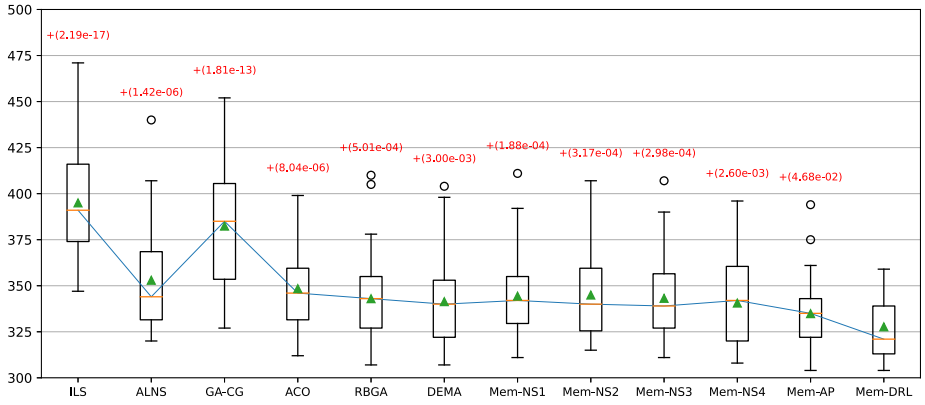


Fig. 17. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #5.

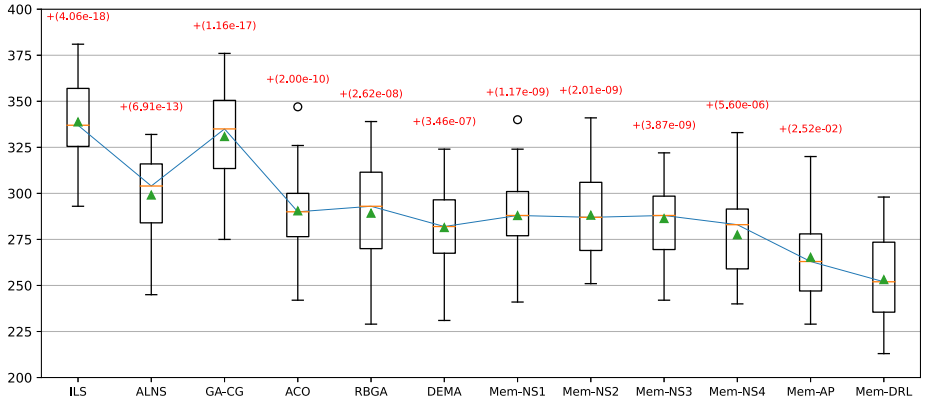


Fig. 18. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #6.

and Mem-NS3 that exchange information within sub-schedules. By integrating the four neighborhood search operators in an adaptive selection schema, Mem-AP exhibits significantly better performance than all four versions with single neighborhood search. Using deep reinforcement learning to better exploit historical information to adaptively select the most appropriate operator for each neighborhood search operation, Mem-DRL further outperforms Mem-AP significantly on the last eight instances.

In summary, the proposed Mem-DRL exhibits the best performance among the ten comparative algorithms on each instance; except the relatively small-size instances #1 and #2, Mem-DRL performs significantly better than all other algorithms on all medium- and large-size instances #3–#10. The performance advantages of the proposed algorithm become more obvious on more difficult instances, i.e., instances with larger number of segment to be deiced and larger number of

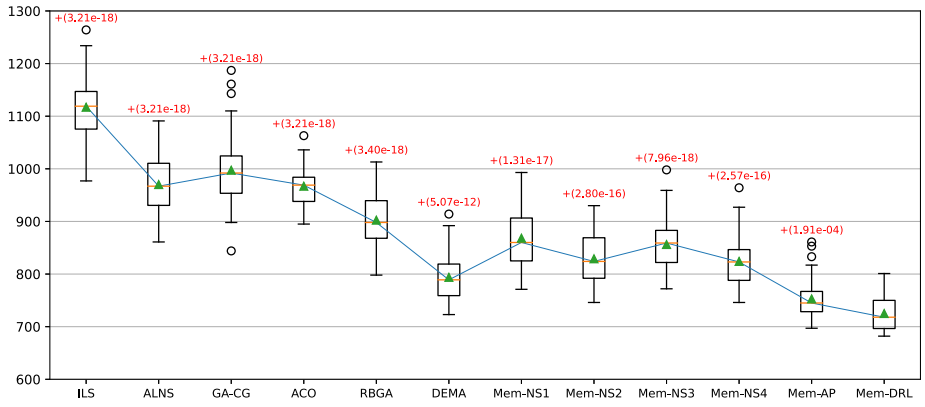


Fig. 19. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #7.

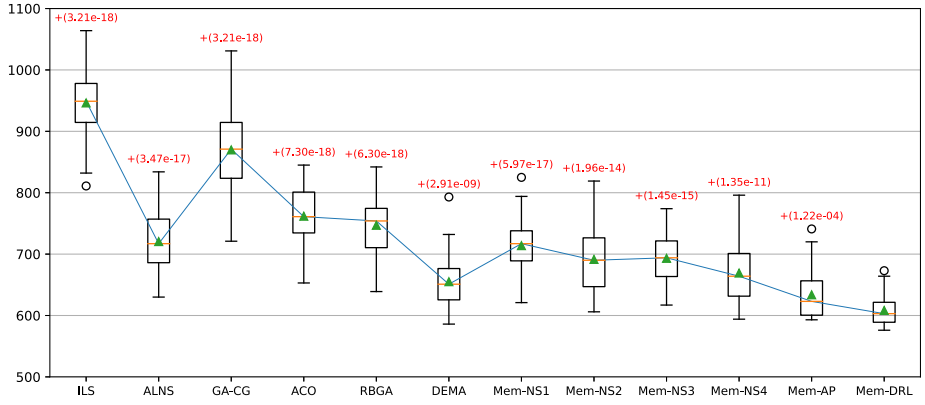


Fig. 20. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #8.

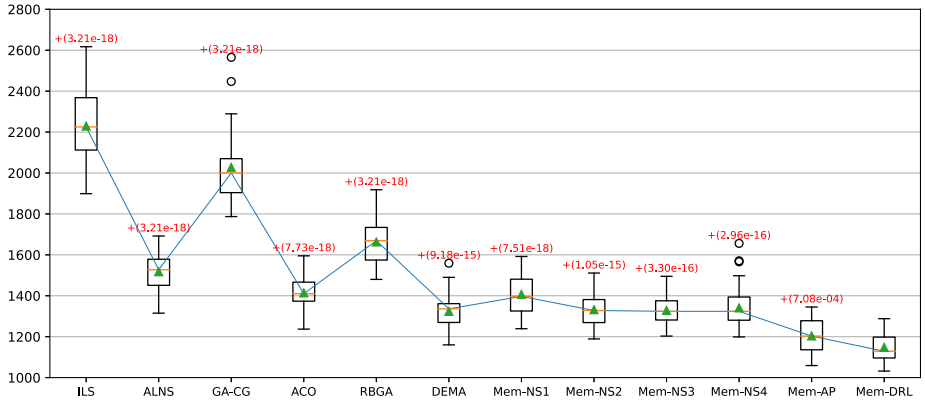


Fig. 21. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #9.

drones. The computational results demonstrate that the proposed algorithm is effective and efficient for solving the considered deicing drone scheduling problem, especially for complex instances.

Table 6 presents the average CPU times consumed by the comparative algorithms on the multi-drone test instances. The time cost of ILS is always the least, but its time-efficiency is at the expense of low solution accuracy. The CPU times consumed by the four versions using single neighborhood search operators are typically longer than those of ILS and ALNS, but shorter than those of GA-CG, RBGA, and DEMA. Compared to these four versions, Mem-AP takes an approximately 3% time overhead; compared to Mem-AP, Mem-DRL takes an approximately 3% time overhead, which is trivial considering the significant improvement of the final solutions. The solution times do not exceed 10 s on the first three instances, do not exceed 120 s on instances #4–#8, and are approximately five to six minutes on the largest instances #9 and #10, which are generally acceptable.

6. Conclusion and discussion

Deicing drones are an effective method in response to freezing disasters striking railway catenary systems. In this study, we present a deicing drone scheduling problem for restoring catenary systems with

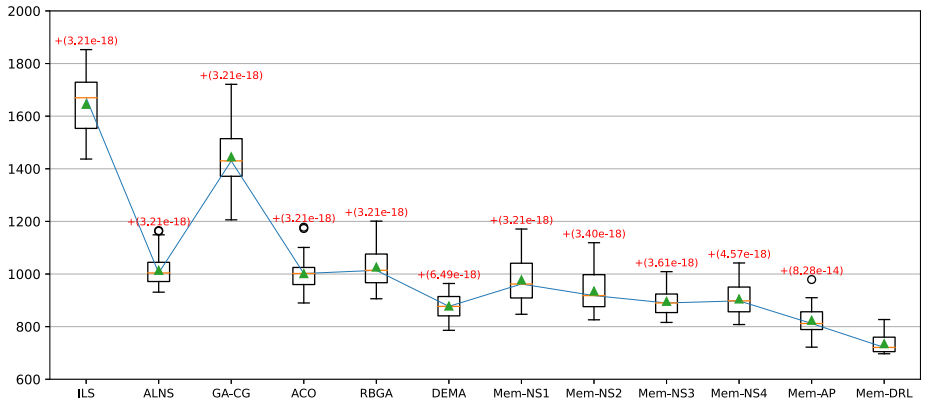


Fig. 22. Box plots of the results obtained by the ten comparative algorithms on multiple drone scheduling instance #10.

Table 5
Ranking of the result (median objective function value) obtained by each algorithm on the test instances of multiple drone scheduling.

#	ILS	ALNS	GA-CG	ACO	RBGA	DEMA	Mem-NS1	Mem-NS2	Mem-NS3	Mem-NS4	Mem-AP	Mem-DRL
1	12	11	10	8	3	2	9	4.5	6	7	4.5	1
2	12	6	7.5	3	5	9	11	7.5	10	4	2	1
3	12	10	11	6	8	5	7	4	9	3	2	1
4	12	9	11	7.5	7.5	3	10	6	5	4	2	1
5	12	9	11	10	8	4.5	6.5	4.5	3	6.5	2	1
6	12	10	11	8	9	3	6.5	5	6.5	4	2	1
7	12	9	11	10	8	3	7	5	6	4	2	1
8	12	7.5	11	10	9	3	7.5	5	6	4	2	1
9	12	9	11	8	10	6	7	5	3.5	3.5	2	1
10	12	9	11	8	10	3	7	6	4	5	2	1
Sum	108	80.5	94.5	70.5	67.5	38.5	71.5	46.5	55	40	20.5	9

Table 6
Median CPU time (in seconds) consumed by each comparative algorithm on the test instances of multiple drone scheduling.

#	ILS	ALNS	GA-CG	ACO	RBGA	DEMA	Mem-NS1	Mem-NS2	Mem-NS3	Mem-NS4	Mem-AP	Mem-DRL
1	0.52	0.57	0.69	0.60	0.66	0.64	0.65	0.64	0.62	0.62	0.70	0.74
2	1.87	2.10	2.38	2.12	2.31	2.27	2.27	2.24	2.19	2.20	2.36	2.48
3	7.32	8.18	9.45	8.38	9.17	9.02	9.06	8.99	8.93	8.97	9.30	9.68
4	12.51	13.97	16.16	14.23	15.69	15.44	15.40	15.38	15.30	15.34	15.79	16.26
5	31.73	34.80	41.25	36.22	40.07	39.44	39.21	39.18	39.01	39.04	39.69	40.67
6	42.23	46.48	55.86	48.27	52.99	52.46	52.20	52.21	51.90	52.02	52.89	54.24
7	73.25	80.48	97.55	84.25	92.96	89.72	89.46	89.57	88.89	89.05	90.45	92.75
8	97.52	105.00	128.65	111.22	124.88	116.64	116.30	116.45	115.67	116.07	117.55	120.27
9	266.42	286.50	349.98	302.51	339.76	318.63	316.36	317.14	316.22	315.71	319.36	323.71
10	322.86	346.96	433.46	376.37	421.21	386.40	382.80	383.97	382.66	382.00	386.06	390.89

the objective to minimize the total negative effect in terms of train delay and cancellation due to the freezing catenary. To efficiently solve the problem, we propose a memetic algorithm integrating global mutation and variable neighborhood search controlled by reinforcement learning, the performance of which is demonstrated by computational results on real-world problem instances compared to selected popular optimization algorithms in the literature.

Our future work considers extending this study in several aspects. First, the current problem only considers stationary depots to provide battery replacement, and ongoing study is integrating stationary depots and movable depots (e.g., ground vehicles), the optimal deployment of which should be combined with the scheduling problem [62]. Second, when the icy rainfall and snowfall is continuing during the decision period, it is required to incorporate the prediction of freezing conditions and deicing workloads into the problem, for which we can use a method integrating machine learning prediction and evolutionary optimization [63,64]. Third, the current objective function evaluation is based on the train delay and cancellation according to the predefined train timetable, while a more challenging task is to enable the trains to change their routes by selecting new railway sections whose catenary lines have been or will be restored according to the deicing solution. Such extensions will significantly improve the complexity of the problem, which is to be solved by more elaborately designed methods, such as surrogate objective functions to reduce the computational complexity.

CRediT authorship contribution statement

Yu-Jun Zheng: Writing – review & editing, Methodology, Funding acquisition, Conceptualization. **Xi-Cheng Xie:** Writing – original draft, Software. **Zhi-Yuan Zhang:** Validation, Data curation. **Jin-Tang Shi:** Resources.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Yu-Jun Zheng reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by National Natural Science Foundation, China (Grants No. 62372148 and No. 61872123) and Zhejiang Provincial Natural Science Foundation, China (Grant No. LR20F030002) of China.

References

[1] W.H. Torres, B.D. Aquino, E.I.O. Rivera, Artificial intelligence and unmanned aerial vehicle applications on electrical power systems, in: IEEE Power & Energy Society General Meeting, 2023, pp. 1–5, http://dx.doi.org/10.1109/PESGM52003.2023.10253090.

[2] Y.-J. Zheng, Y.-C. Du, Z.-L. Su, H.-F. Ling, M.-X. Zhang, S.-Y. Chen, Evolutionary human-UAV cooperation for transmission network restoration, IEEE Trans. Ind. Inform. 17 (3) (2021) 1648–1657, http://dx.doi.org/10.1109/TII.2020.3003903.

[3] X. Jiang, S. Fan, Z. Zhang, C. Sun, L. Shu, Simulation and experimental investigation of DC ice-melting process on an iced conductor, IEEE Trans. Power Deliv. 25 (2) (2010) 919–929, http://dx.doi.org/10.1109/TPWRD.2009.2037632.

[4] J. Laforte, M. Allaire, J. Laflamme, State-of-the-art on power line de-icing, Atmosph. Res. 46 (1) (1998) 143–158, http://dx.doi.org/10.1016/S0169-8095(97)00057-4.

[5] M. Huneault, C. Langheit, R. St-Arnaud, J. Benny, J. Audet, J.-C. Richard, A dynamic programming methodology to develop de-icing strategies during ice storms by channeling load currents in transmission networks, IEEE Trans. Power Deliv. 20 (2) (2005) 1604–1610, http://dx.doi.org/10.1109/TPWRD.2004.838463.

[6] Y. Hou, X. Wang, Y. Zhang, Multi-objective transmission line de-icing outage optimal scheduling framework, IET Gener. Transm. Distrib. 10 (15) (2016) 3865–3874, http://dx.doi.org/10.1049/iet-gtd.2016.0404.

[7] C. Deng, S. Wang, Z. Huang, Z. Tan, J. Liu, Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications, J. Commun. 9 (9) (2014) 687–692, http://dx.doi.org/10.12720/jcm.9.9.687-692.

[8] Z. Zhou, C. Zhang, C. Xu, F. Xiong, Y. Zhang, T. Umer, Energy-efficient industrial internet of UAVs for power line inspection in smart grid, IEEE Trans. Ind. Inform. 14 (6) (2018) 2705–2714, http://dx.doi.org/10.1109/TII.2018.2794320.

[9] R. Atat, M.F. Shaaban, M. Ismail, E. Serpedin, Efficient unmanned aerial vehicle paths design for post-disaster damage assessment of overhead transmission lines, IET Smart Grid 6 (5) (2023) 503–521, http://dx.doi.org/10.1049/stg2.12120.

[10] K. Dorling, J. Heinrichs, G.G. Messier, S. Magierowski, Vehicle routing problems for drone delivery, IEEE Trans. Syst. Man Cybern. 47 (1) (2017) 70–85, http://dx.doi.org/10.1109/TSMC.2016.2582745.

[11] I. Hong, M. Kuby, A.T. Murray, A range-restricted recharging station coverage model for drone delivery service planning, Transp. Res. C 90 (2018) 198–212, http://dx.doi.org/10.1016/j.trc.2018.02.017.

[12] Y. Zheng, Y. Du, H. Ling, W. Sheng, S. Chen, Evolutionary collaborative human-UAV search for escaped criminals, IEEE Trans. Evol. Comput. 24 (2) (2020) 217–231, http://dx.doi.org/10.1109/TEVC.2019.2925175.

[13] M. Pachayappan, V. Sudhakar, A solution to drone routing problems using docking stations for pickup and delivery services, Transp. Res. Rec. 2675 (12) (2021) 1056–1074, http://dx.doi.org/10.1177/03611981211032219.

[14] J. Gómez-Lagos, B. Rojas-Espinoza, A. Candia-Véjar, On a pickup to delivery drone routing problem: Models and algorithms, Comput. Ind. Eng. 172 (2022) 108632, http://dx.doi.org/10.1016/j.cie.2022.108632.

[15] X. Wen, G. Wu, Heterogeneous multi-drone routing problem for parcel delivery, Transp. Res. C 141 (2022) 103763, http://dx.doi.org/10.1016/j.trc.2022.103763.

[16] Y. Guo, Y. Huang, S. Ge, Y. Zhang, E. Jiang, B. Cheng, S. Yang, Low-carbon routing based on improved artificial bee colony algorithm for electric trackless rubber-tyred vehicles, Complex Syst. Model. Simul. 3 (3) (2023) 169–190, http://dx.doi.org/10.23919/CSMS.2023.0011.

[17] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, X. Du, Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system, Swarm Evol. Comput. 69 (2022) 101005, http://dx.doi.org/10.1016/j.swevo.2021.101005.

[18] X. Wang, Z. Liu, X. Li, Optimal delivery route planning for a fleet of heterogeneous drones: A rescheduling-based genetic algorithm approach, Comput. Ind. Eng. 179 (2023) 109179, http://dx.doi.org/10.1016/j.cie.2023.109179.

[19] B. Xu, K. Zhao, Q. Luo, G. Wu, W. Pedrycz, A GV-drone arc routing approach for urban traffic patrol by coordinating a ground vehicle and multiple drones, Swarm Evol. Comput. 77 (2023) 101246, http://dx.doi.org/10.1016/j.swevo.2023.101246.

[20] Y.-J. Zheng, X. Chen, H.-F. Yan, M.-X. Zhang, Evolutionary algorithm for vehicle routing for shared e-bicycle battery replacement and recycling, Appl. Soft Comput. 135 (2023) 110023, http://dx.doi.org/10.1016/j.asoc.2023.110023.

[21] M. Fan, Y. Wu, Z. Cao, W. Song, G. Sartoretti, H. Liu, G. Wu, Conditional neural heuristic for multiobjective vehicle routing problems, in: IEEE Trans. Neural Netw. Learn. Syst., 2024, pp. 1–13, http://dx.doi.org/10.1109/TNNLS.2024.3371706, (in press).

[22] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L.C. Jain, A.K. Nagar, Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning, IEEE Trans. Syst. Man Cybern. 43 (4) (2013) 814–831, http://dx.doi.org/10.1109/TSMCA.2012.2226024.

[23] E. Özcan, J.H. Drake, C. Altıntaş, S. Asta, A self-adaptive multimeme memetic algorithm co-evolving utility scores to control genetic operators and their parameter settings, Appl. Soft Comput. 49 (2016) 81–93, http://dx.doi.org/10.1016/j.asoc.2016.07.032.

[24] B. Peng, Y. Zhang, Y. Gajpal, X. Chen, A memetic algorithm for the green vehicle routing problem, Sustainability 11 (21) http://dx.doi.org/10.3390/su11216055.

[25] J. Cai, D. Lei, J. Wang, L. Wang, A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling, Int. J. Prod. Res. 61 (4) (2023) 1233–1251, http://dx.doi.org/10.1080/00207543.2022.2031331.

[26] F. Zhao, G. Zhou, T. Xu, N. Zhu, Jonrinaldi, A knowledge-driven cooperative scatter search algorithm with reinforcement learning for the distributed blocking flow shop scheduling problem, Expert Syst. Appl. 230 (2023) 120571, http://dx.doi.org/10.1016/j.eswa.2023.120571.

[27] F. Zhao, T. Jiang, L. Wang, A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time, IEEE Trans. Ind. Inform. 19 (7) (2023) 8427–8440, http://dx.doi.org/10.1109/TII.2022.3218645.

[28] F. Zhao, S. Di, L. Wang, A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem, IEEE Trans. Cybern. 53 (5) (2023) 3337–3350, http://dx.doi.org/10.1109/TCYB.2022.3192112.

[29] Z. Pan, L. Wang, J. Zheng, J.-F. Chen, X. Wang, A learning-based multipopulation evolutionary optimization for flexible job shop scheduling problem with finite transportation resources, IEEE Trans. Evol. Comput. 27 (6) (2023) 1590–1603, http://dx.doi.org/10.1109/TEVC.2022.3219238.

[30] R. Li, W. Gong, C. Lu, L. Wang, A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time, IEEE Trans. Evol. Comput. 27 (3) (2023) 610–620, http://dx.doi.org/10.1109/TEVC.2022.3175832.

[31] X. Chang, X. Jia, J. Ren, A reinforcement learning enhanced memetic algorithm for multi-objective flexible job shop scheduling toward industry 5.0, Int. J. Prod. Res. (2024) 1–29, http://dx.doi.org/10.1080/00207543.2024.2357740, (in press).

[32] F. Zhao, C. Zhuang, L. Wang, C. Dong, An iterative greedy algorithm with Q-learning mechanism for the multiobjective distributed no-idle permutation flowshop scheduling, IEEE Trans. Syst. Man Cybern. 54 (5) (2024) 3207–3219, http://dx.doi.org/10.1109/TSMC.2024.3358383.

[33] F. Zhang, R. Li, W. Gong, Deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling with multi-AGV, Comput. Ind. Eng. 189 (2024) 109917, http://dx.doi.org/10.1016/j.cie.2024.109917.

[34] Z. Hu, W. Gong, W. Pedrycz, Y. Li, Deep reinforcement learning assisted co-evolutionary differential evolution for constrained optimization, Swarm Evol. Comput. 83 (2023) 101387, http://dx.doi.org/10.1016/j.swevo.2023.101387.

[35] J. Ou, L. Xing, F. Yao, M. Li, J. Lv, Y. He, Y. Song, J. Wu, G. Zhang, Deep reinforcement learning method for satellite range scheduling problem, Swarm Evol. Comput. 77 (2023) 101233, http://dx.doi.org/10.1016/j.swevo.2023.101233.

[36] M. Fan, Y. Wu, T. Liao, Z. Cao, H. Guo, G. Sartoretti, G. Wu, Deep reinforcement learning for UAV routing in the presence of multiple charging stations, IEEE Trans. Veh. Technol. 72 (5) (2023) 5732–5746, http://dx.doi.org/10.1109/TVT.2022.3232607.

[37] C. Wang, Z. Cao, Y. Wu, L. Teng, G. Wu, Deep reinforcement learning for solving vehicle routing problems with backhauls, IEEE Trans. Neural Netw. Learn. Syst. (2024) 1–15, http://dx.doi.org/10.1109/TNNLS.2024.3371781, (in press).

[38] X. Mao, G. Wu, M. Fan, Z. Cao, W. Pedrycz, DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV, IEEE Trans. Autom. Sci. Eng. (2024) 1–17, http://dx.doi.org/10.1109/TASE.2024.3358894, (in press).

[39] Y.-J. Zheng, Emergency train scheduling on Chinese high-speed railways, Transp. Sci. 52 (5) (2018) 1077–1091, http://dx.doi.org/10.1287/trsc.2017.0794.

[40] L. Hong, Y. Wang, Y. Du, X. Chen, Y. Zheng, UAV search-and-rescue planning using an adaptive memetic algorithm, Front. Inf. Technol. Electron. Eng. 22 (2021) 1477–1491, http://dx.doi.org/10.1631/FITEE.2000632.

[41] B. Li, G. Wu, Y. He, M. Fan, W. Pedrycz, An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem, IEEE/CAA J. Autom. Sin. 9 (7) (2022) 1115–1138, http://dx.doi.org/10.1109/JAS.2022.105677.

[42] Z. Pan, D. Lei, L. Wang, A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling, IEEE Trans. Cybern. 52 (6) (2022) 5051–5063, http://dx.doi.org/10.1109/TCYB.2020.3026571.

[43] Y.-J. Zheng, Water wave optimization: A new nature-inspired metaheuristic, Comput. Oper. Res. 55 (1) (2015) 1–11, http://dx.doi.org/10.1016/j.cor.2014.10.008.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, http://dx.doi.org/10.1038/nature14236.

[45] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015, URL http://arxiv.org/abs/1412.6980.

[46] W.L. Zhang, L, D.-Z. Zheng, An adaptive genetic algorithm with multiple operators for flowshop scheduling, Int. J. Adv. Manuf. Technol. 27 (2006) 580–587, http://dx.doi.org/10.1007/s00170-004-2223-3.

[47] C.-J. Liao, C.-T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (10) (2007) 3099–3111, http://dx.doi.org/10.1016/j.cor.2005.11.017.

[48] P. Dasgupta, S. Das, A discrete inter-species cuckoo search for flowshop scheduling problems, Comput. Oper. Res. 60 (2015) 111–120, http://dx.doi.org/10.1016/j.cor.2015.01.005.

[49] W. Shao, D. Pi, Z. Shao, An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem, Appl. Soft Comput. 61 (2017) 193–210, http://dx.doi.org/10.1016/j.asoc.2017.08.020.

[50] O. Engin, A. Güçlü, A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems, Appl. Soft Comput. 72 (2018) 166–176, http://dx.doi.org/10.1016/j.asoc.2018.08.002.

[51] Y.-J. Zheng, H.-F. Ling, J.-Y. Xue, Ecogeography-based optimization: Enhancing biogeography-based optimization with ecogeographic barriers and differentiations, Comput. Oper. Res. 50 (2014) 115–127, http://dx.doi.org/10.1016/j.cor.2014.04.013.

[52] Y.-J. Zheng, H.-F. Ling, H.-H. Shi, H.-S. Chen, S.-Y. Chen, Emergency railway wagon scheduling by hybrid biogeography-based optimization, Comput. Oper. Res. 43 (3) (2014) 1–8, http://dx.doi.org/10.1016/j.cor.2013.09.002.

[53] Y.-C. Du, M.-X. Zhang, C.-Y. Cai, Y.-J. Zheng, Enhanced biogeography-based optimization for flow-shop scheduling, in: J. Qiao, X. Zhao, L. Pan, X. Zuo, X. Zhang, Q. Zhang, S. Huang (Eds.), Bio-Inspired Computing: Theories and Applications, Commun Comput Inf Sci, Springer, Singapore, 2018, pp. 295–306.

[54] M. Kurdi, A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem, Appl. Soft Comput. 94 (2020) 106458, http://dx.doi.org/10.1016/j.asoc.2020.106458.

[55] M. de Fátima Morais, M.H.D.M. Ribeiro, R.G. da Silva, V.C. Mariani, L. dos Santos Coelho, Discrete differential evolution metaheuristics for permutation flow shop scheduling problems, Comput. Ind. Eng. 166 (2022) 107956, http://dx.doi.org/10.1016/j.cie.2022.107956.

[56] W. Shao, Z. Shao, D. Pi, Multi-local search-based general variable neighborhood search for distributed flow shop scheduling in heterogeneous multi-factories, Appl. Soft Comput. 125 (2022) 109138, http://dx.doi.org/10.1016/j.asoc.2022.109138.

[57] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, G. Laporte, The green mixed fleet vehicle routing problem with partial battery recharging and time windows, Comput. Oper. Res. 101 (2019) 183–199, http://dx.doi.org/10.1016/j.cor.2018.07.012.

[58] M. Keskin, B. Çatay, A metaheuristic method for the electric vehicle routing problem with time windows and fast chargers, Comput. Oper. Res. 100 (2018) 172–188, http://dx.doi.org/10.1016/j.cor.2018.06.019.

[59] C. Wang, C. Guo, X. Zuo, Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm, Appl. Soft Comput. 112 (2021) 107774, http://dx.doi.org/10.1016/j.asoc.2021.107774.

[60] Y.-H. Jia, Y. Mei, M. Zhang, A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem, IEEE Trans. Cybern. 52 (10) (2022) 10855–10868, http://dx.doi.org/10.1109/TCYB.2021.3069942.

[61] J. Xiao, J. Du, Z. Cao, X. Zhang, Y. Niu, A diversity-enhanced memetic algorithm for solving electric vehicle routing problems with time windows and mixed backhauls, Appl. Soft Comput. 134 (2023) 110025, http://dx.doi.org/10.1016/j.asoc.2023.110025.

[62] N. Li, Z. Su, H. Ling, M. Karatas, Y. Zheng, Optimization of air defense system deployment against reconnaissance drone swarms, Complex Syst. Model. Simul. 3 (2) (2023) 102–117, http://dx.doi.org/10.23919/CSMS.2023.0003.

[63] X. Chen, H.-F. Yan, Y.-J. Zheng, M. Karatas, Integration of machine learning prediction and heuristic optimization for mask delivery in COVID-19, Swarm Evol. Comput. 76 (2023) 101208, http://dx.doi.org/10.1016/j.swevo.2022.101208.

[64] Y.-J. Zheng, X. Chen, Q. Song, J. Yang, L. Wang, Evolutionary optimization of COVID-19 vaccine distribution with evolutionary demands, IEEE Trans. Evol. Comput. 27 (1) (2023) 141–154, http://dx.doi.org/10.1109/TEVC.2022.3164260.