An Enhanced Backtracking Search Algorithm for the Flight Planning of a Multi-drones-assisted Commercial Parcel Delivery System

Yiying Zhang, Guanzhong Zhou, Peng Hang, Chao Huang and Hailong Huang

Abstract—**Using drones to carry out commercial parcel delivery can significantly promote the transformation and upgrading of the logistics industry thanks to the saving of human labor source, which is becoming a new component of intelligent transportation systems. However, the flight distance of drones is often constrained due to the limited battery capacity. To address this challenge, this paper designs a multi-drones-assisted commercial parcel delivery system, which supports long-distance delivery by a generalized service network (GSN). Each node of the GSN is equipped with charging piles to provide a charging service for drones. Given the limited number of charging piles at each node and the limited battery capacity of a drone, to ensure the efficient operation of the system, the flight planning problem of drones is converted into a large-scale optimization problem by a priority-based encoding mechanism. To solve this problem, an enhanced backtracking search algorithm (EBSA) is reported, which is inspired by the characteristics of the considered flight planning problem and the weak ability of the backtracking search algorithm to escape from a local optimum. The core components of EBSA are the designed comprehensive learning mechanism and local escape operator. Experimental results prove the validity of the improved strategies and the excellent performance of EBSA on the considered flight planning problem.**

Index Terms—**Parcel delivery, flight planning, multi-drones, generalized service network, backtracking search algorithm.**

I. Introduction

THE rapid development of commercial drones is pounding greatly at the traditional parcel delivery methods. Using drones for parcel delivery can 1) improve delivery efficiency, 2) solve delivery problems in remote areas, 3) reduce operating costs, and 4) effectively promote the transformation and upgrading of the logistics industry. Some famous logistics companies, such as Amazon and SF Express, have made some attempts to employ drones for parcel delivery [1]–[3].

When a drone is regarded as a platform for parcel delivery, its energy should be added multiple times by either replacing the battery or recharging the battery to execute long-distance delivery due to the limited battery capacity [4], [5]. Given this, the flight planning (FP) for the drone is strongly correlated with the operating cost and delivery efficiency. For a multi-drones-assisted parcel delivery system, determining the flights of drones can be regarded as solving an optimization problem, whose objective function is the operating cost that is usually measured by the average travel time or average energy consumption of all drones during parcel delivery. The limitation of the sharing resources, such as service facilities for replacing or recharging the battery, is usually seen as the constraint of the optimization problem. Specifically, solving the optimization problem is to search for the optimal flights for all drones from the search space meeting the constraint.

To offer support for using drones to carry out long-distance delivery, researchers have reported two models, i.e., the drone-vehicle model and the drone-station model. The drone-vehicle model can be divided into two major categories: 1) a vehicle carries some spare batteries to add power to drones by replacing the battery [6], [7], which needs the help of human drivers, and 2) drones travel with public transportation vehicles to perform delivery [8], [9], which is greatly influenced by the public transportation network. The delivery efficiency of the drone-vehicle model has usually a large uncertainty due to unpredictable traffic. The basic idea of the drone-station model is that some service stations (SSs) are deployed, which can provide some professional services (e.g., replacing the battery, recharging the battery, and maintenance) for drones [10]. From this sense, the drone-station model is a more reliable and promising delivery technique than the drone-vehicle model.

For the drone-station model, once the locations of SSs and the delivery missions are fixed, its delivery efficiency mainly depends on the flights of drones. In recent years, scholars have done some research on the FP of drones. However, they mainly pay attention to the obstacle avoidance of drones [11]–[13], which do not cover the long-distance FP of multiple drones. Lee et al. [2] present an approach to finding a set of collision-free paths for delivery missions in a congested flight space. Dorling et al. [14] report an Internet of Things-based drone delivery system. However, the techniques of using drones to execute long-distance delivery missions are not mentioned in [2], [14]. Huang et al. [1] study the FP of a drone used to carry out long-distance delivery, but it does not discuss multi-drones-to-multi-missions. Here, it should be pointed out that using multi-drones to perform long-distance delivery missions is the basic feature of the drone-station model.

To get close to the real parcel delivery situation based on the drone-station model, this paper solves the FP problem of

This work was supported by Department General Research Grant [P0040253] and the Research Institute for Sports Science and Technology [P0043566]. *(Corresponding author: Chao Huang.)*

Y. Zhang and C. Huang are with the Department of Industrial and Systems Engineering, the Hong Kong Polytechnic University, Hong Kong. (E-mail: {yiyingchreo.zhang,hchao.huang}@polyu.edu.hk).

G. Zhou and H. Huang are with the Department of Aeronautical and Aviation Engineering and The Research Institute for Sports Science and Technology (RISports), the Hong Kong Polytechnic University, Hong Kong. (E-mail: guanzhong.zhou@connect.polyu.hk, hailong.huang@polyu.edu.hk).

P. Hang is with the Department of Traffic Engineering, Tongji University. (E-mail: hangpeng@tongji.edu.cn).

a multi-drones-assisted commercial parcel delivery system by the proposed enhanced backtracking search algorithm (EBSA). In the considered system, a drone carries a parcel from a depot to a destination parcel station (DPS), which is the closest parcel station of all the parcel stations to the recipient. After the parcel is dropped into the DPS, its recipient can take it away according to the received electronic message. In general, the delivery distance between the depot and the DSP could be much longer than the maximum range of the drone. To assist drones, some SSs are well deployed. Each SS is equipped with some charging piles, which can offer a charging service to drones. In addition, both the depot and the DPS are also equipped with charging piles to make full use of the facilities. In the system, all facilities including depots, SSs, and DPSs form a generalized service network (GSN). In the GSN, each facility can be seen as a node. A drone can fly from one node to another with the help of the GSN. Here, it should be pointed out that the system can support multi-drones to execute the delivery tasks at the same time. However, given the limited number of charging piles at each node, when the number of drones that need to be charged is more than the number of charging piles at one node, some drones must wait. Under the considered system, the goal of this paper is to determine the optimal flights (the least average travel time consumed by all drones during parcel delivery) on the premise of satisfying the constraints of the limited battery capacity of a drone and the limited number of charging piles at each node.

To solve the considered FP problem, this paper uses the priority-based encoding mechanism (PBEM) [15] to assign a priority sequence (PS) (which generates a flight for a drone) for each drone. That is, flights for all drones can be regarded as a set of PSs. This set can be called the total priority sequence (TPS). In addition, the lengths of PS and TPS are equal to the number of nodes in the GSN and the production of the number of nodes in the GSN and the number of drones (a drone corresponds to a task in the system), respectively. Thus, determining the optimal flights for all drones is to find the optimal TPS from all feasible TPSs. Note that, the length of TPS could be far more than 100 in this system. Given this, the considered FP problem can be seen as a large-scale optimization problem (LSOP) [16]. To solve such an LSOP, EBSA is proposed, which is an improved version of the backtracking searching algorithm (BSA) [17]. BSA is a simple but efficient population-based metaheuristic algorithm. However, it is easy to be trapped into local optimal solutions in solving challenging LSOPs due to the single learning strategy and the weak ability to escape from a local optimum. Motivated by the characteristics of the considered FP problem, the LSOP and BSA, a comprehensive learning mechanism consisting of three learning strategies and a local escape operator are introduced to EBSA. The main contributions of this paper are as follows.

1) This paper presents a multi-drones-assisted commercial parcel delivery system. Unlike the previous research, this system can support simultaneously multi-drones to carry out long-distance parcel delivery, which consists of four core components, i.e., the depot, the drone, the SS, and the DPS. The depot is used to store parcels that need to be delivered. The drone is a parcel delivery tool, which can fly from a depot to a DPS. The DPS is for the temporary storage of parcels, where customers can pick up their parcels. In addition, the depots, SSs, and DPSs are equipped with charging piles for providing charging services for drones, which can be regarded as nodes that form a GSN that can help multi-drones to fly from the depot to their DPSs at the same time.
2) This paper presents a new population-based metaheuristic algorithm, namely EBSA, to determine the optimal flights for drones. As mentioned previously, determining the optimal flights for drones is to find the optimal TPS from all feasible TPSs. Each individual in EBSA denotes a TPS. EBSA is an improved version of BSA, which has two core components that are employed to enhance the global search ability on LSOPs. On the one hand, a comprehensive learning mechanism consisting of three learning strategies is built in EBSA. The first learning strategy is the same as that of BSA; the second learning strategy is based on randomly exchanging PSs between an individual and the obtained best solution; the third learning strategy is driven by the introduced weighted mean position. On the other hand, a local escape operator based on random numbers generated by the standard normal distribution is designed to enhance the ability of EBSA to escape from a local optimum.
3) This paper simulates a realistic multi-drones-assisted commercial parcel delivery scenario based on the geographic information of Hong Kong In this scenario, the GSN consists of 46 nodes, i.e., a depot, 29 SSs, and 16 DPSs, and each node is equipped with a limited number of charging piles. In addition, to be as close as possible to the real parcel delivery situation, this paper designs a grouping delivery mechanism. Following this mechanism, the required delivery tasks are divided into three groups, which cover all DPSs and are executed at regular intervals. Besides, six evaluation indicators including average delivery efficiency, average flight efficiency, average wait efficiency, average charging efficiency, convergence performance, and computational efficiency are employed to evaluate the delivery performance of EBSA and the compared algorithms.

The rest of this paper is divided into five sections. Related work is introduced in Section II. Section III presents problem formulation. Section IV describes the proposed EBSA. Experimental results are discussed in Section V. Section VI concludes this paper and brings forward future research.

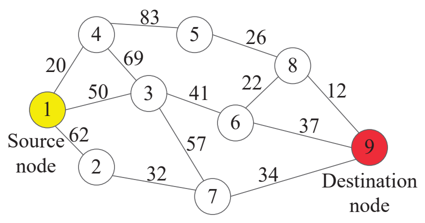


Fig. 1. A random network with 9 nodes. The weights of the connecting edges are shown adjacent to the corresponding edges (as an example, the weights are assigned randomly).

NOMENCLATURE

Symbol	Meaning
$\boldsymbol{X}_{\text{h}}$	Historical population
$\boldsymbol{X}$	Population
η	The penalty factor
θ	The connected factor
ξ	The number of independent runs
D_{c}	The connected distance
E	The residual energy of a battery
$E_{\text{c},k,i}$	The current power of drone k at node i
E_{l}	The consumed energy of a drone during landing
E_{p}	The penalty energy
$E_{\text{r},k,i}$	The required power of drone k at node i
E_{t}	The consumed energy of a drone during take-off
l_k	The number of nodes in the flight planning of drone k
L_{max}	The maximum range of a drone
m	The total number of nodes
n	The number of drones
n_{c}	The number of charging piles
n_{d}	The number of destination parcel stations
n_{p}	The number of depots
n_{s}	The number of service stations
S_{n}	The sequence of numbers of all nodes
t	The charging time
$t_{\text{d},k}$	The travel time of drone k during parcel delivery
t_{l}	The consumed time of a drone during landing
$t_{\text{tc},k}$	The total charging time of drone k during parcel delivery
$t_{\text{tf},k}$	The consumed time of drone k during the normal flight
$t_{\text{tl},k}$	The consumed time of drone k during take-off and landing
$t_{\text{tw},k}$	The total wait time of drone k during parcel delivery
t_{t}	The consumed time of a drone during take-off
v	The speed of a drone during the normal flight
$\boldsymbol{\lambda}_k$	The numbers of nodes in the flight planning of drone k
$\boldsymbol{G}_{\text{GSN}}$	The connected graph of the generalized service network
$\boldsymbol{I}_{\text{task}}$	The task information
$\boldsymbol{O}_{\text{FP}}$	The optimal flight planning
$\boldsymbol{P}_{\text{drone}}$	The parameters of a drone
$\boldsymbol{x}_{*}$	The best solution
$\boldsymbol{x}_{i,\text{FP}}$	The corresponding flight planning of individual i
C_E	Charging efficiency
$E_{\text{rp},\lambda_{k,q}}$	The real penalty power of drone k at node $\lambda_{k,q}$
E_0	The total power of a drone
F_E	Flight efficiency
N	Population size
P_0	The rotated power of a drone
$S_{i,j}$	The occupied time period of charging pile j at node i
W_E	Wait efficiency
$d_{a,b}$	The Euclidean distance between node a and node b
$t_{\text{a},k,i}$	The time of drone k arriving at node i
$t_{\text{e},i,j}$	The occupied end time of charging pile j at node i
$t_{\text{s},i,j}$	The occupied start time of charging pile j at node i
$t_{\text{w},k,i,j}$	The wait time of drone k at jth charging pile of node i
$t_{\text{w},k,i}$	The wait time of drone k at node i

II. RELATED WORK

Over the past three decades, population-based metaheuristic algorithms have developed greatly. A lot of population-based metaheuristic algorithms have been applied to practical engineering problems, such as the traffic signal optimization problem [18], the electric vehicle charge scheduling problem [19], and the sustainable ship routing and bunker management problem [20]. In addition, some researchers have made some attempts to use population-based metaheuristic algorithms to solve the FP problem of unmanned aerial vehicles (UAVs) (a type of drones). Yu et al. [21] propose a hybrid particle swarm optimizer by merging the simulated annealing algorithm to solve the FP problem of UAVs in complex three-dimensional environments. Yu et al. [22] design a constrained differential evolution algorithm to solve the FP problem of a UAV in disaster scenarios. Qu et al. [23] present a novel reinforcement learning-based grey wolf optimizer algorithm to solve the FP problem of UAVs in complicated environments with multiple obstacles. Pehlivanoglu et al. [24] employ an enhanced genetic algorithm to solve the FP problem of autonomous UAV in target converge problems. Phung and Ha [25] use a spherical vector-based particle swarm optimization to solve the safety-enhanced FP problem of a UAV. According to the experimental results, these algorithms show excellent performance in determining the FP of the considered scenarios. However, they pay attention to searching the optimal FP that can avoid the given obstacles, which do not cover the characteristics of the considered FP problem in this paper.

Specifically, the characteristics of the considered FP problem can be summarized as:

1) Problem properties: From the perspective of problem properties, its characteristics can be stated as:

- Multi-hop path. In a real parcel delivery situation, the distance between the depot and the DPS is usually much longer than the maximum range of a drone. The drone must return to the ground serial times to add its energy to complete the delivery task. Thus, the flight route of the drone can be seen as a multi-hop path.
- Multi-drones-to-multi-tasks. The delivery mode of multi-drones to multi-tasks is an essential feature of the modern parcel delivery system based on the drone-station model, which can significantly improve delivery efficiency and effectively reduce inventory pressure.
- Resource sharing. Charging services are the fundamental guarantee of supporting the drone to perform the long-distance delivery task, which are shared by all drones used to perform the delivery tasks.
- Self-performance of a drone. The self-performance of a drone includes the maximum range, the maximum battery capacity, speed, take-off and landing energy consumption, charging characteristics, and the rated power, which has a significant impact on its FP.

2) Problem solving: As mentioned in Section I, the FP problem can be converted into an LSOP by TPS, which is generated by PBEM. PBEM is a common encoding approach to constructing the path between the source node and the destination node in a network topology. Its basic idea is as follows [15]: 1) each node is assigned a random priority value, 2) the larger the priority value, the higher the priority, 3) once a node is selected into the constructed path, the priority value of this node is given a large negative value to make this node no longer selected, and 4) the node with the largest priority value among all nodes connected to the terminal node of the path is selected as a new node into the path. Next, a simple example is used to illustrate the steps of constructing the path by the PBEM. Fig. 1 shows a 9-node random network. Node 1 and node 9 are the source node and destination node in Fig. 1, respectively. Let a random priority sequence $S_{\text{p}} = [5.66, 7.31, 6.71, 7.45, 2.38, 7.24, 3.98, 0.84, 8.24]$ corresponds to the set of priority values from node 1 to node 9. According to S_{p}, the achieved path (which is denoted by node numbers) is $1 \rightarrow 4 \rightarrow 3 \rightarrow 6 \rightarrow 9$. In addition, many algorithms will suffer from the curse of dimension in solving LSOPs, which can be

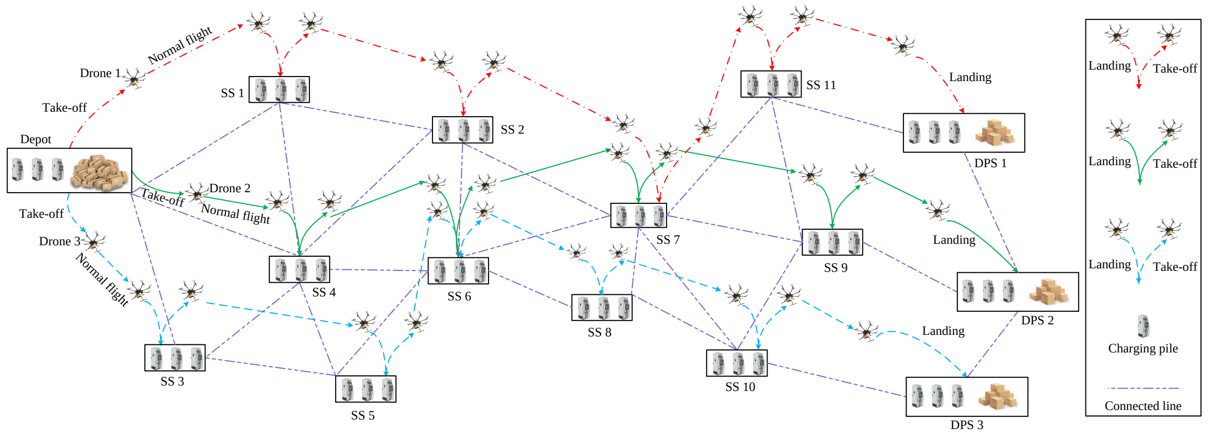


Fig. 2. An example illustrating the built multi-drones-assisted commercial parcel delivery system (“connected line” means that the connected two facilities are reachable).

described as follows. With the increase of the dimension, the number of feasible solutions rises exponentially.

Thus, although the reported algorithms cannot be directly applied to the considered FP problem, the population-based metaheuristic algorithm can get a promising solution to the FP problem of the UAV based on the experimental results presented in these references. Motivated by this, to solve the considered FP problem, this paper designs a new population-based metaheuristic algorithm, namely EBSA, based on the characteristics of the considered FP problem.

III. Problem Formulation

This section first presents the built multi-drones-assisted commercial parcel delivery system. Then, the mathematical model of the considered FP problem is described.

A. *The multi-drones-aided commercial parcel delivery system*

Fig. 2 shows an example of the multi-drones-assisted commercial parcel delivery system. The system contains a depot (to store parcels), three drones (which carry out the delivery tasks from the depot to three DPSs), three DPSs (which are temporary storage areas for customers to pick up their parcels), and 11 SSs (each SS is equipped with a limited number of charging piles). To make full use of the system, the depot and three DPSs are also equipped with a limited number of charging piles. A facility (e.g., a depot, a DPS, or a SS) is regarded as a node. All facilities including a depot, three DPSs, and 11 SSs form a GSN with 15 nodes. Benefiting from the GSN, a drone can perform a long-distance delivery task.

We take drone 1 as an example to describe the operation of the system. Its FP is divided into three stages. Drone 1 takes off from the depot in the first stage. After reaching a given height, it turns into the normal flight state with a constant speed. In the second stage, it flies from SS 1 to SS 11 by charging its battery at SS 1, SS 2, SS 7, and SS 11. In the third stage, it takes off from SS 11 and then lands at DPS 1 after performing the normal flight state. As shown in Fig. 2, SS 7 also offers charging service for drone 2. That is, drone 2 is a potential influencing factor for the FP of drone 1 due to the limited number of charging piles at SS 7. In addition, in

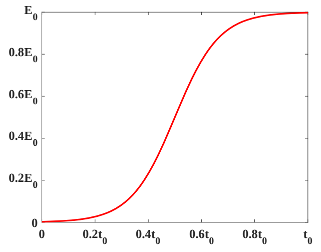


Fig. 3. Charging curve of the battery of a drone (t_0 is the time to fully charge the battery).

Fig. 2, only three drones are considered. In a real scenario, the number of drones is far more than three, which means more drones may perform the delivery tasks via SS 7. Clearly, the FP of drone 1 will bear the greater pressure. The challenge faced by drone 1 also applies to other drones.

B. *Mathematical model of the FP problem*

This section describes the mathematical model of the considered FP problem. Firstly, five assumptions are made:

- All drones used to execute the delivery tasks are exactly the same and their batteries are fully charged at the depot. A drone performs an independent task, which does not collide with other obstacles during the flight.
- A drone has three flight states, i.e., take-off, landing, and normal flight. The drone takes off and lands vertically above the take-off point and landing point, respectively. The drone flies from one node to another node in a straight line at a constant speed and altitude.
- The influencing factors on the energy consumption of a drone include take-off, landing, charging for its battery, and normal flight.
- The influencing factors on the running time of a drone include take-off, landing, normal flight, charging for its battery, and wait.
- The influence factor of the parcel weight on drone performance is not considered.

From Fig. 2, the mathematical model of the considered FP problem is mainly related to the following four parts:

1) Charging function: Each node in the GSN can offer a charging service for drones. At present, a drone is typically equipped with series-connected lithium-ion polymer battery cells [26]. The charging speed for a drone generally follows the "slow-fast-slow" rule during the charging process [27]. Without loss of generality, a charging function is defined by:

$$E = \frac{E_0}{1 + \exp(6 - t/5)}, \tag{1}$$

where t is charging time, E_0 is the total energy capacity of a battery, E is the residual energy of a battery. Fig. 3 shows the charging curve based on (1). To compute the charging time, (1) is rewritten by:

$$t = 30 - 5\ln(\frac{E_0}{E} - 1). \tag{2}$$

Let $E_{\mathrm{c},k,i}$ and $E_{\mathrm{r},k,i}$ be the current power at node i and the required power at node i of drone k, respectively. According to (2), the charging time can be computed by:

$$t_{\mathrm{c},k,i} = 5\ln(\frac{E_0}{E_{\mathrm{c},k,i}} - 1) - 5\ln(\frac{E_0}{E_{\mathrm{r},k,i}} - 1), k \in [1, n], \tag{3}$$

where n is the number of drones and $t_{\mathrm{c},k,i}$ is the charging time of drone k at node i.

2) Connected graph: To describe the positional relations among nodes in the GSN, each node is numbered. Let n_{p}, n_{s}, n_{d}, and S_{n} denote the number of depots, the number of SSs, the number of DPSs, and the sequence of numbers of all nodes, respectively. S_{n} can be represented by:

$$\begin{aligned} S_{\mathrm{n}} = [&1, 2, \ldots, n_{\mathrm{p}}, n_{\mathrm{p}}+1, n_{\mathrm{p}}+2, \ldots, n_{\mathrm{p}}+n_{\mathrm{s}}, \\ &n_{\mathrm{p}}+n_{\mathrm{s}}+1, n_{\mathrm{p}}+n_{\mathrm{s}}+2, \ldots, n_{\mathrm{p}}+n_{\mathrm{s}}+n_{\mathrm{d}}]. \end{aligned} \tag{4}$$

According to (4), the node numbers for depots are from 1 to n_{p}; the node numbers for SSs are from $n_{\mathrm{p}}+1$ to $n_{\mathrm{p}}+n_{\mathrm{s}}$; and the node numbers for DPSs are from $n_{\mathrm{p}}+n_{\mathrm{s}}+1$ to $n_{\mathrm{p}}+n_{\mathrm{s}}+n_{\mathrm{d}}$. That is, each node has its own number. Let m denote the total number of nodes in the GSN, and m meet $m = n_{\mathrm{p}}+n_{\mathrm{s}}+n_{\mathrm{d}}$. To measure the positional relation between two nodes in the GSN, the connected distance D_{c} is defined by:

$$D_{\mathrm{c}} = (1 - \theta) \times L_{\max}, \tag{5}$$

where θ is called a connected factor and $L_{\max}$ is the maximum range of the drone. If the distance between two nodes is less than D_{c}, the two nodes are connected (without recharging the battery, a drone can fly between the two nodes); otherwise, the two nodes are not connected (without recharging the battery, a drone cannot fly between the two nodes). In this paper, θ is set to 0.1. Based on (4) and (5), the positional relations among nodes in the GSN can be described by a connected graph $\boldsymbol{G}_{\mathrm{GSN}}$ based on a 0-1 matrix with m rows and m columns. In the $\boldsymbol{G}_{\mathrm{GSN}}$, "0" means that two nodes are not connected; "1" means that two nodes are connected.

3) Status chart of charging piles: In the GSN, each node is equipped with a limited number of charging piles to provide charging services to drones. Let n_{c} denote the number of charging piles at one node. The rule of using charging piles at one node can be described as follows. All drones that need to be charged line up in the order of arrival. When the number of drones is more than n_{c}, some drones must wait until idle charging piles appear. To describe the status of charging piles, a status chart of charging piles is designed by:

Algorithm 1 The computing method of $t_{\mathrm{w},k,i}$ and $S_{i,j}$

```
Input:
    $S_{i,j}$, $t_{\mathrm{a},k,i}$, $t_{\mathrm{w},k,i}$, $t_{\mathrm{w},k,i,j}$
Output:
    $t_{\mathrm{w},k,i}$, $S_{i,j}$
 1: Initialize the flag bit $f_{\mathrm{b}}$ by $f_{\mathrm{b}} = 0$; // $f_{\mathrm{b}}$ is to determine whether to wait
 2: Initialize $t_{\mathrm{w},k,i,j}$ by $t_{\mathrm{w},k,i,j} = 0$;
 3: for $j = 1$ to $n_{\mathrm{c}}$ do
 4:     if $t_{\mathrm{a},k,i} > t_{\mathrm{e},i,j}$ then
 5:         $f_{\mathrm{b}} = 1$; // there is no need to wait
 6:         $t_{\mathrm{w},k,i} = 0$; // the wait time is 0
 7:         $t_{\mathrm{s},i,j}$=$t_{\mathrm{a},k,i}$; // update the status of charging pile $j$
 8:         $t_{\mathrm{e},i,j} = t_{\mathrm{a},k,i} + t_{\mathrm{c},k,i}$; // update the status of charging pile $j$
 9:         Break;
10:     else
11:         $t_{\mathrm{w},k,i,j} = t_{\mathrm{e},i,j} - t_{\mathrm{a},k,i}$; // compute the wait time
12:     end if
13: end for
14: if $f_{\mathrm{b}} == 0$ then
15:     Find the charging pile $j^*$ with the minimum wait time;
16:     $t_{\mathrm{w},k,i} = t_{\mathrm{w},k,i,j^*}$; // compute the minimum wait time
17:     $t_{\mathrm{s},i,j^*} = t_{\mathrm{a},k,i}$; // update the start time that charging pile $j^*$ is
        occupied
18:     $t_{\mathrm{e},i,j^*} = t_{\mathrm{a},k,i} + t_{\mathrm{c},k,i} + t_{\mathrm{w},k,i}$. // update the end time that charging
        pile $j^*$ is occupied
19: end if
```

$$S_{i,j} = [t_{\mathrm{s},i,j}\ t_{\mathrm{e},i,j}], i \in [1, n_{\mathrm{s}}+n_{\mathrm{d}}+n_{\mathrm{p}}], j \in [1, n_{\mathrm{c}}], \tag{6}$$

where $S_{i,j}$ is the time period that charging pile j at node i is occupied, $t_{\mathrm{s},i,j}$ is the start time that charging pile j at node i is occupied, and $t_{\mathrm{e},i,j}$ is the end time that charging pile j at node i is occupied. Let $t_{\mathrm{a},k,i}$ denote the time of drone k arriving at node i, $t_{\mathrm{w},k,i}$ denote the wait time of drone k at node i, and $t_{\mathrm{w},k,i,j}$ denote the wait time of drone k at the jth charging pile of node i. The computing method of $t_{\mathrm{w},k,i}$ and $S_{i,j}$ can be found in Algorithm 1.

4) Objective function: According to (4), the FP of drone k can be denoted by a sequence of nodes in the GSN. Let $\boldsymbol{\lambda}_k$ denote the node numbers of the sequence corresponding to the FP of drone k and l_k denote the number of nodes in the $\boldsymbol{\lambda}_k$. $\boldsymbol{\lambda}_k$ is expressed by:

$$\boldsymbol{\lambda}_k = [\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,l_k}], l_k \geq 3, \tag{7}$$

$$\lambda_{k,p} \in S_{\mathrm{n}}, p \in [1, l_k]. \tag{8}$$

In (7), l_k is no less than 3. This ensures that at least one node exists between the start node (a depot) and the end node (one DPS). Based on (7) and (8), Fig. 4 presents an example of the flight of drone k. In Fig. 4, the travel time of drone k in the delivery process consists of the following four parts:

- The consumed time by drone k on take-off and landing. From Fig. 4, the number of take-off and landing of drone k is $l_k - 1$ and $l_k - 1$, respectively. Let t_{t} and t_{l} denote the consumed time during take-off and landing, respectively. Let $t_{\mathrm{tl},k}$ denote the consumed time by drone k on take-off and landing. $t_{\mathrm{tl},k}$ can be computed by:

$$t_{\mathrm{tl},k} = (t_{\mathrm{t}} + t_{\mathrm{l}}) \times (l_k - 1). \tag{9}$$

- The consumed by drone k on the normal flight. The constant speed for drone k on the normal flight is set to

v. Let $t_{\mathrm{tf},k}$ denote the consumed time by drone k during the normal flight. $t_{\mathrm{tf},k}$ can be expressed by:

$$t_{\mathrm{tf},k} = \sum_{m=1}^{l_k-1} \frac{d_{\lambda_{k,m},\lambda_{k,m+1}}}{v}, \tag{10}$$

where $d_{\lambda_{k,m},\lambda_{k,m+1}}$ is the Euclidean distance between node $\lambda_{k,m}$ and node $\lambda_{k,m+1}$.

- The consumed time by drone k on charging. According to (3), $t_{\mathrm{c},k,\lambda_{k,q}}$ is related to $E_{\mathrm{c},k,\lambda_{k,q}}$ and $E_{\mathrm{r},k,\lambda_{k,q}}$, where q is an integer meeting $q \in (1, l_k)$. Next, the computing methods of $E_{\mathrm{c},k,\lambda_{k,q}}$ and $E_{\mathrm{r},k,\lambda_{k,q}}$ are introduced. In this paper, to improve the delivery efficiency of drone k, the battery of drone k is charged to $E_{\mathrm{r},k,\lambda_{k,q}}$ rather than E_0 at node $\lambda_{k,q}$. $E_{\mathrm{r},k,\lambda_{k,q}}$ is the minimum power that allows drone k to fly from node $\lambda_{k,q}$ to node $\lambda_{k,q+1}$. Thus, $E_{\mathrm{r},k,\lambda_{k,q}}$ can be computed by:

$$E_{\mathrm{r},k,\lambda_{k,q}} = \frac{P_0 \times d_{\lambda_{k,q},\lambda_{k,q+1}}}{v} + E_{\mathrm{t}} + E_{\mathrm{l}}, \tag{11}$$

where E_{l} is the consumed energy during landing, E_{t} is the consumed energy during take-off, and P_0 is the rated power of drone k. From (11), when drone k lands at node $\lambda_{k,q+1}$, $E_{\mathrm{c},k,\lambda_{k,q+1}}$ is equal to 0. This is extremely harmful to the battery. To avoid this, (11) is rewritten by:

$$E_{\mathrm{r},k,\lambda_{k,q}} = \frac{\mathrm{P}_0 \times d_{\lambda_{k,q},\lambda_{k,q+1}}}{v} + E_{\mathrm{t}} + E_{\mathrm{l}} + E_{\mathrm{p}}, \tag{12}$$

where E_{p} is the penalty energy that can be achieved by:

$$E_{\mathrm{p}} = \eta \times E_0, \tag{13}$$

where η is called the penalty factor that is set to 0.1 in this paper. Note that, $E_{\mathrm{c},k,\lambda_{k,q+1}}$ is much larger than 0 due to the introduction of E_{p}. However, $E_{\mathrm{r},k,\lambda_{k,q}}$ achieved by (12) may be more than E_0 due to the introduction of E_{p}. Thus, $E_{\mathrm{r},k,\lambda_{k,q}}$ needs to be further addressed by:

$$E_{\mathrm{r},k,\lambda_{k,q}} = \begin{cases} E_0, \text{ if } E_{\mathrm{r},k,\lambda_{k,q}} > E_0\ ; \\ E_{\mathrm{r},k,\lambda_{k,q}}, \text{if } E_{\mathrm{r},k,\lambda_{k,q}} \le E_0. \end{cases} \tag{14}$$

For the convenience of describing the computing method of $E_{\mathrm{c},k,\lambda_{k,q}}$, the real penalty power is introduced, which can be represented by:

$$E_{\mathrm{rp},\lambda_{k,q}} = \begin{cases} E_0 - E_{\mathrm{r},k,\lambda_{k,q}}, \text{ if } E_{\mathrm{r},k,\lambda_{k,q}} > \mathrm{E}_0\ ; \\ E_{\mathrm{p}}, \text{if } E_{\mathrm{r},k,\lambda_{k,q}} \le E_0, \end{cases} \tag{15}$$

where $E_{\mathrm{rp},\lambda_{k,q}}$ is the real penalty power of drone k at node $\lambda_{k,q}$. According to (15), $E_{\mathrm{c},k,\lambda_{k,q}}$ is obtained by:

$$E_{\mathrm{c},k,\lambda_{k,q}} = \begin{cases} E_0 - \frac{P_0 \times d_{\lambda_{k,1},\lambda_{k,2}}}{v} - E_{\mathrm{t}} - E_{\mathrm{l}}, \text{ if } q = 2; \\ E_{\mathrm{rp},\lambda_{k,q}}, \text{if } q \in (2, l_k). \end{cases} \tag{16}$$

Based on (3), (14), and (16), $t_{\mathrm{c},k,\lambda_{k,q}}$ can be written by:

$$t_{\mathrm{c},k,\lambda_{k,q}} = 5\ln\left(\frac{E_0}{E_{\mathrm{c},k,\lambda_{k,q}}} - 1\right) - 5\ln\left(\frac{E_0}{E_{\mathrm{r},k,\lambda_{k,q}}} - 1\right). \tag{17}$$

Let $t_{\mathrm{tc},k}$ denote the total charging time of drone k during parcel delivery. $t_{\mathrm{tc},k}$ can be denoted by:

$$t_{\mathrm{tc},k} = \sum_{q=2}^{l_k-1} t_{\mathrm{c},k,\lambda_{k,q}}. \tag{18}$$

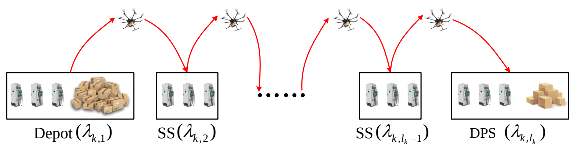


Fig. 4. The FP of drone k from a depot to a DPS.

- The consumed time by drone k on wait. Let $t_{\mathrm{tw},k}$ denote the total wait time of drone k during parcel delivery. From Algorithm 1, $t_{\mathrm{tw},k}$ can be computed by:

$$t_{\mathrm{tw},k} = \sum_{q=2}^{l_k-1} t_{\mathrm{w},k,\lambda_{k,q}}. \tag{19}$$

Let $t_{\mathrm{d},k}$ denote the travel time of drone k during parcel delivery. $t_{\mathrm{d},k}$ can be expressed by:

$$t_{\mathrm{d},k} = t_{\mathrm{tl},k} + t_{\mathrm{tf},k} + t_{\mathrm{tc},k} + t_{\mathrm{tw},k}. \tag{20}$$

Let T_{att} denote the average travel time of all drones during parcel delivery. Solving the considered FP problem is to get the optimal T_{att}. Therefore, the objective function of the considered FP problem can be defined by:

$$\min T_{\mathrm{att}} = f(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_n) = \frac{1}{n}\sum_{k=1}^{n} t_{\mathrm{d},k}. \tag{21}$$

subject to (5), (6), (7), (8), (9), (10), (18), (19), (20), and

$$\boldsymbol{\lambda}_k \in S_{\mathrm{n}}, k = 1, 2, \ldots, n. \tag{22}$$

As shown in (21), the most remarkable characteristic of the designed system is that it can support multi-drones to execute long-distance parcel delivery at the same time.

IV. The Proposed Algorithm

EBSA is an improved version of BSA. This section first introduces BSA and then describes the proposed EBSA.

A. BSA

BSA is a very popular population-based metaheuristic algorithm and its structure consists of four parts [17]:

1) Selection-I: In BSA, the historical population $\boldsymbol{X}_{\mathrm{h}}(\boldsymbol{X}_{\mathrm{h}} = \{\boldsymbol{x}_{\mathrm{h},1}, \boldsymbol{x}_{\mathrm{h},2}, \ldots, \boldsymbol{x}_{\mathrm{h},N}\})$ is used to guide the search direction of the current population $\boldsymbol{X}(\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\})$, where N is the population size. $\boldsymbol{X}_{\mathrm{h}}$ is first obtained by:

$$\boldsymbol{X}_{\mathrm{h}} = \begin{cases} \boldsymbol{X}, \text{ if } \phi_1 > \phi_2; \\ \boldsymbol{X}_{\mathrm{h}}, \text{otherwise}, \end{cases} \tag{23}$$

where ϕ_1 and ϕ_2 are two random numbers in [0,1]. Then, $\varphi(\cdot)$ is used to sort individuals in the $\boldsymbol{X}_{\mathrm{h}}$ randomly, which can be described by:

$$\boldsymbol{X}_{\mathrm{h}} = \varphi\left(\boldsymbol{X}_{\mathrm{h}}\right), \tag{24}$$

2) Mutation operator: This operator is to compute the trail individual of individual i, which can be expressed by:

$$\boldsymbol{z}_i = \boldsymbol{x}_i + F \times \left(\boldsymbol{x}_{\mathrm{h},i} - \boldsymbol{x}_i\right), i \in [1, N], \tag{25}$$

where F is scale factor and $\boldsymbol{z}_i$ is the trail individual of $\boldsymbol{x}_i$. F is computed by $3 \times \alpha$, where α is a random number with standard normal distribution

3) Crossover operator: The crossover operator is further processed for $\boldsymbol{z}_i$ by a binary integer-valued matrix $\boldsymbol{G}$ with N rows and D columns, where D is the dimension of the problem. The method of generating $\boldsymbol{G}$ is based on the mix rate [17]. This operator can be denoted by:

$$z_{i,j} = \begin{cases} x_{i,j}, & \text{if } g_{i,j}=1; \\ z_{i,j}, & \text{if } g_{i,j}=0, \end{cases} \tag{26}$$

where $x_{i,j}$ is the jth element of $\boldsymbol{x}_i$, $z_{i,j}$ is the jth element of $\boldsymbol{z}_i$, and $g_{i,j}$ is the jth element of the ith row in the $\boldsymbol{G}$.

4) Selection-II: This operator is to accelerate the convergence speed of BSA, which can be represented by:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{x}_i, & \text{if } f(\boldsymbol{x}_i)<f(\boldsymbol{z}_i), \\ \boldsymbol{z}_i, & \text{if } f(\boldsymbol{x}_i)\geq f(\boldsymbol{z}_i). \end{cases} \tag{27}$$

Algorithm 2 The flowchart of BSA.

```
Input:
    Population size $N$, the current number of iterations $N_{\mathrm{c}}(N_{\mathrm{c}}=0)$, the maximum number of iterations $N_{\mathrm{m}}$, and the mix rate $M_{\mathrm{rate}}$
Output:
    The optimal solution $\boldsymbol{x}^*$
1: Initialize $\boldsymbol{X}$ and $\boldsymbol{X}_{\mathrm{h}}$;
2: Evaluate $\boldsymbol{X}$, find $\boldsymbol{x}^*$, and update $N_{\mathrm{c}}$ by $N_{\mathrm{c}} = N_{\mathrm{c}} + 1$;
3: while $N_{\mathrm{c}} < N_{\mathrm{m}}$ do
4:     Update $\boldsymbol{X}_{\mathrm{h}}$ by (23-24); // Selection-I
5:     Generate the binary integer-valued matrix $\boldsymbol{G}$;
6:     for $i = 1$ to $N$ do
7:         Execute mutation operator for $\boldsymbol{x}_i$ by (25); // Mutation operator
8:         Execute crossover operator for $\boldsymbol{x}_i$ by (26); // Crossover operator
9:         Generate the next generation of for $\boldsymbol{x}_i$ by (27); // Selection-II
10:    end for
11:    Update $\boldsymbol{x}^*$ and $N_{\mathrm{c}}$ by $N_{\mathrm{c}} = N_{\mathrm{c}} + 1$. // Update the optimal solution
12: end while
```

B. EBSA

Algorithm 2 shows the flowchart of BSA. From Algorithm 2 the disadvantages of BSA are as follows:

- Single learning strategy. BSA only has a learning strategy, which is not helpful for keeping population diversity.
- Unclear search direction. In BSA, the search direction of the population is guided by the selected randomly historical population, which cannot help to accelerate the convergence speed.
- Underutilized the population information. BSA only employs historical information, which does not consider other valuable population information.
- No local escape operator. BSA is easy to converge to the local optima in solving LSOPs due to the lack of a local escape operator.

Motivated by these disadvantages, EBSA is reported to improve the performance of BSA on the considered FP problem. In the proposed EBSA, each individual denotes a TPS of n drones. Let $\boldsymbol{x}_{\mathrm{p},k}$ denote the PS of drone k. The number of elements in each PS is equal to the number of nodes in the GSN. Thus, $\boldsymbol{x}_i$ can be expressed by:

$$\boldsymbol{x}_i = \{\boldsymbol{x}_{\mathrm{p},1}, \boldsymbol{x}_{\mathrm{p},2}, \dots, \boldsymbol{x}_{\mathrm{p},n}\}, i \in [1, N], \tag{28}$$

where $\boldsymbol{x}_{\mathrm{p},k}$ is module k of $\boldsymbol{x}_i$. That is, $\boldsymbol{x}_i$ has n modules and each module includes m elements. The obtained FP based on $\boldsymbol{x}_i$ is denoted by $\boldsymbol{x}_{\mathrm{FP},i}$. EBSA includes two core components:

1) Comprehensive learning mechanism: The built comprehensive learning mechanism includes three learning strategies:

- The first learning strategy is the same as that of BSA.
- The second learning strategy aims to randomly exchange information between $\boldsymbol{x}_i$ and the obtained best solution $\boldsymbol{x}^*$. $\boldsymbol{x}^*$ contains some valuable information for finding the global optimal solution. Motivated by this, β modules are exchanged between $\boldsymbol{x}^*$ and $\boldsymbol{x}_i$. β is obtained by:

$$\beta = \lceil n \times \phi_3 \rceil, \tag{29}$$

where ϕ_3 is a random number in [0,1].

- The basic idea of the third learning strategy is to make full use of the population information. The mean position is a common indicator to measure the overall information of the population. To better keep the population diversity, the weighted mean position $\boldsymbol{M}_{\mathrm{w}}$ is defined by:

$$\boldsymbol{M}_{\mathrm{w}} = \phi_4 \times (\phi_5 \times \boldsymbol{x}_i + (1-\phi_5) \times \boldsymbol{x}_j) + (1-\phi_4) \times \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i, \tag{30}$$

where j is an integer selected randomly from 1 and N, ϕ_4 is a random numbers in [0,1], and ϕ_5 is 0 or 1. The third learning strategy is described as:

$$\boldsymbol{x}_i = \boldsymbol{x}_i + \phi_6 \times (\boldsymbol{x}^* - \boldsymbol{M}_{\mathrm{w}}), \tag{31}$$

where ϕ_6 is a random number in [0,1].

The three learning strategies have the same importance and are assigned the same selected probability, which is adjusted by a switch probability p_{s}. p_{s} is a random number that follows a uniform distribution between 0 and 1.

2) Local escape operator: The basic idea of the designed local escape operator is that using random numbers from the standard normal distribution replaces randomly some elements of some modules in the $\boldsymbol{x}^*$. Let γ^* denote the number of the selected modules in the $\boldsymbol{x}^*$. γ^* can be represented by:

$$\gamma^* = \lceil n \times \phi_7 \rceil, \tag{32}$$

where ϕ_7 is a random number in [0,1]. The number of the replaced elements in a selected module is denoted by:

$$h_s^* = \lceil m \times \phi_8 \rceil, s \in [1, \gamma^*], \tag{33}$$

where h_s^* is the number of the selected elements of the selected sth module in the $\boldsymbol{x}^*$ and ϕ_8 is a random number in [0,1]. Let $\boldsymbol{x}^*_{\mathrm{e},s,j}$ denote the selected jth element of the selected sth module in the $\boldsymbol{x}^*$. The local escape operator is defined by:

$$\boldsymbol{x}^*_{\mathrm{e},s,j} = \mu \times \boldsymbol{x}^*_{\mathrm{e},s,j}, j \in [1, h_s^*], \tag{34}$$

where μ is a random number with the standard normal distribution that is employed to adjust the amplitude of $\boldsymbol{x}^*_{\mathrm{e},s,j}$. In addition, the obtained new individual by performing the local escape operator is denoted by $\boldsymbol{x}^*_{\mathrm{e}}$, whose corresponding FP is $\boldsymbol{x}^*_{\mathrm{e,FP}}$.

In addition, for the considered FP problem, the upper limit and lower limit of each element in the PS are set to m and $-m$, respectively. $\boldsymbol{X}$ can be initialized by:

$$x_{i,j} = -m + 2 \times m \times \phi_9, i \in [1, N], j \in [1, m \times n], \tag{35}$$

Algorithm 3 The implementation of EBSA on the considered FP problem

```
Input:
    Population size $N$, the current number of iterations $N_c$, the maximum number of iterations $N_m$, the mix rate $M_{rate}$, the connected graph of GSN $\boldsymbol{G}_{GSN}$, the task information $\boldsymbol{L}_{task}$, the number of charging pile $n_c$, and the parameters of a drone $\boldsymbol{P}_{drone}$
Output:
    $\boldsymbol{x}^*$ and the optimal FP $\boldsymbol{O}_{FP}$
1: // Initialization parameters (lines 2 to 4)
2: Compute $m$ and $n$ according to $\boldsymbol{G}_{GSN}$ and $\boldsymbol{L}_{task}$;
3: Initialize $\boldsymbol{X}$ by (35), $\boldsymbol{X}_h$ by (34), $M_{rate}$ by $M_{rate}$=1, and $N_c$ by $N_c$=0;
4: Compute the corresponding FP of each individual in the $\boldsymbol{X}$ based on PBEM, $\boldsymbol{G}_{GSN}$ and $\boldsymbol{L}_{task}$;
5: // Evaluate FP and obtain the optimal individual (line 6)
6: Evaluate each FP based on $\boldsymbol{P}_{drone}$ and (22), and get $\boldsymbol{x}^*$ and $\boldsymbol{O}_{FP}$;
7: Update $N_c$ by $N_c = N_c + 1$;
8: // Main loop (lines 9 to 36)
9: while $N_c < N_m$ do
10:     Generate $\boldsymbol{G}$ by $M_{rate}$ and update $\boldsymbol{X}_h$ by (23-24);
11:     for $i = 1$ to $N$ do
12:         // Perform comprehensive learning mechanism (lines 13 to 21)
13:         Generate $p_s$ ;
14:         if $p_s < 1/3$ then
15:             Perform the first learning strategy to update $\boldsymbol{x}_i$ by (25-27);
16:         else if $p_s < 2/3$ then
17:             Get $\beta$ by (29);
18:             Perform the second learning strategy to update $\boldsymbol{x}_i$ by exchanging $\beta$ modules randomly between $\boldsymbol{x}_i$ and $\boldsymbol{x}^*$;
19:         else
20:             Perform the third learning strategy to update $\boldsymbol{x}_i$ by (30-31);
21:         end if
22:         Compute $\boldsymbol{x}_{FP,i}$ of $\boldsymbol{x}_i$ based on PBEM, $\boldsymbol{G}_{GSN}$ and $\boldsymbol{L}_{task}$;
23:         Check $\boldsymbol{x}_{FP,i}$ and evaluate $\boldsymbol{x}_{FP,i}$ based on $\boldsymbol{P}_{drone}$ and (22);
24:         if $f(\boldsymbol{x}_{FP,i})<f(\boldsymbol{O}_{FP})$ then
25:             $\boldsymbol{O}_{FP} = \boldsymbol{x}_{FP,i}$, $\boldsymbol{x}^* = \boldsymbol{x}_i$;
26:         end if
27:         // Perform local escape operator (lines 28 to 32)
28:         Perform the local escape operator to get $\boldsymbol{x}_e^*$ by (32-34);
29:         Compute $\boldsymbol{x}_{e,FP}^*$ of $\boldsymbol{x}_e^*$ based on PBEM, $\boldsymbol{G}_{GSN}$ and $\boldsymbol{L}_{task}$;
30:         Check $\boldsymbol{x}_{e,FP}^*$ and evaluate $\boldsymbol{x}_{e,FP}^*$ based on $\boldsymbol{P}_{drone}$ and (22);
31:         if $f(\boldsymbol{x}_{e,FP}^*)<f(\boldsymbol{O}_{FP})$ then
32:             $\boldsymbol{O}_{FP} = \boldsymbol{x}_{e,FP}^*$, $\boldsymbol{x}^* = \boldsymbol{x}_e^*$;
33:         end if
34:     end for
35:     Update $N_c$ by $N_c = N_c + 1$.
36: end while
```

where ϕ_9 is a random number in [0,1]. Algorithm 3 shows the implementation of EBSA on the considered FP problem. The source code of EBSA can be loaded from https://github.com/jsuzyy/EBSA. The features of EBSA can be summarized as:

- To overcome the mentioned first disadvantage of BSA, EBSA builds a comprehensive learning mechanism.
- To overcome the mentioned second disadvantage of BSA, EBSA makes full use of the obtained best solution in the second and third learning strategies.
- To overcome the mentioned third disadvantage of BSA, EBSA defines a weighted mean position involved to all individuals to design the third learning strategy.
- To overcome the mentioned fourth disadvantage of BSA, EBSA introduces a local escape operator based on the random numbers with standard normal distribution.

As done in [28], from Algorithm 3, the time complexity of each operation in EBSA mainly includes six parts:

- Initializing $\boldsymbol{X}$ and $\boldsymbol{X}_h$ costs $2 * O(N)$.
- Evaluating $\boldsymbol{X}$ costs $O(N)$.
- Comparison between two values costs $O(3 * N + 1)$.
- Generating $\boldsymbol{G}$ costs $O(N * D)$.

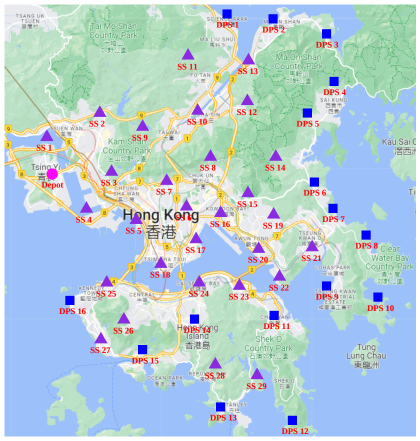


Fig. 5. A real multi-drones-assisted parcel delivery scenario.

- Executing the comprehensive learning mechanism costs $O(N * (D + 1))$ in the worst case.
- Executing the local escape operator costs $O(N * (D + 1))$ in the worst case.

V. Experiments and discussion

A. Experiment preparation

As shown in Fig. 5, this paper simulates a realistic multi-drones-assisted parcel delivery scenario according to the geographic information of Hong Kong. From Fig. 5, n_p, n_s, n_d, and m are equal to 1, 29, 16, and 46, respectively. That is, the GSN consists of 46 nodes (one depot, 29 SSs, and 16 DPSs). Multi-drones can fly from the depot to any DPS at the same time with the help of the GSN. The parameters of drones are extracted from [1], which are summarized as follows: $E_0 = 3.20\times10^5$J, $E_t = 6.25\times10^3$J, $E_l = 6.25\times10^3$J, $t_t = 25$s, $t_l = 25$s, $t_0 = 64$min, $v = 4$m/s, $P_0 = 300$W, and $L_{max} = 4.27$Km.

To be as close as possible to the real parcel delivery situation, a grouping delivery mechanism is designed. Specially, all delivery tasks are divided into three groups that are executed at regular intervals. In addition, as presented in Fig. 5, DPSs are concentrated in the northeast, east, and south, which is the basis of setting delivery tasks. The assigned delivery tasks for each group cover three directions:

- Group 1: Group 1 can be seen as routine tasks and are first performed: drone 1 to drone 16 fly from the depot to DPS 1 to DPS 16, respectively.
- Group 2: Group 2 can be seen as temporary tasks. Half an hour after the start of group 1, group 2 starts: drone 17 to drone 22 fly from the depot to DPS 2, DPS 3, DPS 8, DPS 10, DPS 13, and DPS 15, respectively.
- Group 3: Group 3 also can be seen as temporary tasks. Half an hour after the execution of group 2, group 3 starts: drone 23 to drone 28 fly from the depot to DPS 1, DPS 4, DPS 7, DPS 9, DPS 12, and DPS 14, respectively.

To verify the competitiveness of EBSA, it is compared with BSA, an improved BSA (IBSA) [29], particle swarm optimization (PSO) [30], generalized normal distribution optimization (GNDO) [31], and artificial rabbits optimization (ARO) [32]. Especially, PSO is a very classical and popular population-based metaheuristic algorithm. IBSA is a powerful variant of

TABLE I: The detailed parameters of the utilized system.

Name	Setting
Hardware	
Solid state drive	500G
RAM	16.0G
Frequency	2.30 GHz
CPU	12th Gen Intel(R) Core(TM) i7-12700H
Software	
Language	MATLAB R2017A
Operating system	Windows 11

BSA. GNDO inspired by the normal distribution function and ARO inspired by the survival strategies of rabbits in nature are two novel population-based metaheuristic algorithms. The population size is set to 50 for all the applied algorithms. Note that, as shown in Algorithm 3, each individual in EDDE is evaluated twice in one loop. To make a fair comparison, the maximum number of iterations is set to 500 and 250 for the compared algorithms and EBSA, respectively. In addition, the other control parameters of the compared algorithms are extracted directly from the corresponding references and the detailed parameters of the utilized system are shown in Table I. Besides, n_c is between 1 and 4 at each node and the number of independent runs is set to 30 for each algorithm in each case. Experimental results are shown in Tables (II-V). In these tables, the best results are in bold, and "BEST", "MEAN", and "STD" represent the best value, the mean value, and the standard deviation, respectively.

B. Comparison on the average travel time

According to (21), the designed objective function aims to find the optimal average travel time (average time consumed by all drones during package delivery). To the convenience of description, (21) can be rewritten by:

$$T_{\mathrm{att},i} = \frac{1}{n}\sum_{k=1}^{n} t_{\mathrm{d},k,i} \ , i \in [1, \xi], \tag{36}$$

where $T_{\mathrm{att},i}$ is the average travel time of all drones in the ith run, ξ is the number of independent runs, and $t_{\mathrm{d},k,i}$ is the travel time of drone k in the ith run. From (36), a smaller average travel time means a more efficient delivery efficiency.

Table II presents the statistical results of the average travel time. By observing Table II, EBSA outperforms the compared algorithms in all the considered four cases. This represents that EBSA has better global search ability and stability than the compared algorithms. In addition, ARO and GNDO show strong competitiveness. When n_c is equal to 1, ARO can get the second best BEST; when n_c is larger than 1, GNDO can get the second best BEST. Besides, PSO and BSA are the two worst algorithms in all the considered four cases. In other words, EBSA can determine the better flights for 28 drones than the compared algorithms in each case. From Table II, EBSA can achieve the optimal delivery efficiency on $n_\mathrm{c} = 4$, whose corresponding average travel time is 2.9902 hours. Fig. 6 shows 12 typical flights obtained by EBSA with $n_\mathrm{c} = 4$.

To verify the impact of n_c on delivery efficiency, Fig. 7 is shown based on the data from Table II. Looking at Fig. 7, with the increase of n_c, the optimal delivery efficiency of each algorithm can be improved continuously while the improvement range is significantly decreased. This can be explained as follows. When $n_\mathrm{c} = 1$, charging resources in the GSN is pretty intense and a drone needs to spend a lot of time on wait. Thus, when n_c is increased from 1 to 2, charging resources are obviously relieved and the average travel time of each algorithm shows a sharp decline. With the continuing increase of n_c, the impact of charging resources on determining the flights is getting gradually weaker.

TABLE II: Statistical results of average travel time from 30 independent runs (the unit is hours).

n_c	Indicator	PSO	BSA	ARO	IBSA	GNDO	EBSA
1	BEST	8.6940	8.7891	7.4117	7.9335	7.6650	**4.2621**
	MEAN	10.7535	9.5454	8.6231	8.9100	9.6889	**4.4927**
	STD	1.5111	0.5227	0.5232	0.4534	0.9800	**0.1342**
2	BEST	5.8208	5.5970	5.3706	5.3184	5.0750	**3.3856**
	MEAN	6.5784	6.2060	5.7495	5.8986	6.0580	**3.5256**
	STD	0.4117	0.2677	0.2149	0.2921	0.4577	**0.0933**
3	BEST	4.8287	4.8189	4.5258	4.8063	4.4854	**3.0734**
	MEAN	5.6549	5.3442	4.9833	5.1362	5.1921	**3.2226**
	STD	0.3283	0.2066	0.2034	0.2051	0.3545	**0.0722**
4	BEST	4.7305	4.5632	4.2926	4.5182	4.1825	**2.9902**
	MEAN	5.3708	4.8977	4.6238	4.7795	4.8536	**3.0825**
	STD	0.3158	0.1559	0.1373	0.1367	0.2724	**0.0570**

TABLE III: Statistical results of average flight efficiency from 30 independent runs (the unit is seconds per meter).

n_c	Indicator	PSO	BSA	ARO	IBSA	GNDO	EBSA
1	BEST	1.0387	1.0593	0.9632	0.9724	0.9906	**0.7274**
	MEAN	1.1951	1.1295	1.0615	1.0872	1.1285	**0.7613**
	STD	0.1298	0.0448	0.0530	0.0432	0.0816	**0.0160**
2	BEST	0.6997	0.7158	0.6811	0.6821	0.6756	**0.5902**
	MEAN	0.7490	0.7531	0.7275	0.7334	0.7365	**0.6080**
	STD	0.0281	0.0250	0.0281	0.0256	0.0209	**0.0093**
3	BEST	0.6242	0.6263	0.6213	0.6237	0.6155	**0.5581**
	MEAN	0.6545	0.6535	0.6410	0.6456	0.6416	**0.5698**
	STD	0.0183	0.0151	0.0145	0.0137	0.0150	**0.0081**
4	BEST	0.5966	0.5947	0.5890	0.5881	0.5886	**0.5440**
	MEAN	0.6203	0.6144	0.6070	0.6120	0.6108	**0.5518**
	STD	0.0121	0.0114	0.0100	0.0091	0.0101	**0.0042**

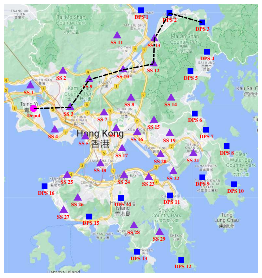
C. Comparison on the average flight efficiency

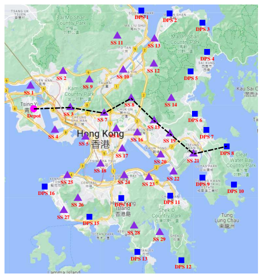
For an ideal flight under the considered scenario, the consumed time on take-off and landing, charging, and wait should be reduced as small as possible. Motivated by this, to measure delivery efficiency, the average flight efficiency is defined by:

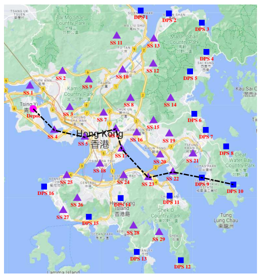
$$F_{\mathrm{E},i} = \frac{1}{n}\sum_{k=1}^{n} t_{\mathrm{d},k,i} / \frac{1}{n}\sum_{k=1}^{n} L_{\mathrm{f},k,i} \ , i \in [1, \xi], \tag{37}$$

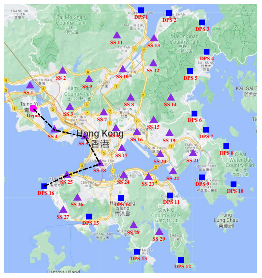
where $F_{\mathrm{E},i}$ is average flight efficiency of all drones in the ith run, $L_{\mathrm{f},k,i}$ is flight length of drone k in the ith run, $t_{\mathrm{d},k,i}$ is travel time of drone k in the ith run. From (37), the average flight efficiency means the required travel time per unit flight length. That is, a smaller average flight efficiency represents a more efficient delivery efficiency.

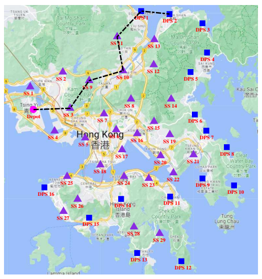
Table III shows the achieved average flight efficiency by EBSA and the compared algorithms. Looking at Table III, in terms of BEST, EBSA is superior to the other algorithms in all considered four cases. That is, EBSA has better flight efficiency than the compared algorithms in each case. In addition, ARO, IBSA, and GNDO have similar BEST in each case. PSO and BSA are the two worst algorithms in terms of the quality

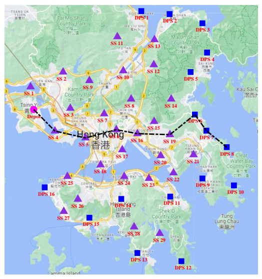

(a) Drone 3 from group 1

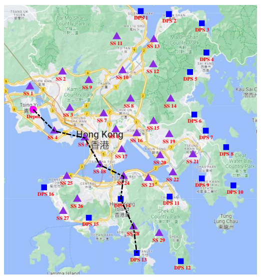

(b) Drone 8 from group 1

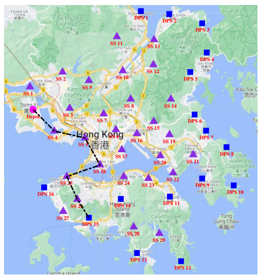

(c) Drone 10 from group 1

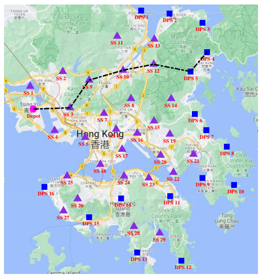

(d) Drone 16 from group 1

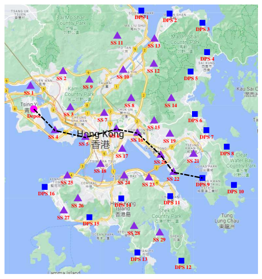

(e) Drone 17 from group 2

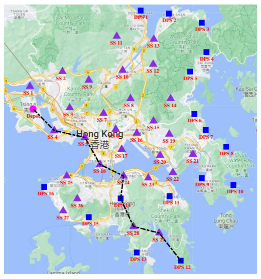

(f) Drone 19 from group 2


(g) Drone 21 from group 2


(h) Drone 22 from group 2


(i) Drone 24 from group 3


(j) Drone 26 from group 3


(k) Drone 27 from group 3

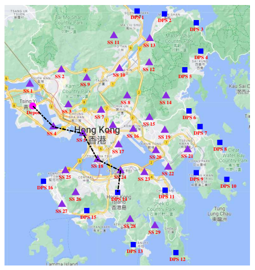

(l) Drone 28 from group 3

Fig. 6. 12 typical flights obtained by EBSA with $n_c = 4$.

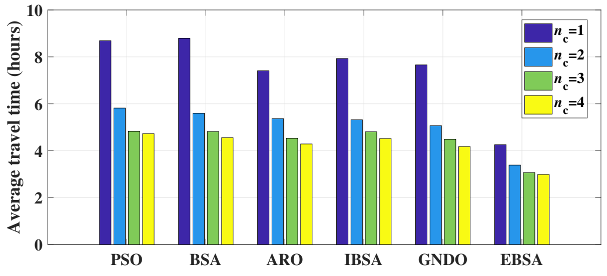


Fig. 7. The statistical results of the optimal average travel time.

TABLE IV: Statistical results of average charging efficiency from 30 independent runs (the unit is seconds per meter).

n_c	Indicator	PSO	BSA	ARO	IBSA	GNDO	EBSA
1	BEST	0.3190	0.3170	0.3142	0.3155	0.3236	**0.2746**
	MEAN	0.3380	0.3306	0.3292	0.3294	0.3353	**0.2831**
	STD	0.0095	**0.0058**	0.0067	0.0080	0.0071	0.0061
2	BEST	0.3153	0.3085	0.3115	0.3097	0.3101	**0.2713**
	MEAN	0.3351	0.3265	0.3235	0.3270	0.3290	**0.2795**
	STD	0.0086	0.0070	0.0063	0.0081	0.0094	**0.0047**
3	BEST	0.3083	0.3103	0.3133	0.3107	0.3126	**0.2653**
	MEAN	0.3332	0.3254	0.3221	0.3234	0.3266	**0.2788**
	STD	0.0087	0.0077	**0.0059**	0.0065	0.0077	0.0068
4	BEST	0.3180	0.3120	0.3058	0.3074	0.3103	**0.2669**
	MEAN	0.3328	0.3222	0.3201	0.3221	0.3247	**0.2733**
	STD	0.0070	0.0056	0.0069	0.0057	0.0071	**0.0033**

of the obtained average flight efficiency. By observing Table III, with the increase of n_c, the obtained optimal average flight efficiency of each algorithm has been falling steadily, which can be explained as follows. The increase of n_c can effectively reduce the wait time of a drone at one node, which can help to reduce the travel time of a drone.

D. *Comparison on the average charging efficiency*

A drone can add its battery capacity by the offered charging service. However, charging also takes some extra time, such as take-off, landing, and wait. To improve delivery efficiency, charging times should be reduced as small as possible. Moti-

TABLE V: Statistical results of average wait efficiency from 30 independent runs (the unit is seconds per meter).

n_c	Indicator	PSO	BSA	ARO	IBSA	GNDO	EBSA
1	BEST	0.4416	0.4589	0.3757	0.3827	0.3874	**0.1827**
	MEAN	0.5906	0.5323	0.4657	0.4912	0.5267	**0.2113**
	STD	0.1255	0.0440	0.0513	0.0429	0.0815	**0.0133**
2	BEST	0.0975	0.1193	0.1009	0.1012	0.0884	**0.0477**
	MEAN	0.1473	0.1599	0.1374	0.1398	0.1410	**0.0616**
	STD	0.0279	0.0243	0.0279	0.0243	0.0183	**0.0082**
3	BEST	0.0354	0.0266	0.0305	0.0343	0.0297	**0.0181**
	MEAN	0.0548	0.0614	0.0522	0.0557	0.0485	**0.0242**
	STD	0.0151	0.0139	0.0123	0.0143	0.0104	**0.0042**
4	BEST	0.0118	0.0086	0.0097	0.0111	0.0071	**0.0064**
	MEAN	0.0209	0.0255	0.0202	0.0233	0.0195	**0.0116**
	STD	0.0077	0.0102	0.0085	0.0075	0.0060	**0.0028**

vated by this, to measure delivery efficiency, average charging efficiency is defined by:

$$C_{\mathrm{E},i} = \frac{1}{n}\sum_{k=1}^{n} t_{\mathrm{tc},k,i} \Big/ \frac{1}{n}\sum_{k=1}^{n} L_{\mathrm{f},k,i}\ , i \in [1, \xi], \tag{38}$$

where $C_{\mathrm{E},i}$ is average charging efficiency of all drones in the ith run and $t_{\mathrm{tc},k,i}$ is charging time of drone k in the ith run. From (38), the average charging efficiency means the required charging time per unit flight length. That is, a smaller average charging efficiency means a more efficient delivery efficiency.

The average charging efficiency obtained by EBSA and the compared algorithms is shown in Table IV. By observing Table IV, according to BEST and MEAN, EBSA can get the best charging efficiency in the considered four cases. In addition, in terms of STD, EBSA can not compete with BSA, ARO, and IBSA on $n_c = 1$, $n_c = 3$, and $n_c = 3$, respectively. However, EBSA is the best of all algorithms on $n_c = 2$ and $n_c = 4$. Besides, GNDO can get the second best BEST on $n_c = 1$, $n_c = 2$, and $n_c = 3$. ARO can obtain the second best BEST on $n_c = 4$. In general, PSO and BSA are the two least competitive algorithms.

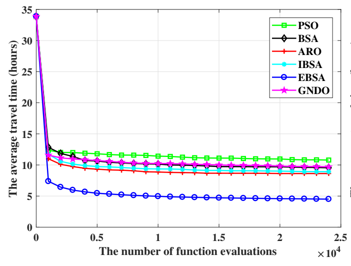
E. Comparison on the average wait efficiency

Thanks to a limited number of charging piles at one node, a drone sometimes needs to wait to charge. However, wait is harmful to delivery efficiency. Motivated by this, to measure delivery efficiency, average wait efficiency is defined by:

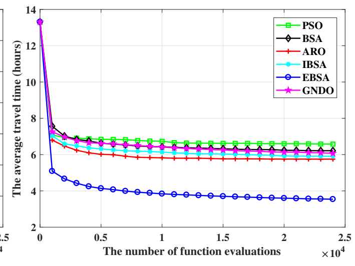
$$W_{\mathrm{E},i} = \frac{1}{n}\sum_{k=1}^{n} t_{\mathrm{tw},k,i} \Big/ \frac{1}{n}\sum_{k=1}^{n} L_{\mathrm{f},k,i}\ , i \in [1, \xi], \tag{39}$$

where $W_{\mathrm{E},i}$ is average wait efficiency of all drones in the ith run and $t_{\mathrm{tw},k,i}$ is wait time of drone k in the ith run. From (39), the average wait efficiency means the required wait time per unit flight length. That is, a smaller average wait efficiency represents a more efficient delivery efficiency.

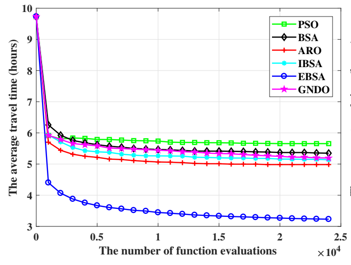
The obtained average wait efficiency by the applied algorithms are presented in Table V. According to the considered scenario, the increase of n_c can effectively reduce the wait time of a drone. From Table V, although the optimal wait efficiency of each algorithm has been a marked decline with the increase of n_c, the compared algorithms still cannot compete with EBSA. That is, EBSA takes the least wait time in each case, which has a more efficient wait efficiency than the other five algorithms in each case. Note that, PSO and BSA are inferior significantly to the other four algorithms in terms of waiting efficiency.



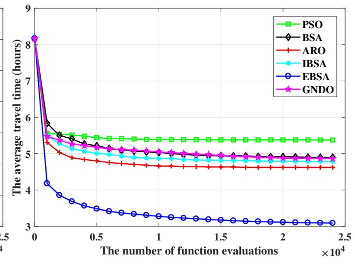
(a) $n_c = 1$



(b) $n_c = 2$



(c) $n_c = 3$



(d) $n_c = 4$

Fig. 8. Average convergence curves from 30 independent runs.

TABLE VI: Average running time (the unit is second) from 30 independent runs of the applied algorithms on each iteration.

n_c	PSO	BSA	ARO	IBSA	GNDO	EBSA
1	1.0906	1.0589	1.0677	**1.0510**	1.1333	1.5354
2	1.1250	1.1047	1.1042	**1.0995**	1.1880	1.5724
3	1.1547	1.1260	**1.1234**	1.1240	1.2427	1.6135
4	1.2036	**1.1500**	1.1646	1.1646	1.2599	1.6563
Avg.	1.1435	1.1099	1.1150	**1.1098**	1.2060	1.5944

F. Comparison on the convergence performance

Fig. 8 shows average convergence curves from 30 independent runs achieved by the applied algorithms in four cases.

Looking at Fig. 8, PSO has the worst convergence ability in the considered four cases. BSA and GNDO have similar convergence performances. IBSA has slightly better convergence performance than BSA and GNDO. Although ARO has obvious convergence advantages over PSO, BSA, IBSA, and GNDO, it is far inferior to EBSA. In addition, PSO, BSA, IBSA, ARO, and GNDO suffer from premature convergence, which converges to locally optimal solutions earlier. In terms of convergence performance, EBSA is significantly superior to PSO, BSA, IBSA, ARO, and GNDO due to two reasons: 1) the built learning mechanism with three learning strategies that can help to keep the population diverse of EBSA, and 2) the designed local escape operator based on random numbers with standard normal distribution can enhance the ability of EBSA to escape from the local optima.

G. Comparison on the running time

Table VI shows the average running time from EBSA and the compared algorithms on each iteration. From Table VI, as the number of charging piles increases, the running time consumed by each algorithm generally has an upward trend due to the structure of Algorithm 1. In addition, according to the average running time shown in the last column of Table VI, the computation efficiency of six algorithms can be sorted from best to worst in the following order: IBSA, BSA, ARO, PSO, GNDO, and EBSA. Clearly, EBSA is the worst of all algorithms in terms of computation efficiency, which can be explained as follows.

Although the excellent performance of EBSA on the considered FP problem has been proven, it requires more computational efforts on executing the local escape operator. In other words, the excellent performance of EBSA is at the cost of reducing computational efficiency. Note that, according to the characteristics of the local escape operator, the number of nodes in the GSN is roughly proportional to the computational overhead of performing the local escape operator.

VI. Conclusion and future directions

This paper focuses on determining the optimal flights for multi-drones to execute long-distance parcel delivery. A multi-drones-assisted commercial parcel delivery system consisting of the depot, the SS, the DPS, and the drone is built. The facilities are equipped with a limited number of charging piles, which form a GSN to provide the charging service for drones. When drones carry out delivery tasks, their FP problem can be converted into an LSOP by PBEM. EBSA is proposed, whose key components are the built comprehensive learning mechanism with three learning strategies and the designed local escape operator with random numbers obeying the standard normal distribution. Experimental results demonstrate that EBSA can determine more efficient flights than the compared algorithms in terms of the considered indicators including average travel time, average flight efficiency, average wait efficiency, average charging efficiency, and convergence performance. That is, experimental results prove the validity of the improved strategies. In addition, as discussed and analyzed in Section V, as the number of charging piles at one node continues to increase, the improvement in delivery efficiency begins to decline. In other words, the number of charging piles at one node needs to be set properly to maximize the comprehensive economic benefits. According to the obtained experimental results presented in Section V, the optimal number of charging piles at one node is three in the considered scenario of this paper.

Note that, in the simulation results, we have not taken into account the issue that the drones can not cross some areas. However, our framework can accommodate the no-fly zone constraint. Specifically, the designed system supports multi-drones to execute long-distance parcel delivery by the connected graph and the deployed charging piles at each node. The connected graph is a 0-1 matrix: "1" means that two nodes are connected (the Euclidean distance between the two nodes is no larger than the given threshold), while "0" means that two nodes are not connected (the Euclidean distance between the two nodes is larger than the given threshold). In the presented simulation results, we used the Euclidean distance for simplicity. If there is some no-fly zone between a pair of nodes, we can simply replace the Euclidean distance with the length of a feasible flight path between the two nodes. This does not impact the operation of the proposed algorithm at all.

As discussed in Section V-G, the excellent performance of EBSA is at the expense of computational efficiency. To overcome this disadvantage, our future research will pay attention to designing a simpler and yet more efficient local escape operator by transfer learning to improve the computational overhead of EBSA on the considered FP problem with a large-scale GSN. In fact, a real multi-drones-assisted commercial parcel delivery system is very complex, whose FP needs to consider two scenarios happening simultaneously. One is that some drones are flying from the depot to their DPSs. Another is that some drones are flying from their DPSs to the depot. In addition, some essential functions also need to be considered, such as a drone needs to pick up parcels returned by customers, a drone needs maintenance after a long flight to its destination parcel station, and a drone needs to avoid obstacles during flight, which also influence the flight planning. Motivated by this, based on the achieved results in this paper, our further work will pay attention to providing the solution to determine the flight planning that covers the round trips of drones between the depot and their destination parcel stations, and some influencing factors, such as picking up parcels returned by customers, maintaining drones after a long-distance flight, and avoiding the obstacles.

References

[1] C. Huang, Z. Ming, and H. Huang, "Drone stations-aided beyond-battery-lifetime flight planning for parcel delivery," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2022.

[2] S. Lee, D. Hong, J. Kim, D. Baek, and N. Chang, "Congestion-aware multi-drone delivery routing framework," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9384–9396, 2022.

[3] S. Poikonen and J. F. Campbell, "Future directions in drone routing research," *Networks*, vol. 77, no. 1, pp. 116–126, 2021.

[4] B. Liu, W. Ni, R. P. Liu, Y. J. Guo, and H. Zhu, "Optimal routing of unmanned aerial vehicle for joint goods delivery and in-situ sensing," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–6, 2022.

[5] W.-C. Chiang, Y. Li, J. Shang, and T. L. Urban, "Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization," *Applied Energy*, vol. 242, pp. 1164–1175, 2019.

[6] S. Kim and I. Moon, "Traveling salesman problem with a drone station," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 42–52, 2019.

[7] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2020.

[8] H. Huang, A. V. Savkin, and C. Huang, "Round trip routing for energy-efficient drone delivery based on a public transportation network," *IEEE Transactions on Transportation Electrification*, vol. 6, no. 3, pp. 1368–1376, 2020.

[9] H. Huang, A. V. Savkin, and C. Huang, "Reliable path planning for drone delivery using a stochastic time-dependent public transportation network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4941–4950, 2021.

[10] H. Huang and A. V. Savkin, "Deployment of charging stations for drone delivery assisted by public transportation vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[11] H. Liu, Q. Chen, N. Pan, Y. Sun, Y. An, and D. Pan, "UAV stocktaking task-planning for industrial warehouses based on the improved hybrid differential evolution algorithm," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 582–591, 2022.

[12] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[13] W. Wang, C. Fang, and T. Liu, "Multiperiod unmanned aerial vehicles path planning with dynamic emergency priorities for geohazards monitoring," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8851–8859, 2022.

[14] K.-W. Chen, M.-R. Xie, Y.-M. Chen, T.-T. Chu, and Y.-B. Lin, "DroneTalk: An Internet-of-Things-based drone system for last-mile drone delivery," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 204–15 217, 2022.

[15] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing*, vol. 8, no. 4, pp. 1643–1653, 2008.

[16] B. Cao, S. Fan, J. Zhao, P. Yang, K. Muhammad, and M. Tanveer, "Quantum-enhanced multiobjective large-scale optimization via parallelism," *Swarm and Evolutionary Computation*, vol. 57, p. 100697, 2020.

[17] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[18] H. Lin, Y. Han, W. Cai, and B. Jin, "Traffic signal optimization based on fuzzy control and differential evolution algorithm," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2022.

[19] A. K. Kalakanti and S. Rao, "A hybrid cooperative method with lévy flights for electric vehicle charge scheduling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 306–14 321, 2022.

[20] A. De, J. Wang, and M. K. Tiwari, "Hybridizing basic variable neighborhood search with particle swarm optimization for solving sustainable ship routing and bunker management problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 986–997, 2020.

[21] Z. Yu, Z. Si, X. Li, D. Wang, and H. Song, "A novel hybrid particle swarm optimization algorithm for path planning of UAVs," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[22] X. Yu, C. Li, and J. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowledge-Based Systems*, vol. 204, p. 106209, 2020.

[23] C. Qu, W. Gai, M. Zhong, and J. Zhang, "A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning," *Applied Soft Computing*, vol. 89, p. 106099, 2020.

[24] Y. V. Pehlivanoglu and P. Pehlivanoglu, "An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems," *Applied Soft Computing*, vol. 112, p. 107796, 2021.

[25] M. D. Phung and Q. P. Ha, "Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization," *Applied Soft Computing*, vol. 107, p. 107376, 2021.

[26] J. Kim, Y. Choi, S. Jeon, J. Kang, and H. Cha, "Optrone: Maximizing performance and energy resources of drone batteries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3931–3943, 2020.

[27] B. Do Valle, C. T. Wentz, and R. Sarpeshkar, "An area and power-efficient analog Li-ion battery charger circuit," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 2, pp. 131–137, 2011.

[28] F. Zhao, J. Zhao, L. Wang, J. Cao, and J. Tang, "A hierarchical knowledge guided backtracking search algorithm with self-learning strategy," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104268, 2021.

[29] S. Nama, A. K. Saha, and S. Ghosh, "Improved backtracking search algorithm for pseudo dynamic active earth pressure on retaining wall supporting c-ϕ backfill," *Applied Soft Computing*, vol. 52, pp. 885–897, 2017.

[30] Z. Ren, A. Zhang, C. Wen, and Z. Feng, "A scatter learning particle swarm optimization algorithm for multimodal problems," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1127–1140, 2014.

[31] Y. Zhang, Z. Jin, and S. Mirjalili, "Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models," *Energy Conversion and Management*, vol. 224, p. 113301, 2020.

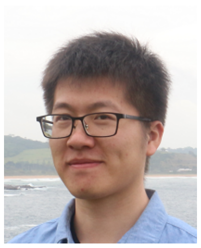
[32] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105082, 2022.

Yiying Zhang received the B.S. degree from Yanshan University, Qinhuangdao, China, in 2014, the M.S. degree from Harbin Institute of Technology, Harbin, China, in 2016, and the Ph.D. degree from Tianjin University, Tianjin, China, in 2021, all in information and communication engineering. He is currently a postdoctoral fellow at the Hong Kong Polytechnic University. His research interests include machine learning, evolutionary computation, and path planning of unmanned aerial vehicles.

Guanzhong Zhou received the B.Eng. degree from the Harbin Institute of Technology, Weihai, China, in 2019. He is now pursuing his master degree at the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong. His research interests include quadruped robots, mobile robots (UGV, UAV), and swarm robotics.

Peng Hang (Member, IEEE) received the Ph.D. degree with the School of Automotive Studies, Tongji University, Shanghai, China, in 2019. He is currently a Research Professor at the Department of Traffic Engineering, Tongji University, Shanghai. He was a Visiting Researcher with the Department of Electrical and Computer Engineering, National University of Singapore, and a Research Fellow with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. His research interests include vehicle dynamics and control, decision making, motion planning, and motion control for autonomous vehicles. He is also an Associate Editor for SAE International Journal of Vehicle Dynamics, Stability, and NVH, and the Guest Editor of the IET Intelligent Transport Systems, and Journal of Advanced Transportation.

Chao Huang the Ph.D. degree in control engineering from the University of Wollongong, Wollongong, NSW, Australia, in 2018. She is currently a Research Assistant Professor with the Department of Industrial and System Engineering, The Hong Kong Polytechnical University (PolyU), Hong Kong. Prior to PolyU, she was a Research Fellow with Nanyang Technological University (NTU), Singapore, and the National Institutes of Informatics (NII), Tokyo, Japan. Her research interests include human–machine collaboration, fault-tolerant control, mobile robot (EV, UAV), and path planning and control. She is an Associate Editor of IEEE Transactions on Intelligent Vehicles and IEEE Transactions on Transportation Electrification.



Hailong Huang the Ph.D degree in Systems and Control from the University of New South Wales, Sydney, Australia, in 2018. He was a post-doctoral research fellow at the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia. He is now an Assistant Professor at the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong. His current research interests include guidance, navigation, and control of UAVs and mobile robots. He is an Associate Editor of IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Vehicles, Intelligent Service Robotics, and International Journal of Advanced Robotic Systems, and an editorial board member of International Journal of Dynamics and Control.