

Drone Stations-aided Beyond-Battery-Lifetime Flight Planning for Parcel Delivery

Chao Huang, Zhenxing Ming and Hailong Huang

Abstract—This paper considers using drones to conduct the last-mile parcel delivery. To enable the beyond-battery-lifetime flight, drone stations are considered to replace or recharge the battery for drones. We focus on the flight planning problem with the goal of minimizing the total travel time from the depot to a customer, a key indicator of the quality of service. We investigate four typical ways for the drone to get extra energy at drone stations: 1) replacing the battery with a fresh one, 2) recharging the battery to the full capacity, 3) recharging the battery to the optimal level, and 4) recharging the battery to the optimal level accounting for the availability of drone stations (i.e., whether a drone station is occupied by other drones). While the first two scenarios can be formulated following the framework of integer linear programming, the last two scenarios turn into mixed-integer nonlinear programming problems. To address the later problems, we present a framework in which discretized state graphs are constructed first and then the optimal paths are found by graph searching algorithms. We propose a dynamic version of Dijkstra's algorithm to deal with the unavailability issue of drone stations. The algorithm can quickly find the optimal flight path for a drone, and extensive computer-based experimental results have been presented to demonstrate the effectiveness of the proposed method.

Note to Practitioners—Multi-rotary unmanned aerial vehicles (UAVs), also known as drones, have been regarded as a promising means to reshape future logistics. To save human labour and reduce cost, many giant logistics companies have been dedicated to developing various drones to deliver light and small parcels during the past decade. However, due to the limitation of payload, the battery capacity is constrained, which prevents drones from long-distance flights. Practitioners have tried the drone-vehicle collaboration method, but this still requires human labour to participate. In this paper, we present a framework where drones autonomously conduct long-distance delivery with the assistance of drone stations. It is worth pointing out that such a framework is not to replace the ground delivery method but to serve as an alternative to the ground counterpart for small and light parcels. A particular focus is on the flight planning from the depot to a destination, which includes not only a sequence of drone stations to stop at but also the corresponding rest time to recharge the battery. Several typical scenarios about battery recharging are discussed, and a dynamic version of Dijkstra's algorithm is presented to deal with the challenging case where drone station resources are limited. The presented approach is able to find out the optimal flight plan quickly.

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

C. Huang is with the Department of Industrial and Systems Engineering, the Hong Kong Polytechnic University, Hong Kong. (E-mail: hchao.huang@polyu.edu.hk).

M. Zhen is with Li Auto Inc. Beijing, China. (E-mail: mingzhenxing@lixiang.com).

H. Huang is 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: hailong.huang@polyu.edu.hk).

Index Terms—Drones, parcel delivery, last-mile delivery, battery recharging, drone stations, path planning, flight planning.

I. INTRODUCTION

DRONE delivery has been widely-researched to alter the current last-mile delivery convention (i.e., door-to-door visiting by postmen) for providing short lead times. Many logistics companies, such as Amazon [1], SF Express [2], DHL [3], and UPS [4], have been dedicated to developing various rotary-wing drones for parcel delivery during the past decade. However, the flight range of most commercial drones, constrained by the currently available battery technology, significantly limits the wide usage of the sole-drone delivery.

To achieve beyond-battery-lifetime delivery, researchers have proposed several approaches. The first approach exploits the collaboration of drones and a ground vehicle such as a van [5]–[11]. The vehicle carries spare batteries for the drones. The basic idea of this approach is to construct a path for the vehicle to visit the target customers and then assign some customers to drones to shorten the path of the vehicle. The conventional travelling salesman problem (TSP) and its variants have been revisited to solve the problems. Such a drone-vehicle collaboration approach can reduce the workload of the vehicle to some extent. But, it still requires the involvement of a human driver.

Aiming at further reducing the participation or even removing human operators from the delivery process, the second approach exploits public transportation vehicles [12]–[15]. The fundamental idea of this approach is that the drone travels with a public transportation vehicle on the roof. The transportation provided by the vehicle can save a significant amount of energy for the drone. The path planning problem has been studied in the time-varying public transportation network. Compared to the first approach, this approach does not require human participation in the delivery process. However, since the public transportation network is with high uncertainty in the timetables of the vehicles, the reliability of this approach is a major concern.

The third approach addresses the reliability issue by deploying drone stations (DSs) where a drone can recharge or replace its used battery with a fresh one [16]–[20]. A key issue in this approach is the positioning problem of DSs. The paper [16] determines the locations of DSs in a given candidate set to maximize the coverage of customers in a certain area. The paper [17] considers the minimization of the average distance between customers and DSs, and DSs are deployed in a continuous space. Differently, the objective of

[18] is to minimize the total system cost including DSs, drone ownership, and service congestion. Another important issue is that given DSs, how to plan the path for drones. The paper [19] proposes to combine public transportation vehicles and DSs to extend the delivery service to remote areas. The paper [20] investigates a problem to minimize the path length and the times of landing at DSs, and proposes a heuristic algorithm based on the ant colony algorithm and A* for path planning.

It is clear that the drone station approach is more reliable than the approach of interacting with public transportation vehicles and more labour-efficient than the approach of drone-vehicle collaboration since no human labour is required. The current paper falls into the drone station approach. We consider a parcel delivery system in an urban area, which consists of a depot, a number of DSs (already deployed according to [16], [17], or [18]) and a number of packstations. The roles of these components are explained as follows. The depot stores parcels to be delivered, and a number of rotary-wing drones are available at the depot. A drone can replace or recharge the onboard battery at a drone station. A packstation temporarily stores parcels for customers. In this system, the normal operation of delivering a parcel to a certain customer can be described as follows. A drone departs the depot with the parcel, flies to the packstation that is the closest to the customer and drops the parcel into the packstation. The drone can land at some DSs for recharging its battery and maintenance during the trip if necessary. Once receiving the parcel, the packstation notices the customer. The customer finally comes to the packstation and collects the parcel.

We pay attention to the flight planning problem from the depot to a packstation that is the closest to a target customer. We call this packstation the target packstation (TPS). Note that although in this paper we consider a TPS as a destination of a delivery drone, the presented methods can be directly applied to the cases where a customer's location is regarded as the destination without any modification. It is also worth pointing out that different from the conventional path planning problem, which targets determining a series of waypoints, the flight planning problem considered here additionally involves the stay at the waypoints, i.e., for recharging or replacing the battery. We formulate the flight planning problem to minimize the total travel time including the time for flight and the time the drone spends at DSs. For the second part of the time, we consider several typical scenarios: 1) the drone replaces the used battery with a fresh one, 2) the drone recharges the battery to the full capacity, 3) the drone recharges the battery to an optimized level, and 4) the drone recharges the battery to an optimized level accounting for the unavailability of DSs. The flight planning problem for the first and second scenarios can be easily addressed by integer linear programming (ILP) techniques. The third and fourth scenarios involve both integer variables (which DSs to stop at) and real-value variables (how long to recharge the battery), which are much more difficult than the first two scenarios.

To address the problems in the third scenario, we propose a discretized solution. The basic idea is to discretize the battery recharging time and the battery residual energy and then construct a state graph. In particular, each possible residual

energy of the drone at each drone station is regarded as a state. Then, the size of the state graph depends on the number of DSs and the number of possible residual energy levels (which further depends on the discretization resolution). With this state graph, we adopt existing graph searching algorithms, such as Dijkstra's algorithm, to find the shortest path from the state corresponding to the depot to a state corresponding to the target packstation, which gives the solution to the third scenario. This method cannot be used to solve the fourth scenario because the state graph does not involve the information on the unavailable intervals of DSs. An unavailable interval tells about an absolute time interval (e.g., from 11:10 to 11:20) during which a drone station cannot be used, while the edge cost associated with an edge in the state graph only tells about the relative information (e.g., it takes 15 minutes to transfer from one state to another). To address this problem, we propose a dynamic version of Dijkstra's algorithm, which amends the cost of an edge of the state graph during the process of computing the shortest path.

The main contributions of this paper are summarized below.

- We formulate the flight planning problems in several typical scenarios to minimize the duration from the instant the drone leaves the depot to the instant the drone arrives at the TPS.
- For the case with sufficient resources of DSs, we propose a discretized method to address the flight planning problem. It firstly discretizes the battery recharging time and the battery residual energy. Then, each possible residual energy of the drone at each drone station is regarded as a state. Dijkstra's algorithm is applied to find the shortest path in the state graph.
- For the case with limited resources of DSs, we propose a dynamic version of Dijkstra's algorithm to address the flight planning problem.
- Extensive simulation results have been shown to demonstrate the effectiveness of the proposed methods, and the impacts of several key parameters have been investigated.

The rest of this paper is organized as follows. In Section II, we discuss the closely relevant work and highlight the main difference between the current paper and the existing publications. In Section III, we describe the flight planning problem and present the mathematical models. In Section IV, we present the proposed solution to address the flight planning problem, and the optimality and the complexity of the proposed approach are discussed. Section V shows the results of computer-based experiments. Finally, Section VI concludes this paper with our research direction.

II. RELATED WORK

The shortest path problem (SPP) is a classical combinatorial optimization problem, which has a great number of applications such as telecommunications, vehicle routing, traffic assignment, and network design. The conventional SPP aims at determining the path from a source position to a destination position so that the summation of the edges' costs is minimized.

So far, numerous algorithms have been proposed to address SPP, including the popular label-setting algorithm of Dijkstra

[21], A* [22], hybrid A* [23], D* [24], Best First Search (BFS) [25], the label-correcting algorithm of Bellman and Ford [26], the random sampling enabled algorithms of Rapidly-exploring Random Tree (RRT) [27] and Probabilistic Roadmap (PRM) [28], and the optimization-based approaches [29]–[31]. While the label-setting and label-correcting-based algorithms can find the optimal path in graphs, the randomized algorithms take random samples from the configuration space to build a graph for a given environment first and then adopt graph searching algorithms to determine the shortest path. The optimal path solved by these algorithms is acceptable under the SPP requirement. Moreover, the shortest paths obtained by these algorithms can generally serve as the input to form a distance matrix for the task assignment when multiple vehicles collaborate; see e.g., [30]–[32].

Over the past few decades, many variants of SPP have been investigated. Reliable shortest path problem (R-SPP) reflects the variability of travel time and is more realistic than standard SPP [33]. In R-SPP, the cost of an edge may change due to congestion, demand variations, and traffic accidents. Such a feature is practical, especially in road networks. R-SPP can be solved by two mainstream methods. The first method maximizes the on-time arrival probabilities or path reliability, and the second method minimizes the mean and standard deviation of travel time of the path.

Under the context that traversing an edge also leads to a certain amount of resource consumption, the constrained shortest path problem (C-SPP) aims at finding the minimum cost path subject to not exceeding a maximum resource consumption [34]. There are two main approaches to address C-SPP: Dynamic Programming (DP) and Lagrangian relaxation [35]. While the main idea of DP-based solutions is to transform the C-SPP into the conventional SPP and solves the latter using a label-setting or label-correcting algorithm, the second major approach is based on the Lagrangian relaxation and solves the relaxed integer programming formulations.

Another variant is the time-dependent shortest path problem (TD-SPP) [36]. Different from the conventional SPPs and some other variants, in TD-SPP, the cost of an edge varies with the starting time to traverse the edge. Dijkstra's algorithm can be used to solve the TD-SPP under the assumption of an unrestricted waiting policy. If waiting at a node leads to an earlier arrival time (e.g., due to relieving congestion), optimal wait times at intermediate nodes can be found and utilized at these nodes [37]. An accepted algorithm to solve this problem is called Bidirectional A* [38].

The studied problems closely relate to TD-SPP but differ from TD-SPP in the following aspects. Unlike TD-SPP where the edges' costs vary with time and there is no cost associated with a node, in the considered problems, the cost of an edge can be roughly regarded as a constant (since it represents the flight time between two physical DSs) and a node (DS) is with a cost representing the time for recharging the battery. Additionally, different from TD-SPP which mainly aims to figure out the optimal path together with the waiting times at the nodes so that the path has the least total travel time, waiting at a node in our context represents the battery recharging process. The waiting time, i.e., battery recharging time is optimized to

not only allow the drone to complete the following edge but also help to avoid the interruption due to the unavailability of DSs. These differences make the existing approaches not applicable to the considered problems. To address the problem of interest, we present mathematical formulations in Section III and our solutions in Section IV.

III. PROBLEM STATEMENT

The considered system consists of a depot, a number of DSs and a number of packstations. We focus on designing the optimal flight plan for the drone in terms of travel time from the depot to the TPS. We do not consider the trivial case where the TPS is within the flight range of a single battery lifetime from the depot. For ease of description, the depot, TPS, and DSs will be simply addressed as stations, if this does not confuse.

To state the problem, some symbols are introduced. Let p_1 denote the location of the depot, p_m denote the location of the TPS, and p_2, \dots, p_{m-1} denote the locations of DSs. Let τ_{tl} and ϵ_{tl} represent the time and energy consumption for take-off and landing operations, respectively. Let ϵ_0 denote the energy capacity of a fully charged battery. Let D_{ij} be the feasible Euclidean distance from p_i to p_j . Suppose the power consumption of the drone is P when it flies at the speed of v . Moreover, we introduce the following functions. Let $f(e_1, e_2)$ be a function returning the time duration needed to recharge the battery from the initial level e_1 to the final level e_2 , where $0 \leq e_1 < e_2 \leq \epsilon_0$. Let $g(e, t)$ be a function returning the final energy of the battery when it starts to be recharged from the energy e for a time duration t , where $0 \leq e \leq \epsilon_0$ and $0 < t \leq R$, where R is a given constant representing the time to fully recharge the battery from level zero. We assume that the recharging process is separable in time, i.e., $g(e, t) = g(g(e, a), t - a)$, where $0 \leq a \leq t$. Note that the functions $f()$ and $g()$ can be obtained from the battery recharging curve. The mainly used symbols are summarized in TABLE I for a quick reference.

We assume that the drone takes the following operations when it flies from station p_i to station p_j , see Fig. 1. It takes off at p_i and climbs to a certain altitude, which should be within the allowed range. It then flies at this altitude at the speed of v towards p_j . When the drone is over p_j , it executes the landing operation. If $j \neq m$, the drone replaces/recharges its battery. Moreover, the necessary condition for a drone to safely reach p_j from s_i is that the overall energy consumption is no larger than the battery capacity:

$$\epsilon_{tl} + \frac{D_{ij}P}{v} \leq \epsilon_0. \quad (1)$$

We can construct a graph $G(V, A)$. The vertex set V consists of the positions of DSs, i.e., $V = \{p_1, \dots, p_m\}$. Any pair of vertices i and j are connected by an arc of A if constraint (1) holds. Each arc is associated with a cost representing the time the drone needs to spend on the arc. Let t_j denote the time a drone spends at station j to replace/recharge its battery. Then, if the vertices i and j are connected by an arc, the corresponding travel time (from the instant the drone leaves

TABLE I: Main symbols and their meanings

Parameter	Meaning
p_i	The location of station i
τ_{tl}	The time for take-off and landing operations
ϵ_{tl}	The energy consumption for take-off and landing
ϵ_0	The energy capacity of a fully charged battery
D_{ij}	The feasible Euclidean distance from p_i to p_j
v	The drone flying speed
P	Power consumption when the drone flies at v .
T_{ij}	Travel time from p_i to p_j including the time for take-off, flight, landing, and battery replacing/recharging
C_{ij}	Travel time from p_i to p_j including the time for take-off, flight and landing
W_{ij}	The energy consumption to travel from p_i to p_j
Q_{sw}	Transfer time from state s to state w in the state graph
Function	Meaning
$f(e_1, e_2)$	The function returning the time duration needed to recharge the battery from the initial level e_1 to the final level e_2
$g(e, t)$	The function returning the final energy of the battery when it starts to be recharged from the energy e for a time duration t ($0 < t \leq R$, where R represents the time to fully recharge the battery from level zero)
$h(I, \tau, t)$	The function returning the instant at which the recharging is completed under that the drone starts to recharge the battery at instant τ and needs to recharge for a duration t , given the unavailability interval I
Variable	Meaning
x_{ij}	A binary variable indicating whether the drone will travel from p_i to p_j
y_i	A real-value variable representing the departure instant from p_i
z_i	A real-value variable representing the drone residual energy when it departs p_i
t_i	A real-value variable representing the battery recharging duration at p_i

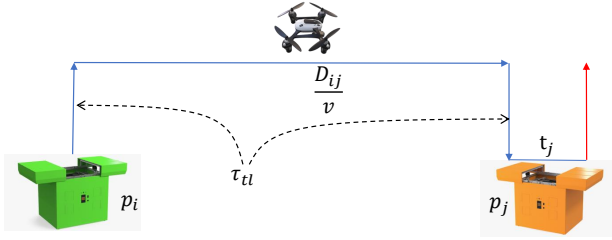


Fig. 1: The operations of a drone from p_i to p_j . The drone takes the time τ_{tl} for take-off and landing, $\frac{D_{ij}}{v}$ for flying, and t_j for recharging the battery.

station i to the instant the drone leaves station j), denoted by T_{ij} , is given by:

$$T_{ij} = \begin{cases} \tau_{tl} + \frac{D_{ij}}{v} + t_j, & \text{if } j \neq m, \\ \tau_{tl} + \frac{D_{ij}}{v}, & \text{if } j = m. \end{cases} \quad (2)$$

In (2), if $j = m$, it means the drone reaches the TPS, where it cannot replace/recharge its battery. In this situation, the drone drops off the parcel, and the corresponding time and energy consumption are ignored.

The proposed scheme, i.e., beyond-battery-lifetime flight, depends on obtaining extra energy at drone stations. In the rest of this section, from simple to complex, we discuss several typical scenarios of battery recharging, detailed as follows.

Scenario 1: Replace the battery.

When spare batteries are available at DSs, the drone can simply replace its used battery with a fresh one instead of recharging it on site. In this scenario, t_j can be considered as a given constant.

Scenario 2: Fully recharge the battery.

When no spare batteries are available at DSs, the drone recharges its battery at DSs. Suppose the drone recharges the battery to its full capacity when it is at a drone station. Then, the time spent at the station depends on the time of fully recharging the battery. When the drone flies from p_i to p_j , since the drone always fully recharges the battery, the residual energy when the drone arrives at p_j is given by $e = \epsilon_0 - \epsilon_{tl} - \frac{D_{ij}P}{v}$. If $j \neq m$, the time the drone spends at p_j is $t_j = f(\epsilon_0 - \epsilon_{tl} - \frac{D_{ij}P}{v}, \epsilon_0)$. In this scenario, the cost of an arc is computed as follows:

$$T_{ij} = \begin{cases} \tau_{tl} + \frac{D_{ij}}{v} + f(\epsilon_0 - \epsilon_{tl} - \frac{D_{ij}P}{v}, \epsilon_0), & \text{if } j \neq m, \\ \tau_{tl} + \frac{D_{ij}}{v}, & \text{if } j = m. \end{cases} \quad (3)$$

In the above two scenarios, the cost of any arc of the graph $G(V, A)$ can be computed according to (2) or (3), or set as ∞ if constraint (1) is unsatisfied. In these cases, the cost of an edge, i.e., T_{ij} , can be known in advance. Let x_{ij} be a binary variable: $x_{ij} = 1$ if the drone travels from station i to station j ; $x_{ij} = 0$, otherwise. Moreover, $x_{ii} = 0, \forall i = 1, \dots, m$. The minimum cost path planning problem from station 1 to station m in the graph $G(V, A)$ is formulated as:

$$\mathbf{P0}: \min \sum_{i,j \in V} T_{ij} x_{ij} \quad (4)$$

subject to

$$\sum_{j=1}^m x_{ij} - \sum_{j=1}^m x_{ji} = \begin{cases} 1, & \text{if } i = 1, \\ -1, & \text{if } i = m, \\ 0, & \text{if } i \neq 1, m. \end{cases} \quad (5)$$

$$\sum_{i,j \in V_1} x_{ij} \leq |V_1| - 1, \forall V_1 \subseteq V, \quad (6)$$

$$x_{ij} \in \{0, 1\}, \forall i, j = 1, \dots, m. \quad (7)$$

The objective function (4) is to minimize the overall time from the depot p_1 to the target packstation p_m . In constraint (5), for $i = 1$, we have $\sum_{j=1}^m x_{ij} - \sum_{j=1}^m x_{ji} = 1$. In other words, the number of the outgoing edges of p_1 is larger than that of the incoming edges by 1. Similarly, for $i = m$, the constraint $\sum_{j=1}^m x_{ij} - \sum_{j=1}^m x_{ji} = -1$ leads to that the number of the incoming edges of p_1 is larger than that of the outgoing edges by 1. For $i \neq 1$ and $i \neq m$, the constraint $\sum_{j=1}^m x_{ij} - \sum_{j=1}^m x_{ji} = 0$ result in the balanced numbers of incoming and outgoing edges. Constraint (6) eliminates subtours, where V_1 is a subset of V , and $|V_1|$ gives the number of elements in the set V_1 . The ILP problem $\mathbf{P0}$ is the standard SPP because the costs at nodes have been added into that of the relevant edges. So, this problem can be addressed by existing algorithms. Note that we do not call this problem the flight planning problem because the battery recharging time is not optimized.

Scenario 3: Optimally recharge the battery.

It is worth pointing out that when the drone needs to recharge its battery at DSs, recharging it to its full capacity may not be the optimal option for minimizing the overall travel time. Consider that it consumes 80% of the battery energy for the drone to fly from one station to the next following a constructed path. The drone can just recharge its battery to 80% at the former drone station instead of 100%. This has the potential to save time to reach the target packstation. Thus, besides optimizing the selection of DSs, it is also necessary to optimize the recharging time at DSs to obtain the minimum cost path.

To formulate this problem, we need to separately model the time the drone travels between stations and spends for recharging. Let C_{ij} denote the time the drone takes to travel between stations i and j . It involves the time for take-off, flying and landing, i.e., $C_{ij} = \tau_{tl} + \frac{D_{ij}}{v}$. Correspondingly, let W_{ij} denote the energy consumption, i.e., $W_{ij} = \epsilon_{tl} + \frac{D_{ij}P}{v}$. Moreover, let $z_j > 0$ be a real-value variable representing the residual energy when it is about to depart from station j (after recharging the battery). When the drone travels from p_i to p_j and recharges the battery at p_j for time t_j , we have the following relationship about the residual energy:

$$z_j = g(z_i - W_{ij}, t_j). \quad (8)$$

Now, the problem of interest is to determine a flight plan including where to stop and how long to recharge the battery so that the drone can reach p_m from p_1 in the shortest time, which is formulated as follows

$$\mathbf{P1:} \min \sum_{i=1}^m \sum_{j=1}^m C_{ij} x_{ij} + \sum_{i=1}^m t_i \quad (9)$$

subject to (5), (6), (7), and

$$x_{ij}(z_j - g(z_i - W_{ij}, t_j)) = 0, \forall i, j = 1, \dots, m, \quad (10)$$

$$x_{ij}(z_i - W_{ij}) \geq 0, \forall i, j = 1, \dots, m, \quad (11)$$

$$z_0 = \epsilon_0, \quad (12)$$

$$z_i \in [0, \epsilon_0], \forall i = 1, \dots, m. \quad (13)$$

$$t_i \in [0, R], \forall i = 1, \dots, m. \quad (14)$$

This formulation can be further explained as follows. The objective function (9) is to minimize the time duration from departing p_1 to the arrival at p_m including the time for flight and that for battery recharging. Constraint (10) gives the relationship of the residual energy when the drone travels between two stations. If the drone does not travel between them, the constraint still holds as x_{ij} will be 0. Constraint (11) specifies that if the drone travels from station i to station j , the departure energy should be no smaller than the amount to be consumed. Constraint (12) defines the initial condition of z_0 . Constraints (13) and (14) set the range of the variables. Clearly, **P1** is a mixed-integer nonlinear programming problem, which is generally difficult to be addressed by the common optimization solvers. In addition, these solvers may not be able to obtain solutions for large-scale instances within a reasonable time.

Scenario 4: Optimally recharge the battery accounting for the unavailability of DSs.

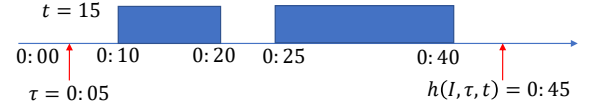


Fig. 2: An illustrative example of $h(I, \tau, t)$. The drone starts to recharge the battery at time 0:05 and requires $t = 15$ minutes. The completing instant is $h(I, \tau, t) = 0:45$. Totally, it takes 40 minutes to complete because the drone station is occupied for 10 minutes from 0:10 to 0:20 and 15 minutes from 0:25 to 0:40.

In Scenario 3, we assume that whenever a drone arrives at a drone station, the battery can get recharged immediately. However, in practice, multiple drones may operate at the same time, and the number of drones a drone station can serve simultaneously may be upper bounded. In this situation, ignoring the unavailability of DSs may result in some waiting time at DSs. In the final part of this section, we formulate a new flight planning problem considering the unavailability of DSs.

Let I_i denote the set of time intervals in which station i is occupied by other drones (which are already assigned to conduct some missions), and I_i is also called the unavailable intervals of station i . Let y_i be a real-value variable indicating the departure instant from station i . Let $h(I, \tau, t)$ be a function returning the time instant at which the recharging is completed under that the drone starts to recharge the battery at instant τ and needs to recharge for a period of t , given the recharging unavailability interval I , where $0 < t \leq R$. An example is shown in Fig. 2. The energy consumption of a drone when it waits at a drone station is ignored. With the function $h(I, \tau, t)$, we can construct the following relationship of the departure instant if the drone travels from station i to station j :

$$y_j = y_i + C_{ij} + h(I_j, y_i + C_{ij}, t_j). \quad (15)$$

Then, the new flight planning problem from p_1 to p_m can be formulated as:

$$\mathbf{P2:} \min \sum_{i=1}^m \sum_{j=1}^m C_{ij} x_{ij} + \sum_{i=1}^m t_i \quad (16)$$

subject to (5), (6), (7), (10), (11), (12), (13), (14), and

$$x_{ij}(y_j - y_i - C_{ij} - h(I_j, y_i + C_{ij}, t_j)) = 0, \forall i, j = 1, \dots, m, \quad (17)$$

$$y_0 = \tau_0, \quad (18)$$

$$y_i \geq \tau_0, \forall i = 1, \dots, m. \quad (19)$$

The objective of **P2** is the same as **P1**. Constraint (17) gives the relationship of the departure instant when the drone travels between two stations. If the drone does not travel between them, the constraint still holds as x_{ij} will be 0. Constraint (18) defines the initial condition of y_0 . Constraint (19) sets the range of y_i . Similar to **P1**, **P2** is also a mixed-integer nonlinear programming problem, which is hard to solve especially in large-scale instances.

IV. PROPOSED SOLUTION

To address **P1** and **P2**, we propose a discretized solution. The basic idea is to discretize the recharging time and the residual energy and then construct a finite state graph. Finally, we search for the optimal path in the state graph.

A. Discretize time and energy

We first discuss the discretization. Let δ_t and δ_e be a unit of time and a unit of energy, respectively, such as 1 minute and 1 Joule. The time duration C_{ij} and the returned values of the functions $f()$ and $h()$ are approximated by the closest values of the set $\{\delta_t, 2\delta_t, 3\delta_t, \dots\}$. Similarly, the battery residual energy z_i , the energy consumption W_{ij} and the returned value of the function $g()$ are approximated by the closest values of the set $N = \{\delta_e, 2\delta_e, \dots, n\delta_e\}$, where $n = \lfloor \frac{\epsilon_0}{\delta_e} \rfloor$. These approximations will be sufficiently accurate if δ_e and δ_t are small enough.

B. Construct state graph and solve **P1**

Secondly, we convert the physical graph to a state graph. Specifically, based on the discretization in Section IV-A, we create n energy states for each station. By doing this, we obtain a $n \times m$ (m is the number of stations) state graph, in which each state characterizes the possible residual energy of the drone at a station.

There are two types of edges in the state graph:

- For two states corresponding to the same station, if the gap of their energy levels is δ_e , they are linked by a Type-1 edge from the state with a lower energy level to the other; otherwise, they are not linked; see the example in Fig. 3. A Type-1 edge represents the battery recharging process, and the cost of a Type-1 edge is the corresponding battery recharging duration, which is computed by the function $f()$ given the energy levels of the two states.
- For two states corresponding to different stations i and j , one state with the higher residual energy is linked with the other state with the lower residual energy by a Type-2 edge from the former to the latter, if the gap of their energy equals the energy consumption when the drone travels from the former station to the latter, i.e., W_{ij} ; see the blue and red arrows in Fig. 3. If the energy gap does not equal the energy consumption, these two states are not linked. A Type-2 edge represents the physical travel between the two corresponding stations, and the cost is given by the corresponding travel duration, i.e., C_{ij} .

We introduce a new symbol, i.e., Q_{sw} , to represent the time for transferring from state s to state w in the state graph. The states s and w can refer to the same physical station i or different stations i and j depending on the type of the edge. It is also worth pointing out that according to the construction process, the state graph is static.

In the state graph, **P1** aims at finding the shortest path from the state $s_{initial} = (p_1, \epsilon_0)$ to one of the states $S_{final} = \{(p_m, \lambda) | \lambda \in N\}$. Note that for the flight planning problem from the depot to the target packstation, there are totally N

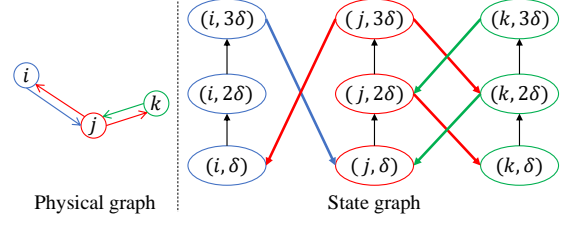


Fig. 3: Converting a physical graph to the state graph. Travelling between stations i and j consumes 2δ , and travelling between stations j and k consumes δ . In the state graph, the black arrows are with the battery recharging times. The blue, red and green arrows are with the times of flying between two stations.

possible final states. If we further consider the return trip, i.e., from the packstation to the depot, λ must be over a value that allows the drone to reach the closest drone station. Having this constraint can simplify the problem because the states that do not satisfy the constraint and the corresponding edges will not be considered in the graph construction phase, resulting in a smaller graph with fewer states and fewer edges. Graph searching algorithms such as Dijkstra's algorithm can be used to address the shortest path problem. The obtained path by solving this problem consists of a number of states, with which we can easily derive the flight plan.

C. Solve **P2**

To solve **P2**, the static state graph is not sufficient as it does not account for the unavailability of DSs. In the state graph, the cost of a Type-1 edge is the corresponding battery recharging duration, which is computed by the function $f()$ with the energy levels of the two states as input. When the unavailability of a drone station is considered, the actual cost of a Type-1 edge becomes time-dependent. If the time period for recharging overlaps with the unavailable interval of the drone station, the actual cost of the edge increases and the amount needs to be computed by the function $h()$ if the instant to start recharging is given; see the example in Fig. 2. However, the unavailability of DSs cannot be incorporated into the state graph because the edge cost refers to the duration (relative information) to pass that edge, not the particular time instant (absolute information) at which the drone appears at the corresponding stations.

To account for the unavailability of DSs in the flight planning process, we propose a dynamic version of Dijkstra's algorithm. The inputs include the starting instant τ_0 , the initial energy ϵ_0 , the unavailability interval of each drone station I_2, \dots, I_{m-1} , and the state graph constructed in Section IV-B. The output is the shortest path consisting of a sequence of states. Following the spirit of Dijkstra's algorithm, the states of the graph are subdivided into three non-overlapping sets:

- A: the states for which the shortest path from the initial state is known. The nodes added to this set will be in the order of the increasing duration from the initial state.

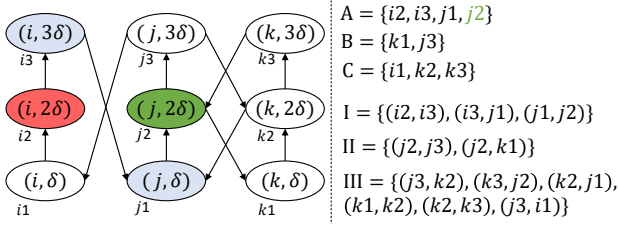


Fig. 4: An illustrative example to demonstrate the definitions of the sets of states and edges used in our algorithm. The state graph on the left is the same as Fig. 3. The initial state is the red one, and we use the representation of $i2$ for convenience. Support the state $j2$ is just visited. For this example, the corresponding state and edge sets are shown on the right.

- B: the states from which the next state to be added to set A will be selected. Note that this set comprises all the states that are linked with at least one state of set A.
- C: all the rest of the states of the graph.

The edges are subdivided into three non-overlapping sets:

- I: the edges appearing on the shortest paths from the initial state to the states in set A.
- II: the edges from which the next edge to be added to set I will be selected.
- III: all the rest of the edges of the graph.

Fig. 4 provides an example to demonstrate these definitions.

With the above definitions, the proposed algorithm is presented in Algorithm 1. To start with, all states are initialized with the label of the earliest arrival instant ∞ , all states are in set C, and all edges are in set III. The initial state, i.e., $s_{initial}$, is then initialized with the actual starting instant τ_0 , denoted by $s_{initial}^{\tau_0}$, where the superscript τ_0 gives the label of the state. This state is moved to set A; see Line 1-3. After initialization, Algorithm 1 repeats a few procedures. Firstly, it finds out all edges E linking the state (denoted by s_o) that was just moved into set A with the states (denoted by s) in sets B and C; see Line 5. For each edge E , it computes the temporary arrival instant at a linked state s ; see Line 6-11. In particular, if E is a Type-1 edge, the temporary arrival instant at s may be updated by the function $h()$ (depending on if the arrival instant falls into an unavailability interval of the corresponding station); see Line 6-8. If E is a Type-2 edge, the temporary arrival instant is updated by adding the edge cost to the arrival instant at state s_o^{τ} ; see Line 9-10. Then, the algorithm selects the set to which the linked state s will be moved. If $s \in$ set C, it is moved to set B and the corresponding edge is moved to set II. The temporary label is attached to s ; see Line 13-14. If $s \in$ set B, it investigates whether using edge E can result in a path with an earlier arrival instant. If yes, edge E replaces the corresponding edge in set II; otherwise, edge E is rejected; see Line 16-22. Note that this procedure is not applied to the case in which $s \in$ set C because the states in set C are all with the label of ∞ . After dealing with all the edges that are linked with state s_o , a new state in set B that has the smallest label (i.e., the arrival instant), is selected and moved to set A. The corresponding edge linking s_o and this

Algorithm 1 Path Planning Algorithm

```

1: Store all states in set C and all edges in set III.
2: Initialize all the arrival instants of the states by  $\infty$ .
3: Move the initial state  $s_{initial}$  from set C to set A and label
   the state by  $s_{initial}^{\tau_0}$ .
4: while not all states belonging to  $S_{final}$  are in set A do
5:   for each  $E$  linking the state  $s_o$  that was just moved to
     set A with the states  $s$  in sets B and C do
6:     if  $E$  is of Type-1 (i.e., states  $s_o$  and  $s$  correspond
       to the same station say  $i$ ) then
7:       Compute the instant the battery recharging is
       completed by  $h(I_i, s_o^{\tau}, Q_{s_o s})$ .
8:       Record  $h(I_i, s_o^{\tau}, Q_{s_o s})$  as the temporary label
       for state  $s$ .
9:     else if  $E$  is of Type-2 then
10:      Record  $s_o^{\tau} + Q_{s_o s}$  as the temporary label for  $s$ .
11:    end if
12:    if state  $s$  belongs to set C then
13:      Move  $s$  from set B and edge  $E$  to set II.
14:      Label  $s$  by the temporary label.
15:    end if
16:    if state  $s$  belongs to set B then
17:      if the temporary label of state  $s$  is smaller than
        the existing label then
18:        Update the label for state  $s$ 
19:        Use edge  $E$  to replace the corresponding
        edge in set II and remove that edge from set II.
20:      else
21:        Reject edge  $E$ .
22:      end if
23:    end if
24:  end for
25:  Look for the state with the minimum arrival instant in
  set B, move it from set B to set A and the corresponding
  edge from set II to set I.
26: end while

```

state is moved from set II to set I; see Line 25. In the next round, the selected state becomes the new s_o . After all states of the set S_{final} are moved to set A, the algorithm terminates with all the edges linking the state $s_{initial}$ and the states in S_{final} . From any final state, we can backtrack all the states leading to $s_{initial}$, and these states form a complete flight plan.

Discussion on the correctness: From the definition of the function of $h(I, \tau, t)$ we know that for a given unavailable interval I and a given charging period t , starting to recharge the battery at an earlier instant τ_1 completes the recharging process at an earlier instant than starting at a later instant τ_2 , i.e., $h(I, \tau_1, t) \leq h(I, \tau_2, t)$. In other words, the function $h(I, \tau, t)$ has the property of first-in-first-out (FIFO), and to bypass any Type-1 edge, an earlier start instant always results in no later instant of completing the edge. This is the reason that we assign the label of a state linked with a Type-1 edge in Line 6-8. Other than this time-dependent cost of Type-1 edges, the rest part of Algorithm 1 shares the spirit of Dijkstra's algorithm. The correctness of Dijkstra's algorithm implies the correctness of Algorithm 1.

Complexity: The computational complexity of implementing Algorithm 1 depends on the data structure in use to store the states and the searching method to find the minimum arrival instant among the states. The states are stored using an ordinary linked list. We do not make any specification on the order of states in the linked list. A linear search through all the states can be used to find the state with the minimum arrival instant. We also assume that the time complexity of computing $h()$ is bounded by a constant. The complexity is proportional to the number of edges and the squared number of states in the state graph. The number of states in the state graph is nm , and the number of edges in the state graph is upper bounded by n^2m^2 . Therefore, the complexity of Algorithm 1 is in the order of $O((nm)^2 + n^2m^2) = O(n^2m^2)$. The complexity of Algorithm 1 can also be analyzed from the algorithm structure. Specifically, at the initial stage, we store the states and edges in the corresponding sets and initialize the labels of all the states. The complexity of such an initialization depends on the number of states and that of edges, which is $O(n^2m^2)$. Regarding the while loop and the for loop except for the state searching operation at the end of the while-loop, i.e., line 25, each state will be put into set A only once and the corresponding edges will also be dealt with only once. Considering that the constructed graph is directional (see Section IV-B), the number of overall operations of the while-loop and the for-loop are right the total number of edges in the graph, whose upper bound is n^2m^2 . Moreover, there are some if-else condition verification and the corresponding operations in the for loop. Generally, the computing complexity of them can be regarded as a constant. Thus, without line 25, the complexity of the while-loop and the for-loop is $O(n^2m^2)$. Now, let us just consider the complexity of the while loop with line 25. In different rounds of the while loop, the number of states in set B will also be different. Thus, it is impossible to accurately estimate the complexity. The worst case is that all the states are in set B, and the number is nm . Applying a linear search leading to a complexity of $O(nm \times nm) = O(n^2m^2)$ for the while-loop with line 25 without the for-loop, where the former nm is the number of while-loops and the latter nm is the number of states in set B in the worst case. So, the overall complexity of the while-loop and the for-loop (together with line 25) is $O(n^2m^2)$. Involving the initialization, the overall complexity of Algorithm 1 is also $O(n^2m^2)$. It is worth noting that the number of edges in a practical state graph should be much smaller than n^2m^2 due to the construction procedures discussed in Section IV-B. Thus, the practical complexity should be smaller than the worst case of $O(n^2m^2)$.

V. COMPUTER-BASED EXPERIMENTAL RESULTS

In this section, a series of computer-based experiments are presented to demonstrate the delivery process and verify the effectiveness of the proposed approach. The locations of the depot, pack stations, and DSs¹ are exhibited in Fig. 5. The red-filled triangle refers to the depot, the red-filled square refers to the packstation, and the blue-filled circle refers to DSs. The

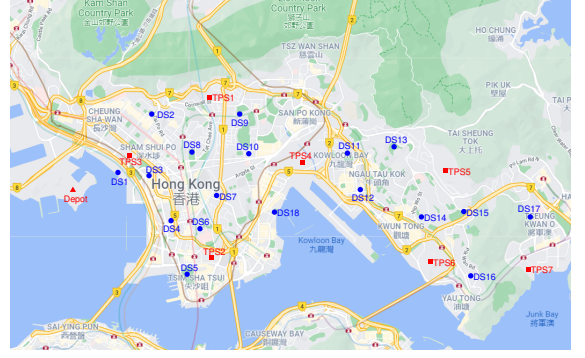


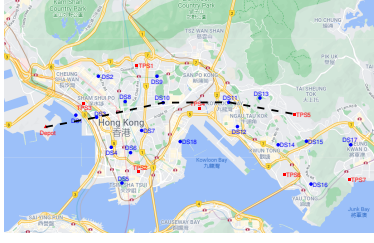
Fig. 5: The locations of the depot, packstations, and DSs.

depot is the only starting point on the map, and TPS is the abbreviation of target packstation. In this experiment, we have $m = 26$ total stations, the average flight speed of the drones are set to $v = 4$ m/s, the fully recharged energy capacity of the battery is set as $e_0 = 32 \times 10^4$ J, the energy consumed during take-off and landing operation is set as $e_{tl} = 1.25 \times 10^4$ J and we assume the time cost for take-off and landing operation is $\tau_{tl} = 50$ s. For the whole model's energy unit and time unit, we set $\delta_e = 5 \times 10^4$ J, $\delta_t = 1$ s. Thus, we have $n = 6$ energy state in total for each DS. We further assume the power consumption rate of the drones during the average speed flight is $P = 300$ W. To simplify the simulation process, the battery recharge function $g(e_0, t)$ is assumed to be linear, and the recharging rate is 5×10^3 J/minute.

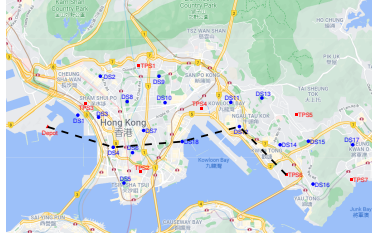
Results without considering DSs' availability: We first simulate our proposed model without taking the DSs' availability into consideration. Applying our approach, the flights from the depot to the seven TPSs shown in Fig. 5 are obtained. We display the global paths to TPS5, TPS6 and TPS7 in Fig. 6. The details of the flights, including flight trajectory, residual energy at each station before and after recharge, total flight length, and time cost during the flight, to the seven TPSs are presented in TABLE II. In the second column of TABLE II, the first element in the parentheses refers to the station that the drone will stop at, the second element indicates the residual energy when the drone arrives, and the third element indicates the residual energy when the drone departs. The sequence in the second column reveals the whole flight trajectory and residual energy variation detail during the flight for different TPSs. Taking TPS1 as an example, the corresponding row in TABLE II should read as: the drone starts from the depot with the initial residual energy of 6; it then flies to DS2 with the arrival residual energy of 2 and recharges the battery to the energy level of 3; finally, it flies to TPS1 with the arrival residual energy of 1. The flight distance is 4.8 km, and the corresponding time is 0.53 hours.

Parameter impacts: We are interested in the impact of different parameters. We take TPS7 as the target packstation to study the impact. The parameter set-ups are shown in TABLE III. The corresponding flight trajectories are exhibited in Fig. 7. In TABLE III, Sim0 is the baseline. Sim1 and Sim2 reveal that

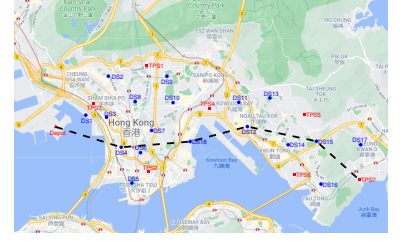
¹Note that the deployment of DSs, belonging to the facility location problem, is out of the scope of this paper. Readers can refer to [16]–[19].



(a) Flight plan to TPS5.



(b) Flight plan to TPS6.



(c) Flight plan to TPS7.

Fig. 6: Flight plans for different target packstations.

TABLE II: Flight Plans for TPS1-TPS7

Destination	Trajectory and Residual Energy	Length (km)	Time Cost (h)
TPS1	(D,6,6)-(DS2,2,3)-(TPS1,1,1)	4.8	0.53
TPS2	(D,6,6)-(DS4,2,3)-(TPS2,1,1)	4.45	0.5
TPS3	(D,6,6)-(TPS3,3,3)	1.88	0.14
TPS4	(D,6,6)-(DS8,1,5)-(TPS4,1,1)	6.68	1.16
TPS5	(D,6,6)-(DS3,3,6)-(DS10,2,6)-(DS11,2,5)-(TPS5,1,1)	10.67	2.46
TPS6	(D,6,6)-(DS4,2,6)-(DS18,2,6)-(DS12,2,6)-(TPS6,1,1)	11.19	2.67
TPS7	(D,6,6)-(DS4,2,6)-(DS18,2,6)-(DS12,2,6)-(DS15,2,4)-(TPS7,1,1)	13.77	3.36

when the power consumption rate P increases, which indirectly indicates the load of the drone increase, the total time cost increases as well because extra time is expended on recharging the battery. In Sim3 and Sim4, a large capacity battery with 50×10^4 J energy and a small capacity battery with 25×10^4 J energy were used to examine our proposed approach. The large capacity battery makes our algorithm generate a path with fewer stations to travel, resulting in the lowest time cost. On the contrary, the small capacity battery makes our proposed approach generate a path with the most number of stations to travel, resulting in a relatively high time cost compared to Sim0, Sim1, and Sim3. In Sim5 and Sim6, we decrease the value of the energy unit δ_e to test its impact on our proposed approach. It turns out that the smaller δ_e would make our approach generate a path with fewer stations to travel due to less numerical error, and the resulting time cost prediction is much more accurate. However, it will put more computational burden on the computer revealed by the run time column in TABLE III. This is because smaller energy units lead to more energy states n in total for each drone station, which significantly increases the complexity of the whole state graph in the end. Since the Dijkstra algorithm needs to loop through all states in the graph to figure out the eventual optimal path, more states lead to a higher computation burden, thus, increasing the computation time. In Table IV, TPS6 is our target packstation. The change of parameter settings in Table IV leads to the same result as analysed in Table III.

Results accounting for DSs' availability: Now, we take the availability of each DS into consideration and simulate our proposed model with Algorithm 1. We assume the simulated drone executes the package delivery task at time 12:00. The

TABLE III: Results under different parameters for TPS7

	P (W)	ϵ_0 (10^4 J)	δ_e (10^4 J)	Station No.	Length (km)	Time Cost (h)	Run Time (s)
Sim0	300	32	5	6	13.8	3.36	0.05
Sim1	350	32	5	8	13.9	3.89	0.04
Sim2	400	32	5	8	14.0	4.57	0.12
Sim3	350	50	5	7	13.5	3.17	0.11
Sim4	350	25	5	9	14.0	4.08	0.04
Sim5	350	32	2.5	6	13.8	4.11	0.21
Sim6	350	32	1	6	13.6	4.05	1.26

TABLE IV: Results under different parameters for TPS6

	P (W)	ϵ_0 (10^4 J)	δ_e (10^4 J)	Station No.	Length (km)	Time Cost (h)	Run Time (s)
Sim0	300	32	5	5	11.19	2.67	0.04
Sim1	350	32	5	7	11.29	3.03	0.05
Sim2	400	32	5	6	11.28	3.36	0.05
Sim3	350	50	5	5	10.57	2.12	0.07
Sim4	350	25	5	8	11.63	3.24	0.04
Sim5	350	32	2.5	5	11.19	3.17	0.12
Sim6	350	32	1	5	11.19	3.17	0.99

TABLE V: DSs' unavailable intervals.

Drone Station	Unavailable Time 1	Unavailable Time 2
DS1	12:00 - 12:08	12:15 - 12:30
DS2	12:17 - 12:34	15:30 - 15:50
DS3	12:02 - 13:23	14:11 - 14:23
DS4	12:20 - 13:00	13:20 - 14:10
DS5	15:27 - 15:35	09:17 - 10:20
DS6	12:45 - 12:55	08:15 - 08:23
DS7	12:15 - 12:28	12:35 - 12:56
DS8	14:05 - 14:23	13:15 - 13:27
DS9	12:36 - 12:49	12:15 - 12:30
DS10	14:45 - 14:58	16:40 - 16:54
DS11	12:15 - 12:28	13:11 - 13:27
DS12	13:23 - 13:40	11:13 - 11:29
DS13	16:21 - 16:44	17:15 - 17:19
DS14	13:15 - 13:22	14:04 - 14:15
DS15	14:46 - 15:23	10:13 - 10:25
DS16	14:13 - 14:25	18:32 - 18:43
DS17	14:00 - 14:13	19:42 - 19:51
DS18	13:32 - 13:54	20:11 - 20:20

parameters set-up remain the same, and each DS associates an availability timetable exhibited in TABLE V, although only two unavailable intervals are considered in TABLE V, the proposed method does not restrict to any number of



Fig. 7: Flight Trajectories for different parameter set-ups



Fig. 8: Flight plans obtained by Algorithm 1.

unavailable intervals. We pick TPS5, TPS6, and TPS7 as our target packstation to conduct the experiments. The global paths generated by Algorithm 1 are presented in Fig.8. Comparing Figs. 6 and 8 we can see that the trajectories from the depot to TPS5, TPS6, and TPS7 become different due to the extra consideration of DSs' availability. To have a better understanding of the effectiveness of Algorithm 1 against the original version that does not account for DSs' availability, we present the full information of the constructed flights in TABLE VI; see the rows with the term of "new". Additionally, we adjust the flight plans shown in TABLE II by adding some waiting time at each DS at which the drone recharges the battery when the arrival time instant falls into the unavailable interval of that DS; see the rows with the term of "old". For each target packstation, we compare the flight plans of the two methods. For TPS3 (and TPS4), the flight plans of the two methods are the same. Looking into TABLE V, we find that the arrival instants at the DSs do not fall into any unavailable intervals. Thus, the generated flight plans by the two methods for TPS3 (and TPS4) are the same. However, for other TPSs, the flight plans of the two methods are different; see TPS1, TPS2, TPS5, TPS6 and TPS7. We find that the new

flight plans may lead to longer flight paths than the old flight plans. However, the benefit is the reduced time to reach the TPSs. For example, to reach TPS1, the new method uses 0.72 hours, while the old version uses 0.81 hours. The main reason for such an advantage comes from the consideration of the availability of DSs in the flight planning phase. Specifically, instead of waiting at an unavailable DS that leads to a shorter flight path, the drone can fly to a relative far but available DS to recharge the battery.

VI. CONCLUSION

In this paper, we investigated the flight planning problem for drone parcel delivery with the assistance of DSs. The objective is to find the optimal flight plan for the drone so that it can reach the TPS in the shortest time. Special attention was paid to the scenarios where the drone recharges its battery at DSs in an optimal manner and when multiple drones use the limited DS resources. We mathematically formulated these problems, but they turn out to be mixed-integer nonlinear programming problems, which are in general difficult to solve. To address the problems, we presented a discretized solution, and a dynamic version of Dijkstra's algorithm was developed

TABLE VI: Comparison of the flight plans obtained by the two proposed methods.

Destination	Trajectory and Residual Energy	Length (km)	Time Cost (h)
TPS1 new	(D,6,6)-(DS8,1,3)-(TPS1,1,1)	5.15	0.72
TPS1 old	(D,6,6)-(DS2,2,3)-(TPS1,1,1)	4.8	0.81
TPS2 new	(D,6,6)-(DS6,2,3)-(TPS2,1,1)	4.45	0.52
TPS2 old	(D,6,6)-(DS4,2,3)-(TPS2,1,1)	4.45	0.83
TPS3 new	(D,6,6)-(TPS3,3,3)	1.88	0.14
TPS3 old	(D,6,6)-(TPS3,3,3)	1.88	0.14
TPS4 new	(D,6,6)-(DS8,1,5)-(TPS4,1,1)	6.68	1.16
TPS4 old	(D,6,6)-(DS8,1,5)-(TPS4,1,1)	6.68	1.16
TPS5 new	(D,6,6)-(DS8,1,3)-(DS10,1,6)-(DS11,2,5)-(TPS5,1,1)	10.74	2.47
TPS5 old	(D,6,6)-(DS3,3,6)-(DS10,2,6)-(DS11,2,5)-(TPS5,1,1)	10.67	3.81
TPS6 new	(D,6,6)-(DS8,1,3)-(DS10,1,6)-(DS12,1,5)-(TPS6,1,1)	11.29	2.67
TPS6 old	(D,6,6)-(DS4,2,6)-(DS18,2,6)-(DS12,2,6)-(TPS6,1,1)	11.19	4.04
TPS7 new	(D,6,6)-(DS8,1,3)-(DS10,1,6)-(DS12,1,4)-(DS14,1,6)-(TPS7,1,1)	13.7	3.52
TPS7 old	(D,6,6)-(DS4,2,6)-(DS18,2,6)-(DS12,2,6)-(DS15,2,4)-(TPS7,1,1)	13.77	4.73

to solve the problem accounting for the unavailability of DSs. The algorithm can quickly find the optimal flight paths, and extensive computer-based experimental results have been presented to demonstrate its effectiveness.

The current approach addresses the optimal flight planning for a single drone. With the current results, the scenario of planning the flights for multiple drones to different TPSs accounting for the unavailable intervals of DSs is worth further study. This will involve the high-level task allocation problem, and the priority of customers can be considered simultaneously.

REFERENCES

- [1] Amazon.com Inc, "Amazon prime air," accessed on 1 Nov. 2021. Online: <http://www.amazon.com/primeair>.
- [2] "SF express approved to fly drones to deliver goods," accessed on 1 Nov. 2021. Online: <https://www.caixinglobal.com/2018-03-28/sf-express-approved-to-fly-drones-to-deliver-goods-101227325.html>.
- [3] "DHL's parcelcopter: changing shipping forever," accessed on 1 Nov. 2021. Online: <https://discover.dhl.com/business/business-ethics/parcelcopter-drone-technology>.
- [4] "UPS testing drones for use in its package delivery system," accessed on 1 Nov. 2021. Online: <https://www.apnews.com/f34dc40191534203aa5d041c3010f6c5>.
- [5] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109, 2015.
- [6] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, Oct 2015.
- [7] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2017.
- [8] 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.
- [9] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "On the min-cost traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 597–621, 2018.
- [10] 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.
- [11] 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.
- [12] A. Trotta, F. D. Andreagiovanni, M. Di Felice, E. Natalizio, and K. R. Chowdhury, "When UAVs ride a bus: Towards energy-efficient city-scale video surveillance," in *IEEE Conference on Computer Communications*, 2018, pp. 1043–1051.
- [13] 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.
- [14] H. Huang, A. V. Savkin, and C. Huang, "Drone routing in a time-dependent network: Toward low-cost and large-range parcel delivery," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1526–1534, 2021.
- [15] 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.
- [16] I. Hong, M. Kuby, and A. T. Murray, "A range-restricted recharging station coverage model for drone delivery service planning," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 198–212, 2018.
- [17] H. Huang and A. V. Savkin, "A method of optimized deployment of charging stations for drone delivery," *IEEE Transactions on Transportation Electrification*, vol. 6, no. 2, pp. 510–518, 2020.
- [18] T. Cokyasar, "Optimization of battery swapping infrastructure for e-commerce drone delivery," *Computer Communications*, vol. 168, pp. 146–154, 2021.
- [19] 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.
- [20] J. Shao, J. Cheng, B. Xia, K. Yang, and H. Wei, "A novel service system for long-distance drone delivery using the 'Ant Colony+A*' algorithm," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3348–3359, 2021.
- [21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [22] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [23] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.
- [24] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles*. Springer, 1997, pp. 203–220.
- [25] A. Felner, S. Kraus, and R. E. Korf, "Kbfs: K-best-first search," *Annals of Mathematics and Artificial Intelligence*, vol. 39, no. 1, pp. 19–39, 2003.
- [26] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.
- [27] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Report No. TR 98-11, Computer Science Department, Iowa State University*, 1998.
- [28] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [29] M. Dorigo, V. Maniezzo, and A. Colormi, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.
- [30] X. Bai, W. Yan, S. S. Ge, and M. Cao, "An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field," *Information Sciences*, vol. 453, pp. 227–238, 2018.
- [31] X. Bai, W. Yan, and M. Cao, "Clustering-based algorithms for multi-vehicle task assignment in a time-invariant drift field," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2166–2173, 2017.
- [32] X. Bai, W. Yan, M. Cao, and D. Xue, "Distributed multi-vehicle task assignment in a time-invariant drift field with obstacles," *IET Control Theory & Applications*, vol. 13, no. 17, pp. 2886–2893, 2019.
- [33] Y. Zhang and A. Khani, "An algorithm for reliable shortest path problem with travel time correlations," *Transportation Research Part B: Methodological*, vol. 121, pp. 92–113, 2019.

- [34] L. Lozano and A. L. Medaglia, "On an exact method for the constrained shortest path problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 378–384, 2013.
- [35] L. D. P. Pugliese and F. Guerriero, "A survey of resource constrained shortest path problems: Exact solution approaches," *Networks*, vol. 62, no. 3, pp. 183–200, 2013.
- [36] M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Tabu search for the time-dependent vehicle routing problem with time windows on a road network," *European Journal of Operational Research*, vol. 288, no. 1, pp. 129–140, 2021.
- [37] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, 2008, pp. 205–216.
- [38] G. Nannicini, D. Delling, L. Liberti, and D. Schultes, "Bidirectional a search for time-dependent fast paths," in *International Workshop on Experimental and Efficient Algorithms*. Springer, 2008, pp. 334–346.



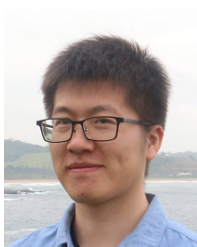
Chao Huang received the B.Sc. degree in automation from the China University of Petroleum, Beijing, China, in 2012, and the Ph.D. degree in control engineering from the University of Wollongong, Wollongong, NSW, Australia, in 2018.

She is currently a Research Assistant Professor with the Department of Industrial and System Engineering, The Hong Kong Polytechnic University (PolyU), Hong Kong. Prior to PolyU, she was a Research Fellow with Nanyang Technological University (NTU), Singapore, and the National Institute of Informatics (NII), Tokyo, Japan. Her research interests include human-machine collaboration, fault tolerant control, mobile robot (EV, UAV), and path planning and control.

Dr. Huang is an Associate Editor of IEEE Transactions on Intelligent Vehicles and IEEE Transactions on Transportation Electrification.



Zhenxing Ming received the B.Sc. degree in Electrical Engineering, from Hubei University of Technology, Wuhan, China, in 2018, and received master degree in Systems and Control from the University of New South Wales, Sydney, Australia, in 2021. He is now an Autonomous-Vehicle engineer at Li Auto Inc, Beijing. His current research interests include vision-based environment perception, navigation, and control of mobile robots.



Hailong Huang received the B.Sc. degree in automation, from China University of Petroleum, Beijing, China, in 2012, and received 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 mobile robots, multi-agent systems, and distributed control.

His current research interests include guidance, navigation, and control of mobile robots, multi-agent systems, and distributed control.