An Efficient Position Reconfiguration Approach for Maximizing Lifetime of Fixed-wing Swarm Drones

Han Liu[1], Tian Liu[1], Mingyue Cui[1], Yunxiao shan[2], Shuai Zhao[1] and Kai Huang[1]

***Abstract*— With the development and application of swarm drones, some researchers have tried to replicating the migration patterns of geese in drones swarm formation to extend their lifetime. However, the problem of performing appropriate position reconfiguration based on the battery energy still remains an unsolved issue. This paper proposes an efficient position reconfiguration approach that reduces the energy consumption imbalance of the swarm and prolongs the lifetime. The approach includes: (1) a two-step MIP (mixed-integer programming)-based optimization method. (2) a two-step heuristic algorithm that can run in pseudo-polynomial time and without the need for optimization solver. The approach provides a complete position reconfiguration solution that determines (i) the number of position reconfiguration; (ii) which drones need to exchange positions in every position reconfiguration; (iii) the length of time to maintain each position before next reconfiguration. Finally, the approach is compared with other three methods in experiments which demonstrate the effectiveness of it.**

I. INTRODUCTION

In recent years, there has been growing interest in aerial swarm drones due to their wide range of potential applications, such as search and rescue missions, transportation, and surveillance [1]–[3]. Swarm drones demonstrate higher efficiency and greater flexibility compared to individual drones during mission execution. However, the flight lifetime of drones is limited due to the constrained battery energy they can carry. Thus, extending the lifetime of swarm drones is crucial for successful mission execution.

In the literature, various approaches have been developed to extend the lifetime of swarm. One aspect involves optimizing the energy management system [4], [5]. Another aspect entails designing a more efficient aircraft structure to extend lifetime [6]. Path planning can also extend lifetime [7], [8]. Last but not least, optimizing embedded system computing and flight systems can extend lifetime [9], [10].

There are many researches that seek inspiration from biomimetics. One example is to mimic the flight process of geese migration [11], [12]. During the migration, the strong lead geese takes the front position in a V-shaped formation, while the weaker geese take the side wing positions [13]. This allows the weaker geese to benefit from the lift created by the lead geese, enabling them to complete the long-distance flight. By mimicking the flight formation of geese migration together with intelligent position reconfiguration mechanism, researches show the overall lifetime of the swarm can be improved [14]–[16].

Fig. 1: Geese flying in formation.

However, applying the mechanism of geese flight to actual drones is challenging. When to perform a reconfiguration and who to change positions in each reconfiguration are not obvious. On one hand, staying too long in high-power locations would cause their batteries to drain more quickly, thus shortening the lifetime of swarm. On the other hand, the reconfiguration itself carries a cost and frequent reconfiguration will also shorten the lifetime. Additionally, the position selection process should consider the specific battery energy and the corresponding power values required for each position. Intuitively, the problem can be resolved by allocating drones with lower battery energy to positions demanding less power consumption and drones with higher energy to positions requiring more power consumption. But how to strike a balance between the lifetime gain of reconfiguration and the energy cost of reconfiguration is still unsolved.

To address these challenges, we propose an efficient position reconfiguration approach. The essence of our approach is to eliminate the imbalance of swarm energy consumption through position reconfiguration, thereby extending the lifetime. Firstly, we formulate the whole problem into an optimization problem by using the time segments. Then, we propose a two-step iterative framework to solve this problem. The two-step consists of position selection and time optimization to produce new position configuration and to allocate time for the new configuration, respectively. For the position selection and time optimization, we construct two MIP (mixed-integer programming)-based optimization methods to address them. Moreover, we propose two heuristic algorithms that can run in pseudo-polynomial time and without the need for optimization solver. Finally, a detailed experimental analysis of our approach is carried out. The main contributions of this paper are as follows:

- We present a formulation of the whole problem by introducing the concept of time segments. So as to facilitate subsequent optimization analysis.
- We propose a two-step iterative framework which

[1]H. Liu, T. Liu, M. Cui, S. Zhao and K. Huang are with the School of Computer Science, Sun Yat-sen University, Guangzhou, China.

[2]Y. Shan is with the School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China.

E-mail: liuh386@mail2.sysu.edu.cn, huangk36@mail.sysu.edu.cn

Manuscript 283 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 10, 2024.

consists of the position selection and time optimization to solve this optimization problem.
- We construct two MIP-based optimization methods to address the position selection and time optimization, respectively.
- We propose two heuristic algorithms that can run in pseudo-polynomial time and without the need for optimization solver.

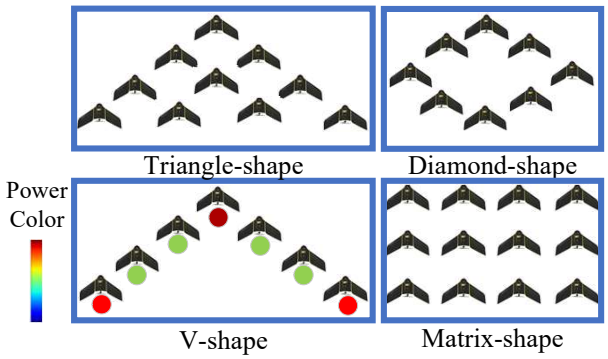


Fig. 2: Some typical swarm formations

II. Related Work

There are various methods to increase the lifetime of swarm drones and improve their energy utilization efficiency. These methods can be categorized as the single drone perspective and the swarm perspective.

From the perspective of a single drone, the methods can be divided into many kinds. First is to improve the energy management systems of the drone. Gang *et al.* [4] introduce a hybrid electric energy management system equipped with power switching. Leonard *et al.* [5] propose a proficient energy management system that incorporates safer electrical contacts and a balancer. Second is to design more efficient aircraft structures. Ma *et al.* [6] present a smart morphing method for adjusting the long-lifetime design of drones and flight mode. Moreover, lifetime can be improved from the perspective of path planning. Ware *et al.* [7] propose a wind-aware planner that exploits wind fields for improved flight performance. Hong *et al.* [8] present an energy-efficient path planning twin-delayed deep deterministic policy gradient model for real-time execution. Last but not least, the lifetime can be improved from the aspect of embedded system computing. Boroujerdian *et al.* [9], [10] examine the role of computing in drones, present analytical models and benchmarks for energy optimization.

From the perspective of swarm, one way is to apply the V-shaped formation mechanism observed in the geese migration. Mirzaeinia *et al.* [13] analyze the aerodynamics of V-shaped flocking in Canada geese and find that switching the lead and tail birds can increase flight endurance and range. Li *et al.* [14] explore the formation of V-shaped flight patterns in bird flocks from a control engineering perspective. Beaver *et al.* [15] present a constraint-driven control algorithm for achieving formation while minimizing energy consumption. Mirzaeinia *et al.* [16] propose a load balancing algorithm that replaces drones within the swarm to optimize energy consumption, thereby enhancing both flight time and range. The papers all attempt to apply the mechanism of geese migration to actual drones flight in order to extend lifetime. But how to achieve a reasonable arrangement and allocation in position reconfiguration still remains an unsolved issue.

III. Problem Statement

A. The model of swarm

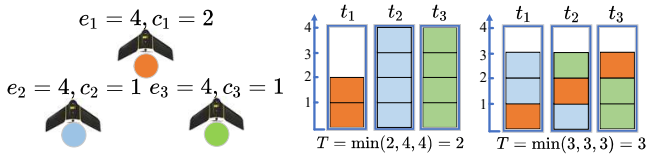
Consider a swarm system consisting of N identical drones, where i denotes the ith drone. Drone i has an initial battery energy e_i and a lifetime t_i. Let $\boldsymbol{e} = \{e_1, ..., e_N\}$ represent the set of battery energy and $\boldsymbol{t} = \{t_1, \ldots, t_N\}$ represent the set of lifetime of all drones. Then, the lifetime of the swarm can be defined as $T = \min\{t_1, \ldots, t_N\}$, which is the lifetime of the drone that depletes its battery energy first. During the flight from the start to the destination, the swarm needs to maintain a formation which consists of N relative positions and can be denoted as $\boldsymbol{rp} = \{rp_1, ..., rp_N\}$. The corresponding power set is $\boldsymbol{c} = \{c_1, ..., c_N\}$.

Fig.2 shows some typical swarm formations like diamond-shape, matrix-shape, triangle-shape and V-shape. Due to the influence of airflow, different positions within the formation require varying power levels. For example, in a V-shaped formation, the leading drone and the trailing drones on both sides require higher power, while the middle section requires less power [17]. In this paper, we do not study the specific relationship between formation positions and power values. What we focus on is the position reconfiguration scheme where the power value is just input. The model about formation positions and power could refer to [18], [19].

Thus, the problem to investigate is how to reduce the energy imbalance of the swarm by reconfiguring positions, in order to prolong the lifetime. To solve this problem, we need to consider: (1) the number of position reconfiguration, (2) which drones need to exchange positions during position reconfiguration, (3) the length of time to maintain each position before next reconfiguration.

B. Motivation example

Fig.3 gives a motivation example of 3 drones in V-shape. Fig.3a shows the battery and power setup. The battery energy values of three drones are all 4. The power value in the head position (orange) is 2 and the values in the tail positions (blue and green) are 1. In case-1 (Fig.3b), the drones keep their positions unchanged, and the head drone runs out of battery energy firstly because of its high power. The vertical axis



(a) battery and power setup (b) case-1 (c) case-2

Fig. 3: A motivation example of 3 drones in V-shape

represents time. A color block means that the drone stays at this position for a time. Thus, the lifetime result is 2. In case-2 (Fig.3c), position reconfiguration is performed during the flight. Three drones take turns flying at the head position for a time and the lifetime result is 3 which is extended compared to case-1.

As shown by this example, the lifetime T can be effectively extended by allocating drones with lower battery energy to positions demanding less power consumption. For complex formations and setup (e.g., different battery energy, increased number of drones and the cost of reconfiguration itself), how to execute position reconfiguration is not obvious. Therefore, a sophisticated allocation scheme is needed to achieve longer lifetime.

IV. Problem Formulation

In this section, we present an optimization formulation of our problem by introducing the time segments.

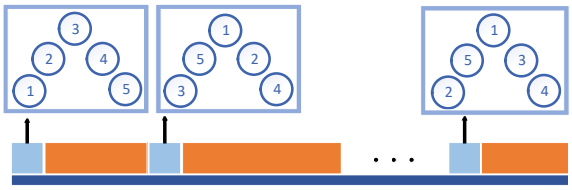


Fig. 4: Time segments with position reconfiguration and flight parts.

The lifetime T could be divided into multiple time segments, each consisting of a position reconfiguration part and a flight part. Let Q represent the total number of segments. Denote j as the jth time segment, so that T_j represents its duration. Fig.4 shows a graphical representation of the time segments. The blue and orange represent the position reconfiguration and the flight parts, respectively. Let t_j^{sta} represent the time point at which segment T_j starts and $\boldsymbol{t}^{sta} = \{t_1^{sta}, ..., t_Q^{sta}\}$ represent the set of time points. Thus, $T_j = t_{j+1}^{sta} - t_j^{sta}$ represents the time interval between two consecutive time points. Denote the $cx_{i,j}$ as the power of drone i in the j time segment.

1) Position reconfiguration part: Let $p_{i,j}$ represent the relative position in formation of drone i in the jth segment. $\boldsymbol{p}_j = \{p_{1,j}, ..., p_{N,j}\}$ represents the set of position information in the jth segment, and $\boldsymbol{p}^{all} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_Q\}$ represents the position information of all time segments. During the position reconfiguration part, the goal for drone i is to move from $p_{i,j}$ to $p_{i,j+1}$. After the position reconfiguration process, the power of the drones will change according to the airflow effect of current position. Let $e_{i,j}^{cha}$ represent the energy consumed by drone i in the position reconfiguration part during the jth segment, while $t_{i,j}^{cha}$ represent the time drone i needs to complete position reconfiguration in the jth segment.

Collision avoidance is critical during the reconfiguration process [20]. The energy consumed by the drones include not only the consumption of position change, but also the consumption caused by collision avoidance. Some drones may consume extra energy even if same positions are maintained in consecutive time segments. Meanwhile, some drones located at the edge of the formation may not consume energy due to collision avoidance.

The time it takes for each drone to complete reconfiguration is also different. Some drones have arrived at the positions ahead of time, while some drones are still in motion. Therefore, we define the reconfiguration time of j segment equals the time when the last drone arrives at the goal position,

$$t_j^{cha} = \max(t_{i,j}^{cha}), \forall i \in \{1, ..., N\}. \tag{1}$$

For the drones that move due to collision avoidance or position change,

$$e_{i,j}^{cha} = \mu \left\| p_{i,j-1} - p_{i,j} \right\| + e^{avoi} + (t_j^{cha} - t_{i,j}^{cha}) cx_{i,j}, \tag{2}$$

where the first two terms represent the energy consumed by position change and collision avoidance. And the last term is the energy of regular flight waiting after early arrival. μ and e^{avo} are constants. For the drones that maintain regular flight during position reconfiguration, with neither position change nor collision avoidance,

$$e_{i,j}^{cha} = t_j^{cha}\, cx_{i,j}, \tag{3}$$

which is the power of the current position multiplied by the reconfiguration time.

2) Flight part: The flight part refers to the time period in which the swarm drones maintain a constant flight speed. During this period, the drone will not perform position reconfiguration, so the power of the position remains unchanged. The energy consumed by drone i during the jth time segment in the flight part is defined as:

$$e_{i,j}^{fly} = cx_{i,j}(t_{j+1}^{sta} - t_j^{sta} - t_j^{cha}). \tag{4}$$

Let $t_{i,j}^{fly}$ represent the time of flight part of drone i in the jth segment. Then we have $t_{i,j}^{fly} = t_{j+1}^{sta} - t_j^{sta} - t_i^{cha}$. After such modeling, the objective function could be formulated as:

$$\underset{Q, \boldsymbol{p}^{all}, \boldsymbol{t}^{sta}}{\arg\max}\ f(Q, \boldsymbol{p}^{all}, \boldsymbol{t}^{sta}) = \min\{t_1, \ldots, t_n\}. \tag{5}$$

It indicates that the core of this problem is to find the optimal Q segments, position information $\boldsymbol{p}^{all}$ and the length of time segments $\boldsymbol{t}^{sta}$ in order to maximize the lifetime T.

V. Proposed Approach

A. The two-step iterative framework

This section introduces the framework of our approach. Fig.5 depicts the architecture diagram of the framework. We use two steps to decompose the problem into two sub-problems. The first step is position selection which determines the relative locations of each drone in the swarm under the current Q segments. And the second step is time optimization where time is allocated to each time segment to achieve the optimal lifetime result. Before the two-step, the Q needs to be determined. We use a for loop to test different Q values which start from 1 and increase 1 at each step. For each Q, the two-step approach is used to compute lifetime result. The inputs are: the relative

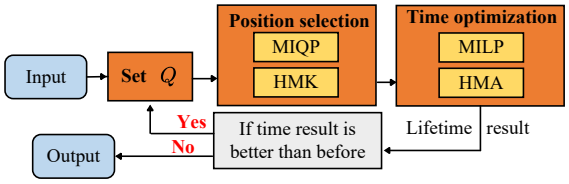


Fig. 5: Architecture diagram of the approach.

positions, power information in the formation, battery energy of each drone, and the time and energy required for position reconfiguration. The outputs are: time segments with corresponding information. As the for loop progresses, the value of T will increase with the increase of Q, but it will reach a peak and then start to decrease. When we observe that T starts to decrease, we can stop the loop and output the peak result.

The utilization of a for loop in the approach has some following reasons. A larger Q means more position reconfiguration which will cost more energy. Initially, as Q increases, the swarm benefits from position reconfiguration. However, as Q continues to increase, the lifetime gains are offset by the associated costs. Therefore, it is essential to find the optimal Q value that strikes the best balance between benefits and costs.

In the next two sections, we will discuss the details of position selection and time optimization.

VI. Position Selection

The selection of positions is the core of our approach. An intuitive idea is that drones with lower remaining energy should be positioned in locations that require less power. However, this greedy algorithm cannot achieve better results.

Our approach for position selection will arrange positions based on the power and proportion of the initial battery energy. This will ensure that the sum of power ratio is as close as possible to the battery ratio, making subsequent optimization easier and yielding better result. Define the sum of power of drone i as:

$$cs_i = \sum_{j=1}^{Q} cx_{i,j}, \forall i \in \{1, \ldots, N\}. \tag{6}$$

Then, our goal could be represented as:

$$\frac{cs_i}{cs_{i-1}} \rightarrow \frac{e_i}{e_{i-1}}, \forall i \in \{2, \ldots, N\}. \tag{7}$$

Equ.(7) aims to align the sum of power ratio as closely as possible with the battery ratio.

In this section, we will introduce the MIQP-based method and the HMK algorithm. The ideas of them to address the problem are the same. The MIQP-based method can obtain better results, but it takes a long time. The HMK algorithm runs in pseudo-polynomial time $\mathcal{O}(QN\log(N))$.

A. Mixed-integer quadratic programming (MIQP)

Firstly, we will discuss the details of MIQP-based method. Since the positions are mutually exclusive, once a drone selects a specific position, other drones can only choose from the remaining positions. Therefore, binary variables need to be introduced to describe this characteristic. Let $z_{i,j,k}$ represent the binary variable in which i and j denote the ith drone and the jth time segments. k is a new order variable. The binary variables can only take the value 0 or 1. $cx_{i,j}$ is the power of drone i in the j time segment. Thus, we can derive the following equation:

$$p_{i,j} = \sum_{k=1}^{N} z_{i,j,k} rp_k, \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., Q\}, \tag{8}$$

$$cx_{i,j} = \sum_{k=1}^{N} z_{i,j,k} c_k, \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., Q\}, \tag{9}$$

which indicate that the power of the drone i in the j time segment has N power options corresponding to the positions of the formation. There are N drones in the swarm, so we need $N \times N$ binary variables to describe these power choices in the j time segment. And all binary variables could be written into the form of a matrix $\boldsymbol{Z}_j$ as:

$$\boldsymbol{Z}_j = \begin{bmatrix} z_{1,j,1} & z_{1,j,2} & \cdots & z_{1,j,N} \\ z_{2,j,1} & z_{2,j,2} & \cdots & z_{2,j,N} \\ \vdots & \ddots & \ddots & \vdots \\ z_{N,j,1} & z_{N,j,2} & \cdots & z_{N,j,N} \end{bmatrix}. \tag{10}$$

where $\boldsymbol{Z}_j \in \mathbb{R}^{N \times N}$. For the all Q time segment $\boldsymbol{Z}^{all} = \{\boldsymbol{Z}_1, ..., \boldsymbol{Z}_Q\} \in \mathbb{R}^{N \times N \times Q}$. In order to describe the mutual exclusion of the formation positions, the following constraints on $\boldsymbol{Z}_j$ are introduced,

$$\sum_{i=1}^{N} z_{i,j,k} = 1, \forall k \in \{1, ..., N\}, \forall j \in \{1, ..., Q\}, \tag{11}$$

$$\sum_{k=1}^{N} z_{i,j,k} = 1, \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., Q\}. \tag{12}$$

The Equs. (11), (12) indicate that both the sum of each row and each column of the $\boldsymbol{Z}_j$ is 1. Each row represents the position selection of the drone, The binary variable equals 1 means the drone selects this position while 0 means the drone doesn't select. Therefore, the mutual exclusion of the formation is reasonably expressed.

Denote set $\boldsymbol{S}^{battery}$ as,

$$\boldsymbol{S}^{battery} = \{s_1, ..., s_N\}, s_i = e_i / e_{\min}. \tag{13}$$

Then, a new vector $\boldsymbol{st} = (st_1, ..., st_N)$ can be obtained,

$$st_i = \frac{\sum_{j=1}^{Q} cx_{i,j}}{s_i} = \frac{cs_i}{s_i}, \forall i \in \{1, ..., N\}, \tag{14}$$

which is the ratio of sum of power and battery energy. Next, it needs to calculate the variance of $\boldsymbol{st}$,

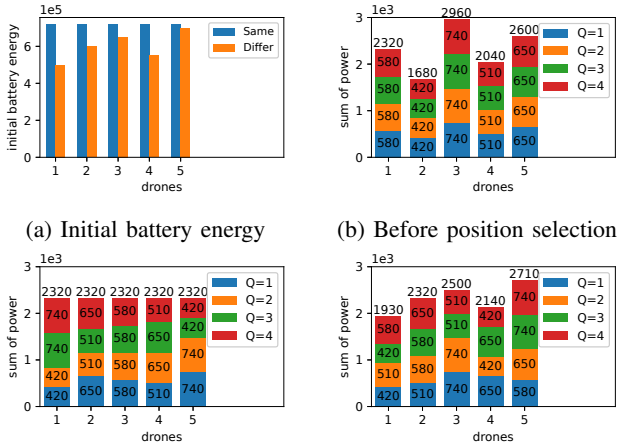
$$\mathbb{D}(\boldsymbol{st}) = \mathbb{E}((\boldsymbol{st})^2) - \mathbb{E}^2(\boldsymbol{st}). \tag{15}$$

Putting things together, a complete mixed-integer quadratic programming (MIQP) model is obtained, summarized below.

$$\arg\min_{\boldsymbol{Z}^{all}} f(\boldsymbol{Z}^{all}) = \mathbb{D}(\boldsymbol{st}) \tag{16}$$

Manuscript 283 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 10, 2024.

subject to: Equs. (9), (10), (11), (12), (13), (14), (15). After this MIQP process, the specific binary values in $\boldsymbol{Z}^{all}$ could be obtained, which represent the position selection result in every time segment.



(a) Initial battery energy (b) Before position selection

(c) After-(same battery) (d) After-(different battery)

Fig. 6: A position selection example

Fig.6 shows an example of the position selection under the same battery energy and the different battery energy. The example input consists of $\boldsymbol{c} = \{580, 420, 740, 510, 650\}W$, $Q = 4$, $\boldsymbol{e} = \{7.2, 7.2, 7.2, 7.2, 7.2\} \times 10^5 J$-(same), $\boldsymbol{e} = \{5, 6, 6.5, 5.5, 7\} \times 10^5 J$-(Differ). Fig.6a shows the initial battery energy. Fig.6b shows the sum of power under no position selection. Fig.6c and Fig.6d show the sum of power after the process in the same battery energy and the different battery energy respectively. It can be seen that after this selection, the sum of power ratio is close to initial battery energy. The essence of this approach is somewhat similar to building with blocks. Each time, it selects a block from the power set $\boldsymbol{c}$ and relies on stacking it Q times to achieve a ratio close to the desired value.

B. Heuristic multiple knapsack algorithm (HMK)

To achieve the goal of Equ.(7), we could transform the position selection problem into a multiple knapsack problem.

The details of the multiple knapsack setup are as follows. The items are the power set $\boldsymbol{c}$ of all Q segments. The knapsacks correspond to the drones. Denote the c^{ave} and e^{ave} as:

$$c^{ave} = \frac{\sum_{i=1}^{N} c_i}{N} = \frac{c^{\text{sum}}}{N}, e^{ave} = \frac{\sum_{i=1}^{N} e_i}{N} = \frac{e^{\text{sum}}}{N}. \quad (17)$$

Then, the capacity of each knapsack Ca_i could be set as:

$$\boldsymbol{Ca} = \{Ca_1, \ldots, Ca_N\}, Ca_i = \frac{e_i}{e^{ave}} c^{ave} Q = \frac{e_i c^{\text{sum}} Q}{e^{\text{sum}}}. \quad (18)$$

The goal of the HMK is

$$\min \max_{i=1,\ldots N} \{ct_i - Ca_i\}, \quad (19)$$

where we focus on minimizing the maximum exceeded knapsack capacity and allocating all items. The items are divided into Q groups. Each group has the power set $\boldsymbol{c} = \{c_1, ..., c_N\}$. When allocating the items, the constraints are that: (1) Only after one group is allocated can the next group be allocated. (2) Each item in a group can only choose one knapsack at a time. And each item must choose one knapsack. This is because the items stand for the positions. Once a drone selects a specific position, other drones can only choose from the remaining positions.

The allocation of items is done greedily by sequentially assigning items (sorted from heavier to lightest) to the knapsacks with the most free capacity. Therefore, the HMK runs in pseudo-polynomial time $\mathcal{O}(QN\log(N))$, which corresponds to the greedy assignment of items. After the allocation, the items (power) packed by each knapsack (drone) represent the positions each drone should go in each segment. Thus, we get the $\boldsymbol{p}^{all}$ result. Equ.(18) indicates that the capacities are the product of the battery ratio and Q times the c^{ave}. This setting allows the sum of the capacities to be equal to the weight of all items,

$$\sum_{i=1}^{N} Ca_i = \sum_{i=1}^{N} \frac{e_i c^{\text{sum}} Q}{e^{\text{sum}}} = c^{\text{sum}} Q. \quad (20)$$

When we allocate, the best result is that no knapsack has excess capacity. At this time, the sum of power ratio is equal to the battery ratio. Through the above setting, the goal of Equ.(7) is equivalently transformed into the goal of Equ.(19). Fig.7 shows a HMK example of 3 drones. The power set is

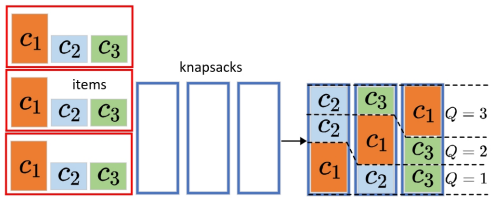


Fig. 7: A HMK example

$c_1 = 2, c_2 = c_3 = 1$. The initial battery is set the same which is $1:1:1$. The number of reconfiguration Q is set to 3. Thus, there has three groups items. Since the ratio of batteries is $1:1:1$, the capacities of three knapsacks are the same (Equ.(18)). According to the constraints, the allocation is done group by group. The heaviest item always choose the knapsack with the most free capacity. After allocation, we find that the sum of power ratio is equal to the battery ratio. For the first knapsack, it contains the items of c_1, c_2, c_2 which means that the first drone will choose the position c_1, c_2, c_2 in $Q = 1, 2, 3$ segment, respectively.

VII. Time Optimization

The next question after receiving the position selection result $\boldsymbol{p}^{all}$ is how to allocate the flight time ($\boldsymbol{t}^{sta}$) for each drone and what the optimal allocation method would be. To solve this, time optimization can be employed where the quality of the position information is evaluated and time is allotted accordingly.

In this section, we will introduce a MILP-based method and the HMA algorithm. The MILP could get the better results with a long computation time. While the HMA could get the results with a time complexity $\mathcal{O}(N)$.

A. Mixed-integer linear programming (MILP)

First, let t_i be the lifetime of drone i that can be achieved before the battery is exhausted. Then it could be expressed as:

$$t_i = \frac{e_i - e_{i,Q}^{cha} - \sum_{j=1}^{Q-1}(e_{i,j}^{cha} + e_{i,j}^{fly})}{cx_{i,Q}} + t_Q^{cha} + t_Q^{sta}. \quad (21)$$

After the $Q-1$ time segments, the remaining energy of drone i is divided by the power of the Qth time segment which is the remaining battery lifetime. And add it to the sum of $t^{cha} + t_Q^{sta}$ which will get the lifetime of drone i. In this way, t_i can be written as a function of $\boldsymbol{t}^{sta}$.

Next, it needs to introduce some constraints,

$$e_i - e_{i,Q}^{cha} - \sum_{j=1}^{Q-1}(e_{i,j}^{cha} + e_{i,j}^{fly}) \geqslant 0. \quad (22)$$

The Equ. (22) is to make sure the remaining energy is greater than zero. Then we need some time constraints like,

$$t_j^{sta} = 0, j = 1, \quad (23)$$

$$t_j^{sta} + t_j^{cha} \leqslant t_{j+1}^{sta}, \forall j \in \{1, ..., Q-1\}, \quad (24)$$

$$t_j^{sta} + t_j^{cha} \leqslant t_i, \forall i \in \{1, ..., N\}, j = Q. \quad (25)$$

The time constraints ensure that the first time segment starts at 0 (23), the start point of each time segment does not exceed the start point of the following segment (24), and the start point of the final time segment does not exceed the lifetime of each drone in the swarm (25). These constraints are based on practical experience and serve to prevent unrealistic results such as negative time values. Finally, the objective function and constraints could be written together and we obtain a model summarized below:

$$\arg\max_{\boldsymbol{t}^{sta}} f(\boldsymbol{t}^{sta}) = \min\{t_1, \ldots, t_n\} \quad (26)$$

subject to: Equs. (21), (22), (23), (24), (25). Note that this problem is a maximize minimum problem that could be solved by mixed-integer linear programming (MILP) [21]. After this MILP process, the specific values of $\boldsymbol{t}^{sta}$ which represent the time point of each time segment can be obtained. Thus, we know when the positions should be changed, and how long to keep it before next time.

B. Heuristic minimum allocation algorithm (HMA)

The idea of HMA is to let each drone stay in its power position for the same time, and then choose the minimum lifetime as the basis for $\boldsymbol{t}^{sta}$ allocation.

Firstly, we let each drone spend an equal amount of time in each power position. The lifetime of each drone is,

$$t_i = \frac{Q(e_i - \sum_{j=1}^{Q} e_{i,j}^{cha})}{cs_i} + \sum_{j=1}^{Q} t_{i,j}^{cha}, \forall i \in \{1, ..., N\} \quad (27)$$

Equ.(27) means that after subtracting the energy consumed by reconfiguration of all segments, the remaining energy is allocated to each power position in equal proportions.

Denote i_δ as the drone with the minimum lifetime. Then, the minimum lifetime t_{i_δ} is selected as the basis for each t_j^{sta},

$$t_{i_\delta} = \min(t_i), \forall i \in \{1, ..., N\}, \quad (28)$$

$$t_j^{sta} = \frac{t_{i_\delta}(j-1)}{Q}, \forall j \in \{1, ..., Q\}. \quad (29)$$

Thus, we get the results of $\boldsymbol{t}^{sta}$. The HMA runs in polynomial time $\mathcal{O}(N)$ which corresponds to the process computation of Equ.(27). The reasons for choosing the t_{i_δ} are that: With such allocation, the other drones will not run out of energy before the last Q segment. Therefore, the final results must be reasonable.

VIII. Experiments

This section provides an experimental analysis of our approach. Our experiments are conducted on a PC with an Inter Core-i7 12700 CPU under 32GB of RAM. The methods are as follows:

(1) Greedy_one: A greedy algorithm. The number of reconfiguration Q is fixed to 3. Each reconfiguration, the top 1/3 of the drones with the most remaining energy exchange positions with the 1/3 of the drones with the least remaining energy. The time of exchange is one quarter of battery time. For example, if the battery energy can sustain normal flight time for 12 minutes, the three exchange times are 3, 6, and 9 minutes respectively.

(2) Greegy_two: A greedy algorithm. The Q is not fixed. The reconfiguration will happen when two conditions are satisfied: (a) The energy gap between the drone with the most energy and the drone with the least energy is higher than a value (we set 50000J in our experiments). (b) The power of the drone with the most energy is less than the power of the drone with the least energy.

(3) RM_A: A replacement mechanism proposed by [16].

(4) Two_mip: Our MIQP-MILP approach solved by Gurobi [22]. (In order to prevent the calculation of MIQP from taking too long, we set an upper limit of 5 minutes. When 5 minutes is reached, the currently found optimal solution in MIQP is directly output.)

(5) HMK_HMA: Our heuristic algorithm.

We first define the experiment setup where the number of drones N are evaluated. We evaluate how the lifetime T changes according to the number of drones N. A total of 12 different numbers from 3 to 100 for the number N (3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100) are selected. For comparison, a "diff" and a "same" battery energy datasets are generated. After specifying an upper

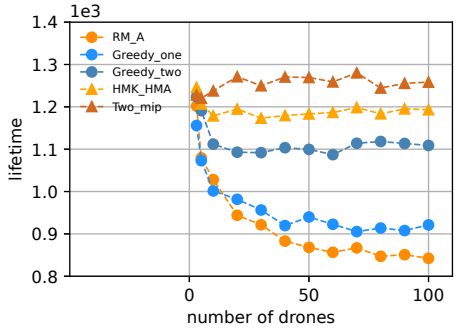
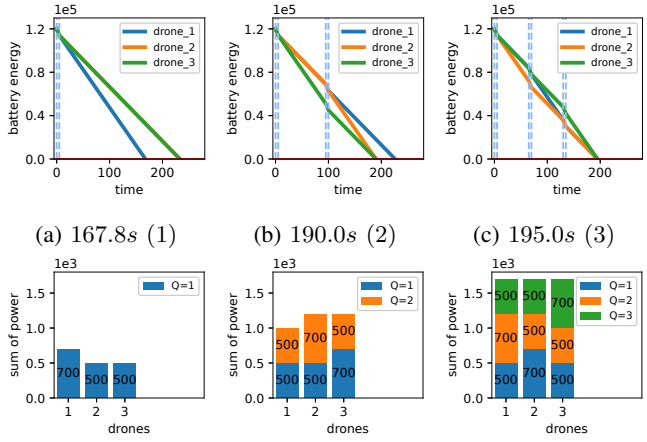


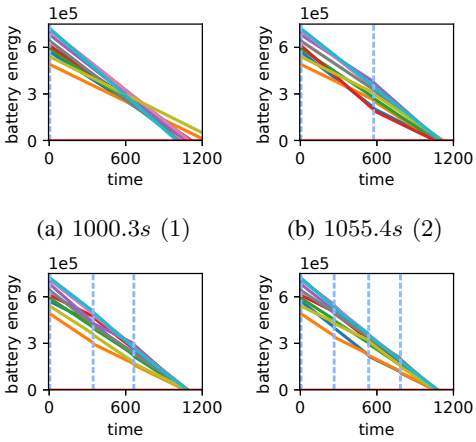
Fig. 8: Lifetime results w.r.t. number of drones N



(a) $167.8s$ (1) (b) $190.0s$ (2) (c) $195.0s$ (3)

(d) $Q = 1$ (e) $Q = 2$ (f) $Q = 3$

Fig. 9: The battery energy change and sum of power under different Q of 3 drones. "$195.0s$ (3)" indicates that $T = 195.0s, Q = 3$.



(a) $1000.3s$ (1) (b) $1055.4s$ (2)

(c) $1066.3s$ (3) (d) $1057.2s$ (4)

Fig. 10: The energy change under different Q of 10 drones.

bound and a lower bound, the "diff" battery energy values are randomly generated using a uniform distribution within the two bounds. The "same" battery energy values are all equal to the mean of the two bounds. The battery energy time of the mean of the two bounds is 20 minutes. The power set is also randomly generated using a uniform distribution with an upper bound and a lower bound. The formation positions are randomly generated using a uniform distribution within an area. For each selected value of N, we generate 20 times in total. Thus, we have a 480 ($12 \times 2 \times 20$) group of datasets. To more accurately simulate the reconfiguration process, we employ the Reciprocal Velocity Obstacles (RVO) [23] algorithm enabling us to compute the energy and time consumed by each drone. The energy consumed by position changes is set 1.5∼1.8 times the energy consumed by regular flight. This will drive the approach to minimize the Q to obtain good results.

Fig.8 shows the lifetime results with respect to the number of drones. For every N, mean values are reported for each group of 40 randomly generated datasets. From the figures, we can make the following observations: (1) When the N is small (3, 5), there is not much difference between the lifetime of five methods. This shows that the greedy algorithm still works well when N is small. (2) As N increases, the lifetime of Greegy_one and the RM_A decreases faster. This shows that Greegy_one and RM_A can not cope with the larger N. (3) The lifetime of Greegy_two is better than the Greegy_one. This is because we do not limit the Q in Greegy_two. Once the condition is satisfied, the reconfiguration will happen in Greegy_two. The disadvantage of the Greegy_two is that it considers the problem from an individual perspective rather than from a swarm perspective. (4) The lifetime of the Two_mip is better than the HMK_HMA. The Two_mip takes a lone time for computation which indeed leads to better results.

Fig.9 demonstrate the role of position selection and time optimization of 3 drones case. Fig.9a, 9b, 9c show the change in battery energy over time. The blue dashed lines indicate position reconfiguration parts, and the portions between them represent flight parts. Fig.9d, 9e, 9f show the sum of power of each Q segment. In Fig.9d, 9e, there is still a gap between the battery energy ratio and the ratio of sum of power. But in Fig.9f, the ratio of sum of power is equal to the ratio of battery energy which is 1:1:1. In Fig.9c, three drones run out of battery energy simultaneously and $T = 195s$ which is the best. The time optimization is just like a hand that adjusts the positions of the blue dashed lines. The closer the ratio, the better the lifetime the time optimization can get.

Fig.10 show the process of lifetime results rising first and then falling. The figures are the variation of energy from 1 to 4 Q segments of 10 drones. We can observe that: (1) As the Q increases, the issue of unbalanced energy consumption is resolved, and the lifetime results are getting closer. (2) The T increases first, but it will reach a peak at $Q = 3$ and then start to decrease. This is because, with the increase of Q, the imbalance of energy consumption can be effectively addressed. However, due to the cost of reconfiguration, excessive reconfiguration may lead to a decrease in lifetime. The max lifetime of Q is determined by the initial unevenness of battery energy, the value of power, and the cost of reconfiguration. If we can make the ratio of sum of power as close as possible to the ratio of the battery energy when Q is small, then a larger Q is a waste. If the cost of reconfiguration becomes higher, even if the lifetime results of drones become more balanced, the lifetime of swarm may not necessarily be better.

Table.I shows the computation time with respect to the number of drones N. It can be seen that the computation time of the Greegy_one, the Greegy_two and the RM_A are all within 1s to 10s. They increase almost linearly with N. The lifetime results of Two_mip are all around 300s. This is because we set a limit of 300s for MIQP in the initial setup. We find that the calculation of MIQP takes a long time in experiments. The branch-and-bound algorithm for solving the mixed integer programming will consume a lot of time to find the optimal solution. Compared to the

TABLE I: The computation time(s) w.r.t. number of drones

	The number of drones N				
	20	40	60	80	100
Greedy_one	1.276	2.756	4.657	5.480	7.527
Greedy_two	1.952	3.876	5.564	7.494	8.990
RM_A	1.545	2.796	3.669	4.482	6.764
Two_mip	901.26	1203.58	902.18	1500.94	1201.71
HMK_HMA	0.01	0.02	0.03	0.04	0.05

TABLE II: The average improvement

	The average improvement		
	Greedy_one	Greegy_two	RM_A
Two_mip	28.90%	11.50%	33.25%
HMK_HMA	22.35%	6.71%	27.54%

Two_mip, the computation time results of HMK_HMA are greatly reduced. The computation result of HMK_HMA is almost 10000x times lower than Two_mip and 100x times lower than Greegy_one, the Greegy_two and the RM_A.

Table.II shows the average improvement of the Two_mip and the HMK_HMA compared with the other three methods. We can observe that:(1) The average improvement of the Two_mip is 28.90%, 11.50% and 33.25% respectively compared to the other three methods. (2) The average improvement of the HMK_HMA is 22.35%, 6.71% and 27.54% respectively compared to the other three methods. The Two_mip is better than the HMK_HMA in lifetime.

Therefore, the Two_mip method is more suitable for tasks that run on the central server and have high requirements on lifetime results. The HMK_HMA is more suitable for tasks on embedded computing platforms where sub-optimal results are acceptable.

IX. Conclusion

In this paper, an efficient position reconfiguration approach is proposed. A two-step iterative framework which consists of position selection and time optimization is presented. For the two problems, we construct two MIP-based optimization methods and two heuristic algorithms to address them. The experimental results demonstrate the effectiveness of the proposed approach.

References

[1] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho *et al.*, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert-like environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1576–1583, 2018.

[2] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[3] M. Petrlík, V. Vonásek, and M. Saska, "Coverage optimization in the cooperative surveillance task using multiple micro aerial vehicles," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 4373–4380.

[4] B. G. Gang and S. Kwon, "Design of an energy management technique for high endurance unmanned aerial vehicles powered by fuel and solar cell systems," *International Journal of Hydrogen Energy*, vol. 43, no. 20, pp. 9787–9796, 2018.

[5] J. Leonard, A. Savvaris, and A. Tsourdos, "Energy management in swarm of unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 74, pp. 233–250, 2014.

[6] T. Ma, Y. Liu, D. Yang, Z. Zhang, X. Wang, and S. Hao, "Research on the design of smart morphing long-endurance uavs," *The Aeronautical Journal*, vol. 125, no. 1283, pp. 22–41, 2021.

[7] J. Ware and N. Roy, "An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1507–1514.

[8] D. Hong, S. Lee, Y. H. Cho, D. Baek, J. Kim, and N. Chang, "Energy-efficient online path planning of multiple drones using reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 9725–9740, 2021.

[9] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, "Mavbench: Micro aerial vehicle benchmarking," in *IEEE/ACM international symposium on microarchitecture (MICRO)*, 2018, pp. 894–907.

[10] B. Boroujerdian, H. Genc, S. Krishnan, B. P. Duisterhof, B. Plancher, K. Mansoorshahi, M. Almeida, W. Cui, A. Faust, and V. J. Reddi, "The role of compute in autonomous micro aerial vehicles: Optimizing for mission time and energy efficiency," *ACM Transactions on Computer Systems (TOCS)*, vol. 39, no. 1-4, pp. 1–44, 2022.

[11] J. Martinez-Ponce, C. Urban, S. F. Armanini, R. K. Agarwal, and M. Hassanalian, "Aerodynamic analysis of v-shaped flight formation of flapping-wing drones: Analytical and experimental studies," in *AIAA SCITECH Forum*, 2022, p. 1979.

[12] E. J. Billingsley, M. Ghommem, R. Vasconcellos, and A. Abdelkefi, "Biomimicry and aerodynamic performance of multi-flapping wing drones," in *AIAA Scitech Forum*, 2021, p. 0227.

[13] A. Mirzaeinia and M. Hassanalian, "Energy conservation of v-shaped flocking canada geese through leader and tail switching," in *AIAA Propulsion and Energy Forum*, 2019, p. 4152.

[14] X. Li, Y. Tan, J. Fu, and I. Mareels, "On v-shaped flight formation of bird flocks with visual communication constraints," in *International Conference on Control & Automation (ICCA)*. IEEE, 2017, pp. 513–518.

[15] L. E. Beaver, C. Kroninger, M. Dorothy, and A. A. Malikopoulos, "A constraint-driven approach to line flocking: The v formation as an energy-saving strategy," *arXiv preprint arXiv:2209.11664*, 2022.

[16] A. Mirzaeinia, M. Hassanalian, K. Lee, and M. Mirzaeinia, "Energy conservation of v-shaped swarming fixed-wing drones through position reconfiguration," *Aerospace Science and Technology*, vol. 94, p. 105398, 2019.

[17] H. Thien, M. A. Moelyadi, and H. Muhammad, "Effects of leaders position and shape on aerodynamic performances of v flight formation," *arXiv preprint arXiv:0804.3879*, 2008.

[18] Q. Zhang and H. H. Liu, "Aerodynamics modeling and analysis of close formation flight," *Journal of Aircraft*, vol. 54, no. 6, pp. 2192–2204, 2017.

[19] G. Yuan, J. Xia, and H. Duan, "A continuous modeling method via improved pigeon-inspired optimization for wake vortices in uavs close formation flight," *Aerospace Science and Technology*, vol. 120, p. 107259, 2022.

[20] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annual Reviews in Control*, vol. 48, pp. 147–164, 2019.

[21] J. C. Smith and Z. C. Taskin, "A tutorial guide to mixed-integer programming models and solution techniques," *Optimization in Medicine and Biology*, pp. 521–548, 2008.

[22] G. Optimization, *Gurobi Optimizer Reference Manual*, 2021. [Online]. Available: https://www.gurobi.com/documentation/

[23] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE international conference on robotics and automation (ICRA)*, 2008, pp. 1928–1935.