drones

MDPI

Article

Coverage Path Planning for UAVs: An Energy-Efficient Method in Convex and Non-Convex Mixed Regions

Li Wang [1], Xiaodong Zhuang [1], Wentao Zhang [1], Jing Cheng [2] and Tao Zhang [1,*]

[1] School of Software, Northwestern Polytechnical University, Xi'an 710072, China; wang.li@nwpu.edu.cn (L.W.); zxdong@mail.nwpu.edu.cn (X.Z.); wentao_zhang@mail.nwpu.edu.cn (W.Z.)
[2] School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China; chengjing@xatu.edu.cn
* Correspondence: tao_zhang@nwpu.edu.cn

Abstract: As an important branch of path planning, coverage path planning (CPP) is widely used for unmanned aerial vehicles (UAVs) to cover target regions with lower energy consumption. Most current works focus on convex regions, whereas others need pre-decomposition to deal with non-convex or mixed regions. Therefore, it is necessary to pursue a concise and efficient method for the latter. This paper proposes a two-stage method named Shrink-Segment by Dynamic Programming (SSDP), which aims to cover mixed regions with limited energy. First, instead of decomposing and then planning, SSDP formulates an optimal path by shrinking the rings for mixed regions. Second, a dynamic programming (DP)-based approach is used to segment the overall path for UAVs in order to meet energy limits. Experimental results show that the proposed method achieves less path overlap and lower energy consumption compared to state-of-the-art methods.

Keywords: coverage path planning; unmanned aerial vehicle; non-convex region; shrinking rings; energy consumption

Citation: Wang, L.; Zhuang, X.; Zhang, W.; Cheng, J.; Zhang, T. Coverage Path Planning for UAVs: An Energy-Efficient Method in Convex and Non-Convex Mixed Regions. *Drones* **2024**, *8*, 776. https://doi.org/10.3390/drones8120776

Academic Editor: Oleg Yakimenko

Received: 29 October 2024
Revised: 19 December 2024
Accepted: 19 December 2024
Published: 20 December 2024

1. Introduction

Unmanned aerial vehicles (UAVs) have become widely used in applications such as aerial photography [1] and rescue [2–4]. Due to restrictions on permitted flight regions and battery capacity, there has been extensive research on UAV path planning [5–7]. Coverage path planning (CPP) is a critical subset of UAV path planning. The primary challenge lies in designing feasible routes for UAVs within target regions to ensure thorough coverage of the entire space while optimizing energy consumption. Key objectives include minimizing redundant coverage, avoiding collisions between UAVs, and achieving specific tasks such as monitoring crop conditions or detecting targets.

Most existing approaches to CPP focus on obstacle avoidance [8–10], region decomposition [11,12], energy consumption [13,14], and UAV collaboration [15,16]. The methods for covering a region in these studies can be broadly categorized into two patterns, namely, back-and-forth (B&F) and spiral. Mu et al. [8] introduced the continuity constraint function within B&F to reduce coverage overlap. Nielsen et al. [11] showed that the travel direction in B&F should be aligned vertically relative to the shortest edge of the polygon. Cabreira et al. [13] proposed a spiral coverage method and optimized the speed along straight segments to minimize energy consumption. However, these methods are only suited for convex polygons; when applied to non-convex polygons, region decomposition is required. For example, Choset et al. [9] used a sweep line to partition regions, and Nielsen et al. [11] extended the edges of concave angles to decompose non-convex regions.

The studies mentioned above focused solely on single-region scenarios; however, in tasks such as disaster relief, the presence of rivers and mountains often leads to the division of the target area into several smaller regions. Using UAVs to cover multiple regions is generally divided into two steps: determining the regional access order, and determining

the path within each region. The first step is similar to the traveling salesman problem (TSP). Based on this, Vasquez-Gomez et al. [17] proposed a two-step method called TSPP. First, TSPP estimates the cost using the distance between the centroids of polygons and applies a genetic algorithm (GA) to determine the visiting order. Second, it computes the B&F paths for each region based on this order. Xiao et al. [18] proposed a clustering algorithm called CSCA to assign regions to UAVs. In CSCA, UAV starting points are initialized as cluster centers. By adding regions to the clusters and updating them, each UAV is assigned at least one region. Then, the nearest-to-end policy and B&F pattern are used to determine the regional access order and intra-region paths. Du et al. [19] proposed a method that mixes B&F and spiral paths to ensure complete coverage of regions. In this method, the B&F path is generated by dividing the polygon into multiple strips based on the sensor's width, while the spiral path is created using the perpendicular bisectors of the polygon's edges. However, these algorithms only address energy consumption in the results, and do not take into account the constraints imposed by the drone's battery capacity.

In fact, the energy constraint should be included in the problem's assumptions. Xie et al. [20] defined the problem as an energy-constrained multiple TSP-CPP (EMTSP-CPP). This EMTSP-CPP requires the UAVs to collaboratively complete the coverage task under limited energy, with each UAV departing from the warehouse and returning after completing its task. Xie et al. used a GA combined with grouping to assign regions to each UAV. To achieve a similar objective, Ahmed et al. [21] employed a greedy algorithm integrated with a heuristic approach. Jiang et al. [22] used an improved fuzzy c-means clustering algorithm to assign regions, but did not account for intra-region paths. Unfortunately, the coverage paths resulting from these methods are not optimal. For one, if a UAV finishes covering a region and is closer to the next region than to its starting point, it should continue to the next region if it has enough energy. In addition, a single UAV may not be able to cover a large region of interest on its own. To address these issues, Luna et al. [23] proposed the BINPAT algorithm, which first designs a globally optimal coverage path and then divides it among the UAVs. Compared to regional division, route division considers UAV incorporation rather than merely ensuring that they do not interfere with each other. While BINPAT is implemented using the Jonker–Volgenant algorithm [24], its time complexity of $O(n^3)$ results in significant computational time when the number of waypoints is large. Furthermore, BINPAT simplifies the energy constraint to a distance constraint without accounting for the energy consumption caused by UAVs needing to make turns.

This paper aims to determine the optimal coverage paths for energy-constrained UAVs in convex and non-convex mixed regions. To achieve this goal, a two-stage method called Shrink-Segment by Dynamic Programming (SSDP) is proposed. In its first stage, SSDP uses a polygon offset-based shrink method to generate shrinking rings. With the objective of minimizing energy consumption, shrinking rings from each region are then connected sequentially from the outermost to the innermost, resulting in an optimal coverage path. In the second stage, a dynamic programming (DP)-based approach is used to segment the overall path for UAVs. The principal contributions of this research are as follows:

1. A shrink method is proposed which is suitable for both convex and non-convex regions. This method reduces operational complexity is directly applicable to non-convex regions, thereby avoiding the need for pre-decomposition.
2. A DP-based approach is proposed to optimally segment the overall path among UAVs, allowing some UAVs to remain on standby rather than adhering to the principle of equal or maximum load distribution.
3. The proposed SSDP ensures that UAVs meet their respective energy constraints while minimizing overall energy consumption.

The remainder of this paper is organized as follows: the problem formulation and method overview are introduced in Section 2; Section 3 provides details on the shrink method, followed by the DP-based approach in Section 4; experimental results and discussion are presented in Section 5; finally, Section 6 presents our conclusion.

2. Problem Definition and Method Overview

This section employs mathematical symbols to describe the coverage path planning problem for UAVs. First, the inputs and outputs of the problem are defined. Second, the constraints are established to ensure the feasibility of path planning and comprehensive region coverage. Finally, the overall framework of the proposed SSDP method is outlined.

2.1. Problem Formulation

In this study, we deploy N homogeneous UAVs $\{u_n, n \in [1, N]\}$, which all start from a warehouse w_0, to scan M regions $\{\boldsymbol{R}_m, m \in [1, M]\}$. Region $\boldsymbol{R}_m = (\boldsymbol{v}_m^1, \boldsymbol{v}_m^2, ..., \boldsymbol{v}_m^{r_m})$ can be of any type, either convex or non-convex, with $\boldsymbol{v}_m^{r_m} = (x_m^{r_m}, y_m^{r_m})$ indicating the coordinates of the r_m-th vertex. Here, $x_m^{r_m}$ and $y_m^{r_m}$ denote the horizontal and vertical coordinate, respectively. The number of vertices in each region is not fixed, with r_m being determined case-by-case.

We assume that UAV u_n flies at a constant speed V and has an energy constraint E_n. Because all UAVs are of the same type, their energy consumption is identical. Let $E = E_1 = E_2 = \cdots = E_n$ represent this common energy consumption. The sensor model of UAV is shown as Figure 1, where L_1 represents the longitudinal scanning length of the sensor camera; correspondingly, L_2 represents the lateral scanning length. To ensure thorough scanning of these target regions, it is required that the images captured by the UAVs have a certain overlap length, denoted as $L_1^{'}$ and $L_2^{'}$ for the longitudinal and lateral directions, respectively.

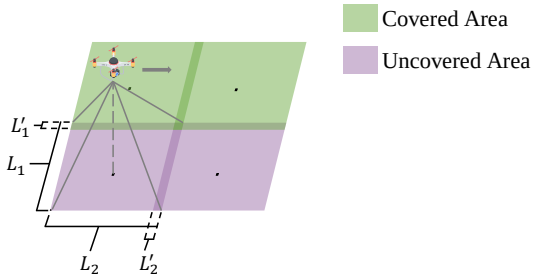


Figure 1. UAV sensor model.

Each UAV is authorized to access all regions. Additionally, considering that the core objective of the coverage task is to obtain comprehensive ground data, the UAVs fly at a sufficient altitude, eliminating the need to avoid any obstacles.

The task result is to distribute a series of waypoints $\{\boldsymbol{W}_n, n \in [1, N]\}$ to the UAV swarm that provides coordinated coverage of all target regions. In this context, $\boldsymbol{W}_n = (\boldsymbol{w}_n^1, \boldsymbol{w}_n^2, ..., \boldsymbol{w}_n^{k_n})$ represents the flight path for UAV u_n, with $\boldsymbol{w}_n^{k_n} = (x_n^{k_n}, y_n^{k_n})$ being the k_n-th waypoint.

2.2. Problem Constraints

This section analyzes the constraints that must be followed when UAVs cover mixed regions. Moreover, an optimization model is developed to enable the UAVs to complete the coverage task with minimal energy consumption.

Constraint 1 (C1): All UAVs are required to depart from the warehouse $\boldsymbol{w}_0$ before executing their tasks and return to the warehouse upon completion. Therefore, $\boldsymbol{W}_n = (\boldsymbol{w}_n^1, \boldsymbol{w}_n^2, ..., \boldsymbol{w}_n^{k_n})$ must satisfy the following condition:

$$\forall n \in [1, N], \boldsymbol{w}_n^1 = \boldsymbol{w}_n^{k_n} = \boldsymbol{w}_0 \tag{1}$$

where $\boldsymbol{w}_n^1$ represents the first waypoint of UAV u_n and $\boldsymbol{w}_n^{k_n}$ represents its last waypoint.

Constraint 2 (C2): To ensure complete coverage, each region must be covered by at least one UAV. We define a variable $c_{n,m}$ to indicate whether u_n is involved in the coverage of $\boldsymbol{R}_m$. The value of $c_{n,m}$ is defined as shown below.

$$\exists \boldsymbol{w}_n^i \in \boldsymbol{W}_n, i \in [2, k_n - 1], c_{n,m} = \begin{cases} 1, & \boldsymbol{w}_n^i \text{ located within } \boldsymbol{R}_m \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Furthermore, $c_{n,m}$ must satisfy the following condition:

$$\forall m \in [1, M], \sum_{n=1}^{N} c_{n,m} \geq 1. \tag{3}$$

Constraint 3 (C3): To avoid multiple coverage, a UAV can cover a maximum of M regions; thus, the variable $c_{n,m}$ must satisfy the following constraint:

$$\forall n \in [1, N], 0 \leq \sum_{m=1}^{M} c_{n,m} \leq M. \tag{4}$$

Constraint 4 (C4): Due to the maximum energy consumption E of the UAV, the waypoint set $\boldsymbol{W}_n$ of u_n must satisfy the following constraint:

$$\forall n \in [1, N], \lambda_1 \times \sum_{i=1}^{k_n-1} \mathrm{d}(\boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1}) + \lambda_2 \times \sum_{i=2}^{k_n-1} \mathrm{a}(\boldsymbol{w}_n^{i-1}, \boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1}) \leq E \tag{5}$$

where λ_1 represents the weight of distance in energy consumption and λ_2 denotes the weight of turning in energy consumption; according to [25], the values of the coefficients λ_1 and λ_2 have been determined as 0.1072 and 0.0104, respectively. Additionally, $\mathrm{d}(\boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1})$ represents the path distance for u_n from $\boldsymbol{w}_n^i$ to $\boldsymbol{w}_n^{i+1}$, while $\mathrm{a}(\boldsymbol{w}_n^{i-1}, \boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1})$ represents the turning angle for u_n travels from $\boldsymbol{w}_n^{i-1}$ through $\boldsymbol{w}_n^i$ to $\boldsymbol{w}_n^{i+1}$.

In this paper, we aim to plan a series of paths that allows the UAVs to cover these target regions while minimizing energy consumption as much as possible. The total energy consumption of the UAV swarm is determined by the sum of the path distances and the turning angles of each UAV, expressed as follows:

$$p = \lambda_1 \times \sum_{n=1}^{N} \sum_{i=1}^{k_n-1} \mathrm{d}(\boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1}) + \lambda_2 \times \sum_{n=1}^{N} \sum_{i=2}^{k_n-1} \mathrm{a}(\boldsymbol{w}_n^{i-1}, \boldsymbol{w}_n^i, \boldsymbol{w}_n^{i+1}). \tag{6}$$

Finally, the mission optimization process can be formulated as

$$\begin{aligned} &\min \ \ p \\ &\text{s.t. } \ \mathrm{C1}, \mathrm{C2}, \mathrm{C3}, \mathrm{C4} \end{aligned} \tag{7}$$

Calculating the Euclidean distance between waypoints is required in Equation (6). This introduces a nonlinear objective function, making it a nonlinear programming (NLP) problem. Furthermore, the selection of coverage paths involves combinatorial complexity. As the number of regions increases, the number of possible path combinations grows exponentially. Therefore, this problem is also classified as NP-hard, indicating high computational complexity.

2.3. Method Overview

We propose a two-stage method to address this problem, which we name SSDP. In the first stage, the shrink method generates an optimal coverage path through three steps: first, shrinking rings for each region are generated using the polygon offset method; second, the shrinking rings are broken into waypoints; and finally, these waypoints are connected to form the optimal coverage path. In the second stage, we employ a DP-based approach combining a two-dimensional matrix with memoization to segment the path into individual flight paths for the UAVs. The flowchart of SSDP is shown in Figure 2.

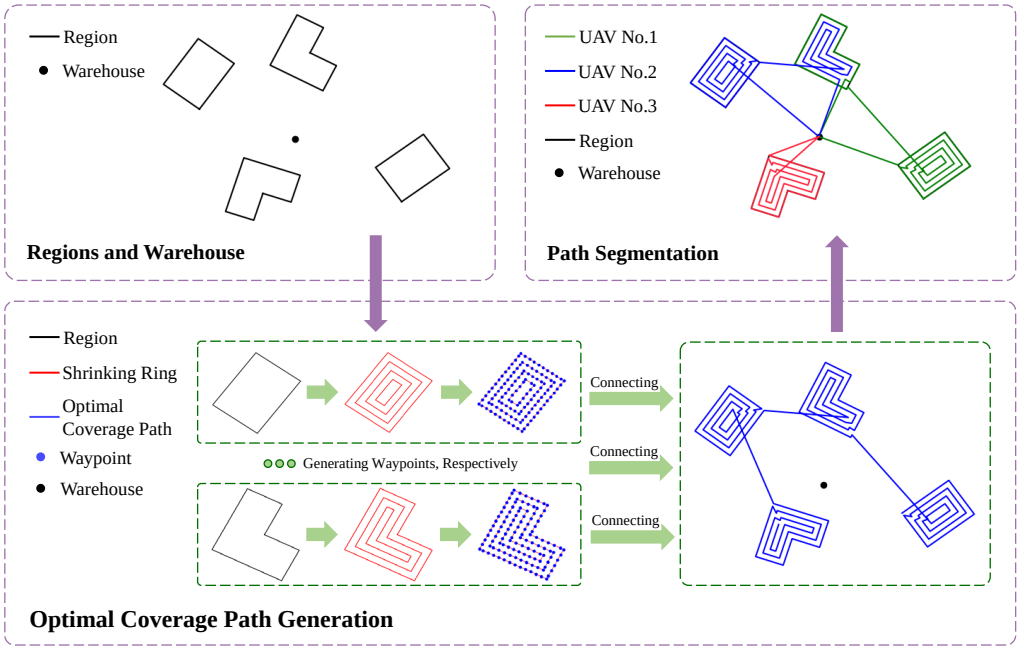


Figure 2. Flowchart of SSDP.

3. Optimal Coverage Path Generation

This section provides a detailed explanation of the first stage of SSDP. The core idea is to use iterative polygon offsetting operations to generate an optimal coverage path for all target regions, whether convex or non-convex. The following subsections discuss the principles of generating shrinking rings through polygon offset, strategies for evenly distributing waypoints, and methods for orderly connection of waypoints between rings. These steps lay a solid foundation for the subsequent path segmentation and allocation processes.

3.1. Shrinking Ring Generation

First, we take the boundary of the region as the initial shrinking ring. Using the polygon offset method, the first ring is shifted inward by a distance of L_1–$L_1^{'}$ to generate the second shrinking ring. The same method is then applied to generate the third shrinking ring. In certain cases, offsetting the previous shrinking ring may result in multiple new shrinking rings. We use a depth-first search strategy to continue generating these rings. The search process continues until a UAV flying along a specific shrinking ring can fully observe all of the information within the ring, at which point that shrinking ring is considered the final one.

The polygon offset method we employ here is our improvement of the work by Lee et al. [26]. The polygon offset method generates offset vertices by using the vertices of the previous shrinking ring and organizes them into an offset vertex list $\boldsymbol{OVL}$. Next, adjacent offset vertices in $\boldsymbol{OVL}$ are connected to form offset edges, which are then organized into an offset edge list $\boldsymbol{OEL}$. By comparing the directional and positional information, each offset edge is assigned an off_dir and off_pos attribute; $off\_dir = 0$ indicates that the direction is invalid, meaning that the offset edge formed by the two offset vertices is oriented oppositely to the original edge created by the corresponding two original vertices, while $off\_pos = 0$ indicates that the entire offset edge lies within the scanning area of a UAV flying along the previous shrinking ring, and as such contributes nothing to the current shrinking ring. On the other hand, $off\_dir = 1$ and $off\_pos = 1$ respectively signify that the direction and position attributes are valid.

We then begin traversing the $\boldsymbol{OEL}$ from the start. We set the first offset edge with both off_dir and off_pos equal to 1 as the backward edge and set the second offset edge with both off_dir and off_pos equal to 1 as the forward edge. The inv_dir and inv_pos attributes are used to record the invalid direction and position information of the offset

edges between the backward and forward edges. The core of the polygon offset method consists of applying corresponding procedures based on *inv_dir* and *inv_pos* to extract the required vertices from the offset vertices. In the subsequent process, the current forward edge is set as the backward edge and the next offset edge with both off_dir and off_pos equal to 1 is set as the forward edge. This loop continues until all offset edges have been considered as potential backward edges. All extracted vertices are then organized into an extracted vertex list $\boldsymbol{EVL}$.

Lee et al. classified the possible distributions of *inv_dir* and *inv_pos* into five categories: CASE 1.1, CASE 1.2, CASE 2, CASE 3.1, and CASE 3.2; however, through extensive exploration, we found that the existing cases do not encompass all possible situations. For instance, CASE 3.2 was designed to address scenarios where more than two directionally invalid offset edges occur during the offset process, but it only handles cases where the backward edge and forward edge are parallel. In practice, the backward and forward edges are often not parallel. Even when the backward and forward edges are parallel, the method used in CASE 3.2 can still result in incomplete coverage. As shown in Figure 3a, the vertices of the previous shrinking ring are labeled as v_1 to v_6. Among these, the edges (v_5v_6) and (v_3v_4) are parallel. The offset vertices corresponding to these are labeled as $\boldsymbol{ov}_1$ to $\boldsymbol{ov}_6$. When $(\boldsymbol{ov}_5\boldsymbol{ov}_6)$ is the backward edge and $(\boldsymbol{ov}_3\boldsymbol{ov}_4)$ is the forward edge, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are the intersection points chosen by the CASE 3.2 strategy. In Figure 3b, the current shrinking ring consists of vertices $v_1^{'}$ to $v_4^{'}$. By flying along these two shrinking rings, the UAV is unable to fully cover the intermediate area.

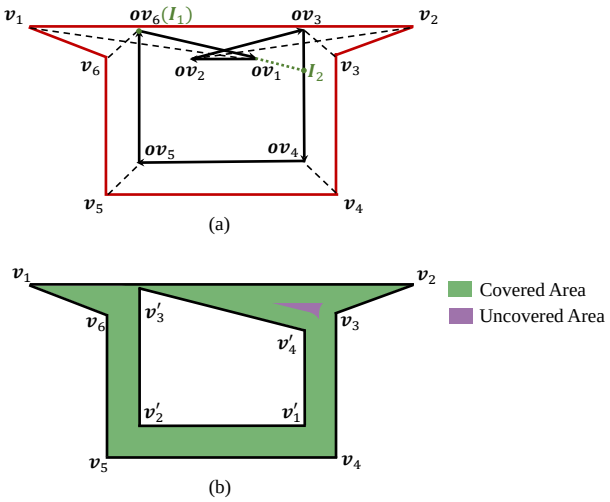


Figure 3. An example of incomplete coverage in CASE 3.2: (**a**) Offset vertices (**b**) Uncovered area.

Based on these observations, we have reorganized the cases into CASE 1, CASE 2, CASE 3.1, CASE 3.2, and CASE 3.3, improving the algorithm to address all scenarios during the offset process.

CASE 1: $inv\_dir = 0$ and $inv\_pos = 0$. There are no invalid offset edges in terms of direction or position between the backward edge and the forward edge. For this case, it is sufficient to include the terminal offset vertex of the backward edge in $\boldsymbol{EVL}$. As illustrated in Figure 4a, $(\boldsymbol{ov}_1\boldsymbol{ov}_2)$ represents the backward edge and $(\boldsymbol{ov}_2\boldsymbol{ov}_3)$ represents the forward edge; therefore, only $\boldsymbol{ov}_2$ should be added to $\boldsymbol{EVL}$.

CASE 2: $inv\_dir = 0$ and $inv\_pos \geq 1$. All offset edges between the backward edge and the forward edge are valid in terms of direction, but there is at least one positionally invalid offset edge. In this case, these offset edges result in globally invalid loops. We begin by sequentially adding the terminal offset vertex of the backward edge, all intermediate offset vertices, and the initial offset vertex of the forward edge to the $\boldsymbol{EVL}$. Trimming is performed subsequently. As depicted in Figure 4b, $(\boldsymbol{ov}_2\boldsymbol{ov}_3)$ represents the backward edge and $(\boldsymbol{ov}_4\boldsymbol{ov}_5)$ represents the forward edge; thus, $\boldsymbol{ov}_3$ and $\boldsymbol{ov}_4$ are added to $\boldsymbol{EVL}$ in

sequence. Figure 4c shows the globally invalid loop generated by $(\boldsymbol{ov_2ov_3})$, $(\boldsymbol{ov_3ov_4})$, $(\boldsymbol{ov_4ov_5})$, and $(\boldsymbol{ov_7ov_8})$.

CASE 3.1: $inv\_dir = 1$. There is only one directionally invalid offset edge between the backward edge and the forward edge. The strategy for handling this case is to determine the intersection point between the backward edge and the forward edge and add it to $\boldsymbol{EVL}$. As shown in the Figure 4d, $(\boldsymbol{ov_1ov_2})$ represents the backward edge and $(\boldsymbol{ov_3ov_4})$ represents the forward edge. Because $(\boldsymbol{ov_2ov_3})$ is a directionally invalid edge, the intersection point $\boldsymbol{I}$ between $(\boldsymbol{ov_1ov_2})$ and $(\boldsymbol{ov_3ov_4})$ is added to $\boldsymbol{EVL}$.

CASE 3.2: $inv\_dir \geq 2$ and the backward edge intersects the forward edge. The approach for this scenario is the same as in CASE 3.1. Figure 4e illustrates that $(\boldsymbol{ov_4ov_5})$ is the backward edge and $(\boldsymbol{ov_7ov_1})$ is the forward edge. The edges $(\boldsymbol{ov_5ov_6})$ and $(\boldsymbol{ov_6ov_7})$ situated between them are directionally invalid. We calculate the intersection point $\boldsymbol{I}$ of $(\boldsymbol{ov_4ov_5})$ and $(\boldsymbol{ov_7ov_1})$ and add it to $\boldsymbol{EVL}$.

CASE 3.3: $inv\_dir \geq 2$ and the backward edge does not intersect the forward edge. The method here involves first sequentially traversing the directionally invalid offset edges between the backward and forward edges, then calculating their intersection point $\{\boldsymbol{I}_1^1, \boldsymbol{I}_1^2, ..., \boldsymbol{I}_1^i, ...\boldsymbol{I}_1^{inv\_dir}\}$ with the backward edge. Subsequently, the minimum distance $d_{\boldsymbol{I}_1^i}$ between $\boldsymbol{I}_1^i$ and all original edges is calculated. Then, $\boldsymbol{I}_1^i$ is added to the preselected list only if it lies inside the shrinking ring and $d_{\boldsymbol{I}_1^i} \geq L_1 - L_1^{'}$. The shortest distance from the points in the preselected list to the original edges is expressed as $d^* = \min\{d_{\boldsymbol{I}_1^i} | d_{\boldsymbol{I}_1^i} \geq L_1 - L_1^{'}\}$. Finally, the $\boldsymbol{I}_1^i$ corresponding to d^* from the preselected list is selected and added to $\boldsymbol{EVL}$. The intersection point $\boldsymbol{I}_2$ with the forward edge is then added to $\boldsymbol{EVL}$ using the same procedure. As shown in Figure 4f, $(\boldsymbol{ov_5ov_6})$ represents the backward edge and $(\boldsymbol{ov_3ov_4})$ represents the forward edge. The intersection points of $(\boldsymbol{ov_6ov_1})$, $(\boldsymbol{ov_1ov_2})$, and $(\boldsymbol{ov_2ov_3})$ with $(\boldsymbol{ov_5ov_6})$ are $\boldsymbol{I}_1^1$,$\boldsymbol{I}_1^2$,$\boldsymbol{I}_1^3$, respectively. Given that $d_{\boldsymbol{I}_1^1} < L_1 - L_1^{'}$ and $d_{\boldsymbol{I}_1^2} < d_{\boldsymbol{I}_1^3}$, $\boldsymbol{I}_1^2$ is set as $\boldsymbol{I}_1$ and added to $\boldsymbol{EVL}$. Similarly, $\boldsymbol{I}_2$ from the intersection points with $(\boldsymbol{ov_3ov_4})$ is added to $\boldsymbol{EVL}$.

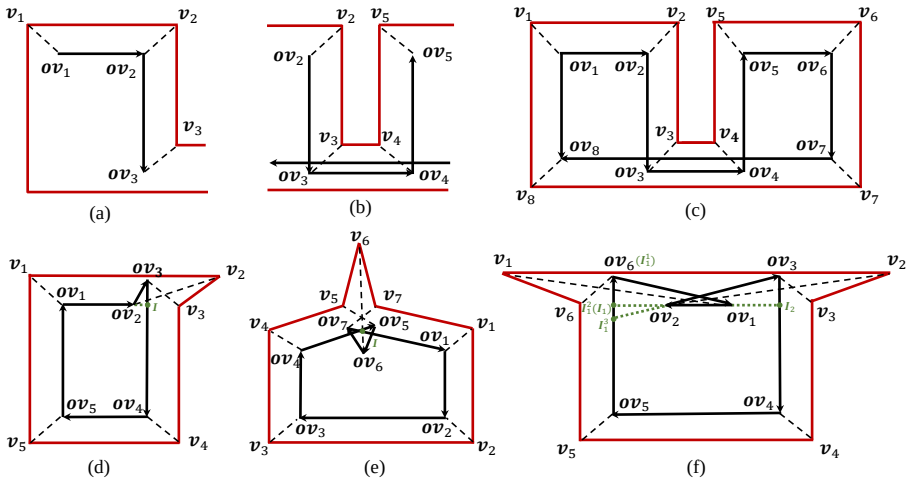


Figure 4. Cases that may occur in the offset process. (**a**) Case 1. (**b**) Case 2. (**c**) Globally invalid loop generated by case 2. (**d**) Case 3.1. (**e**) Case 3.2. (**f**) Case 3.3.

According to CASE 2, $\boldsymbol{EVL}$ may form globally invalid loops. These loops fall entirely within the UAV's photography range as it follows the previous shrinking ring. This results in no improvement in coverage and increases the path length. To eliminate these globally invalid loops, we identify the self-intersection points within the overall loop formed by $\boldsymbol{EVL}$. Based on these points, we trim the line segments and ultimately extract the shrinking rings.

To facilitate a clearer understanding of the process of generating shrinking rings, the full pseudocode of the algorithm is provided in Algorithm 1.

Algorithm 1 Generate the Shrinking Ring List for the Region

Input: Region $\boldsymbol{R}_m = (\boldsymbol{v}_m^1, \boldsymbol{v}_m^2, ..., \boldsymbol{v}_m^{r_m})$, the longitudinal scanning length L_1, the overlap length in the longitudinal direction $L_1^{'}$

Output: The shrinking ring list $\boldsymbol{SRL}_m = (\boldsymbol{S}_m^1, \boldsymbol{S}_m^2, ..., \boldsymbol{S}_m^{idx}, ..., \boldsymbol{S}_m^{n_m})$, where n_m denotes the number of shrinking rings, varying by region.

$\boldsymbol{SRL}_m \leftarrow ()$
Initialize the list $\boldsymbol{T}_1$ to store the shrinking rings to be offset: $\boldsymbol{T}_1 \leftarrow (\boldsymbol{R}_m)$
while $\boldsymbol{T}_1 \neq ()$ **do**
  Set the previous shrinking ring: $\boldsymbol{S} \leftarrow$ the head of $\boldsymbol{T}_1$
  Append $\boldsymbol{S}$ to the tail of $\boldsymbol{SRL}_m$
  Generate $\boldsymbol{OVL}$ from $\boldsymbol{S}$
  Generate $\boldsymbol{OEL}$ from $\boldsymbol{OVL}$
  Set the off_pos and off_div attributes for each offset edge in $\boldsymbol{OEL}$
  Iterate through $\boldsymbol{OEL}$ and extract $\boldsymbol{EVL}$ based on different cases.
  Connect adjacent vertices in $\boldsymbol{EVL}$ to form extracted edge list $\boldsymbol{EEL}$.
  Insert intersection points into the $\boldsymbol{EVL}$ based on the intersections of the edges in the $\boldsymbol{EEL}$.
  Reconstruct $\boldsymbol{EEL}$ based on the new $\boldsymbol{EVL}$.
  Remove edges from the $\boldsymbol{EEL}$ that are within a minimum distance of less than $L_1 - L_1^{'}$ from each edge of $\boldsymbol{S}$
  Initialize the list $\boldsymbol{T}_2$ to store the offset edges: $\boldsymbol{T}_2 \leftarrow ()$
  for $i \leftarrow 1$ to size of $\boldsymbol{EEL}$ **do**
    Append $\boldsymbol{EEL}[i]$ to the tail of $\boldsymbol{T}_2$
    $end \leftarrow$ size of $\boldsymbol{T}_2$
    $begin \leftarrow end - 1$
    while $begin \geq 1$ **do**
      if the start vertex of $\boldsymbol{T}_2[begin]$ = the end vertex of $\boldsymbol{T}_2[end]$ **then**
        Form $\boldsymbol{S}_m^{idx}$ by extracting the start vertices of edges indexed from *begin* to *end*
        Append $\boldsymbol{S}_m^{idx}$ to the head of $\boldsymbol{T}_1$
        Remove edges in $\boldsymbol{T}_2$ with indices from *begin* to *end*
        break
      else
        $begin \leftarrow begin - 1$
      end if
    end while
  end for
end while

3.2. Waypoint Generation

After the shrinking ring has been generated, waypoints must be created along each segment to ensure subsequent coverage by multiple UAVs. For a segment of length l in a shrinking ring, the number of waypoints l_n is calculated as follows:

$$l_n = \left\lceil \frac{l - \frac{L_2}{2}}{l_g} \right\rceil \tag{8}$$

where $l_g = L_2 - L_2^{'}$ denotes the distance between waypoints. To ensure that waypoints are evenly spaced, $L_2^{'}$ should be adjusted according to the value of l_n. This adjustment is expressed as follows:

$$\overline{L_2^{'}} = \frac{\left(l_n - \frac{1}{2}\right) L_2 - l}{l_n - 1}. \tag{9}$$

Accordingly, the interval distance l_g between waypoints should be revised to $\overline{l_g}$. This revised interval is expressed as

$$\overline{l_g} = L_2 - \overline{L_2^{'}}. \tag{10}$$

3.3. Connecting Waypoints

To obtain the optimal coverage path, the first step is to determine the regional access order. A widely-used strategy involves treating the center of each region as a representative point and optimizing the visit sequence of these centers using a TSP solver [17–20,27]. Therefore, we utilize the Self-Organizing Map (SOM) [28] algorithm to establish the visiting order of regions, which is a well-documented method used in previous studies [29,30]. After the visit order of the regions is determined, the corresponding $\boldsymbol{SRL}_m$ is arranged accordingly. Then, the optimal coverage path can be formed by sequentially connecting the waypoints in each shrinking ring $\boldsymbol{S}_m^{idx}$ of $\boldsymbol{SRL}_m$.

For $\boldsymbol{S}_m^{idx} = (\boldsymbol{w}_{m,idx}^1, \boldsymbol{w}_{m,idx}^2, ..., \boldsymbol{w}_{m,idx}^{n_{idx}})$, n_{idx} represents the number of waypoints in the shrinking ring, while $\boldsymbol{w}_{m,idx}^{n_{idx}}$ denotes the n_{idx}-th waypoint of the idx-th shrinking ring generated by region R_m. Thus, there are n_{idx} possible ways to integrate $\boldsymbol{S}_m^{idx}$ into the optimal coverage path: $\boldsymbol{P}_{m,idx} = \{(\boldsymbol{w}_{m,idx}^1, \boldsymbol{w}_{m,idx}^2, ..., \boldsymbol{w}_{m,idx}^{n_{idx}}), (\boldsymbol{w}_{m,idx}^2, \boldsymbol{w}_{m,idx}^3, ..., \boldsymbol{w}_{m,idx}^1), ..., (\boldsymbol{w}_{m,idx}^{n_{idx}}, \boldsymbol{w}_{m,idx}^1, ..., \boldsymbol{w}_{m,idx}^{n_{idx}-1})\}$. Suppose that in the n_{idx}-th sub-path of $\boldsymbol{P}_{m,idx}$, the first waypoint is denoted as $\boldsymbol{P}_{m,idx}^{n_{idx},1}$ and the last waypoint is denoted as $\boldsymbol{P}_{m,idx}^{n_{idx},n_{idx}}$. Then, the cost of $\boldsymbol{P}_{m,idx}^{n_{idx}}$ is computed based on the waypoint $\boldsymbol{P}_{m,idx}^{n_{idx},1}$ and $\boldsymbol{P}_{m,idx}^{n_{idx},n_{idx}}$ using the following formula:

$$\begin{aligned}\boldsymbol{P}_{m,idx}^{n_{idx}} = \lambda_1 \times (\mathrm{d}(\boldsymbol{P}_{m,idx}^{n_{idx},1}, \boldsymbol{w}_b) - \mathrm{d}(\boldsymbol{P}_{m,idx}^{n_{idx},1}, \boldsymbol{P}_{m,idx}^{n_{idx},n_{idx}})) \\ + \lambda_2 \times (\mathrm{a}(\boldsymbol{w}_b, \boldsymbol{P}_{m,idx}^{n_{idx},1}, \boldsymbol{P}_{m,idx}^{n_{idx},2}) - \mathrm{a}(\boldsymbol{P}_{m,idx}^{n_{idx},n_{idx}}, \boldsymbol{P}_{m,idx}^{n_{idx},1}, \boldsymbol{P}_{m,idx}^{n_{idx},2}))\end{aligned} \tag{11}$$

where $\boldsymbol{w}_b$ refers to the backward reference point, typically the last waypoint of the previous-level shrinking ring. However, for the first-level shrinking ring, $\boldsymbol{w}_b$ corresponds to the warehouse. This indicates that when forming the optimal coverage path, the connection between $\boldsymbol{P}_{m,idx}^{n_{idx},1}$ and $\boldsymbol{P}_{m,idx}^{n_{idx},n_{idx}}$ in the current shrinking ring $\boldsymbol{S}_m^{idx}$ is broken and $\boldsymbol{P}_{m,idx}^{n_{idx},1}$ is connected to $\boldsymbol{w}_b$, resulting in new distance and turning costs. Thus, the connection way is determined as follows:

$$\boldsymbol{P}_{m,idx}^{*} = \underset{m \in [1,M]}{\operatorname{argmin}} \boldsymbol{P}_{m,idx}^{n_{idx}}. \tag{12}$$

Based on this, the pseudocode for the first stage of SSDP is presented in Algorithm 2.

Algorithm 2 Generate the Optimal Coverage Path

Input: Region set $\{\boldsymbol{R}_m\}$, the longitudinal and lateral scanning length L_1, L_2, the overlap length in the longitudinal and lateral direction $L_1^{'}, L_2^{'}$, warehouse $\boldsymbol{w}_0$, the weights for distance and turning in energy consumption λ_1 and λ_2
Output: A optimal coverage path $\boldsymbol{G} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_g, \ldots, \boldsymbol{w}_G)$

for $m \leftarrow 1$ to M **do**
  Generate $\boldsymbol{SRL}_m = (\boldsymbol{S}_m^1, \boldsymbol{S}_m^2, ..., \boldsymbol{S}_m^{n_m})$ by $\boldsymbol{R}_m$ using Algorithm 1
  for $\boldsymbol{S}_m^{idx} \in \boldsymbol{SRL}_m$ **do**
    Traverse each edge of $\boldsymbol{S}_m^{idx}$ and insert waypoints
  end for
end for
Determine the region visitation order using a TSP solver (SOM)
$\boldsymbol{G} \leftarrow ()$
for $m \leftarrow 1$ to M **do**
  Retrieve the $\boldsymbol{SRL}_m$ corresponding to the m-th region in the visitation order
  for $\boldsymbol{S}_m^{idx} \in \boldsymbol{SRL}_m$ **do**
    Merge $\boldsymbol{S}_m^{idx}$ into the tail of $\boldsymbol{G}$ according to $\boldsymbol{P}_{m,idx}^{*}$, calculated by Equation (12)
  end for
end for

4. Path Segmentation

Given that the optimal coverage path $\boldsymbol{G} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_g, \ldots, \boldsymbol{w}_G)$ is formed by sequenced waypoints, multiple UAVs can achieve collaborative coverage by having each

take on non-overlapping segments of the path. When using waypoints as the units of segmentation, the minimum energy consumption problem for the UAV swarm transforms into a combinatorial optimization problem. Specifically, we regard each waypoint as a decision stage; during the solution process, each stage must be considered sequentially, with each stage having its own optimal solution. Additionally, to determine the optimal solution for a stage, we must consider the optimal solutions of all preceding stages. Put another way, this problem is characterized by optimal substructure and overlapping subproblems, making the DP-based method the appropriate solution.

4.1. State Transition Matrix Design

The state transition matrix crucial for correctly solving problems and finding the optimal solution with the DP algorithm. Here, we design a two-dimensional matrix $\boldsymbol{D}$ as the state transition matrix. The length of $\boldsymbol{D}$ corresponds to the number of waypoints in $\boldsymbol{G}$ and its width is determined by the maximum number of available drones N. The value $\boldsymbol{D}[n][g]$ in the matrix represents the subproblem, that is, the minimum energy consumption for visiting the first g waypoints with n drones.

4.2. Initialization and State Transition

Before initialization, the distances and angles between waypoints are precomputed and stored in the arrays $\boldsymbol{l}$ and $\boldsymbol{a}$. These precomputations are defined as follows:

$$\boldsymbol{l}[g] = \begin{cases} 0, & g = 1 \\ \boldsymbol{l}[g-1] + \mathrm{d}(\boldsymbol{w}_{g-1}, \boldsymbol{w}_g), & 1 < g \leq G \end{cases} \tag{13}$$

$$\boldsymbol{a}[g] = \begin{cases} 0, & g = 1 \\ \boldsymbol{a}[g-1] + \mathrm{a}(\boldsymbol{w}_{g-1}, \boldsymbol{w}_g, \boldsymbol{w}_{g+1}), & 1 < g < G \end{cases} \tag{14}$$

where $\boldsymbol{l}[g]$ represents the total path distance from waypoint $\boldsymbol{w}_1$ to waypoint $\boldsymbol{w}_g$ in the optimal coverage path and $\boldsymbol{a}[g]$ represents the total path angle from waypoint $\boldsymbol{w}_1$ to waypoint $\boldsymbol{w}_{g+1}$. Here, $\mathrm{d}(\cdot,\cdot)$ denotes the Euclidean distance between two waypoints and $\mathrm{a}(\cdot,\cdot,\cdot)$ represents the turning angle of the path formed by three sequential waypoints.

We initialize each element in the first row of $\boldsymbol{D}$ using the following formula.

$$\boldsymbol{D}[1][g] = \begin{cases} \lambda_1 \times 2 \times \mathrm{d}(\boldsymbol{w}_0, \boldsymbol{w}_1) + \lambda_2 \times \mathrm{a}(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_0), & g = 1 \\ \lambda_1 \times \big(\mathrm{d}(\boldsymbol{w}_0, \boldsymbol{w}_1) + \boldsymbol{l}[g] + \mathrm{d}(\boldsymbol{w}_g, \boldsymbol{w}_0)\big) & \\ \quad + \lambda_2 \times \big(\mathrm{a}(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) + \boldsymbol{a}[g-1] + \mathrm{a}(\boldsymbol{w}_{g-1}, \boldsymbol{w}_g, \boldsymbol{w}_0)\big), & 1 < g \leq G \end{cases} \tag{15}$$

This indicates that the first drone starts from depot $\boldsymbol{w}_0$, visits $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_G$ in sequence, and then returns to the depot. After this calculation, the values must be checked to eliminate infeasible scenarios, as expressed below:

$$\boldsymbol{D}[1][g] = \begin{cases} \boldsymbol{D}[1][g], & \boldsymbol{D}[1][g] \leq E \\ +\infty, & \boldsymbol{D}[1][g] > E \end{cases} \tag{16}$$

where $\boldsymbol{D}[1][g]$ represents the energy cost for UAV u_1 to travel sequentially from the warehouse $\boldsymbol{w}_0$ through intermediate waypoints $\boldsymbol{w}_1, \boldsymbol{w}_2, ...$, finally reaching $\boldsymbol{w}_g$ before returning to the warehouse. If this energy cost exceeds its energy constraint, the process is deemed infeasible and the cost is set to infinity. At this point, the reachable waypoints are temporarily assigned to u_1, while the unreachable waypoints are assigned to other UAVs for completion.

Then, the first column of $\boldsymbol{D}$ is initialized as follows:

$$\boldsymbol{D}[n][1] = \lambda_1 \times 2 \times \mathrm{d}(\boldsymbol{w}_0, \boldsymbol{w}_1) + \lambda_2 \times \mathrm{a}(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_0), 1 \leq n \leq N. \tag{17}$$

This shows that when there is only one waypoint $\boldsymbol{w}_1$ in the path; the drone departs from the warehouse $\boldsymbol{w}_0$, visits $\boldsymbol{w}_1$, and immediately returns to $\boldsymbol{w}_0$. Furthermore, because

$\boldsymbol{D}[1][1] = \boldsymbol{D}[2][1] = ... = \boldsymbol{D}[N][1]$, only the first drone is required for this task, while the other drones can remain on standby.

Continuing with the matrix solution, we focus on the remaining components; $\boldsymbol{D}[n][g]$ can be derived from the subproblem $\boldsymbol{D}[n-1][k]$, with its value calculated as follows:

$$t_k = \boldsymbol{D}[n-1][k] + \lambda_1 \times t_k^1 + \lambda_2 \times t_k^2, \tag{18}$$

$$1 \le k < g, 2 \le g \le G, 2 \le n \le N$$

where t_k is a candidate for $\boldsymbol{D}[n][g]$ composed of $\boldsymbol{D}[n-1][k]$, t_k^1, t_k^2. Specifically, t_k^1 represents the distance cost for the n-th UAV, expressed as

$$t_k^1 = \mathrm{d}(\boldsymbol{w}_0, \boldsymbol{w}_{k+1}) + \boldsymbol{l}[g] - \boldsymbol{l}[k+1] + \mathrm{d}(\boldsymbol{w}_g, \boldsymbol{w}_0), 1 \le k < g. \tag{19}$$

Meanwhile, t_k^2 denotes the turning cost for the n-th UAV, described as follows.

$$t_k^2 = \begin{cases} \mathrm{a}(\boldsymbol{w}_0, \boldsymbol{w}_{k+1}, \boldsymbol{w}_{k+2}) + \boldsymbol{a}[g-1] \\ -\boldsymbol{a}[k+1] + \mathrm{a}(\boldsymbol{w}_{g-1}, \boldsymbol{w}_g, \boldsymbol{w}_0), & 1 \le k < g-1 \\ \mathrm{a}(\boldsymbol{w}_0, \boldsymbol{w}_g, \boldsymbol{w}_0), & k = g-1 \end{cases} \tag{20}$$

This formulation assumes that the first k waypoints are handled by the first $n-1$ UAVs. The n-th UAV begins its route at the warehouse, sequentially visits waypoints $\boldsymbol{w}_{k+1}, \boldsymbol{w}_{k+2}, ..., \boldsymbol{w}_g$, and returns to the warehouse. The value of $\boldsymbol{D}[n][g]$ is then determined as the smallest t_k among all candidates.

Therefore, the optimal solution for $\boldsymbol{DP}[n][g]$ can be obtained from these values as follows:

$$\boldsymbol{D}[n][g] = \min_{1 \le k < g} \{t_k \mid t_k - \boldsymbol{D}[n-1][k] \le E\}. \tag{21}$$

Alternatively, if the value of $\boldsymbol{D}[n][g]$ calculated from the above formula is greater than $\boldsymbol{D}[n-1][g]$, then $\boldsymbol{D}[n][g]$ should be set to $\boldsymbol{D}[n-1][g]$. This indicates that all g waypoints are to be handled by the first $n-1$ UAVs, meaning that the n-th UAV is not needed, that is, the optimal solution can be achieved with only the first $n-1$ UAVs. The final expression for $\boldsymbol{D}[n][g]$ is as follows:

$$\boldsymbol{D}[n][g] = \min\left(\boldsymbol{D}[n-1][g], \min_{1 \le k < g} \{t_k \mid t_k - \boldsymbol{D}[n-1][k] \le E\}\right). \tag{22}$$

Finally, the values in the last row and last column of the matrix signify the minimum energy consumption for the UAV swarm. The time complexity of the path segment algorithm is primarily determined by this part. The time complexity is $O(NG^2)$, where N represents the number of drones and G denotes the number of waypoints in the optimal coverage path. In practice, the computation can stop when $\boldsymbol{D}[n][G]$ equals $\boldsymbol{D}[n-1][G]$, which indicates that the n-th drone and all subsequent drones are not required for coverage.

4.3. Tracing the Optimal Solution

After the optimal solution is determined, we can trace back through the state transition matrix to allocate waypoints to each UAV. We start from the last element in the final row, $\boldsymbol{D}[N][G]$; if $\boldsymbol{D}[N][G] = \boldsymbol{D}[N-1][G]$, it implies that the N-th UAV was not employed and no waypoints were assigned to it. Otherwise, we should trace back using Equation (21) to determine that it was derived from $\boldsymbol{D}[N-1][k]$ and assign the waypoints $(\boldsymbol{w}_{k+1}, ..., \boldsymbol{w}_G)$ to the N-th drone. Next, we determine the waypoints assigned to the $(N-1)$-th drone in the same manner from $\boldsymbol{D}[N-1][k]$. Finally, when $n = 1$, all remaining waypoints are allocated to the first UAV. The steps of the algorithm are presented in Algorithm 3.

Algorithm 3 Path Segmentation

Input: the optimal coverage path $\boldsymbol{G} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_g, \ldots, \boldsymbol{w}_G)$, warehouse $\boldsymbol{w}_0$, the weights for distance and turning in energy consumption λ_1 and λ_2, energy constraint E
Output: the flight paths for UAVs $\{\boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_n\}$

for $g \leftarrow 1$ to G **do**
  Calculate the elements of array $\boldsymbol{l}$ and $\boldsymbol{a}$ by Equations (13) and (14), respectively.
end for
Initialize the first row of $\boldsymbol{D}$ by Equation (15).
Check whether the first row of $\boldsymbol{D}$ satisfies the energy constraints using Equation (16).
Initialize the first column of $\boldsymbol{D}$ by Equation (17).
for $n \leftarrow 2$ to N **do**
  for $g \leftarrow 2$ to G **do**
    Calculate the value of $\boldsymbol{D}[n][g]$ by Equation (22).
  end for
end for
$g \leftarrow G$
for $n \leftarrow N$ to 1 **do**
  if $n = 1$ **then**
    Assign the waypoint sequence $\boldsymbol{W}_1 = (\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_g, \boldsymbol{w}_0)$ to UAV u_1.
    break
  end if
  if $\boldsymbol{D}[n][g] = \boldsymbol{D}[n-1][g]$ **then**
    Assign the waypoint sequence $\boldsymbol{W}_n = ()$ to UAV u_n.
  else
    for $k \leftarrow 1$ to $g - 1$ **do**
      if $\boldsymbol{D}[n][g]$ is derived from $\boldsymbol{D}[n-1][k]$ by Equation (21) **then**
        Assign the waypoint sequence $\boldsymbol{W}_n = (\boldsymbol{w}_0, \boldsymbol{w}_{k+1}, \ldots, \boldsymbol{w}_g, \boldsymbol{w}_0)$ to UAV u_n.
        $g \leftarrow k$
      end if
    end for
  end if
end for

5. Experiments and Discussion

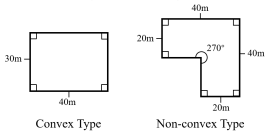
In this section, we describe some simulation scenarios created for our computational experiments. Visual Studio Code 1.87.0 was the simulation experiment platform software, and the hardware setup of the experimental system included a 2.4 GHz 4-Core Intel Core i5 processor, with the programming language being C++. The analysis of tables and line charts was conducted using OriginPro 2021. The path simulation diagrams were created using the matplotlib library in Python.

5.1. Parameter Setting

For the experiments, the field of view for the UAV cameras was configured to 4 m × 4 m. The images captured by the drones were required to have an overlap of 1 m in both length and width. The maximum energy consumption for each UAV was capped at 1000 kJ. Each drone was required to depart from the warehouse, complete the coverage task, and return to the warehouse.

The specific parameter settings of the map along with the target regions are shown in Table 1. These regions are either convex or non-convex. In order to exclude the influence of the shape of the region, all convex regions had the same shape and all non-convex regions had the same shape. Each region was equal in size; in addition, the warehouse coordinate was not located within any region, and all regions did not intersect with each other.

Table 1. Parameter settings of the map and target regions.

Parameter	Value
Map coordinate (m, m)	((0, 0), (0, 400), (400, 400), (400, 0))
Warehouse coordinate (m, m)	(200, 200)
Region shape	 Convex Type Non-convex Type
Number of regions	Refer to specific experiments
Locations of regions	Randomly distributed in the map
Angles of regions	$\frac{\pi}{\text{Number of regions}} \times \text{index}$

In order to verify the effectiveness and superiority of SSDP, we set up the following three experimental scenarios and used the total energy consumption of the UAV swarm as the experimental evaluation metric:

1. Increasing the proportion of non-convex regions in the total regions; the experimental details are provided in Section 5.2.
2. Increasing the total area while maintaining the ratio of convex to non-convex regions; the experimental details are provided in Section 5.3.
3. Increasing the number of UAVs; the experimental details are provided in Section 5.4.
4. Adjusting the decomposition strategy; the experimental details are provided in Section 5.5.

The following models were selected as comparison models:

1. Benchmark model: B&F-Avg. In this model, the SOM method is used to determine the visitation order of regions, after which the B&F pattern coverage paths are generated for each region according to the visitation order. These paths are then connected end-to-end to form a optimal coverage path. Finally, the optimal coverage path is segment equally among the UAVs based on length.
2. Ablation model: B&F-DP. The only difference between this method and B&F-Avg is the use of the DP-based method to segment the optimal coverage path among the UAVs.
3. State-of-the-art methods: GASC [20] and BINPAT [23].

Because the B&F pattern cannot be applied directly to non-convex regions, these methods require pre-decomposing the regions into convex parts.

5.2. Increasing the Proportion of Non-Convex Regions

In this experiment, a total of 20 convex regions were randomly generated. These regions were gradually replaced by non-convex regions in increments of 2. The number of UAVs was set at 3. Tables 2–4 list the statistical results for the UAV swarm's total path distance, total path angle, and energy consumption, respectively. Figure 5 illustrates the changes in energy consumption as the proportion of non-convex regions increases. Figure 6 shows the UAV swarm paths generated by each model when all regions are non-convex.

From Tables 2 and 3 and Figure 5, it can be observed that the path generated by GASC is consistently the shortest among those produced by methods based on the B&F pattern, which include B&F-Avg, B&F-DP, and BINPAT. This holds true across different proportions of non-convex regions. However, when the proportion exceeds 10%, the total turning angle of GASC becomes greater than that of B&F-Avg and B&F-DP. When the proportion exceeds 60%, it also surpasses BINPAT. Consequently, when the proportion exceeds 70%, the energy consumption of GASC surpasses that of B&F-DP and BINPAT. This indicates that the shortest path does not necessarily result in the lowest energy consumption. Additionally, because the B&F pattern requires decomposing concave regions for coverage, the path length of GASC exceeds that of SSDP when the proportion of non-convex regions surpasses 40%.

Table 2. The UAV swarm's total path distance (m) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	9453.91	8852.40	8877.79	9875.32	8662.26
10	9635.92	9113.07	8915.15	9768.66	8765.68
20	9926.45	9329.50	8977.68	9976.71	8882.51
30	10,139.30	9695.32	9010.03	10,097.40	9005.72
40	10,458.70	9799.72	9035.02	10,322.20	9253.09
50	10,728.44	10,104.42	9059.61	10,432.50	9388.63
60	10,807.22	10,427.86	9076.22	10,445.20	9572.77
70	11,145.19	10,583.82	9102.90	9979.21	9751.16
80	11,362.74	10,850.81	9149.90	10,100.30	9802.40
90	11,569.86	11,096.35	9172.03	10,261.30	9912.20
100	11,921.05	11,343.37	9208.48	10,339.50	10,150.06

Table 3. The UAV swarm's total path angle (°) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	27,511.61	27,202.47	41,881.81	31,130.10	26,961.34
10	28,603.41	28,242.96	42,004.70	32,988.10	29,468.20
20	29,497.60	29,150.81	41,615.97	35,089.50	32,155.66
30	30,130.43	30,313.16	41,798.76	35,965.50	34,801.66
40	31,200.70	31,031.21	41,616.96	38,758.70	37,027.66
50	32,888.65	32,068.54	41,535.77	42,083.30	39,871.62
60	33,684.15	33,343.08	41,398.59	42,127.00	42,634.80
70	34,446.90	34,050.46	41,173.71	42,015.30	45,167.94
80	35,363.35	35,101.21	40,996.62	44,351.90	48,558.80
90	36,326.30	36,109.54	41,051.14	45,502.80	50,310.08
100	37,637.10	36,977.18	40,844.73	47,887.80	53,941.52

Table 4. The UAV swarm's total energy consumption (kJ) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	1299.58	1231.88	1387.26	1382.38	1208.99
10	1330.44	1270.64	1392.55	1390.27	1246.15
20	1370.89	1303.29	1395.21	1434.43	1286.62
30	1400.28	1354.59	1400.58	1456.48	1327.35
40	1445.65	1373.25	1401.37	1509.63	1377.01
50	1492.13	1416.70	1403.16	1556.03	1421.12
60	1508.84	1464.63	1403.51	1557.84	1469.60
70	1553.01	1488.71	1404.03	1506.73	1515.07
80	1585.86	1528.25	1407.23	1544.01	1555.82
90	1618.08	1565.06	1410.17	1573.24	1585.81
100	1669.36	1600.57	1411.93	1606.42	1649.07

Unlike other methods, BINPAT uses the nearest-neighbor approach to determine the order of region visits. This approach is highly influenced by the shape and location of the regions, meaning that its energy consumption does not always increase with the proportion of non-convex regions. However, it still shows an overall upward trend. As the proportion of non-convex regions increases, the number of subregions generated by decomposition also rises, narrowing the gap between the nearest-neighbor method and the TSP solver. Consequently, when the proportion exceeds 70%, BINPAT's energy consumption is lower than that of B&F-Avg and GASC, although it still falls short compared to B&F-DP and SSDP. This suggests that in order to achieve the lowest energy consumption, a TSP solver should be used to determine the order of region visits.

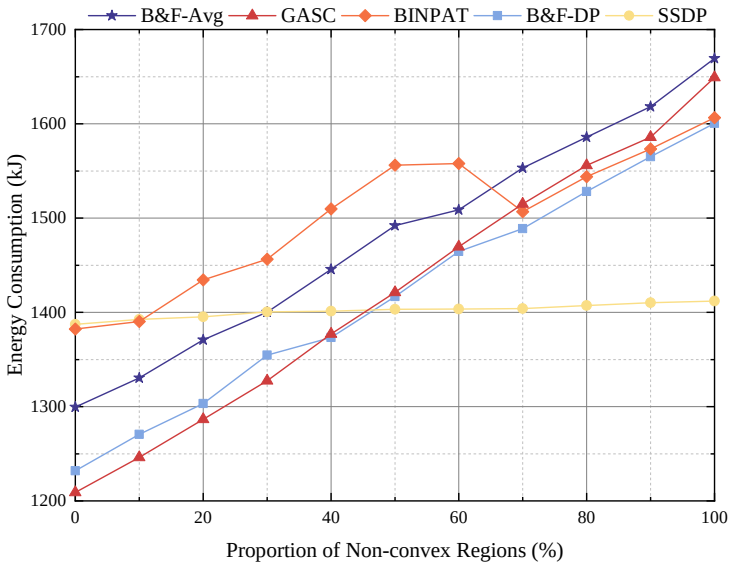


Figure 5. Visual results showing the UAV swarm's total energy consumption (kJ) for each model with different proportions of non-convex regions.

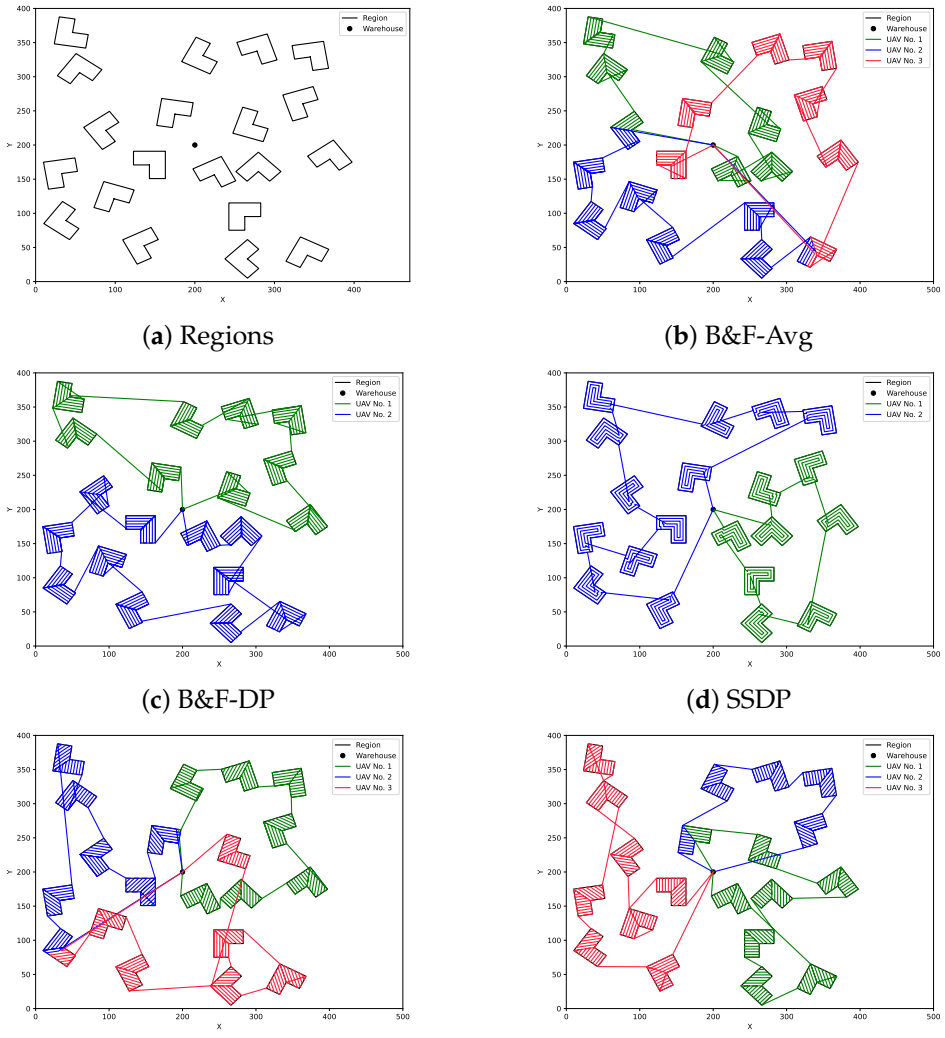


Figure 6. Paths generated by each model when the target regions are all of non-convex type.

The energy consumption of SSDP falls below that of BINPAT when the proportion of non-convex regions reaches 20%, below B&F-Avg at 40%, and subsequently falls below B&F-DP and GASC at 50%. This demonstrates the superiority of optimal coverage path generation in handling non-convex regions. B&F-DP consistently consumes less energy than B&F-Avg and BINPAT, and its energy consumption drops below that of GASC when the proportion of non-convex regions exceeds 40%, proving the effectiveness of the DP-based approach. Additionally, as shown in Figure 6, SSDP and B&F-DP require only two UAVs to complete the coverage task in all scenarios. This indicates that the DP-based approach not only reduces the total energy consumption of the UAV swarm but also conserves UAV resources.

5.3. Increasing the Total Area

In this section, the ratio of convex to non-convex regions was set to 1:1, then the number of regions was increased from 4 to 28 in increments of 4. The number of UAVs was 3. Tables 5–7 list the statistical results for the UAV swarm's total path distance, total path angle, and energy consumption, respectively. Figure 7 illustrates the changes in the energy consumption as the number of regions increases.

Table 5. The UAV swarm's total path distance (m) for each model as the number of regions increases.

Region Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
4	3148.27	2556.98	2347.72	2968.14	2515.72
8	5207.67	4530.11	4132.77	4918.77	4307.10
12	7276.99	6542.73	6005.10	6969.96	6208.62
16	8881.28	8345.74	7511.54	7977.63	7754.37
20	10,619.30	10,131.65	9037.13	9788.74	9562.48
24	12,397.00	11,905.30	10,583.57	11,641.30	11,091.75
28	14,413.50	13,825.00	12,268.36	13,245.60	12,971.58

Table 6. The UAV swarm's total path angle (°) for each model as the number of regions increases.

Region Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
4	6556.39	6330.11	8350.53	7952.71	8273.67
8	12,932.64	12,420.60	16,233.07	15,184.80	16,940.40
12	19,662.40	19,182.14	24,236.37	24,824.80	24,649.30
16	26,106.15	25,728.09	32,823.99	30,887.40	34,105.06
20	32,725.56	32,303.09	41,502.89	38,714.90	40,179.80
24	39,467.30	38,654.70	49,728.27	46,694.80	49,080.14
28	45,639.40	44,847.40	58,060.76	57,175.40	57,104.98

Table 7. The UAV swarm's total energy consumption (kJ) for each model as the number of regions increases.

Region Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
4	405.68	339.94	338.52	400.89	355.73
8	692.76	614.80	611.86	685.21	637.90
12	984.58	900.87	895.81	1005.36	921.92
16	1223.58	1162.24	1146.61	1176.43	1185.96
20	1478.73	1422.07	1400.41	1451.99	1442.97
24	1739.42	1678.26	1651.73	1733.57	1699.47
28	2019.78	1948.45	1919.00	2014.55	1984.45

As shown in Table 5, SSDP reduces total path distance by approximately 14.63–25.43% compared to B&F-Avg, by about 8.18–11.26% compared to B&F-DP, by around 5.84–20.90% compared to BINPAT, and by approximately 3.13–6.68% compared to GASC. This is because SSDP can directly address non-convex regions, significantly shortening the coverage path.

Combined with the data from Table 6, it can be observed that B&F-DP reduces the total path distance by approximately 3.97–18.78% and decreases the total turning angles by about 1.29–3.96% compared to B&F-Avg. This indicates that the DP-based approach can effectively reduce both path distance and turning angle, thereby reducing energy consumption.

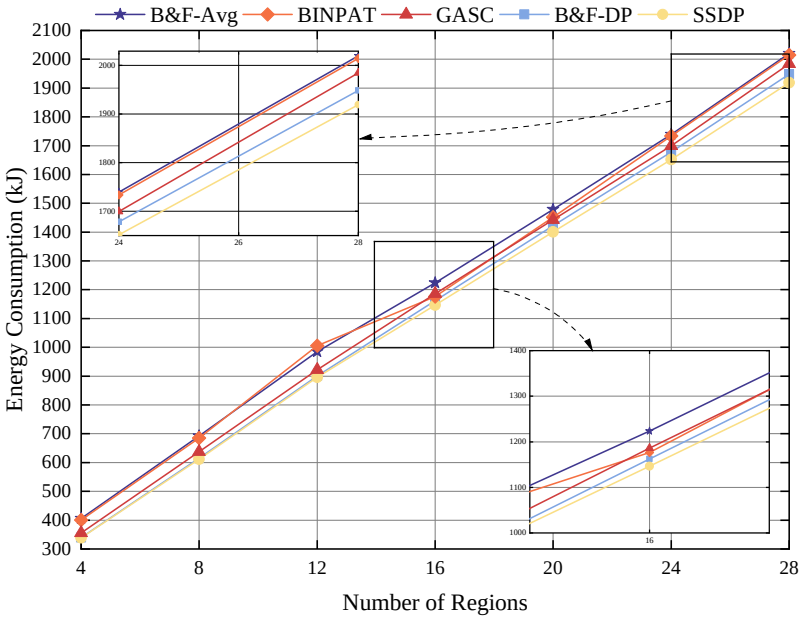


Figure 7. Visual results showing the UAV swarm's total energy consumption (kJ) for each model as the number of regions increases.

As shown in Figure 7, when the number of regions is less than 16, the energy consumption gap between B&F-Avg, BINPAT, and SSDP gradually increases. However, at 16 regions this gap narrows, only to resume its widening trend afterward. This is because SSDP deploys more UAVs when the number of regions reaches 16, leading to higher energy consumption, primarily due to the additional path distance and turns required for the new UAV to reach its waypoints and return to the depot. Similarly, when the number of regions is less than 16, the energy consumption gap between B&F-DP and SSDP also gradually increases. However, after reaching 16 regions, B&F-DP also deploys more UAVs, resulting in an even greater energy consumption gap compared to SSDP as the number of regions exceeds 16. Likewise, the energy consumption gap between GASC and SSDP consistently increases with the number of regions. The energy consumption of SSDP remains the lowest, demonstrating its significant energy efficiency in scenarios that include both convex and non-convex regions during coverage tasks.

5.4. Increasing the Number of UAVs

In this section, the total number of regions was set to 20, with 10 convex regions and 10 non-convex regions. The number of UAVs was increased from 2 to 8 in increments of 1. Tables 8–10 list the statistical results for the UAV swarm's total path distance, total path angle, and energy consumption, respectively. Figure 8 illustrates the changes in the energy consumption as the number of UAVs increases.

From Tables 8 and 9, it is evident that both the total path distance and total turning angle of B&F-Avg and BINPAT increase with the number of UAVs; consequently, their energy consumption also rises, as shown in Figure 8. This suggests that in most cases the distance and turning cost for the previous UAV to return to the warehouse and for the next UAV to fly to the subsequent waypoint are higher than the cost for the previous UAV to fly directly to the next target waypoint. On the other hand, due to GASC's strategy of allocating target regions to UAVs before planning their paths, the total path angle is maintained at a certain level. However, the path length still increases with the number

of UAVs, leading to a gradual increase in energy consumption. In contrast, B&F-DP and SSDP do not necessarily deploy more UAVs as the number of available UAVs increases. Instead, they prioritize minimizing energy consumption; as a result, their total path length, total turning angle and energy consumption remain stable. This highlights that in coverage tasks, increasing the number of UAVs involved does not necessarily result in lower energy consumption.

Table 8. The UAV swarm's total path distance (m) for each model as the number of UAVs increases.

UAV Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
2	10,258.51	10,169.73	9042.86	9628.14	9355.40
3	10,619.30	10,131.65	9037.13	9788.74	9562.48
4	10,834.91	10,120.82	9044.29	10,227.90	9592.84
5	11,210.03	10,139.32	9042.22	10,443.00	9748.43
6	11,475.07	10,141.74	9077.67	10,756.10	9893.69
7	11,696.74	10,155.17	9081.18	11,009.70	10,080.61
8	12,069.84	10,145.89	9062.40	11,313.80	10,208.25

Table 9. The UAV swarm's total path angle (°) for each model as the number of UAVs increases.

UAV Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
2	32,407.56	32,327.15	41,492.66	38,526.20	40,750.50
3	32,725.56	32,303.09	41,502.89	38,714.90	40,179.80
4	32,782.24	32,300.21	41,707.80	38,884.50	40,745.82
5	32,787.89	32,150.89	41,533.94	38,842.30	39,823.04
6	33,031.91	32,393.80	41,473.40	39,152.30	39,681.48
7	33,111.04	32,345.40	41,172.50	39,220.20	40,352.08
8	33,220.21	32,370.22	41,293.46	39,601.50	40,382.98

Table 10. The UAV swarm's total energy consumption (kJ) for each model as the number of UAVs increases.

UAV Number	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
2	1436.75	1426.40	1400.92	1432.81	1426.70
3	1478.73	1422.07	1400.41	1451.99	1442.97
4	1502.44	1420.87	1403.31	1500.83	1452.11
5	1542.71	1421.30	1401.28	1523.45	1459.19
6	1573.66	1424.09	1404.45	1560.24	1473.29
7	1598.25	1425.03	1401.70	1588.13	1500.30
8	1639.38	1424.29	1400.94	1624.69	1514.31

5.5. Adjusting the Decomposition Strategy

In this experiment, we used the same region settings as in Section 5.2 except that the region decomposition method was changed from symmetric to asymmetric. By making this adjustment, we aimed to analyze how region decomposition increases energy consumption during CPP. Tables 11–13 present the UAV swarm's total path distance, path angles, and energy consumption, respectively. Figure 9 shows energy consumption trends as the proportion of non-convex regions increases, while Figure 10 illustrates new UAV paths generated by B&F-Avg, B&F-DP, BINPAT, and GASC for fully non-convex regions.

Comparing the results from Tables 2 and 11, it is evident that the path lengths for B&F-Avg and B&F-DP decreased across all proportions after modifying the decomposition method. For BINPAT, path lengths decrease when the proportion of non-convex regions is less than or equal to 60%, but increase when the proportion exceeds 60%. For GASC, path lengths increase at proportions of 20%, 30%, and 80%, and decrease under other proportions. Comparing the results from Tables 3 and 12, the path angles indicate that

after modifying the decomposition method, B&F-Avg and B&F-DP exhibit increases across all proportions. BINPAT shows increases at proportions of 10%, 20%, 30%, 70%, and 90%, while it decreases at other proportions. For GASC, path angles increase at proportions of 10%, 20%, and 40%, but decrease under other proportions.

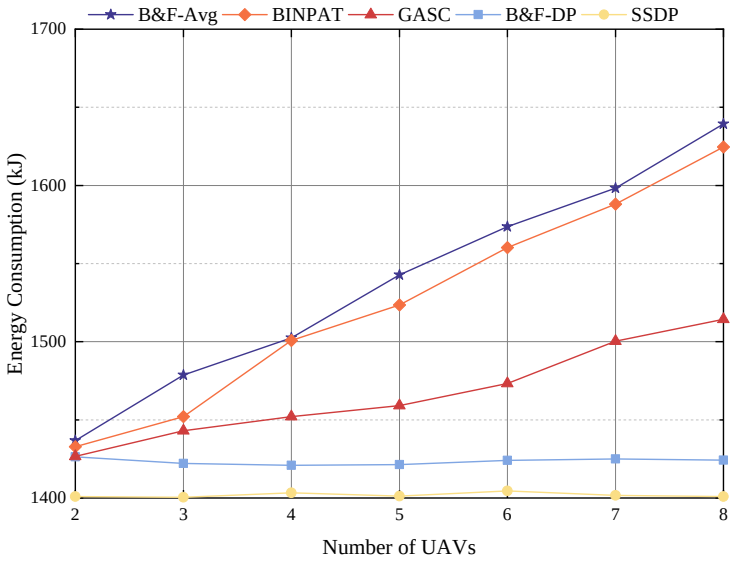


Figure 8. Visual results showing the UAV swarm's total energy consumption (kJ) for each model as the number of UAVs increases.

Table 11. The UAV swarm's total path distance (m) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	9453.91	8852.40	8877.79	9875.32	8662.26
10	9554.93	9049.76	8915.15	9468.79	8717.08
20	9671.34	9162.90	8977.68	9590.18	8928.98
30	9805.27	9378.74	9010.03	9598.00	9053.81
40	9936.69	9427.77	9035.02	9758.37	9195.45
50	10,090.56	9527.23	9059.61	9778.28	9272.76
60	10,404.48	9900.69	9076.22	9818.35	9407.13
70	10,529.18	10,054.75	9102.90	10,403.00	9623.58
80	10,579.82	10,205.74	9149.90	10,422.30	9814.67
90	10,765.14	10,338.96	9172.03	10,524.20	9900.18
100	11,172.68	10,496.22	9208.48	10,489.50	10,044.86

Table 12. The UAV swarm's total path angle (°) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	27,511.61	27,202.47	41,881.81	31,130.10	26,961.34
10	29,172.86	28,936.58	42,004.70	35,152.10	32,317.70
20	30,622.54	30,286.64	41,615.97	35,876.80	32,806.12
30	32,049.60	31,849.18	41,798.76	36,991.20	33,753.66
40	33,593.80	33,143.72	41,616.96	37,143.30	37,605.42
50	35,003.90	34,537.62	41,535.77	37,982.90	39,524.84
60	36,988.22	36,433.38	41,398.59	37,930.00	41,917.56
70	38,528.92	38,209.40	41,173.71	45,940.20	42,943.28
80	39,601.14	39,233.78	40,996.62	44,206.30	43,451.20
90	40,947.40	40,610.08	41,051.14	46,280.20	45,800.46
100	42,841.46	41,667.24	40,844.73	47,706.80	47,061.18

Table 13. The UAV swarm's total energy consumption (kJ) for each model with different proportions of non-convex regions.

Proportion	B&F-Avg	B&F-DP	SSDP	BINPAT	GASC
0	1299.58	1231.88	1387.27	1382.39	1208.99
10	1327.69	1271.07	1392.55	1380.64	1270.57
20	1355.24	1297.24	1395.21	1401.19	1298.37
30	1384.44	1336.63	1400.58	1413.61	1321.61
40	1414.59	1355.35	1401.37	1432.39	1376.85
50	1445.75	1380.51	1403.16	1443.25	1405.10
60	1500.04	1440.26	1403.52	1447.00	1444.39
70	1529.43	1475.25	1404.04	1592.98	1478.26
80	1546.01	1502.09	1407.23	1577.02	1504.02
90	1579.88	1530.68	1410.17	1609.51	1537.62
100	1643.26	1558.53	1411.93	1620.63	1566.25

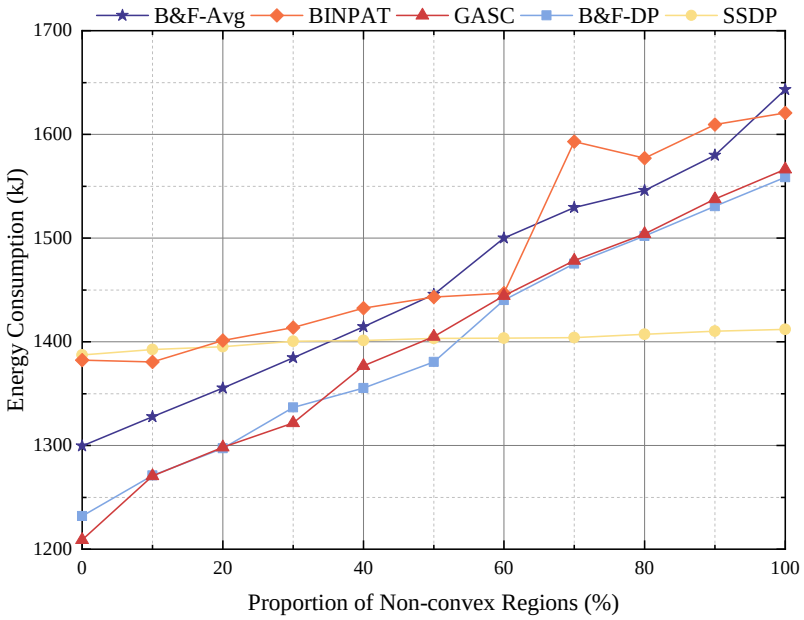
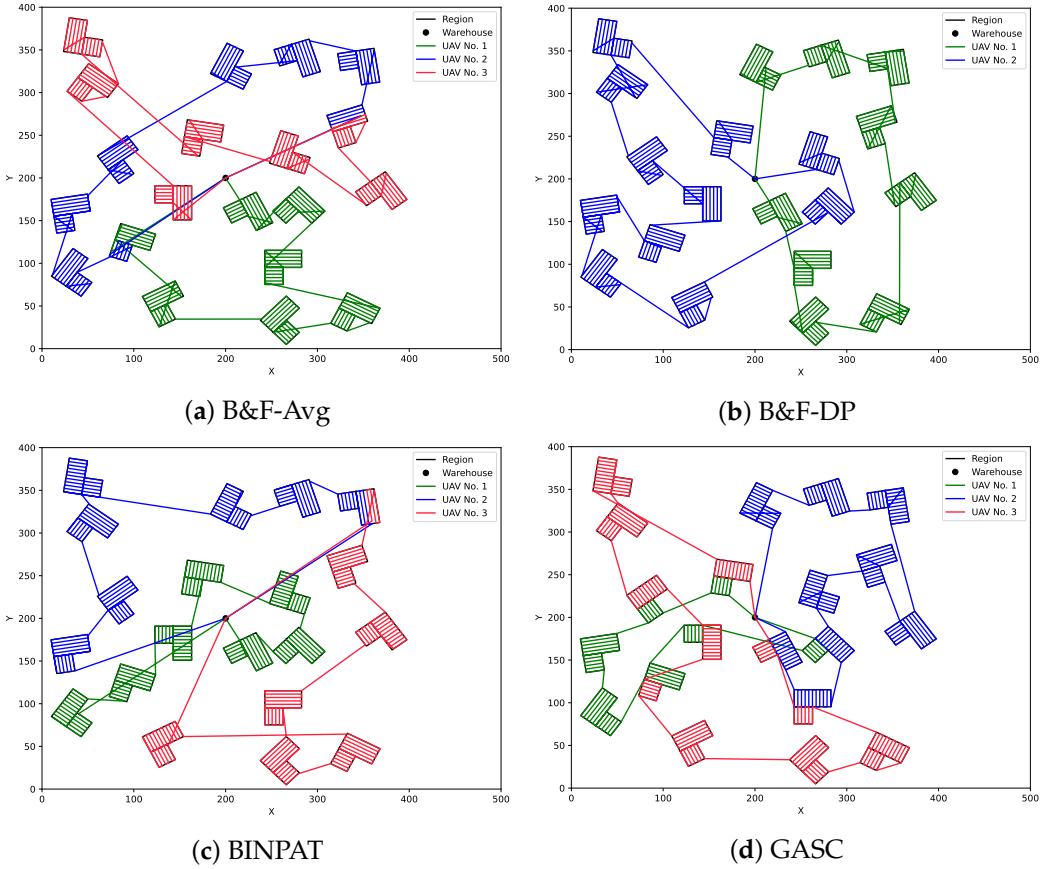


Figure 9. Visual results showing the UAV swarm's total energy consumption (kJ) for each model with different proportions of non-convex regions.

As demonstrated in Tables 4 and 13 and Figures 5 and 9, the results regarding energy consumption indicate that B&F-Avg and B&F-DP achieve consistent reductions in energy consumption, with the maximum decrease not exceeding 50 kJ. For BINPAT, energy consumption decreases when the proportion of non-convex regions is less than 60%, with a noticeable increase observed after this threshold is exceeded. In the case of GASC, energy consumption initially increases for proportions below 20% followed by a subsequent decrease, with the maximum reduction reaching 82.83 kJ. Similar to Section 5.2, B&F-DP consistently remains lower than B&F-Avg, BINPAT, and GASC after reaching 40%. The relative performance of B&F-Avg, BINPAT, and GASC compared to SSDP remains unchanged; for B&F-DP, its energy consumption surpasses that of SSDP starting at a proportion of 50%, with this threshold shifting to 60% after the modifications.

These results indicate that as the proportion of non-convex areas increases, the number of subregions also increases, regardless of the decomposition approach; while non-convex regions slightly increase the energy consumption of SSDP, this increase is significantly smaller than the additional energy required by other methods due to the growing number of subregions. These findings support the conclusion that direct coverage without decomposition effectively reduces energy consumption.



(**a**) B&F-Avg (**b**) B&F-DP

(**c**) BINPAT (**d**) GASC

Figure 10. The new paths generated by B&F-Avg, B&F-DP, BINPAT, and GASC when the target regions are all of non-convex type.

6. Conclusions

This paper introduces an approach named Shrink-Segment by Dynamic Programming (SSDP) for coverage path planning (CPP) with unmanned aerial vehicles (UAVs) in both convex and non-convex mixed regions. SSDP addresses the challenge of covering non-convex regions without requiring pre-decomposition. Specifically, SSDP is a two-stage method. In the first stage, an optimal coverage path is generated using a shrinking ring-based approach, which can be directly applied to both convex and non-convex regions. In the second stage, a dynamic programming-based method is employed to segment the path for UAVs. This process utilizes a two-dimensional matrix and memoization techniques to ensure that energy constraints are met while minimizing total energy consumption.

Experimental results in scenarios with a high proportion of non-convex regions (a common occurrence) or densely distributed regions demonstrate that SSDP achieves lower energy consumption by reducing path length and turning angles compared to state-of-the-art models such as BINPAT and GASC as well as benchmark (B&F-Avg) and ablation (B&F-DP) models. Moreover, our experimental results show that simply increasing the number of UAVs does not necessarily guarantee reduced energy consumption.

While SSDP proves effective in static environments with clearly defined regions, its performance may face challenges in dynamic or obstacle-rich environments. During actual flight, UAVs must account for real-time dynamic avoidance of obstacles such as birds or other flying objects. These obstacles may necessitate fine-tuning of preplanned paths during flight. Real-time dynamic obstacle avoidance will be a key focus of our future research.

Author Contributions: Conceptualization, L.W. and X.Z.; methodology, L.W. and X.Z.; software, X.Z.; validation, L.W., W.Z., and T.Z.; formal analysis, L.W.; investigation, L.W.; resources, L.W. and T.Z.; data curation, L.W. and X.Z.; writing—original draft preparation, L.W. and X.Z.; writing—review and editing, T.Z. and W.Z.; visualization, L.W. and X.Z.; supervision, T.Z., W.Z., and J.C.; project

administration, T.Z.; funding acquisition, T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant 61901388), Shaanxi Province Key R&D Program General Project–Industrial Field (No. 2024GX-YBXM-139), and Innovation Foundation for Doctoral Dissertation of Northwestern Polytechnical University (No. CX2024019).

Data Availability Statement: Experimental data are already included in the article.

Acknowledgments: The authors would like to thank the editors and reviewers for their constructive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cao, Y.; Cheng, X.; Mu, J. Concentrated coverage path planning algorithm of UAV formation for aerial photography. *IEEE Sensors J.* **2022**, *22*, 11098–11111. [CrossRef]
2. Harikumar, K.; Senthilnath, J.; Sundaram, S. Multi-UAV oxyrrhis marina-inspired search and dynamic formation control for forest firefighting. *IEEE Trans. Autom. Sci. Eng.* **2018**, *16*, 863–873. [CrossRef]
3. Koutras, D.I.; Kapoutsis, A.C.; Kosmatopoulos, E.B. Autonomous and cooperative design of the monitor positions for a team of uavs to maximize the quantity and quality of detected objects. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4986–4993. [CrossRef]
4. Adam, M.S.; Nordin, R.; Abdullah, N.F.; Abu-Samah, A.; Amodu, O.A.; Alsharif, M.H. Optimizing Disaster Response through Efficient Path Planning of Mobile Aerial Base Station with Genetic Algorithm. *Drones* **2024**, *8*, 272. [CrossRef]
5. Qin, H.; Shao, S.; Wang, T.; Yu, X.; Jiang, Y.; Cao, Z. Review of autonomous path planning algorithms for mobile robots. *Drones* **2023**, *7*, 211. [CrossRef]
6. Jin, Z.; Nie, L.; Li, D.; Tu, Z.; Xiang, J. An autonomous control framework of unmanned helicopter operations for low-altitude flight in mountainous terrains. *Drones* **2022**, *6*, 150. [CrossRef]
7. Li, J.; Xiong, Y.; She, J. UAV path planning for target coverage task in dynamic environment. *IEEE Internet Things J.* **2023**, *10*, 17734–17745. [CrossRef]
8. Mu, X.; Gao, W.; Li, X.; Li, G. Coverage Path Planning for UAV Based on Improved Back-and-Forth Mode. *IEEE Access* **2023**, *11*, 114840–114854. [CrossRef]
9. Choset, H.; Pignon, P. Coverage path planning: The boustrophedon cellular decomposition. In *Field and Service Robotics*; Springer: London, UK, 1998; pp. 203–209.
10. Palacios-Gasós, J.M.; Talebpour, Z.; Montijano, E.; Sagüés, C.; Martinoli, A. Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1321–1327.
11. Nielsen, L.D.; Sung, I.; Nielsen, P. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. *Sensors* **2019**, *19*, 4165. [CrossRef]
12. Keil, J.M. Polygon Decomposition. In *Handbook of Computational Geometry; Elsevier: Amsterdam, The Netherlands*, 2000; Volume 2, pp. 491–518.
13. Cabreira, T.M.; Di Franco, C.; Ferreira, P.R.; Buttazzo, G.C. Energy-aware spiral coverage path planning for uav photogrammetric applications. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3662–3668. [CrossRef]
14. Modares, J.; Ghanei, F.; Mastronarde, N.; Dantu, K. UB-ANC planner: Energy efficient coverage path planning with multiple drones. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 6182–6189.
15. Luna, M.A.; Ale Isaac, M.S.; Ragab, A.R.; Campoy, P.; Flores Peña, P.; Molina, M. Fast multi-UAV path planning for optimal area coverage in aerial sensing applications. *Sensors* **2022**, *22*, 2297. [CrossRef]
16. Ramezani, M.; Amiri Atashgah, M.; Rezaee, A. A Fault-Tolerant Multi-Agent Reinforcement Learning Framework for Unmanned Aerial Vehicles–Unmanned Ground Vehicle Coverage Path Planning. *Drones* **2024**, *8*, 537. [CrossRef]
17. Vasquez-Gomez, J.I.; Herrera-Lozada, J.C.; Olguin-Carbajal, M. Coverage path planning for surveying disjoint areas. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 899–904.
18. Xiao, P.; Li, N.; Xie, F.; Ni, H.; Zhang, M.; Wang, B. Clustering-Based Multi-Region Coverage-Path Planning of Heterogeneous UAVs. *Drones* **2023**, *7*, 664. [CrossRef]
19. Du, L.; Fan, Y.; Gui, M.; Zhao, D. A Multi-Regional Path-Planning Method for Rescue UAVs with Priority Constraints. *Drones* **2023**, *7*, 692. [CrossRef]
20. Xie, J.; Chen, J. Multiregional coverage path planning for multiple energy constrained UAVs. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17366–17381. [CrossRef]
21. Ahmed, G.; Sheltami, T.; Mahmoud, A. Energy-Efficient Multi-UAV Multi-Region Coverage Path Planning Approach. *Arab. J. Sci. Eng.* **2024**, *49*, 13185–13202. [CrossRef]

22. Jiang, Y.; Bai, T.; Wang, D.; Wang, Y. Coverage Path Planning of UAV Based on Linear Programming—Fuzzy C-Means with Pigeon-Inspired Optimization. *Drones* **2024**, *8*, 50. [CrossRef]
23. Luna, M.A.; Molina, M.; Da-Silva-Gomez, R.; Melero-Deza, J.; Arias-Perez, P.; Campoy, P. A multi-UAV system for coverage path planning applications with in-flight re-planning capabilities. *J. Field Robot.* **2023**, *41*, 1480–1497. [CrossRef]
24. Jonker, R.; Volgenant, T. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR Zusammen mit der NSOR*; Springer: Berlin/Heidelberg, Germany, 1988; p. 622.
25. Fu, Z.; Yu, J.; Xie, G.; Chen, Y.; Mao, Y. A heuristic evolutionary algorithm of UAV path planning. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 2851964. [CrossRef]
26. Lee, C.S.; Phan, T.T.; Kim, D.S. 2D curve offset algorithm for pockets with islands using a vertex offset. *Int. J. Precis. Eng. Manuf.* **2009**, *10*, 127–135. [CrossRef]
27. Luna, M.A.; Isaac, M.S.A.; Fernandez-Cortizas, M.; Santos, C.; Ragab, A.R.; Molina, M.; Campoy, P. Spiral coverage path planning for multi-uav photovoltaic panel inspection applications. In Proceedings of the 2023 International Conference on Unmanned Aircraft Systems (ICUAS), Warsaw, Poland, 6–9 June 2023; pp. 679–686.
28. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
29. Gao, M.; Wei, P.; Liu, Y. Competitive self-organizing neural network based UAV path planning. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 2376–2381.
30. Zhang, J.; Huang, H. Occlusion-aware UAV path planning for reconnaissance and surveillance. *Drones* **2021**, *5*, 98. [CrossRef]