processes

MDPI

Article

A Molecular Force Field-Based Optimal Deployment Algorithm for UAV Swarm Coverage Maximization in Mobile Wireless Sensor Network

Xi Wang, Guan-zheng Tan *, Fan-Lei Lu, Jian Zhao and Yu-si Dai

School of Automation, Central South University, Changsha 410083, China; wx198982@csu.edu.cn (X.W.); 164611087@csu.edu.cn (F.L.L.); 174612234@csu.edu.cn (J.Z.); 184611036@csu.edu.cn (J.-s.D.)

* Correspondence: tgz@csu.edu.cn; Tel.: +86-189-7313-8432

Received: 13 February 2020; Accepted: 16 March 2020; Published: 22 March 2020

Abstract: In the mobile wireless sensor network (MWSN) field, there exists an important problem—how can we quickly form an MWSN to cover a designated working area on the ground using an unmanned aerial vehicle (UAV) swarm? This problem is of significance in many military and civilian applications. In this paper, inspired by intermolecular forces, a novel molecular force field-based optimal deployment algorithm for a UAV swarm is proposed to solve this problem. A multi-rotor UAV swarm is used to carry sensors and quickly build an MWSN in a designated working area. The necessary minimum number of UAVs is determined according to the principle that the coverage area of any three UAVs has the smallest overlap. Based on the geometric properties of a convex polygon, two initialization methods are proposed to make the initial deployment more uniform, following which, the positions of all UAVs are subsequently optimized by the proposed molecular force field-based deployment algorithm. Simulation experiment results show that the proposed algorithm, when compared with three existing algorithms, can obtain the maximum coverage ratio for the designated working area thanks to the proposed initialization methods. The probability of falling into a local optimum and the computational complexity are reduced, while the convergence rate is improved.

Keywords: UAV swarm; MWSN; deployment algorithm; molecular force; coverage maximization

1. Introduction

A mobile wireless sensor network (MWSN) is a wireless network formed by a large number of mobile sensor nodes deployed in a designated area. Over recent decades, MWSNs have evolved rapidly, and many new applications have emerged, including target search and detection, disaster rescue, environmental monitoring, indoor positioning, and so on. However, for most situations, the random deployment of mobile sensors cannot guarantee the required coverage ratio and may cause overlapping coverage or coverage holes. Thus, how to deploy the mobile sensors is a fundamental problem in MWSN [1], as it greatly influences the performance of the coverage of the network. Usually, it is difficult to manually deploy the MWSN in special environments, such as unknown, hostile, or disaster areas [2]. To solve this problem, it is possible to use a multi-rotor unmanned aerial vehicle (UAV) swarm to carry the sensors and build the network autonomously. The reason for choosing multi-rotor UAVs (abbreviated as UAVs below) is that they have some key characteristics, such as high flexibility [3,4], high speed, airborne hovering ability [5,6], and being unrestricted by terrain or obstacles. A typical application is in a military action or disaster situation, where the existing communication facilities have been destroyed. The UAVs can carry the communication sensors by hovering in the air and automatically building an MWSN to provide communication

services. Therefore, designing a deployment algorithm to deploy the UAVs and cooperatively form an MWSN with maximum coverage ratio is a key issue in our study.

Recently, many studies on cooperative work in UAV swarms were carried out by scholars, as the performance given by a single UAV is limited. For example, in [7], a UAV swarm was used to build a queuing network. In [8], to find the optimal position of each UAV in areas with large obstacles, a method using the PSO algorithm was proposed. In [9], to maximize the detection area formed by a swarm of UAVs, a combination of an ant colony-based algorithm and chaotic dynamics (CACOC) was proposed. In [10], a UAV swarm was used for target tracking. In [11], optimal deployment and movement schemes for a UAV swarm were studied.

At present, many intelligent algorithms have been used to solve the MWSN coverage problem [12], such as those presented in [13,14], in which, based on the multi-objective genetic algorithm (MOGA), the coverage area was maximized and the movement distance was optimized. In [15] and [16], in order to increase the coverage ratio and save energy consumption, the multi-objective evolutionary algorithm (MOEA) and the multi-objective immune algorithm (MOIA) were proposed, respectively. Moreover, in [17], a coverage model considering the maximum coverage ratio and the minimum redundancy were given, and an optimization strategy based on the quantum-inspired cultural algorithm was proposed. In [18], The enhanced deployment algorithms (EDA-I and EDA-II) were proposed to achieve a high sensing coverage ratio in the monitored field. In [19], a virtual force-based deployment algorithm—the virtual force algorithm (VFA)—was proposed, which was then improved in [20–22]. The Voronoi diagram is a famous computational geometric structure [23], which has been widely used for sensor deployment problems [24–29]. Guiling Wang et al. [30] designed and evaluated distributed self-deployment protocols for mobile sensors based on the Voronoi diagram; three algorithms (VECtor, VORonoi, and Minimax) were proposed to optimize this problem. In [31,32], the centroid (geometric center) was introduced to improve the original Voronoi deployment algorithm. Combining particle swarm optimization (PSO) and the Voronoi diagram, the PSO_Voronoi and the WSNPSOcon algorithms were proposed in [33,34]. In [29,35], the proposed movement strategies were based on the distances of each sensor and the points inside its Voronoi polygon from the edges or the vertices of the polygon. In [27], based on a rigorous mathematical analysis, a Delaunay-based coordinate-free mechanism (DECM) was proposed for full coverage. In [28], a distributed greedy algorithm was proposed by Sung and Yang, which can improve the effective field coverage of directional sensor networks. In order to save energy and extend the lifetime, the authors proposed the Centralized Immune-Voronoi deployment Algorithm (CIVA) [36] to maximize coverage. In [37], a gradient-based non-linear optimization approach was applied to find a target point for each sensor, following which, the local coverage increases as much as possible when the sensor moves to this point. Mahboubi et al. proposed a set of distributed deployment algorithms based on the distances of each sensor and the points inside its co-ordinate Voronoi polygon from the edges or the vertices of the polygon [35,38]. In [39], the authors modeled the sensor deployment problem as a constrained source coding problem and designed Lloyd-like algorithms to provide a trade-off between sensing coverage and energy consumption.

In this study, we found the VFA-based algorithm to be suitable for solving the coverage problem in our application scenario, as it is simple and efficient and does not require a centralized computing mechanism during implementation as long as the UAVs can perform distributed computing in the same connected MWSN. However, there are still some shortcomings in traditional VFA-based algorithms:

(1) In the traditional VFA-based algorithm, the magnitude of the virtual force is only inversely proportional to the distance between the UAVs. However, we found that the distance and the virtual force should not be in a linear relationship. The closer the distance between each pair of UAVs is, the greater the impact will be. Thus, we should give a higher weight to the closer UAVs, such that the interference of some of the farther UAVs can be filtered out.
(2) The final coverage result is affected by the initialization result in a single test as, for most traditional VFA-based deployment algorithms, the initialization is random; the only requirement is that the UAVs must be deployed in the working area. In this case, the UAVs in

some areas may be significantly denser or sparser than those in other areas due to the randomness. If the random initialization is applied to the convex polygon coverage, it may cause slower convergence or even become stuck in a local optimum. Only by repeating the experiments (re-initializing for each experiment) to obtain the optimal deployment scheme can the robustness of the traditional VFA-based algorithms be embodied.

(3) Most traditional VFA-based algorithms can only be used to cover a rectangular area; some even do not have a boundary to constrain the position of the UAVs but just maximize the coverage area of the UAVs.

In this paper, to overcome the shortcomings of the traditional VFA-based deployment algorithms, a deployment algorithm for a UAV swarm is proposed, which can maximize the ground coverage of an MWSN formed by the UAVs in a designated area. Its main feature is the introduction of the molecular force field to make the force model more accurate. We also designed two initialization methods, which can make the initial distribution of the nodes more uniform, such that the probability of falling into a local optimum and the computational complexity is reduced, while the convergence rate is improved. In addition, convex polygon boundary constraints are added.

2. Formulation and Preliminaries

2.1. Problem Formulation

In this paper, we consider the communication coverage problem of an MWSN in the environment of a three-dimensional space where the existing communications facilities are destroyed (e.g., by war or natural disaster). Our goal is to use many communication sensors to form the MWSN and cover a working area on the ground, providing communication services for ground users, as shown in Figure 1. The communication sensors are carried by hovering multi-rotor UAVs. The proposed deployment algorithm must solve how to optimally deploy these UAVs to maximize the coverage ratio of the working area.

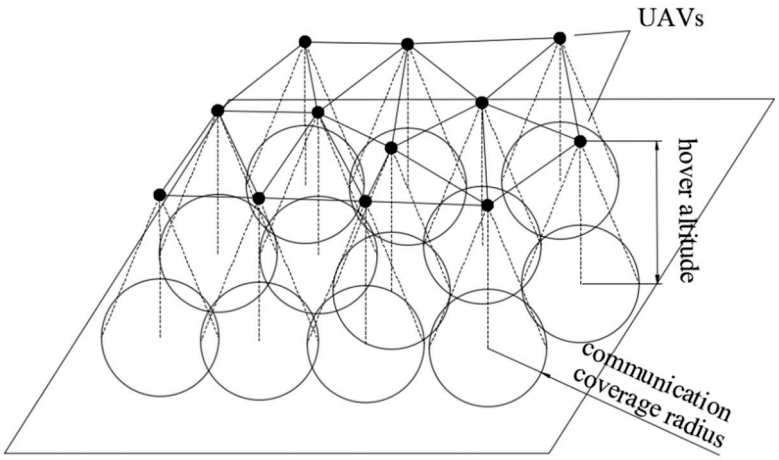


Figure 1. The system model of the communication coverage problem of the mobile wireless sensor network (MWSN). The black points represent the unmanned aerial vehicles (UAVs), and their communication coverage areas on the ground are the circles.

The coverage sensor model used in our study is the binary model, also known as the Boolean disk coverage sensor model, which is the most widely used sensor coverage model [13,15,30]. The communication model of the MWSN is defined as follows:

Definition 1. Each sensor can communicate with other sensors through single or multiple hops in the same connected MWSN.

Definition 2. When a UAV hovers in the air with a communication sensor, the communication range between the communication sensor and the users is denoted as Rc. The interior of the sphere with

the UAV as the center and Rc as the radius are considered the communication space. In this case, the hover altitude (h) should be lower than Rc, such that there is an intersecting circular plane between this sphere and the ground, which is defined as the communication coverage area of the sensor on the ground, the radius of which is denoted by R. The communication sensor can provide communication services to users in this coverage area on the ground. It forms a right-angled triangle with the known Rc (hypotenuse) and hover altitude h (cathetus) and, thus, the value of R can be obtained by the Pythagorean theorem, as shown in Figure 2.

Definition 3. To guarantee network connectivity, the communication range between the sensors is set by at least 2Rc.

Definition 4. All sensors in the MWSN are treated as nodes. There is no obstacle in the working area, and collision avoidance is not considered.

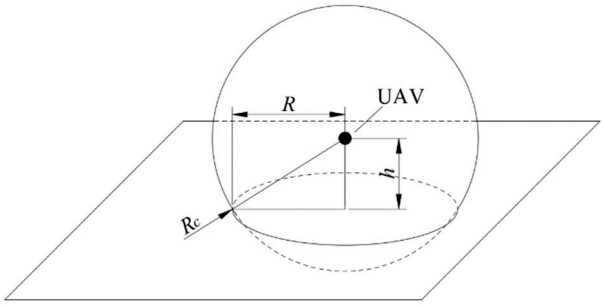


Figure 2. The schematic of the communication coverage area of a UAV on the ground.

The application scenario settings in this paper are defined as follows:

Definition 5. All the UAVs hover at the same altitude, yielding the same coverage area as a result. A suitable hover altitude can maximize the coverage area without being affected by ground obstacles. In this paper, in order to simplify the calculation, the value of R is directly assigned. Moreover, R must be smaller than the radius of the largest inscribed circle of the convex polygon. Otherwise, there is a lot of wasted coverage area, which makes the coverage meaningless.

Definition 6. The working area is a convex polygon with M sides and area S. A convex polygon is defined as a polygon with all interior angles less than 180°; this means that all vertices of the polygon point outwards, away from the interior of the shape. Its vertex coordinates are set as [(xv_1, yv_1), (xv_2, yv_2), …, (xv_M, yv_M)], which are sorted clockwise. The position information of the UAVs, which are defined as [(xu_1, yu_1), (xu_2, yu_2), …, (xu_n, yu_n)], can be exchanged through the MWSN.

Definition 7. The deployment algorithm adopted in our study requires multiple iterations to continuously improve the coverage ratio. Then, the optimal deployment scheme can be obtained, where the optimal position represents where the UAVs should be located in the optimal deployment scheme. In order to avoid wasting energy, all UAVs directly fly to the optimal position from the base, after the iterative calculation is completed and the optimal deployment scheme is obtained, instead of moving once per iteration. This movement strategy was adopted in most relevant studies.

Definition 8. The working area information is sent by the base. After the UAVs receive the working area information, a UAV can independently calculate its optimal position based on the positions of other UAVs obtained through the MWSN and its sequence number. This process is calculated online and is adjusted in real-time according to situation changes. For example, if the working area changes or some UAVs malfunction, the UAVs can still obtain the required information through the MWSN and update the deployment scheme independently.

The coverage problem can be described as follows. There are n UAVs used to cover the working area co-operatively. The optimal deployment scheme is obtained by the proposed algorithm (including initialization and the molecular force field-based algorithm). After the optimal deployment scheme is calculated, the UAVs fly directly to their optimal positions. Next, if the working area changes or some UAVs malfunction, the proposed molecular force algorithm can continuously update the position of UAVs and adapt the deployment scheme to these new changes.

2.2. Coverage Ratio Calculation

In the MWSN field, the coverage ratio (P) is the criterion used for evaluating the performance of an algorithm. The value of P is equal to the ratio of the area covered to the working area (see Equation (1)) The area covered by the sensors is denoted by S_c.

$$P = \frac{S_c}{S} \times 100\%. \tag{1}$$

2.3. Estimate of the Ideal Number of the UAVs

When a UAV swarm is covering a convex polygon working area, blind-spot and overlapping areas cannot be avoided. As is well-known, if we wish to minimize the blind-spot and the overlapping coverage areas, every three circles should intersect at one point, and the center distance of each two circles should be equal to $\sqrt{3}R$. This situation is called an ideal deployment, as shown in Figure 3.

In an ideal deployment, we can see that, compared to the case where a UAV is adjacent to the edge, if a UAV is only adjacent to other UAVs, the effective coverage area (S_s) of this UAV is higher, and there is less wasted area. At this time, the definition of S_s is the inscribed hexagon area in the coverage area of a sensor (i.e., the overlapping area has been removed), as shown in Equation (2) and Figure 3. Therefore, to calculate the average effective coverage area (S_r) of each UAV requires compensation for the wasted area; based on many experiments, the ratio of the compensation was set to 90%. Finally, S_r can be obtained by Equation (3):

$$S_s = 6 \times \frac{1}{2} \times \frac{\sqrt{3}}{2} R \times R, \tag{2}$$

$$S_r = S_s \times 90\% = 1.35\sqrt{3}R^2. \tag{3}$$

Thus, the estimated number of UAVs required can be obtained by Equation (4). The MATLAB function ceil indicates rounding upward (i.e., toward positive infinity):

$$n = \text{ceil}\left(\frac{S}{S_r}\right). \tag{4}$$

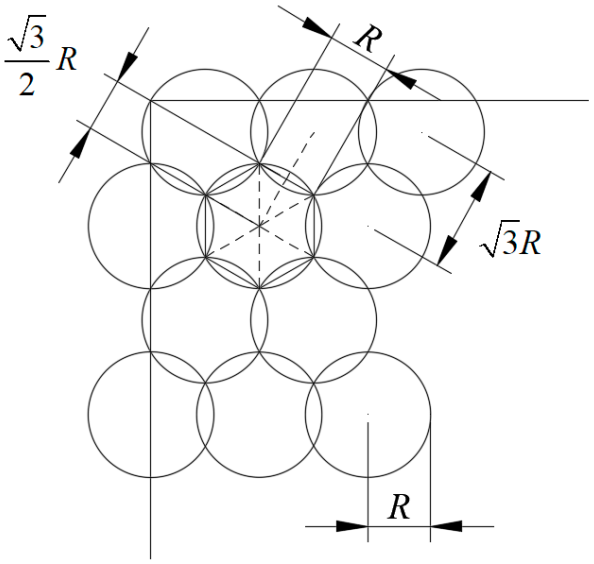


Figure 3. The ideal deployment scheme when using equal circles to cover, where the center distance of each two circles is $\sqrt{3}R$, and the definition of S_s is shown.

2.4. Centroid of the Convex Polygon

The algorithm proposed in this paper must calculate the centroid of the convex polygon. The calculation method is as follows:

Given that a convex polygon has M vertices, its vertex coordinates are denoted in as (xv_i, yv_i), I = 1, 2, …, M+1, which are sorted (either clockwise or counterclockwise), and (xv_1, yv_1) is equal to (xv_{M+1}, yv_{M+1}). The centroid (c_x, c_y) is the average position of all the points of the polygon, which can be obtained by Equations (5–7).

$$S = \frac{1}{2}\sum_{i=1}^{M}\left(xv_i yv_{i+1} - xv_{i+1} yv_i\right), \tag{5}$$

$$c_x = \frac{1}{6S}\sum_{i=1}^{M}\left(xv_i + xv_{i+1}\right)\left(xv_i yv_{i+1} - xv_{i+1} yv_i\right), \tag{6}$$

$$c_y = \frac{1}{6S}\sum_{i=1}^{M}\left(yv_i + yv_{i+1}\right)\left(xv_i yv_{i+1} - xv_{i+1} yv_i\right). \tag{7}$$

2.5. Equal Division Method of the Triangle

The proposed initialization method in this paper requires dividing a triangle into equal areas. The division method is as follows:

In Figure 4, the three vertices of the triangle are denoted by (tx_1, ty_1), (tx_2, ty_2), and (tx_3, ty_3). The length of these three sides are a, b, and c, respectively. The incenter co-ordinate O (x_c, y_c) of this triangle can be calculated by Equation (8). The dashed lines OG, OH, and OI represent the perpendiculars of each side through point O, such that OG = OH = OI. The dotted lines OA, OB, and OC represent the angle bisectors.

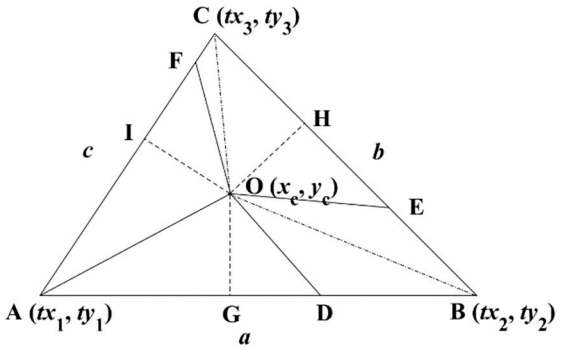


Figure 4. This sample shows the meaning of each symbol in this triangle, including sides, vertices, incenter, equal division points, and perpendiculars.

$$\begin{cases} x_c = \dfrac{btx_1 + ctx_2 + atx_3}{a+b+c} \\ y_c = \dfrac{bty_1 + cty_2 + aty_3}{a+b+c} \end{cases}, \tag{8}$$

where A, D, E, and F are defined as the division points, and OA, OD, OE, and OF divide the triangle ABC into four parts, including the polygons AOD, DOEB, EOFC, and FOA. The areas of the above four polygons are S_1, S_2, S_3, and S_4, respectively, as shown in Equation (9):

$$\begin{cases} S_1 = \frac{1}{2}\mathrm{AD}\times\mathrm{OG} \\ S_2 = \frac{1}{2}\mathrm{DB}\times\mathrm{OG}+\frac{1}{2}\mathrm{BE}\times\mathrm{OH} \\ S_3 = \frac{1}{2}\mathrm{EC}\times\mathrm{OH}+\frac{1}{2}\mathrm{CF}\times\mathrm{OI} \\ S_4 = \frac{1}{2}\mathrm{AF}\times\mathrm{OI} \end{cases}. \quad (9)$$

If we can make AD = DB + BE = EC + CF = FA, then $S_1 = S_2 = S_3 = S_4$ can be easily concluded by the precondition that OG = OH = OI. Therefore, based on the above triangle properties, a new method is proposed to divide any triangle into d_n equal parts. The steps can be listed as follows. First, calculate the perimeter of the triangle and the co-ordinate of each division point. In the example shown in Figure 5, AD = DE = EF = FA. Then, connect the incenter and the division points. Finally, we can obtain d_n polygons with equal area.

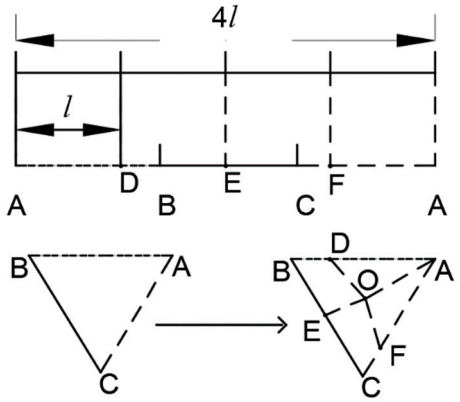


Figure 5. Steps of dividing a triangle into several polygons of equal area; different line types are used to distinguish different sides.

The detailed implementation is given in Algorithm 1, where D(*i*), *i*=1, 2, …, d_n, represent the division points; *anum*, *bnum*, and *cnum* represent the number of the division points in sides *a*, *b*, and *c*, respectively; $l = (a + b + c)/d_n$, as shown in Figures 4 and 5; and poly(*i*), *i*=1, 2, …, d_n, means the *i*-th polygon in the triangle.

Algorithm 1 Dividing a triangle into equal areas

```
for I = 1: dn
  if 0 <l *I ≤ a
    if anum == 0
      Calculate the co-ordinate of point D (i) on the line AB, where the distance between point D (i) and point A is l.
      poly (i) = polygon AOD (i);
    else
      Calculate the co-ordinate of point D (i) on the line AB, where the distance between point D (i) and point A is (l *i).
      Poly (i) = polygon OD (i) D (i-1);
    end
    anum=anum + 1;
  end
  if a < l *I ≤a + b
    if bnum == 0
```

```
14:          Calculate the co-ordinate of point D (i) on
the line BC, where the distance between point D (i)
and point B is (l *i-a).
15:          if anum ≠ 0
16:            poly (i) = polygon OD(i) BD (i-1);
17:          else
18:            poly (i) = polygon AOD (i) B;
19:          end
20:        else
21:          Calculate the co-ordinate of point D (i) on
the line BC, where the distance between point D (i)
and point D (i-1) is l.
22:          poly(i) = polygon OD (i) D (i-1);
23:        end
24:      bnum=bnum + 1;
25:    end
26:    if a + b < l *I ≤ a + b + c
27:      if cnum == 0
28:          Calculate the co-ordinate of point D (i) on
the line CA, where the distance between point D (i)
and point C is (l *i-a-b).
29:          if bnum ≠ 0
30:            poly (i) = polygon OD (i) CD (i-1);
31:          else
32:            poly(i) = polygon OD (i) CBD (i-1);
33:          end
34:        else
35:          Calculate the co-ordinate of the point D (i)
on the line CA, where the distance between point D
(i) and point D (i-1) is l.
36:          poly (i) = polygon OD (i) D (i-1);
37:        end
38:      cnum = cnum + 1;
39:    end
40:  end
```

For any triangle, when $d_n \geq 3$, the generated polygons must be one of three shapes: triangle, quadrangle, or pentagon. However, this leads to generating concave polygons when still using the above method with $d_n = 2$. Therefore, another approach is used for dealing with this situation, which is stated in Section 3.1. If $d_n = 1$, the triangle does not need to be divided.

3. Molecular Force Field-Based Deployment Algorithm for UAV Swarm

3.1. Initialization

To solve the initialization problem, an initialization method is designed. First, the required number of UAVs is determined according to Equation (4). Next, the centroid of the working area, which is denoted as Cen, is obtained by Equations (5–7). After that, the first UAV is deployed at Cen, and the M-gon is divided into M triangles by connecting Cen with each vertex. Finally, the remaining $n-1$ UAVs are put into the M triangles according to the relative area of each triangle. The detailed steps are as follows:

Algorithm 2 The allocation method for the remaining n–1 UAVs

```
for I = 1: M
    Compute the area (s(i)) of tri (i);
    Compute the number (trinum(i)) of UAVs in tri(i) by trinum (i)=round ((n-1) *s(i)/S);
end
n_a = sum (trinum(i));
```

In Algorithm 2, tri(i) means the i-th triangle, and n_a represents the sum of the calculated number of UAVs in these M triangles. The sum function returns the sum of the elements in the array, and the round function means rounding to the nearest integer; therefore, there may be a difference between n_a and $n-1$. In order to eliminate the difference, a solution is proposed at the end of this section.

After Algorithm 2, the working area is divided into M triangles. The number of allocated UAVs in each triangle is proportional to the area of the triangle. This can ensure that the average area that each UAV needs to cover is almost equal. Then, the n_a UAVs are deployed by the following two methods:

Algorithm 3 Obtain the initial deployment positions of the UAVs.

```
Divide each of the M triangles into trinum (M) equal area polygons by the method in Section 3.5 (Algorithm 1).
for i = 1: M
    Compute the perimeter of tri (i);
    Compute the incentre of tri (i);
    if trinum (i) == 1
        poly (i,1) = tri (i);
    end
    if trinum (i) == 2
Connect the midpoint of the i-th side and Cen;
Then, tri (i) is divided into two equal area triangles poly (i,1) and poly (i,2);
    end
    if trinum (i) > = 3
Compute the coordinates of the point that equally divides all sides of tri (i) into trinum (i) parts;
Connect each division point to the incenter of tri (i);
Then, tri (i) is divided into trinum (i) equal area polygons poly (i,1), poly (i,2), …, poly (i, trinum(i));
    end
end
for i=1: M
    for j = 1: trinum (i)
Randomly generate a set of two-dimensional coordinates in poly (i, j) as the position of the q-th UAV (only running in method 1);
Place the q-th UAV in the centroid of poly (i, j) (only running in method 2);
    q=q + 1;
    End
End
```

In Algorithm 3, poly(i, j) means the j-th polygon in the i-th triangle, and q is the sequence number of the UAV. The triangles are divided into n_a polygons, and the polygons in each triangle have the same area.

Figure 6 shows a collection of these examples, where 51 UAVs are used to cover a working area of 6450 m^2, and the working area is a convex polygon with seven sides. From this, we can see the division approach of each initialization method above and the specific meaning of each variable and symbol. The details of Algorithm 2 are shown in the tri(1) part, and the tri(3) part shows a division sample.

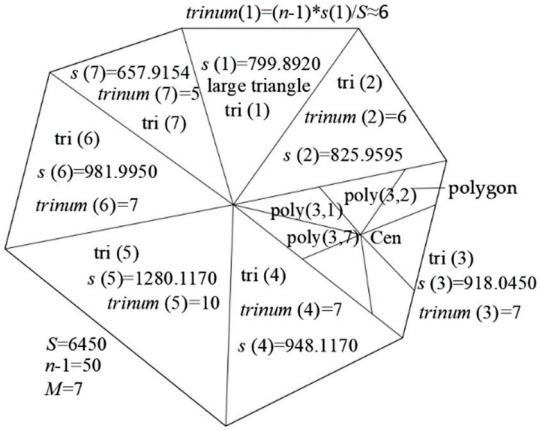


Figure 6. A collection of these examples, which shows the allocation method and the allocated number of the UAVs in each triangle.

After an *M*-gon is divided into *M* triangles, the assigned number of UAVs in each triangle is an approximate value, such that there is some difference between n_a and n–1. In this circumstance, we designed an experiment to observe the magnitude of the difference value. The experimental environment was as follows. The working area was a square of dimensions 100 m (length) × 100 m (width), inside which 1000 convex polygons were randomly generated. There were 30 UAVs in each polygon. The remaining parameter settings were the same as those in Section 4.1. The number of convex polygons with each difference value is shown in Figure 7. We can see that, in the 1000 convex polygons, there were 574 convex polygons whose difference value between n_a and n–1 was zero and, for most convex polygons, the difference value was either −1, 0, or 1. The typical difference value between n_a and n–1 was quite small, and all of them fell in the range [−2,2].

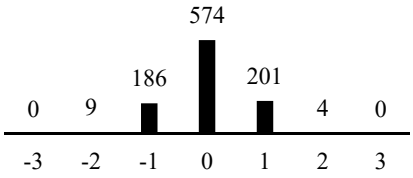


Figure 7. The number of convex polygons with each difference value.

To eliminate such differences, the following strategies were adopted: (1) if $n_a > n$–1, then the unnecessary UAVs (i.e., at the end of the sequence of the UAVs) are discarded; (2) if $n_a = n$–1, then do not make any change; (3) if $n_a < n-1$, then generate the number of missing UAVs randomly in the *M*-gon. After many experiments, we found that this strategy can guarantee the number of deployed UAVs is the same as the predetermined value and, if UAVs are discarded or generated randomly, the results are not influenced very much. However, the proposed strategy still has a shortcoming—the round function is used, thus the number of the UAVs must be corrected after initialization. We will resolve this issue in future research.

3.2. Molecular Force Field Deployment Algorithm

The virtual force model of the UAVs in our study is similar to the molecular force field model, in which molecules must be uniformly distributed within a designated working area. Ideally, the

distance between the UAVs is neither too close nor too far, just as molecules are always uniformly distributed in solids or liquids—neither discrete nor over-compressed. If the shape changes, the molecules automatically adjust the distances between them to maintain a uniform distribution. Obviously, the use of the concept of a molecular force field in analyzing the virtual forces between UAVs provides more accuracy. Therefore, in our study, the virtual force model is improved by introducing the molecular force field.

A molecular force field includes the attractive and the repulsive forces produced by interactions between molecules. Changes in the distances between molecules cause changes in their resultant forces. The rules are as follows: (1) r_0 represents the intermolecular balance distance when the resultant force is zero; (2) if the intermolecular distance is greater than r_0, the resultant force is attraction; and (3) if the intermolecular distance is less than r_0, the resultant force is repulsion.

Inspired by molecular force fields, the implementation steps of the proposed algorithm are defined as follows:

Step 1: Send the information about the working area from the base to the UAVs. According to Equations (2–4), the minimum number of UAVs required can be estimated.

Step 2: Virtually deploy the UAVs in the working area using the proposed initialization method.

Step 3: Calculate the resultant force of each UAV separately, which consists of three parts:

(1) The virtual interaction between UAVs.

The distance between UAVs i and u is denoted by $\boldsymbol{D}_u(i,u)$, $i \in [1,\ 2,\ldots,n]$, $u \in [1,2,\ldots,n]$, which can be calculated by Equation (11):

$$\boldsymbol{D}_u(i,u) = \sqrt{\left(x_i - x_u\right)^2 + \left(y_i - y_u\right)^2}\ . \tag{11}$$

The vector $\boldsymbol{V}_u(i,u)$ goes from UAV i to UAV u, as shown in Equation (12):

$$\boldsymbol{V}_u(i,u) = \left(x_u - x_i, y_u - y_i\right). \tag{12}$$

Then, the force between UAVs i and u can be calculated by Equation (13):

$$\boldsymbol{F}_u(i,u) = \frac{k_1 \boldsymbol{V}_u(i,u)}{\boldsymbol{D}_u(i,u)^2}, \tag{13}$$

where the balance distance is set to $\sqrt{3}R$. If $\boldsymbol{D}_s(i,u) > \sqrt{3}R$, the relationship between UAV i and u is mutually attractive ($k_1 = k_1(\mathrm{g}) > 0$); otherwise, it is mutually repulsive ($k_1 = k_1(\mathrm{r}) < 0$). Here, k_1, k_2, and k_3 (mentioned below) are the control gains; their values are determined by the working area and the coverage radius.

For UAV i, the resultant force of the other UAVs can be obtained by Equation (14):

$$\boldsymbol{F}_{ur}(i) = \sum_{u=1}^{n} \boldsymbol{F}_u(i,u). \tag{14}$$

(2) The interaction between UAVs and boundaries.

First, the distance between UAV i and boundary b, which is denoted as $\boldsymbol{D}_b(i,b)$ $i \in [1,\ 2,\ldots,n]$, $b \in [1,2,..,M]$, should be calculated by Equations (15–18):

$$\begin{cases} \left[\left(xv_1, yv_1\right),\ \left(xv_2, yv_2\right), \cdots, \left(xv_N, yv_N\right), \left(xv_{M+1}, yv_{M+1}\right)\right] \\ \left(xv_{M+1}, yv_{M+1}\right) = \left(xv_1, yv_1\right) \end{cases}, \tag{15}$$

$$\boldsymbol{temp}(i,b) = \frac{\left(x_i - xv_b\right)\left(xv_{b+1} - xv_b\right) + \left(y_i - yv_b\right)\left(yv_{b+1} - yv_i\right)}{\left(xv_{b+1} - xv_b\right)^2 + \left(yv_{b+1} - yv_i\right)^2}, \tag{16}$$

$$\begin{cases} \boldsymbol{px}(i,b) = xv_b + \boldsymbol{temp}(i,b)\left(xv_{b+1} - xv_b\right) \\ \boldsymbol{py}(i,b) = yv_b + \boldsymbol{temp}(i,b)\left(yv_{b+1} - yv_b\right) \end{cases}, \tag{17}$$

where $\boldsymbol{temp}(i,b)$ is an intermediate variable, and the foot of the perpendicular of UAV i on boundary b is denoted as $[px(i,b), py(i,b)]$.

$$\boldsymbol{D}_b(i,b) = \sqrt{\left(x_i - \boldsymbol{px}(i,b)\right)^2 + \left(y_i - \boldsymbol{py}(i,b)\right)^2}. \tag{18}$$

The vector $\boldsymbol{V}_b(i,b)$ goes from UAV i to boundary b, as shown in Equation (19):

$$\boldsymbol{V}_b(i,b) = \left[\boldsymbol{px}(i,b) - x_i, \boldsymbol{py}(i,b) - y_i\right]. \tag{19}$$

Then, the force between UAV i and boundary b can be calculated by Equation (20):

$$\boldsymbol{F}_b(i,b) = \frac{k_2 \boldsymbol{V}_b(i,b)}{\boldsymbol{D}_b(i,b)^2}, \tag{20}$$

where the balance distance is set to $R/2$. If $\boldsymbol{D}_b(i,b) > R/2$, the relationship between UAV i and boundary b is mutually attractive ($k_2 = k_2(\mathrm{g}) > 0$); otherwise, it is mutually repulsive ($k_2 = k_2(\mathrm{r}) < 0$).

For UAV i, the resultant force of other boundaries can be obtained by Equation (21):

$$\boldsymbol{F}_{br}(i) = \sum_{b=1}^{M} \boldsymbol{F}_b(i,b). \tag{21}$$

(3) Interaction between UAVs and vertices.

$D_v(i,v)$ represents the distance between UAV i and vertex v, $i \in [1, 2, \ldots, n]$, $v \in [1, 2, \ldots, M]$, which can be calculated by Equation (22):

$$\boldsymbol{D}_v(i,v) = \sqrt{\left(x_i - xv_v\right)^2 + \left(y_i - yv_v\right)^2}. \tag{22}$$

The vector $\boldsymbol{V}_v(i,v)$ goes from vertex v to UAV i, as shown in Equation (23):

$$\boldsymbol{V}_v(i,v) = \left(x_i - xv_v, y_i - yv_v\right). \tag{23}$$

Then, the force between UAV i and vertex v can be calculated by Equation (24):

$$\boldsymbol{F}_v(i,v) = \frac{k_3 \boldsymbol{V}_v(i,v)}{\boldsymbol{D}_v(i,v)^2}, \tag{24}$$

where the setting of balance distance and k_3 are consistent with the interaction between UAVs.

For UAV i, the resultant force of all vertices can be obtained by Equation (25):

$$\boldsymbol{F}_{vr}(i) = \sum_{v=1}^{M} \boldsymbol{F}_v(i,v). \tag{25}$$

Step 4: All UAVs should be substituted into Equations (11–25) to calculate their respective resultant forces. The resultant force of UAV i (i.e., $\boldsymbol{F}(i)$) can be obtained by Equation (26):

$$\boldsymbol{F}(i) = \boldsymbol{F}_{ur}(i) + \boldsymbol{F}_{br}(i) + \boldsymbol{F}_{vr}(i). \tag{26}$$

Step 5: Then, UAV i moves a distance $l = k_p \left|\boldsymbol{F}(i)\right|$ in the direction of $\boldsymbol{F}(i)$. Similarly to k_1, k_2, and k_3, k_p is also a control gain, which is used to control the rate and the accuracy of convergence. Its value needs to be adjusted according to the experimental environment. If the value of k_p is too large, an optimal solution is not found, and the positions of the UAVs enter a state of continuous oscillation; however, if the value of k_p is too small, the convergence rate is slower.

Step 6: All UAVs should be substituted into Steps 3–5, thus these steps iterate *gen* times, during which, the positions of the UAVs are continuously optimized.

Step 7: Select the deployment scheme with the highest coverage ratio as the optimal deployment scheme, and all UAVs fly to their optimal position directly from the base. In this way, the working area can be properly covered.

Step 8: Finally, if the working area changes or some UAVs malfunction, then repeat Steps 3–7 to dynamically update the optimal deployment scheme. For each change, the optimal deployment scheme is selected after iterating the above algorithm *gen* times, following which, the UAVs fly to their newly obtained optimal positions.

3.3. Computational Complexity of the Proposed Algorithm

The computational complexity is denoted as T (M, gen, n). The entire proposed algorithm includes two main parts: (1) initialization; (2) finding the optimal deployment scheme for the UAVs. $T1$ and $T2$ correspond to the computational complexities of the above two parts, respectively:

$$T\left(M, gen, n\right) = T_1\left(M, n\right) + T_2\left(M, n, gen\right). \quad (27)$$

The first part—initialization—contains three steps: (1) calculating the number of the UAVs in the M triangles separately; (2) generating the n polygons (according to Algorithm 1 in Section 2.5); and (3) calculating the centroids of these polygons (method 2) or randomly deploying them into these polygons (method 1). Method 2 is more complicated than method 1, thus the complexity of this part is equal to method 2. Therefore, for initialization, $T1(M, n) = O(M + n + nM)$. The second part—finding the optimal deployment scheme—consists of two steps: (1) updating the position of the UAVs (this step contains two layers of the nested loop and thus its computational complexity is $O(n^2 + nM + nM)$); and (2) iterating step 1 *gen* times. Therefore, $T2(M, n, gen) = O[(n^2 + nM + nM)*gen]$. Finally, the entire computational complexity is obtained by Equation (28):

$$T\left(M, gen, n\right) = T_1\left(M, n\right) + T_2\left(M, n, gen\right) = O(M + n + nM) + O[(n^2 + nM + nM)gen)] = O(n^2 gen). \quad (28)$$

From that, we can conclude that the amount of computation in the initialization process is significantly smaller than in the subsequent deployment algorithm, even less than the required amount in a single iteration. Therefore, the initialization process plays a large role in reducing the amount of computation.

4. Simulation and Results

In order to verify the performance of the proposed algorithm, many simulation experiments were conducted in our study. The simulation environment was MATLAB 2018b.

The coverage ratio was computed using an image manipulation program. During the process of each iteration, the program generates a coverage picture and counts the number of pixels in the covered area and the whole working area (the overlap coverage area is only counted once). The coverage ratio P is then calculated by Equation (1).

4.1. Performance of the Proposed Algorithm

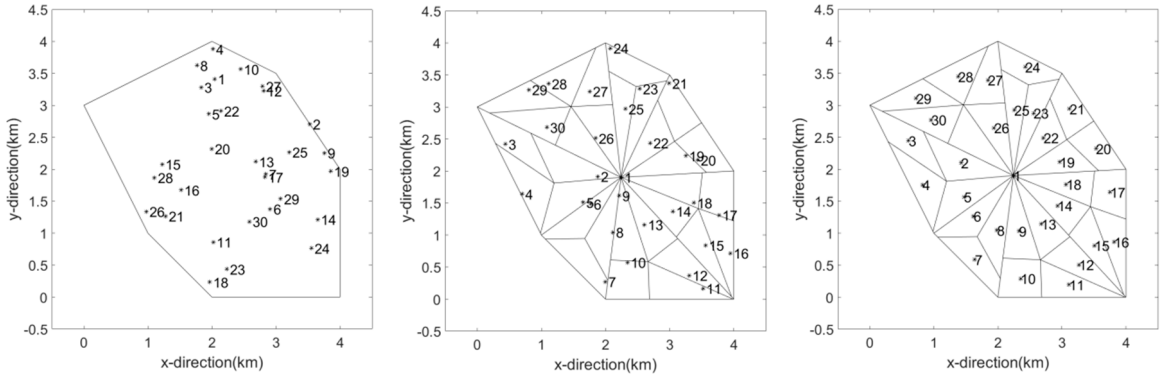
For fair comparison, all simulation experiments were conducted in the same environment. Each set of experiments ran 50 (gen = 50) iterations to ensure convergence. In order to observe the impact of randomness in experiments where the random function was used, they were repeated 20 times.

The working area of these experiments was a convex polygon with seven sides (M = 7). The coordinates of the vertices (in kilometers) were (0, 3), (1, 1), (2, 0), (4, 0), (4, 2), (3, 3.5), and (2, 4), and its area was 11 km^2. According to Equation (4), 30 UAVs are needed to cover the working area. The parameters of the experimental environment were set as follows: R = 0.4 km, k_1(g) = 0.001, k_1(r) = −0.5, k_2(g) = 0.2, k_2(r) = −0.5, k_3(g) = 0.15, k_3(r) = −0.05, and k_r = 0.025.

The experimental records include the coverage ratio P in each iteration. The criteria for better performance were: (1) higher coverage ratio and (2) faster convergence.

We designed three sets of experiments to compare the performance of the two initialization methods in Section 4.1. Test 1 was the control group using the proposed algorithm with random initialization. The distinction among all experiments was the initialization method; Tests 2 and 3 correspond to initialization methods 1 and 2 in Algorithm 3, respectively.

According to the experimental environment and the parameters above, the three sets of experiments were conducted separately. The initialization deployment results of these three methods are shown in Figure 8. As the random function was used in Tests 1 and 2, Figures 8a,b show the results of one of the 20 rounds of the respective experiments. The random function was not used in Test 3 (which is shown in Figure 8c), and thus the outcome is only related to the working area and the number of UAVs.



(**a**) Test 1 (**b**) Test 2 (**c**) Test 3

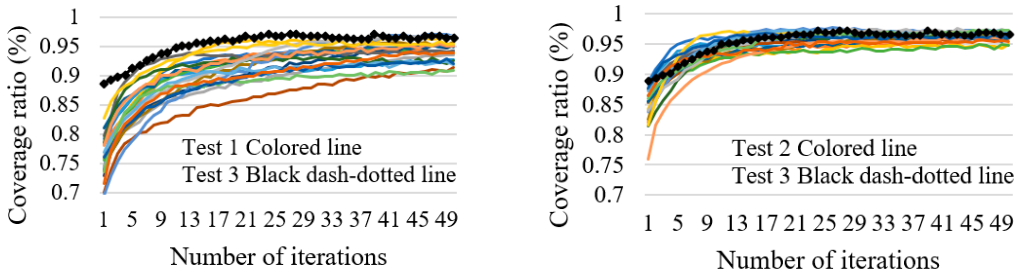
Figure 8. The initial position of UAVs obtained by three initialization methods: (**a**) is the result of random initialization; (**b**,**c**) are the results of the proposed initialization methods 1 and 2, respectively. The black "*" points represent the positions of the UAVs in the working area. The lines represent the dividing lines and boundaries.

After initialization, the proposed algorithm in Section 4.2 was used to maximize the coverage ratio. The simulation results are shown in Figures 9 and 10 and Table 1, where *Best* represents the maximum coverage ratio in all experiments, and *Avg* means the average of the maximum coverage ratio of every round. We can draw the following conclusions from the results: (1) in Table 1, compared with Test 1, Tests 2 and 3 had no obvious advantage in *Best*, but their *Avg* values were significantly higher than those of Test 1. In other words, in Test 1, the results in some rounds were close to the maximum coverage ratios of Tests 2 and 3. However, in most rounds, the maximum coverage ratio was lower; that is to say, in these rounds, they fell into a local optimum. Therefore, using the designed initialization method can reduce the possibility of falling into a local optimum, resulting in a higher *Avg* value. (2) In Figure 9, the black dash-dotted lines represent the coverage ratio of Test 3, and the colored lines in (a,b) represent the coverage ratios of Test 1 and Test 2, respectively. We can see that the average coverage ratio of Test 1 per each round was the lowest, the convergence rate of Test 1 was the slowest, and for Tests 2 and 3, there was not much difference. This means that, if the number of the iterations used in one round experiment was equal, Tests 2 and 3 had a higher probability of finding the deployment scheme with a higher coverage ratio. (3) In Figure 10, compared with Test 1, Test 2 had a higher probability of finding a higher coverage ratio when the number of rounds in the repeated experiments was equal. Test 3 only needed to run a round of experiments and find a higher coverage ratio in this round, which significantly reduced the amount of computation.

Table 1. The simulation results of the *best* and *avg* coverage ratio of Tests 1–3.

Test	*P* (%)		
	Initial	***Best***	***Avg***
1	76.51	97.15	94.47

2	84.78	97.81	96.67
3	88.82	97.16	-



(**a**) (**b**)

Figure 9. For ease of observation, two figures are used to show the comparison of the three experiments in the relationship between coverage ratio and number of iterations. The comparison between Tests 1 and 3 is shown in (**a**), and the comparison between Tests 2 and 3 is shown in (**b**).

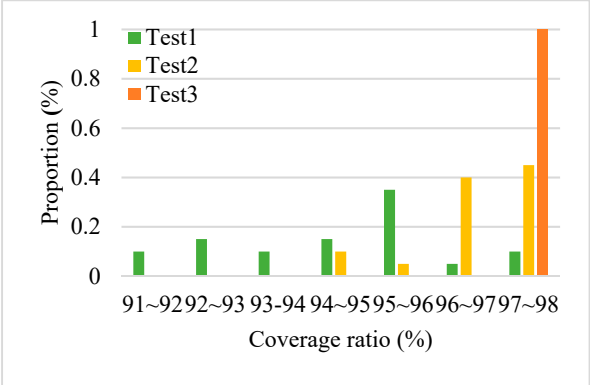


Figure 10. The proportion of the maximum coverage ratio in each numerical interval, in which the distribution proportion of the maximum coverage ratio in the three tests is shown, including 20 rounds of Tests 1 and 2, respectively, and a round of Test 3.

The optimal deployment scheme of Test 3 (method 2) is shown in Figure 11, where the "*" symbols represent the initial positions of the UAVs, the "+" symbols represent the optimal positions of the UAVs, and the circles represent the covered area when the UAVs are at their optimal positions. We can observe that, for most UAVs, the distance between each pair of UAVs was near the ideal value ($\sqrt{3}R$). The distribution of UAVs was close to the ideal deployment scheme, and the boundaries and the vertices were basically covered.

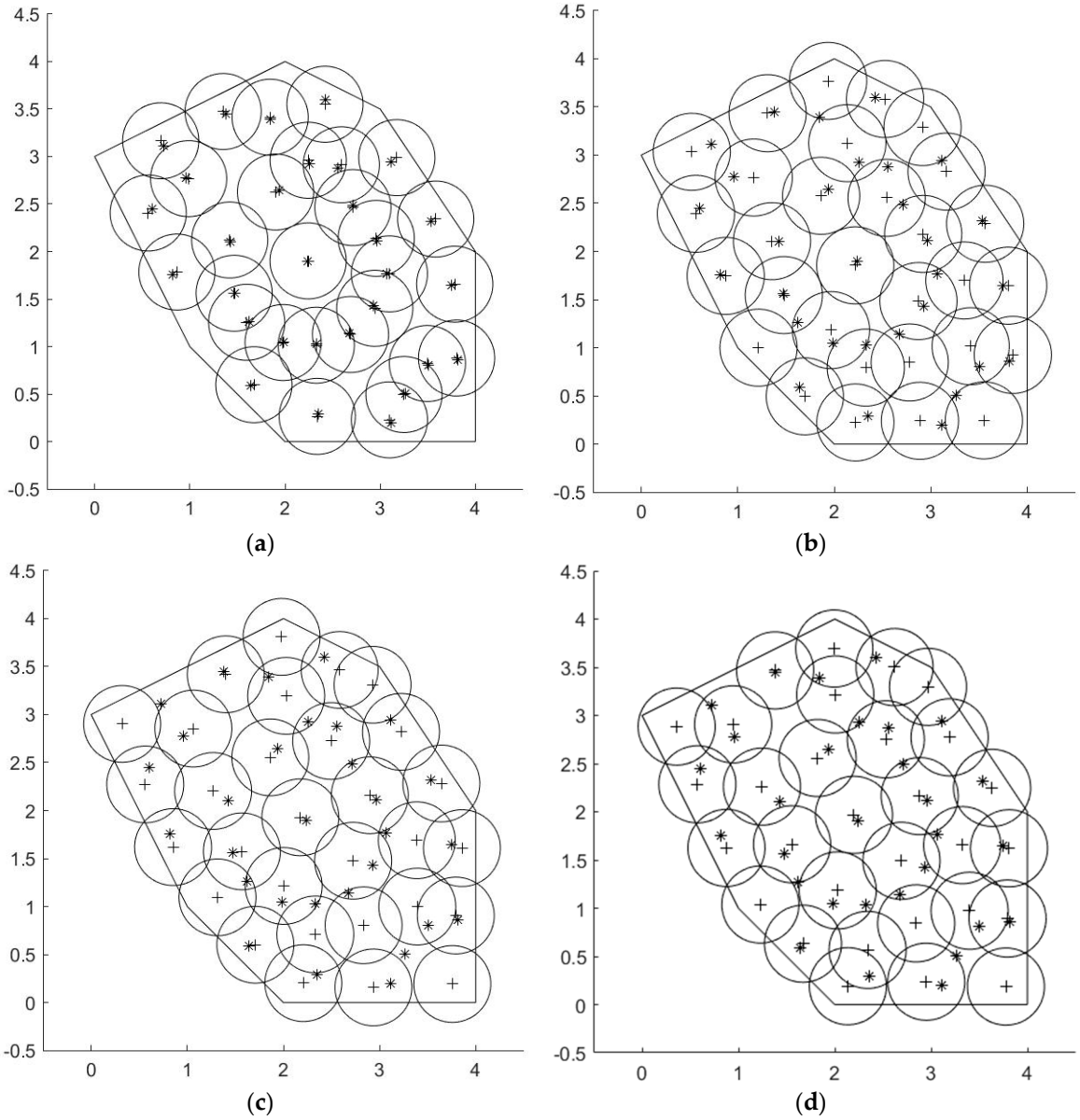


Figure 11. The optimal deployment scheme obtained by the proposed deployment algorithm, where method 2 (Test 3) was used to initialize the positions of the UAVs: (**a**) represents the position of the UAVs in iteration #1; (**b**) represents the position of the UAVs at iteration #17; (**c**) represents the position of the UAVs at iteration #34; and (**d**) represents the position of the UAVs at iteration #50.

4.2. Comparison with Related Algorithms in Terms of the Coverage Ratio

Three related coverage algorithms were selected for comparison: PSO_Voronoi, WSNPSOcon, and CIVA. The traditional VFA algorithm cannot cover a designated working area, causing it to be unfairly compared with other algorithms; therefore, we did not take it into account.

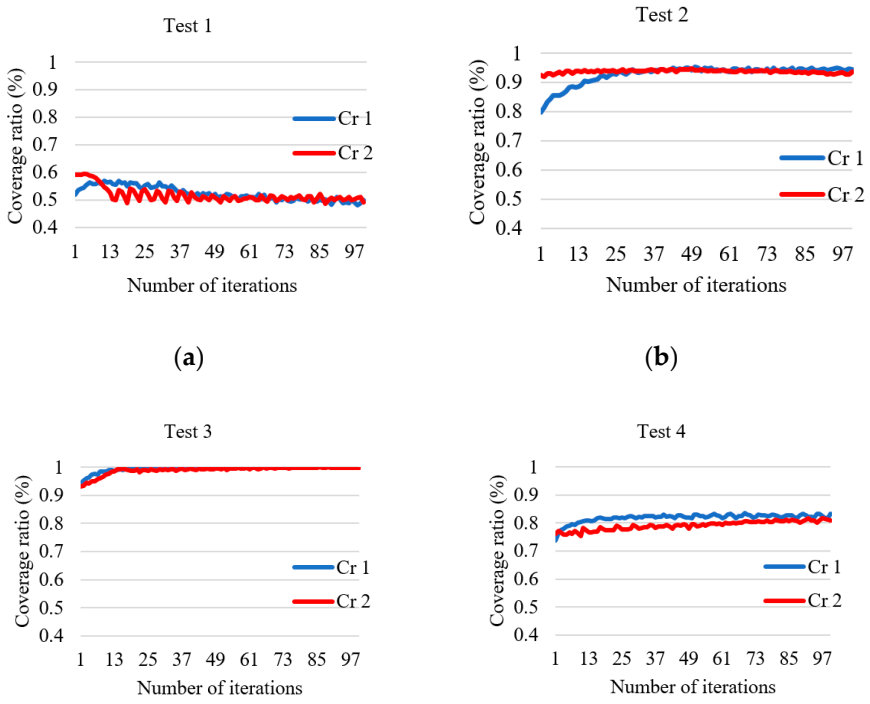
For our proposed deployment algorithm, 50 iterations were performed in each round, and R was set as 7 m. The parameter settings of each set of experiments are shown in Table 2. Additionally, $k_1(g) = 0.015$, $k_1(r) = -0.5$, $k_2(g) = 0.25$, $k_2(r) = -0.5$, $k_3(g) = 0.125$, $k_3(r) = -0.05$, and $k_r = 10$ (Tests 1–3) or $k_r = 15$ (Tests 4–6). Before the proposed molecular force deployment algorithm was used, the two initialization methods in Algorithm 3 were used to initialize the positions of UAVs. For method 1, each experiment was repeated for 20 rounds under different initial conditions; the maximum coverage ratio of all rounds is denoted by Cr 1. For method 2, it only needed to be run once; the maximum coverage ratio in this case is denoted by Cr 2.

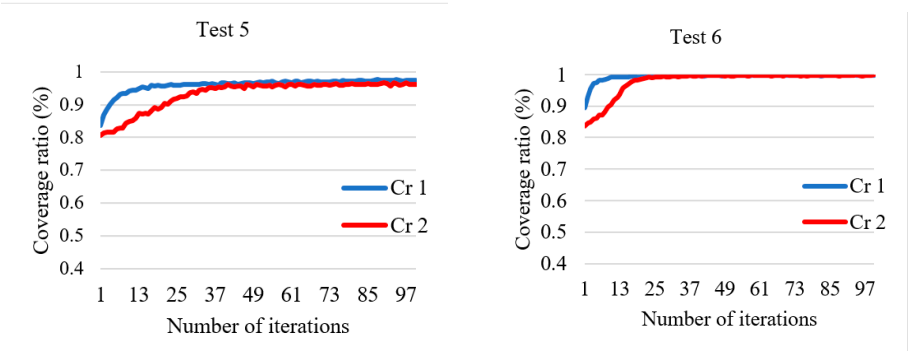
Table 2. Comparison of the coverage ratio of the four deployment algorithms in six tests.

Test	Size	*n* (used)	*n* (ideal)	Coverage ratio (%)				
				PSO_Voronoi	WSNPSOcon	CIVA	Cr 1	Cr 2
1	50 × 50	10	21	58.36	57.96	60.24	56.97	59.83
2	50 × 50	20	21	94.23	93.34	93.55	95.17	94.75
3	50 × 50	30	21	98.80	98.64	97.37	100.00	100.00
4	100 × 100	60	86	77.62	73.56	78.59	83.48	81.74
5	100 × 100	80	86	88.13	81.66	88.94	97.66	96.68
6	100 × 100	100	86	92.70	84.90	92.47	99.87	99.91

From Table 2, we can see that, in Test 1, the coverage ratio of the proposed algorithm was lower than that of CIVA, but the difference between them was small. The main reason for this is that the number of the used UAVs was too small (less than half of the ideal value). In Tests 2–6, the coverage ratio of the proposed algorithm was higher than those of the others. Comparing Cr 1 with Cr 2, we can see that: (1) Cr 1 was lower than Cr 2 only in Test 1; (2) in Tests 2, 4, and 5, Cr 1 was higher than Cr 2; (3) the maximum coverage ratios in Tests 3 and 6 were almost the same.

The relationship between the coverage ratio and the number of iterations of the proposed algorithm is shown in Figure 12, which corresponds to the above Tests (1–6). The two curves in the figures represent the results of the two rounds of the experiments where Cr 1 and Cr 2 were obtained, respectively.





(**e**) (**f**)

Figure 12. The relationship between the coverage ratio and the number of iterations of the proposed algorithm in the six tests. Figures (**a**–**f**) represent Tests 1–6, respectively.

4.3. Results Analysis

From the experimental results described in Section 4.1, we can observe that the proposed algorithm can uniformly distribute the UAVs in the working area. Compared with the control group (random initialization), the convergence rate was significantly improved, and the computational complexity of the algorithm was reduced. Observing the small difference between the value of *Best* in the three tests as well as the significant increase of the *Avg* value in Table 1, we can conclude that the proposed initialization method had a small improvement in *best*, as the algorithm was robust with repeated experiments. However, thanks to the proposed initialization method, the probability of falling into a local optimum was significantly reduced. In other words, the probability of finding the maximum coverage ratio was significantly increased.

From Table 2 and Figure 12 in Section 4.2, we can observe that our proposed algorithm obviously surpassed the other three deployment algorithms in terms of coverage ratio, especially in a high-density environment when the number of the UAVs approached or exceeded the ideal value. As for Test 1, because the number of used UAVs was too small, the coverage ratio was not improved by the proposed deployment algorithm; however, relying only on the initialization methods, an acceptable deployment scheme could also be obtained, whose coverage ratio was about the same as the other three algorithms. For Tests 2–6, with the help of the proposed molecular force field-based deployment algorithm, the coverage ratio increased after initialization. Comparing Cr 1 with Cr 2, it seems that Cr 1 converged faster, but it was selected after 20 rounds of experiments. Meanwhile, only one round was needed to obtain Cr 2, thus reducing the required computation.

Based on the two experiments detailed above, we can draw the following conclusions. If we need a quick deployment, we should use method 2 for initialization, followed by running the entire deployment algorithm once to obtain an optimal deployment scheme. If we need a higher coverage ratio deployment, we should use method 1 for initialization and then repeat the experiment to obtain the optimal deployment scheme. However, this incurs more computation. In either case, the number of the used UAVs needs to approach or exceed the ideal value.

5. Conclusions

In this paper, a coverage maximization deployment algorithm for a UAV swarm in an MWSN is proposed. Using this algorithm, the UAVs can maximize the coverage of the designated working area. Its main feature is the introduction of a molecular force field to make the force model more accurate. To reduce the impact of initialization on the optimal deployment scheme, we also designed two initialization methods, which can make the initial deployment of the UAVs more uniform by dividing the working area into several polygons of equal area. Unlike some deployment algorithms, which can only be used to cover a rectangular area, the proposed algorithm can be used for convex polygonal areas. The simulation experiments show that the proposed algorithm can maximize the

coverage ratio and improve the convergence rate, while the probability of falling into a local optimum and the computational complexity are reduced. The proposed algorithm demonstrated a significant increase in coverage ratio compared to several selected algorithms, especially in high-density environments.

Author Contributions: X.W. came up with ideas, synthesized data, designed and simulated extrusion dies, and wrote the original manuscript; G.-z.T. has instructed, supervised and edited the content; J.Z., F.-L.L., and Y.-s.D. analyzed the results, reviewed, and edited the manuscript. 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 (No.61403422, No.61973320)

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

Data Availability: The data used to support the findings of this study are available from the corresponding author upon request.

References

1. Senouci, M.R.; Mellouk, A.; Asnoune, K.; Bouhidel, F.Y. Movement-Assisted Sensor Deployment Algorithms: A Survey and Taxonomy. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2493–2510, doi:10.1109/COMST.2015.2407954.
2. Mohamed, S.M.; Hamza, H.S.; Saroit, I.A. Coverage in mobile wireless sensor networks (M-WSN): A survey. *Comput. Commun.* **2017**, *110*, 133–150.
3. Kothari, M.; Postlethwaite, I.; Gu, D.W. UAV path following in windy urban environments. *J. Intell. Robot. Syst.* **2014**, *74*, 1013–1028.
4. Kothari, M.; Postlethwaite, I. A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. *J. Intell. Robot. Syst.* **2013**, *71*, 231–253.
5. Kapoor, D.; Deb, D.; Bangar, H.; Sahai, A. Adaptive Failure Compensation for Coaxial Rotor Helicopter under Propeller Failure. In Proceedings of the American Control Conference 2012, Montreal, Canada, 27–29 June 2012.
6. Kapoor, D.; Sodhi, P.; Deb, D. Adaptive Failure Compensation of a Single Main-Rotor Helicopter. In Proceedings of the IFAC Workshop on Embedded Guidance, Navigation and Control (EGNCA), Bangalore, India, 13–15 February 2012.
7. Kirichek, R.; Paramonov, A.; Koucheryavy, A. Swarm of public unmanned aerial vehicles as a queuing network. In *Communications in Computer and Information Science*; Springer: Cham, Switzerland,2016; Volume 601, pp. 111–120.
8. Spanogianopoulos, S.; Zhang, Q.; Spurgeon, S. Fast Formation of Swarm of UAVs in Congested Urban Environment. *IFAC-PapersOnLine* **2017**, *50*, 8031–8036, doi:10.1016/j.ifacol.2017.08.1228.
9. Rosalie, M.; Dentier, J.E.; Danoy, G.; Bouvry, P.; Kannan, S.; Olivares-Mendez, M.A.; Voos, H. Area exploration with a swarm of UAVs combining deterministic chaotic ant colony mobility with position MPC. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 1392–1397.
10. Brust, M.R.; Zurad, M.; Hentges, L.; Gomes, L.; Danoy, G.; Bouvry, P. Target tracking optimization of UAV swarms based on dual-pheromone clustering. In Proceedings of the 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), Exeter, UK, 21–23 June 2017.
11. Koyuncu, E.; Khodabakhsh, R.; Surya, N.; Seferoglu, H. Deployment and trajectory optimization for UAVs: A quantization theory approach. In Proceedings of the Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
12. Zou, Y.; Chakrabarty, K. Sensor Deployment and Target Localization Based on Virtual Forces. In Proceedings of the IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), San Francisco, CA, USA, 30 March–3 April 2003 2003; Volume 2, pp. 1293–1303, doi:10.1109/INFCOM.2003.1208965.

13. Qu, Y.; Georgakopoulos, S.V. Relocation of wireless sensor network nodes using a genetic algorithm. In Proceedings of the Wireless and Microwave Technology Conference (WAMICON), Clearwater Beach, FL, USA, 18–19 April 2011; pp. 1–5.
14. Jia, J.; Chen, J.; Chang, G.; Wen, Y.; Song, J. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Comput. Math. Appl.* **2009**, *57*, 1767–1775, doi:10.1016/j.camwa.2008.10.037.
15. Jin, L.; Jia, J.; Sun, D. Node Distribution Optimization in Mobile Sensor Network Based on Multi-Objective Differential Evolution Algorithm. In Proceedings of the 2010 Fourth International Conference on Genetic and Evolutionary Computing, Shenzhen, China, 13–15 December 2010; pp. 51–54, doi:10.1109/ICGEC.2010.21.
16. Abo-Zahhad, M.; Ahmed, S.M.; Sabor, N.; Sasaki, S. Coverage maximization in mobile Wireless Sensor Networks utilizing immune node deployment algorithm. In Proceedings of the Canadian Conference on Electrical and Computer Engineering; Toronto, ON, Canada, 4–7 May 2014.
17. Guo, Y.N.; Liu, D.; Liu, Y.; Chen, M. The coverage optimization for wireless sensor networks based on quantum-inspired cultural algorithm. In *Lecture Notes in Electrical Engineering*; Springer: Berlin, Heidelberg, 2013; Volume 254 LNEE, pp. 87–96.
18. Lin, T.Y.; Santoso, H.A.; Wu, K.R.; Wang, G.L. Enhanced deployment algorithms for heterogeneous directional mobile sensors in a bounded monitoring area. *IEEE Trans. Mob. Comput.* **2017**, *16*, 744–758.
19. Zou, Y.; Chakrabarty, K. Sensor Deployment and Target Localization in Distributed Sensor Networks. *ACM Trans. Embed. Comput. Syst. (TECS)* **2004**, *3*, 61–91.
20. Loscrí, V.; Natalizio, E.; Mitton, N. Performance evaluation of novel distributed coverage techniques for swarms of flying robots. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, 6–9 April 2014; pp. 3278–3283.
21. Sallam, G.; Baroudi, U.; Al-Shaboti, M. Multi-Robot Deployment Using a Virtual Force Approach: Challenges and Guidelines. *Electronics* **2016**, *5*, 34.
22. Wang, X.; Tan, G.; Liu, X.; Zhao, Z. A Molecular Force-Based Deployment Algorithm for Flight Coverage Maximization of Multi-Rotor UAV. *J. Intell. Robot. Syst.* **2019**, *95*, 1063–1078.
23. Aurenhammer, F.; Klein, R.; Lee, D.-T.; Klein, R. *Voronoi Diagrams and Delaunay Triangulations*; World Scientific: Singapore 2013.
24. Han, Y.H.; Kim, Y.H.; Kim, W.; Jeong, Y.S. An energy-efficient self-deployment with the centroid-directed virtual force in mobile sensor networks. *Simulation* **2012**, *88*, 1152–1165, doi:10.1177/0037549711411314.
25. Pavone, M.; Arsie, A.; Frazzoli, E.; Bullo, F. Distributed algorithms for environment partitioning in mobile robotic networks. *IEEE Trans. Automat. Control* **2011**, *56*, 1834–1848, doi:10.1109/TAC.2011.2112410.
26. Qu, Y. *Wireless Sensor Network Deployment, 2013*; FLORIDA International University, Miami, FL, USA, 2013.
27. Qiu, C.; Shen, H. A delaunay-based coordinate-free mechanism for full coverage in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 828–839.
28. Sung, T.W.; Yang, C.S. Voronoi-based coverage improvement approach for wireless directional sensor networks. *J. Netw. Comput. Appl.* **2014**, *39*, 202–213.
29. Bartolini, N.; Bongiovanni, G.; La Porta, T.; Silvestri, S.; Vincenti, F. Voronoi-based deployment of mobile sensors in the face of adversaries. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 532–537.
30. Wang, G.; Cao, G.; La Porta, T.F. Movement-assisted sensor deployment. *Proc. IEEE INFOCOM* **2004**, *4*, 2469–2479.
31. Lee, H.J.; Kim, Y.H.; Han, Y.H.; Park, C.Y. Centroid-based movement assisted sensor deployment schemes in wireless sensor networks. In Proceedings of the IEEE Vehicular Technology Conference, Anchorage, AK, USA, 20–23 September 2009.
32. Fang, W.; Song, X.; Wu, X.; Sun, J.; Hu, M. Novel efficient deployment schemes for sensor coverage in mobile wireless sensor networks. *Inf. Fusion* **2018**, *41*, 25–36, doi:10.1016/j.inffus.2017.08.001.
33. Aziz, N.A.B.A.; Mohemmed, A.W.; Alias, M.Y. A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi diagram. In Proceedings of the 2009 International Conference on Networking, Sensing and Control, Okayama, Japan, 26–29 March 2009; pp. 602–607.
34. Azlina, N.; Aziz, A. Wireless Sensor Networks Coverage-Energy Algorithms Based on Particle Swarm Optimization. *Emir. J. Eng. Res*. **2013**, *18*, 41–52.

35. Mahboubi, H.; Vaezi, M.; Labeau, F. Mobile Sensors Deployment Subject to Location Estimation Error. *IEEE Trans. Veh. Technol*. **2017**, *66*, 668–678, doi:10.1109/TVT.2016.2537403.
36. Abo-Zahhad, M.; Sabor, N.; Sasaki, S.; Ahmed, S.M. A centralized immune-Voronoi deployment algorithm for coverage maximization and energy conservation in mobile wireless sensor networks. *Inf. Fusion* **2016**, *30*, 36–51, doi:10.1016/j.inffus.2015.11.005.
37. Habibi, J.; Mahboubi, H.; Aghdam, A.G. A gradient-based coverage optimization strategy for mobile sensor networks. *IEEE Trans. Control Netw. Syst.* **2017**, *4*, 477–488, doi:10.1109/TCNS.2016.2515370.
38. Mahboubi, H.; Moezzi, K.; Aghdam, A.G.; Sayrafian-Pour, K.; Marbukh, V. Distributed deployment algorithms for improved coverage in a network of wireless mobile sensors. *IEEE Trans. Ind. Inform.* **2014**, *10*, 163–174.
39. Guo, J.; Jafarkhani, H. Movement-efficient sensor deployment in wireless sensor networks. 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.