IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 22, 2020, accepted January 4, 2021, date of publication January 8, 2021, date of current version January 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049892

A Deep Learning Trained by Genetic Algorithm to Improve the Efficiency of Path Planning for Data Collection With Multi-UAV

YUWEN PAN[1], YUANWANG YANG[1,2], (Member, IEEE), AND WENZAO LI[3,4], (Member, IEEE)

[1]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]Science and Technology on Communication Networks Laboratory, Shijiazhuang, China
[3]College of Communication Engineering, Chengdu University of Information Technology, Chengdu 610225, China
[4]Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Yuanwang Yang (yuanwangyang@uestc.edu.cn)

This work was supported in part by the Science and Technology on Communication Networks Laboratory Foundation Project under Grant 6142104190202, and in part by the Network and Data Security Key Laboratory of Sichuan Province, UESTC, under Grant NDS2021-7.

ABSTRACT To collect data of distributed sensors located at different areas in challenging scenarios through artificial way is obviously inefficient, due to the numerous labor and time. Unmanned Aerial Vehicle (UAV) emerges as a promising solution, which enables multi-UAV collect data automatically with the preassigned path. However, without a well-planned path, the required number and consumed energy of UAVs will increase dramatically. Thus, minimizing the required number and optimizing the path of UAVs, referred as multi-UAV path planning, are essential to achieve the efficient data collection. Therefore, some heuristic algorithms such as Genetic Algorithm (GA) and Ant Colony Algorithm (ACA) which works well for multi-UAV path planning have been proposed. Nevertheless, in challenging scenarios with high requirement for timeliness, the performance of convergence speed of above algorithms is imperfect, which will lead to an inefficient optimization process and delay the data collection. Deep learning (DL), once trained by enough datasets, has high solving speed without worries about convergence problems. Thus, in this paper, we propose an algorithm called Deep Learning Trained by Genetic Algorithm (DL-GA), which combines the advantages of DL and GA. GA will collect states and paths from various scenarios and then use them to train the deep neural network so that while facing the familiar scenarios, it can rapidly give the optimized path, which can satisfy high timeliness requirements. Numerous experiments demonstrate that the solving speed of DL-GA is much faster than GA almost without loss of optimization capacity and even can outperform GA under some specific conditions.

INDEX TERMS Genetic algorithm, deep learning, optimization, multi-UAV path planning, data collection.

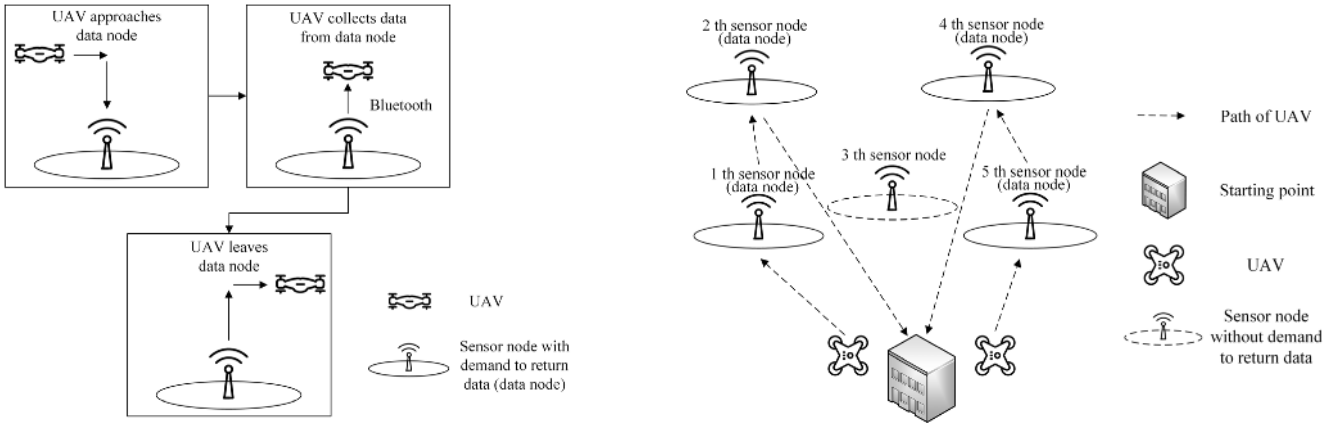
I. INTRODUCTION

A. BACKGROUND AND MOTIVATION

In urban scenarios, it is very convenient to collect data from sensor nodes because of the dense deployment of communication base stations, cable networks and so on. Data can be sent from sensor nodes to the base stations and finally transmitted to the data center in IoT application scenarios. However, in some challenging scenarios without the infrastructure networks like desert and ocean [26], collecting data with unmanned aerial vehicle (UAV) [1]–[3] is obvious the choice of solution. In such data collection scenarios, the process of collecting data from a sensor node is shown in FIGURE 1 (a). UAV will approach to the sensor node and then it will hover over them and use near-field communication protocols such as RFID and Bluetooth to connect with them to collect data. Finally, UAV will leave sensor node. (As the bandwidth and data size vary according to the actual situation of the application system, the data transmission time and energy consumption during the data collection are relatively complex. In addition, in this paper, we focus on improving efficiency of path planning for multi-UAV. So, we do not consider the time and energy consumption in the process of hovering to collect data and the physical communication channel is ignored in this paper.)

In the aforementioned scenarios, there are many sensor nodes deployed at different positions. When UAVs collect data, we may just need the data from a few certain sensor

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.



(a) Collecting data from data node with UAV

(b) Covering all data nodes with UAVs

FIGURE 1. Data collection with UAVs.

nodes according to our requirements. Therefore, we will determine which the sensor nodes need to be collected by UAVs. In other words, just a part of sensor nodes needs to be collected by UAVs, which means they have demand to return data (These sensor nodes are called data nodes.). And those sensor nodes that are not selected by us will not be taken into consideration during the data collection, which means they have no demand to return data. The destination of UAVs is to collaborate to cover all the data nodes in the scenarios with the pre-designed path, whose process is shown as FIGURE 1 (b). (Here, due to the complex concepts of sensor nodes and date nodes, we clarify the difference of them. In FIGURE 1 (b), there are 5 sensor nodes in the scenario. However, only 4 sensor nodes (1,2,4 and 5 th sensor nodes) have the demand to return data, which are called data nodes and will be selected to be collected by UAVs (1,2,4 and 5 th sensor nodes are data nodes). And 3 th sensor node has no demand to return data. Therefore, it is a sensor node but not a data node. All the sensor nodes that have demand to return data will be called data nodes in the rest of this paper.)

However, without a reasonable path to collect data, required resource such as number and energy of UAVs will be increased, causing the inefficient data collection. So, an efficient optimization algorithm is necessary to decide the proper path to collect data. Since not each sensor node has the demand to return data, the algorithm needs to be executed to plan the path every time UAVs collect data. Based on the aforementioned analysis, we know that this problem has high requirement for timeliness. To conclude, we know that there are two key requirements for our algorithm, strong optimization capacity and fast convergence speed (or short solving time).

B. LIMITATION OF RELATED WORK

Multi-UAV path planning is one of the key challenges in the process of data collection. And there are several methods for multi-UAV path planning. Song *et al.* [4] proposed a cellular automata to generate coverage trajectory of UAV with given width of footprint on the ground. But this research focused on designing a path to cover an area, not points. Furthermore, it did not take into account the limitation of energy of UAVs. Sun *et al.* [26] proposed an optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems, which aimed at design a 3D-trajectory and allocate resource for UAV to provide the communication services in the challenging scenarios. Cai *et al.* [27] also proposed a trajectory and resource allocation design for energy-efficient secure UAV communication systems to achieve the maximum system energy efficiency. Although they did not ignore the energy consumption of UAV and achieve great results, they were also not suitable for our problem because they both focused on designing the trajectory of UAV to achieve the optimal area coverage and resource allocation with maximum efficiency. And there are many other proposed algorithms to dynamically control UAVs [5], [6]. But these methods are not feasible for our problem.

Genetic Algorithm (GA) and Ant Colony Algorithm (ACA) are two of the classical heuristic algorithms, which are usually used to plan the path of UAVs to cover all the points in the specific scenarios. Chen *et al.* [7] proposed a GA to minimize total distance of UAV. And Daryanavard and Harifi [8] also proposed an ACA to solve this problem. Besides, there are many different improved GA and ACA [9], [10] to achieve better results. However, no matter GA, ACA or their improved methods, the key shortcoming is their poor performance of convergence speed, which will cause the waste of much time and make it not suitable to optimize multi-UAV's path for data collection that has the high requirement for timeliness. Li and Wu [11] proposed a deep reinforcement learning, which considered the deep neural network as the optimizer and train it by the reward through the interaction with environment. Once deep neural network is trained, it will rapidly give the results of path planning without the convergence process during the optimization.

Therefore, the solving time of optimization process of deep reinforcement learning is much less than GA and ACA, which can satisfy the requirement for timeliness. Nevertheless, deep reinforcement learning is not the supervised learning. Before deep neural network converges, it usually takes many steps to do the random explorations, which will cost much time during training.

In general, for our problem, multi-UAV path planning requires the timeliness for algorithms. Many researches do not take convergence speed into consideration. In other words, they ignore the solving time of algorithms. GA and ACA have the strong optimization capacity but their performance of convergence speed is poor, which makes them not suitable for this problem. Although deep reinforcement learning can rapidly give the multi-UAV path planning results, its training time is not acceptable.

Thus, building a model to resolve multi-UAV path planning to collect data in the challenging scenarios to achieve shortest path and shorter solving time is valuable.

C. OUR METHOD

In the challenging scenarios, it is uncertain which sensor nodes have the demand to return data. GA and ACA have no memory mechanism. In other words, they can not learn the past path planning experiences. So, every time UAVs collect data, GA or ACA needs to be carried out with low computation efficiency. Deep neural network has the ability to learn from the past experiences. And while it is trained, no convergence process is needed during the optimization, which makes its solving time much shorter than GA and ACA. However, deep neural network needs the path planning experiences first.

So, focus on this problem, we propose a Deep Learning Trained by Genetic Algorithm (DL-GA), which considers GA as the ''instructor'' to provide the path planning experiences to deep neural network and guide it to learn from the experiences. We first execute GA to gain the path planning results (including states of scenario and path) and store it in an experience replay. Then, we sample from experience replay to train deep neural network. Finally, the trained deep neural network has the path planning capacity and it can give the optimized results rapidly.

D. CHALLENGES AND CONTRIBUTIONS

There are two main challenges for the multi-UAV path planning in the challenging scenarios. First, due to the limited energy of UAV, without a proper path to collect data will not only waste time and energy but also increase the required number of UAVs to finish the data collection. Second, in the scenario, different states of all sensor nodes (their positions and which sensor nodes have demand to return data) make it have to execute the optimization algorithm to design the trajectory every time. If this algorithm converges slowly (or its solving time is long), then the timeliness requirement cannot be satisfied. It is very complicated to figure out an algorithm with short solving time and strong optimization capacity at the same time.

Based on the above challenges, we propose a DL-GA, where GA will obtain the path planning experiences and then guide deep neural network to learn from it. This algorithm makes path planning do not need any convergence process during optimization and retain the optimization ability from GA, which satisfy the timeliness requirement. Numerical simulation results in section VI will show the superiority of DL-GA.

Our proposed algorithm focuses on covering points in the scenario by multi-UAV with the optimal path. By contrast, [4], [26], [27] both pay attention to design a trajectory for UAV to achieve the optimal area coverage. Compared with [5] and [6], they aim at designing a dynamically control method for UAV. And [7]–[10] are the different heuristic algorithms to solve the UAV path planning. However, their convergence speed is not ideal in practical applications. On the contrary, our algorithm can achieve 300 to 2000 times faster than GA and the optimization capacity can outperform GA under some specific conditions, which makes our algorithm more suitable to solve the multi-UAV path planning for data collection in the challenging scenarios. And finally, [11] proposed a deep reinforcement learning, which will take many steps to train the deep neural network and waste more time compared with our algorithm.

The rest of this paper is organized as follows. Multi-UAV path planning based on GA and DL-GA will be presented respectively in section II and section III. Then, the simulation results will be discussed to prove the performance of DL-GA in section IV. And section V will conclude this paper.

II. MULTI-UAV PATH PLANNING BASED ON GA

In DL-GA, GA plays the role to provide multi-UAV path planning experiences. Since deep neural network will draw on the experiences of GA to design the path, the quality of path planning experiences will influence the results to a large extent. Therefore, an efficient GA is indispensable in DL-GA to get the optimization results.

In this section, we will construct the multi-UAV path planning model based on GA whose processes include fitness computation, selection, crossover and mutation operations in order. System model and all these operations and process of GA will be introduced in follows.

A. SYSTEM MODEL

In the challenging scenarios, data nodes can be seen as points. The destination is to cover all the points by UAVs with the shortest path. Considering there is the energy restriction of UAVs, we suppose there are n data nodes and m sensor nodes, and the data collection order can be expressed as $T = (t_1, t_2, \ldots, t_{n_v})$, where t represents the serial number of data nodes ($t \in [1, m], t \in Z$) and v is required number of UAVs to collect data. And a distance vector that demonstrates the distance between data nodes is given as $G = (g_{1,1}, g_{1,2}, \ldots, g_{v,n_v-n_w+2})$ and the energy consumption

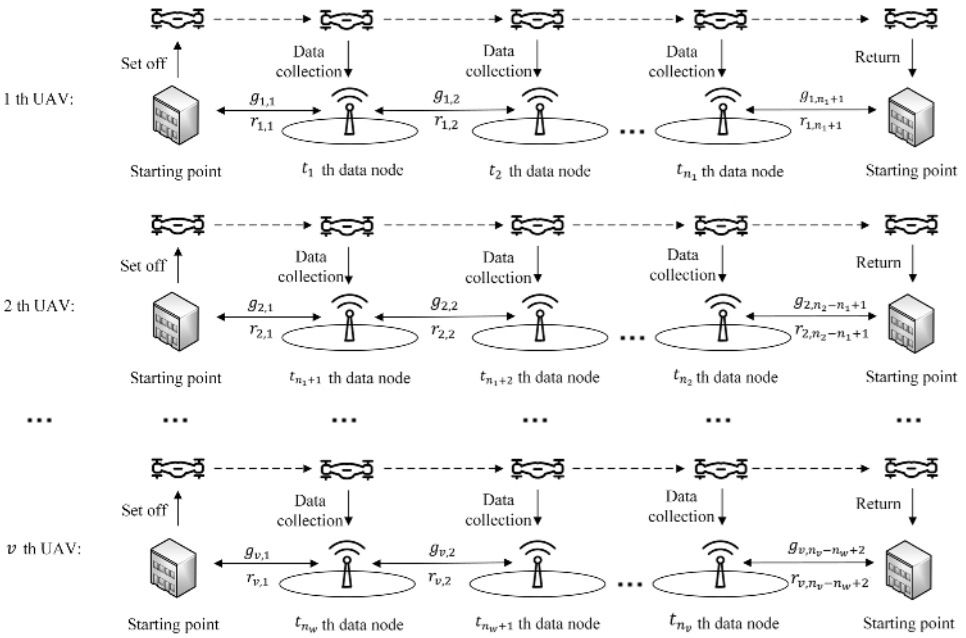


FIGURE 2. Distance vector and energy consumption vector.

vector that denotes the energy consumption between data nodes is $R = (r_{1,1}, r_{2,1}, \ldots, r_{v,n_v-n_w+2})$, where $g_{i,j}$ represents the distance and $r_{i,j}$ is the energy consumption, which are shown in FIGURE 2. Then, this problem can be seen as a complicated Traveling Salesman Problem (TSP) [12], [13], where the number of salesmen may be more than one and there is a restriction to limit the travel distance of every salesman. Based on the above analysis, we suppose each UAV's energy is δ_r. Combined with TSP, this model can be described by equation (1).

$$\begin{cases} min \sum_{i=1}^{v} \sum_{j} g_{i,j} \\ s.t. \begin{cases} \sum_{j} r_{1,j} < \delta_r \\ \sum_{j} r_{2,j} < \delta_r \\ \cdots \\ \sum_{j} r_{v,j} < \delta_r \end{cases} \end{cases} \tag{1}$$

where $\sum_j g_{i,j}$ represents the total distance of i th UAV and $\sum_j r_{i,j}$ is the total energy consumption of i th UAV. And TSP has proven to be a non-deterministic polynomial (NP) problem [29]. Thus, we choose GA to solve this problem in this section.

B. FITNESS COMPUTATION

Fitness can be considered as the quality of path (referred as individual in GA). The destination of GA is to find the individual that has the biggest fitness. Thus, it is important to design a reasonable fitness computation model.

First, supposing there are n data nodes and m sensor nodes in the scenario. Then, a matrix P is given to express the population which consists of M individuals (each row is an individual).

$$P = \begin{bmatrix} \tau_{1,1} & \tau_{1,2} & \cdots & \tau_{1,n} \\ \tau_{2,1} & \tau_{2,2} & \cdots & \tau_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \tau_{M,1} & \tau_{M,2} & \cdots & \tau_{M,n} \end{bmatrix} \tag{2}$$

where $\tau_{k,i} \neq \tau_{k,j} (k \in [1, M]. i \neq j, i, j \in [1, n]. i, j, k \in Z.)$ and $\tau_{k,i} \in [1, m], \tau_{k,i} \in Z$ and $\tau_{k,i}$ is a serial number which denotes $\tau_{k,i}$ th data node. Because data nodes are the sensor nodes that have the demand to return data, $n \leq m$. $P_k = [\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,n}]$ represents an individual that can be understood as the trajectory of required UAVs (See FIGURE 3). Combining with equation (1), fitness and represented trajectory of all individuals can be obtained by Algorithm 1.

Where we suppose the starting point is $\tau_{k,0} = 0$. Then, the energy consumption of UAVs can be expressed as:

$$C_{\tau_{k,i},\tau_{k,j}} = \mu \times D_{\tau_{k,i},\tau_{k,j}} \tag{3}$$

where $D_{\tau_{k,i},\tau_{k,j}}$ represents the distance traveled by UAV between data nodes $\tau_{k,i}$ and $\tau_{k,j}$, which is timed by a coefficient μ to denote energy consumption. And the initial energy of every UAV is δ_m and the remaining energy is δ.

To help it understand Algorithm 1, we present the following example. Supposing k th individual in population P is $P_k = [\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,n}]$. In Algorithm 1, UAV will set off from starting points $\tau_{k,0}$. Then equation (3) will be carried out to calculate the energy consumption of UAV from $\tau_{k,0}$ th data node to $\tau_{k,1}$ data node, which is $C_{\tau_{k,0},\tau_{k,1}}$. And if $C_{\tau_{k,0},\tau_{k,1}} + C_{\tau_{k,1},\tau_{k,0}}$ is less than the current energy of UAV δ,

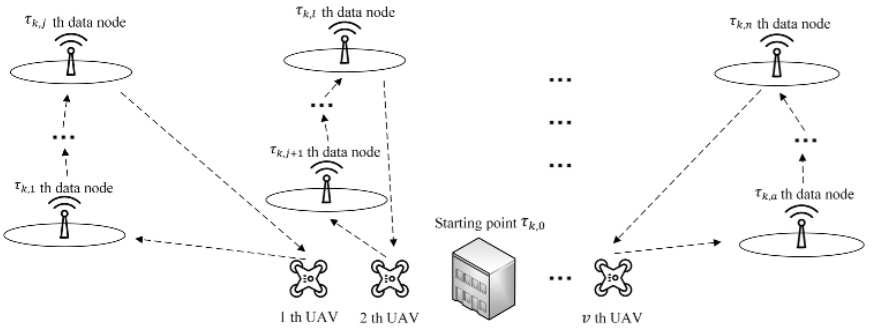


FIGURE 3. Example of covering data nodes by UAVs.

Algorithm 1 Fitness and Trajectory of Required UAVs Computation

Initialize fitness matrix F and trajectory matrix T_r
Initialize energy of UAV $\delta = \delta_m$
for $k = 1, M$ **do**
  Initialize the total distance of UAV $S = 0, i, t = 1, 0$
  Initialize T_t to represent the trajectory of an UAV
  T_t.append($\tau_{k,0}$)
  while True **do**
    if $t == 0$ **do**
      $C = \mu \times D_{\tau_{k,0},\tau_{k,i}}, d = D_{\tau_{k,0},\tau_{k,i}}$
    else do
      $C = \mu \times D_{\tau_{k,i-1},\tau_{k,i}}, d = D_{\tau_{k,i-1},\tau_{k,i}}$
    if $\delta > C + \mu \times D_{\tau_{k,i},\tau_{k,0}}$ **do**
      $\delta = \delta - C, S = S + d$
      T_t.append($\tau_{k,i}$)
      if $i == n$ **do**
        $S = S + D_{\tau_{k,i},\tau_{k,0}}$
        T_t.append($\tau_{k,0}$), T_{r_k}.append(T_t)
        break
      $i = i + 1, t = 1$
    else do
      Initialize energy of UAV $\delta = \delta_m$
      $S = S + D_{\tau_{k,i-1},\tau_{k,0}}, t = 0$
      T_t.append($\tau_{k,0}$), T_{r_k}.append(T_t), $T_t = [\tau_{k,0}]$
  F_k.append(-S)
return F, T_r

which means the UAV has the sufficient energy to cover $\tau_{k,1}$ th data node to collect data and back to the starting point $\tau_{k,0}$, it will cover $\tau_{k,1}$ th data node. Otherwise, it will back to the starting point $\tau_{k,0}$. Perform above steps with the sequence of P_k until the energy of this UAV is not enough and backs to the starting point $\tau_{k,0}$. Then, the trajectory of this UAV is $T_t = [\tau_{k,0}, \tau_{k,1}, \ldots, \tau_{k,j}, \tau_{k,0}]$. And repeat the above steps until all the data nodes are covered by UAVs. Then, the represented trajectory of k th individual P_K can be denoted as:

$$T_{r_k} = \left\{ \begin{array}{c} \left[\tau_{k,0}, \tau_{k,1}, \ldots, \tau_{k,j}, \tau_{k,0}\right]_1 \\ \left[\tau_{k,0}, \tau_{k,j+1}, \ldots, \tau_{k,l}, \tau_{k,0}\right]_2 \\ \cdots \\ \left[\tau_{k,0}, \tau_{k,a}, \ldots, \tau_{k,n}, \tau_{k,0}\right]_v \end{array} \right\} \tag{4}$$

where v denotes the required number of UAVs. And it is shown in FIGURE 3.

According to T_{r_k}, we can gain the trajectory, number and total distance of required UAVs. Because the destination of GA is to find the individual that has the biggest fitness, Algorithm 1 considers the minus total distance of UAVs $(-S)$ as the fitness so that GA can seek the shortest path.

C. SELECTION

After obtaining the fitness of population, selection operation will be carried out to choose the individual according to its fitness. Roulette Algorithm (RA) [14], [15] is used in our algorithm. First, $\gamma(j)$ is calculated by the following equation to represent the probability of j th individual being selected.

$$\gamma(j) = \frac{F_j}{\sum_{i=1}^{M} F_i} \tag{5}$$

where F is the fitness matrix obtained by Algorithm 1. Second, the new matrix ϑ will be established by choosing individuals for M times from P by the probability $\gamma(j)$. Then, Elitism Strategy (ES) [16], [17] is used to select the maximum fitness individual and spliced with ϑ to form the new population $\bar{P}$ so that the best individual can be kept during the selection operation.

D. CROSSOVER AND MUTATION

In our GA, crossover operation is to change the individual substantially, which makes it to search the global optimal solution. And Mutation operation is to change the individual slightly so that it can search the local optimal solution. GA combines the two type of operations to ensure its global and local search capacity. The process of them is as followed.

Crossover: Crossover operation randomly chooses two individuals from population $\bar{P}$ (except the maximum fitness individual) [18]. Supposing they are $P_k = [\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,n}]$ and $P_h = [\tau_{h,1}, \tau_{h,2}, \ldots, \tau_{h,n}]$. Randomly select a position number e (where $e \in [1, n], e \in Z$) and swap $\tau_{k,e}$ and $\tau_{h,e}$. Then, find the position numbers e_1 of $\tau_{k,e}$ in P_h and e_2 of $\tau_{h,e}$ in P_k. Finally, swap $\tau_{k,e2}$ and $\tau_{h,e1}$.

Mutation: We set k as 0. Mutation operation will choose individuals by mutation probability m_o [19]. Every time a

random number $\pi \in [0, 1]$ is generated and if $\pi < m_o$, the current kth individual will be chosen. Each selected individual $P_k = [\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,n}]$ will perform the following operations. First, two position numbers e_1 and e_2 are randomly chosen. Then, $\tau_{k,e1}$ and $\tau_{k,e2}$ are swapped. Finally, perform $k \leftarrow k + 1$ and repeat above steps until $k > m$.

As conclusion, GA takes maximizing fitness as its destination, adjusting the population structure by selection operation and searching the optimal solution by crossover and mutation operations. Multi-UAV path planning based on GA can be summarized as follow.

Algorithm 2 Multi-UAV Path Planning Based on GA

Initialize population P
Block:
  Perform Algorithm 1 to get the fitness matrix F
  Input F and perform selection operation
  Repeat crossover operation for α times
  Perform mutation operation
  Delete the individual whose fitness is the smallest
if maximum fitness has no change for ε iterations **do**
  return to **Block**
else do
  return maximum fitness individual
  break

III. MULTI-UAV PATH PLANNING BASED ON DL-GA

GA has the advantage of strong optimization ability. However, its performance of convergence speed is very poor which will cause the waste of time. It is not suitable to optimize the problems with high requirement for timeliness like multi-UAV path planning to collect data. So, in this section, we propose a DL-GA to solve this problem because its excellent performance in terms of solving speed.

In DL-GA, first, GA obtains the states and paths from the scenario and stores them in an experience replay. Second, deep neural network will study the experiences of path planning from experience replay. And finally, after training, deep neural network can rapidly give the results of path planning without the convergence process, which can satisfy timeliness requirements. The specific processes will be introduced in the follows.

A. EXPERIENCE REPLAY COLLECTION

Supposing there are n data nodes and m sensor nodes in the scenario. Their coordinate arrays can be expressed as $\rho_x = [x_1, x_2, \ldots, x_m, \hat{x}]$ and $\rho_y = [y_1, y_2, \ldots, y_m, \hat{y}]$, where x_i and y_i represent the coordinate of i th sensor nodes. $\hat{x}$ and $\hat{y}$ denote the coordinate of starting point. Then, a matrix $s = [\epsilon_1, \epsilon_2, \ldots, \epsilon_m, \hat{\epsilon}]$ is given to denote the states of all sensor nodes, where ϵ_i is the state of i th sensor node and $\hat{\epsilon}$ represents the state of starting point (If i th sensor nodes has been covered or it has no demand to return data, its state ϵ_i is equal to 0, otherwise equal to 1 and $\hat{\epsilon} \equiv 0$). We use σ to represent the serial number of the sensor nodes where the current UAV is located. So, the states of the scenario can be denoted as $\varphi = [\rho_x, \rho_y, s, \delta, \sigma]$. The path section is the next serial number of the data node that the UAV is going to, which is gotten by GA. And the experience replay can be represented as $E_X = [\bar{\varphi}; \bar{\tau}]$, where $\bar{\varphi}$ is the set of φ and $\bar{\tau}$ is the set of path sections. The process of experience replay collection is shown in Algorithm 3.

Algorithm 3 Experience Replay Collection

Obtain the maximum fitness individual from Algorithm 2 (Supposing it is P_k)
Initialize number of data nodes n
Initialize energy of UAV $\delta = \delta_m$
Initialize state array φ
Obtain the experience replay E_X
$i = 1, t = 0$
while True **do**
  if $t == 0$ **do**
    $C = \mu \times D_{\tau_{k,0}, \tau_{k,i}}$
  else do
    $C = \mu \times D_{\tau_{k,i-1}, \tau_{k,i}}$
  if $\delta > C + \mu \times D_{\tau_{k,i}, \tau_{k,0}}$ **do**
    Store state φ and path section $\tau_{k,i}$ in the E_X
    $\delta = \delta - C$, $\epsilon_{\tau_{k,i}} = 0$, $\sigma = \tau_{k,i}$, update φ
    if $i == n$ **do**
      break
    $i = i + 1, t = 1$
  else do
    Store state φ and path $\tau_{k,0}$ in the E_X
    Initialize $\delta = \delta_m$, $\sigma = \tau_{k,0}$, update φ
    $t = 0$
return E_X

Algorithm 3 can store the states of the scenario and path sections in the experience replay. During storing, not only the number of data nodes n ($n \in [o, m]$.$n \in Z$) but also which sensor nodes have the demand to return data will be randomly changed so that the experience replay can collect the various samples to train the deep neural network plenty.

B. EXPERIENCES LEARNING

After getting the experience replay, the next step is to train the neural network. States of the scenario φ and the path sections are the input features and labels of neural network. We choose convolution neural network (CNN) to study experiences because its superior features extraction capacity [20]–[23]. Before training, the input features φ should be preprocessed including normalization and reshaping, which are given as follows.

Normalization: Supposing $\varphi = [\rho_x, \rho_y, s, \delta, \sigma]$ and the size of the scenario is $z_x \times z_y$. Then the normalization can be expressed as followed equation.

$$\begin{cases} \rho_x' = \rho_x / z_x \\ \rho_y' = \rho_y / z_y \\ s' = s \\ \delta' = \delta / \delta_m \\ \sigma' = \sigma / m \end{cases} \tag{6}$$

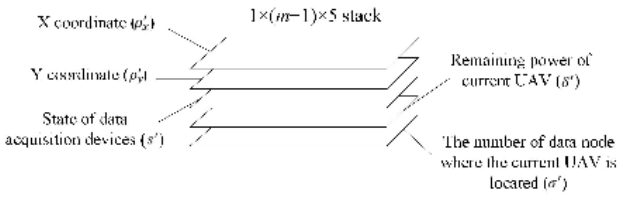


FIGURE 4. Reshaping input features.

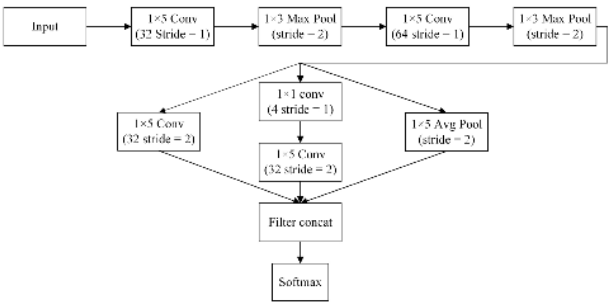


FIGURE 5. Architecture of CNN.

The normalized input features can be represented as $\varphi' = [\rho'_x, \rho'_y, s', \delta', \sigma']$.

Reshaping: Since we choose CNN as neural network, we reshape the normalized input features φ' into ψ, which is shown in FIGURE 4.

After the data preprocessing, we are going to draw on the experiences of Network in Network (NIN) [24] and inception model [25] to build the CNN, whose architecture is shown in FIGURE 5.

Here, the classification result denotes the path section of the UAV (For instance, if the input feature is ψ and result of classification is i, then the current UAV will set off to i th data node.). And finally, the data will be randomly sampled from the experience replay to train the CNN.

C. MULTI-UAV PATH PLANNING BY CNN

From multi-UAV path planning based on GA, experience replay collection and experience learning, we know that GA provides the multi-UAV path planning experiences and store it in the experience replay, and then we will sample from the experience replay and use it to train CNN. Therefore, the trained CNN has the capacity to plan the multi-UAV's path. And because there is no convergence process during optimization with CNN, it can quickly figure out the multi-UAV's path, which saves time and reduce the computation load. The process is given as Algorithm 4.

According to Algorithm 4, we can get the trajectory of UAVs T_r. Finally, we display the whole process of multi-UAV path planning with DL-GA in FIGURE 6 (supposing $n = 4$, $m = 5$ and $v = 2$). Compared with GA, the computational complexity of DL-GA is much lower. Therefore, it is not hard to understand that the solving time of DL-GA is less than GA (See APPENDIX for full computational process.).

Algorithm 4 Multi-UAV Path Planning by CNN

```
Initialize number of data nodes n
Initialize the state of scenario φ
Initialize path array T_r
Initialize a temporary array T_t
T_t. append (0)
while True do
    Normalize φ and reshape it into ψ
    Input ψ to CNN and get the path section i
    if i == 0 do
        T_t. append (0)
        T_r. append (T_t)
        Initialize the temporary array T_t
        T_t. append (0), update φ
    else do
        T_t.append(i)
        Update φ
    if all data nodes are covered do
        break
return T_r
```

TABLE 1. Experimental parameters.

Parameters	Value
δ_m	3×10^5J
M	100
μ	20J/m
ε	500
α	2
m_o	0.6

IV. SIMULATION RESULTS

We evaluate DL-GA based on three factors: total distance of UAVs, required number of UAVs and solving time. First, we collect experience replay based on GA according to Algorithm 3. Then, normalization and reshaping are employed to preprocess the input features, which is used to train CNN. And finally, Algorithm 4 is applied to the multi-UAV path planning for data collection. To prove the superiority of DL-GA, we also use GA and random choosing as the contrast.

There are up to 20 sensor nodes in the scenario ($m = 20$), and the minimum number of data nodes is 5 and the maximum number of data nodes is 20 ($n \in [5, 20]$). The size of the scenario is set as $5000m \times 5000m$ [28] ($z_x = 5000m, z_y = 5000m$). Other important parameters are shown in the TABLE 1 [30].

To verify the validity of DL-GA, we set the number of data nodes from 5 to 20. And in every condition, 100 simulations are carried out. We record the average total distance of UAVs with DL-GA, GA and random choosing, average require number of UAVs with DL-GA, GA and random choosing, average solving time with DL-GA and GA under the different conditions.

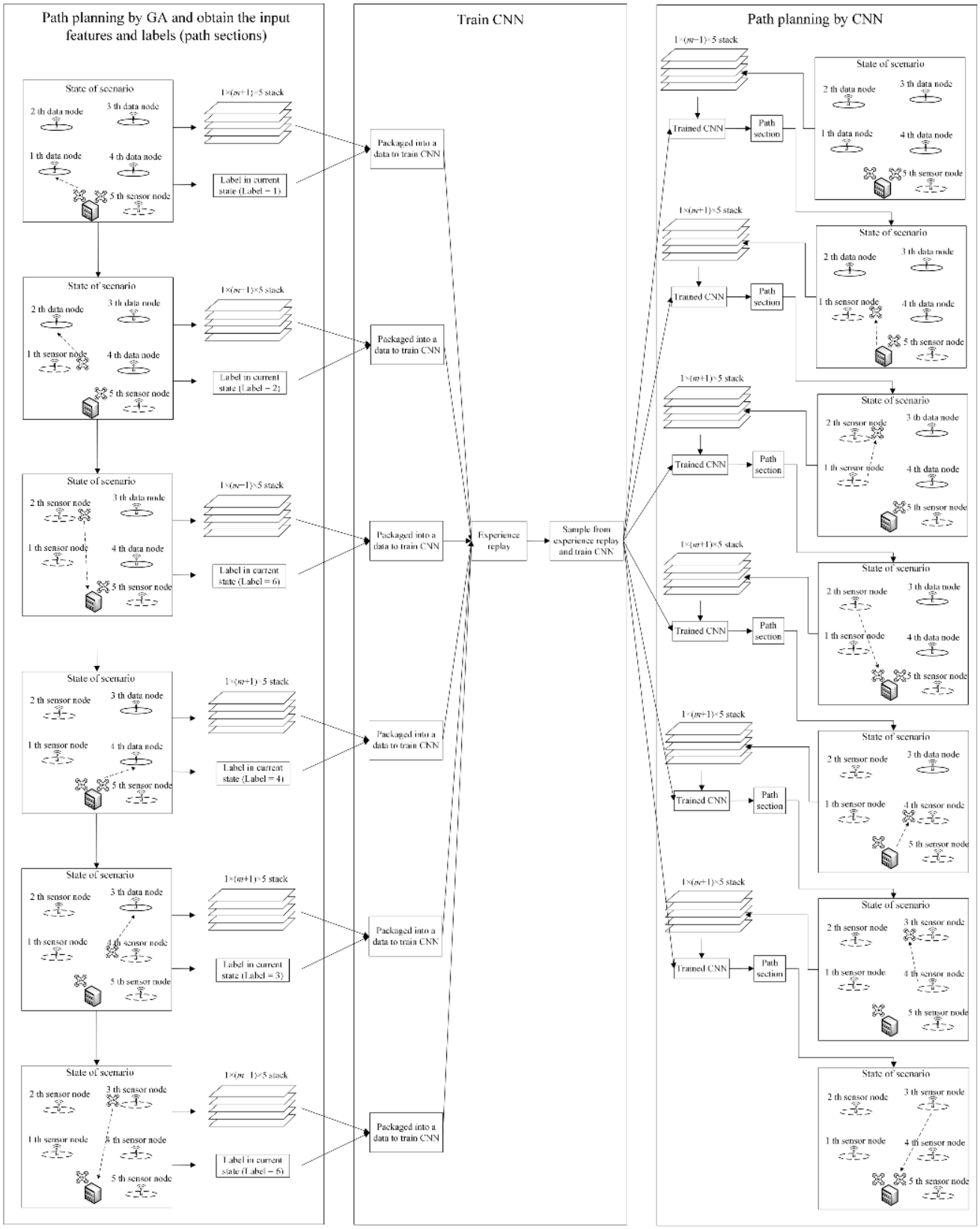


FIGURE 6. Process of multi-UAV path planning with DL-GA.

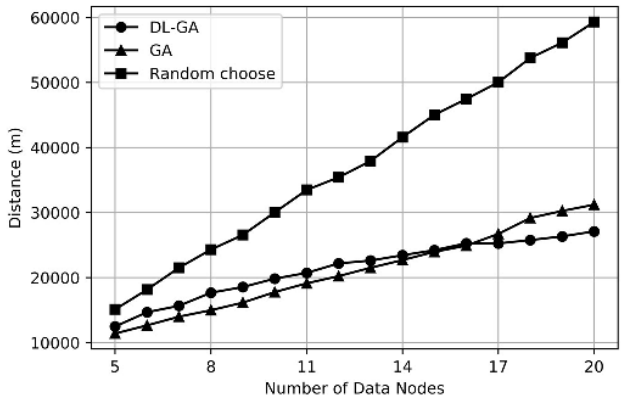


FIGURE 7. Average total distance of UAVs with DL-GA, GA and random choosing under the different conditions.

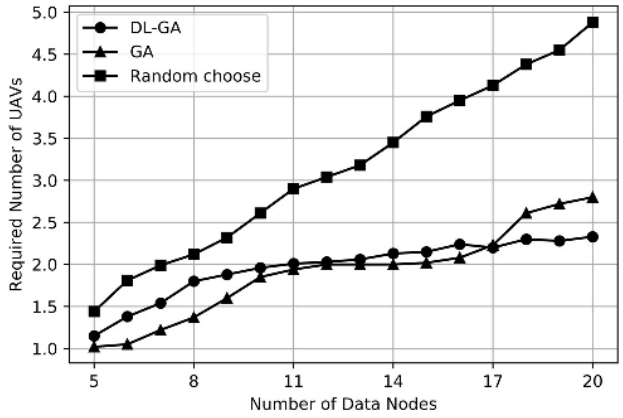


FIGURE 8. Average required number of UAVs with DL-GA, GA and random choosing under the different conditions.

FIGURE 7 and TABLE 2 display the average total distance with above three algorithms. We can see that the total distance of UAVs with DL-GA and GA are much shorter than random choosing. And the total distance of UAVs with GA is a little shorter than DL-GA while the number of data nodes is less than 17. However, while the number of data nodes is more than 17, the total distance of UAVs with DL-GA is shorter than GA.

FIGURE 8 and TABLE 3 present the average required number of UAVs of above three algorithms. It shows that the average required number of UAVs obtained by DL-GA and GA are obviously small than random choosing. And like FIGURE 7, when the number of data nodes is less than 17, the average required number of UAVs obtained by GA is a little less than DL-GA. Nevertheless, when the number of data nodes is more than 17, the average required number of UAVs obtained by DL-GA is less than GA.

From FIGURE 7 and FIGURE 8, we can see that when the number of data nodes is less than 17, no matter the average total distance or the average required number of UAVs, the performance of DL-GA is a little worse than GA. However, when it reached to 17, the performance of DL-GA is better than GA, which means the optimization capacity of DL-GA is better than GA. Here, we will give the explanation.

Explanation: To explain this phenomenon clearly, we give the following assumptions. We suppose that at a moment in the scenario, there are n data nodes, and the multi-UAV path which is obtained by GA can be expressed as $T_n = (t_1, t_2, .., t_n)$, where $n \leq m$. Supposing there are two states of scenario in the different moments, and the number of data nodes are n_1 and n_3. Then, we use GA to plan the path, in which they can be expressed as $T_{n_1} = (t_1, t_2, \ldots, t_{n_1})$ and $Q_{n_3} = (q_1, q_2, \ldots, q_{n_2}, \ldots, q_{n_3})$, where $n_1 < n_3$ and $T_{n_1} \in (q_{n_2}, q_{n_2+1}, \ldots, q_{n_3})$, $t_1, t_2, \ldots, t_{n_1} \cap (q_1, q_2, \ldots, q_{n_2-1}) = \emptyset$.

FIGURE 9 displays T_{n_1} and Q_{n_3}. Q_{n_3} is divided into two parts, $Q^1_{n_3}$ and $Q^2_{n_3}$, where $Q^1_{n_3}$ includes $n_2 - 1$ data nodes and $Q^2_{n_3}$ includes $n_3 - n_2 + 1$ data nodes. Supposing there are two objective global optimal solutions under the two conditions which can be presented as θ_{n_1} and θ_{n_3} (θ_{n_1} and θ_{n_3} are the distance of path). Since $n_3 > n_1$, the path under the condition where there are n_3 data nodes is more complicated to plan than the path under the condition where there are n_1 data nodes. Therefore, if GA is carried out to plan the two paths, then the difference between distance of T_{n_1} and θ_{n_1} is less than the difference between distance of Q_{n_3} and θ_{n_3}. Furthermore, the distance of T_{n_1} is less than the distance of $Q^2_{n_3}$. In other words, we can consider T_{n_1} is better than $Q^2_{n_3}$ (The above values mean average value).

While CNN learns the experiences of T_{n_1} and Q_{n_3}, CNN can only learn the experiences of path planning of $(q_1, \ldots, q_{n_2-1})$ from $Q^1_{n_3}$ because there are no such experiences in T_{n_1}. So, the distance of $(q_1, \ldots, q_{n_2-1})$ obtained by trained CNN is more than $Q^1_{n_3}$. However, when CNN learns the experiences of path planning of $(q_{n_2}, q_{n_2+1}, \ldots, q_{n_3})$, it can not only learn it from $Q^2_{n_3}$ but also T_{n_1}. And because the distance of T_{n_1} is less than $Q^2_{n_3}$, then the distance of $(q_{n_2}, q_{n_2+1}, \ldots, q_{n_3})$ obtained by trained CNN is more than the distance of T_{n_1} but less than the distance of $Q^2_{n_3}$. In general, supposing the number of data nodes is n_3, and we divide it into two parts as shown in FIGURE 9. Under this condition,

TABLE 2. Average total distance of UAVs with DL-GA, GA and random choosing under the different conditions.

Number of data nodes	5	8	11	14	17	20
DL-GA(m)	11980	17560	20771	23824	25275	27104
GA(m)	11112	14779	19526	22641	26969	31474
Random choose(m)	14786	23848	32956	41756	51043	58968

TABLE 3. Average required number of UAVs with DL-GA, GA and random choosing under the different conditions.

Number of data nodes	5	8	11	14	17	20
DL-GA	1.15	1.8	2.01	2.13	2.2	2.33
GA	1.02	1.37	1.94	2.00	2.23	2.8
Random choose	1.44	2.12	2.9	3.45	4.13	4.88

TABLE 4. Average solving time with DL-GA and GA under the different conditions.

Number of data nodes	5	8	11	14	17	20
DL-GA(s)	0.034	0.038	0.034	0.037	0.034	0.035
GA(s)	8.6	14.7	24.8	43.1	59.7	77.6

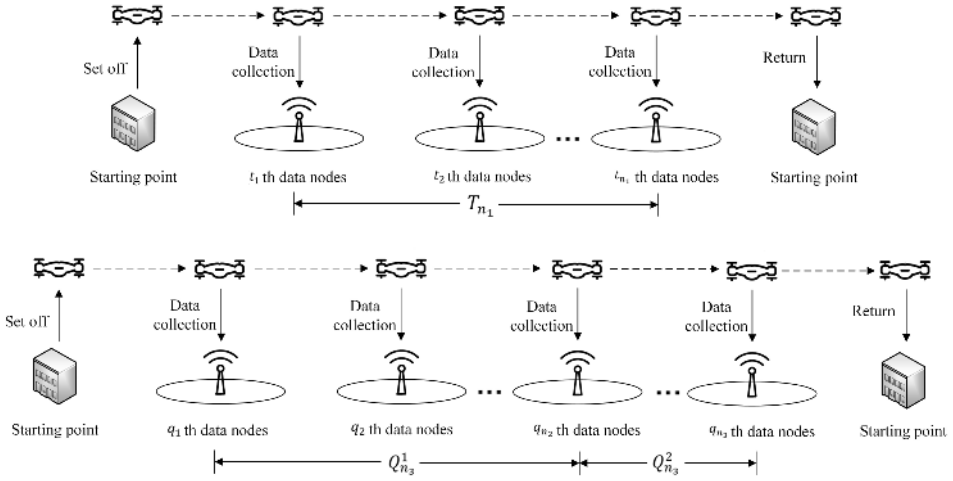


FIGURE 9. Path planning under the conditions of different data nodes.

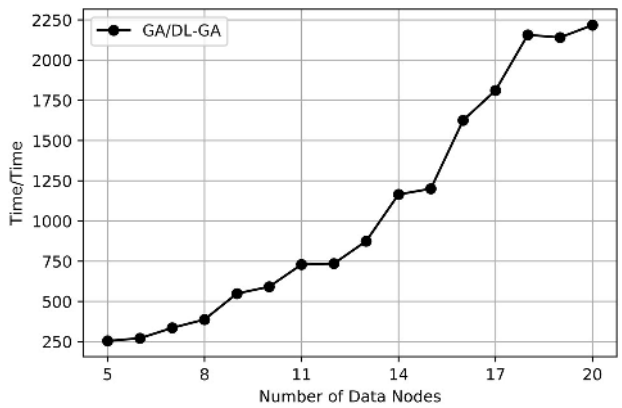


FIGURE 10. Average solving time of GA over DL-GA under the different conditions.

compared path planning of CNN with GA, the distance of $(q_{n_2}, q_{n_2+1}, \ldots, q_{n_3})$ obtained by CNN is less than GA and the distance of $(q_1, \ldots, q_{n_2-1})$ obtained by CNN is more than GA. Therefore, the total distance obtained by CNN may be less than GA while the number of data nodes is relatively large. And this trend will be more obvious with the increasement of the number of data nodes according to the simulation results.

The average solving time with DL-GA and GA are shown in FIGURE 10 and TABLE 4.

FIGURE 10 and TABLE 4 display the average solving time with DL-GA and GA under the different conditions. We can see that with the increasement of the number of data nodes, average solving time with GA shows a rising trend and average solving time with DL-GA has a little fluctuation (This is because the average solving time of DL-GA is very small so that the error caused by software has the effect on it.). Average solving time with DL-GA is about 300 to 2000 times less than GA, which proves the superiority of solving time with DL-GA.

Therefore, from the above comparison results, DL-GA can efficiently design the path of multi-UAVs and reduce the total distance and required number of UAVs. Compared with GA, the optimization capacity of DL-GA is better than GA under the specific conditions, and the solving time of DL-GA is much shorter than GA, which proves the superior performance of DL-GA.

V. CONCLUSION

Aim to overcome the poor performance of convergence speed of GA, we proposed a DL-GA for multi-UAV data collection in the challenging scenarios in this paper. In the proposed

Description of GA for Multi-UAV Path Planning

While True **do**
  for $i = 1, M$ **do**
    for $j = 1, n$ **do**
      See Algorithm 1 for specific process
  for $i = 1, M$ **do**
    Perform selection operation
  for $i = 1, \alpha$ **do**
    Perform crossover operation
  for $i = 1, M$ **do**
    Perform mutation operation
  if maximum fitness has no change for ε iterations **do**
    Break

strategy, GA obtains the states of the scenario and path sections, which are stored in an experience replay to train CNN. Once it is trained, CNN will rapidly give the path planning results draw on the experiences of GA. Numerical experiments are carried out to prove the superiority of DL-GA. First, when the number of data nodes is relatively small, the performance of total distance and required number of UAVs obtained by DL-GA is close to GA but its performance of solving time is much more outstanding than GA. Second, when the number of data nodes is large, the performance of total distance, required number of UAVs and solving time of DL-GA surpass GA. The above conclusions express that DL-GA can design the path for multi-UAV rapidly, which not only solves the problem caused by the waste of energy and required number of UAVs but also overcomes the shortcoming of slow convergence speed of GA.

DL-GA is suitable not only in this scenario but also in the scenarios which have the similar modeling method. Compared with GA, DL-GA will reduce the computation load and solving time to a large extent and almost without loss of optimization capacity. Furthermore, when the optimization problems are complicated, the optimization capacity of DL-GA may exceed GA. Obviously, DL-GA is suitable to solve those complicated optimization problems and problems with high requirement for timeliness.

APPENDIX

TIME COMPLEXITY COMPUTATION FOR GA AND DL-GA

We can summarize the GA for multi-UAV path planning as follow:

Description of DL-GA for Multi-UAV Path Planning

Get states from scenario
for $i = 1, n$ **do**
  Input states to CNN and get path section
  Get states from new scenario
  if energy of current UAV is not enough **do**
    Replace it with a new UAV
  if all data nodes are covered **do**
    Break

Supposing the number of data nodes is n. From the above description and combining with Algorithm 1, we can see that the time complexity of GA in the part of fitness computation is $O(\varepsilon Mn)$. And we summarize the DL-GA for multi-UAV path planning as follow:

From the above description, we can obtain the time complexity DL-GA for multi-UAV path planning are $O(n)$.

ACKNOWLEDGMENT

The authors thank all the reviewers and editors who have contributed to the quality of this article.

REFERENCES

[1] Y. Wang, Z. Hu, X. Wen, Z. Lu, and J. Miao, "Minimizing data collection time with collaborative UAVs in wireless sensor networks," *IEEE Access*, vol. 8, pp. 98659–98669, 2020.
[2] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet Things J.*, early access, Jul. 29, 2020, doi: 10.1109/JIOT.2020.3012835.
[3] V. K. Chawra and G. P. Gupta, "Multiple UAV path-planning for data collection in cluster-based wireless sensor network," in *Proc. 1st Int. Conf. Power, Control Comput. Technol. (ICPC2T)*, Raipur, India, Jan. 2020, pp. 194–198.
[4] Z. Song, H. Zhang, F. Liu, S. Chen, and F. Zhang, "Unmanned aerial vehicle coverage path planning algorithm based on cellular automata," in *Proc. Int. Conf. Inf. Syst. Comput. Aided Edu. (ICISCAE)*, Changchun, China, 2018, pp. 371–374.
[5] X. Chen and J. Zhang, "The three-dimension path planning of UAV based on improved artificial potential field in dynamic environment," in *Proc. 5th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Hangzhou, China, Aug. 2013, pp. 144–147.
[6] L. Wang and Y. Li, "A multi-objective optimization method based on dimensionality reduction mapping for path planning of a HALE UAV," in *Proc. Chin. Autom. Congr. (CAC)*, Hangzhou, China, Nov. 2019, pp. 3189–3194.
[7] J. Chen, F. Ye, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *Proc. Prog. Electromagn. Res. Symp. Fall*, Singapore, Nov. 2017, pp. 832–837.
[8] H. Daryanavard and A. Harifi, "UAV path planning for data gathering of IoT nodes: Ant colony or simulated annealing optimization," in *Proc. 3rd Int. Conf. Internet Things Appl. (IoT)*, Isfahan, Iran, Apr. 2019, pp. 1–4.
[9] Z. Cheng and D. Li, "Improved GASA algorithm for mutation strategy UAV path planning," in *Proc. 10th Int. Conf. Commun. Softw. Netw. (ICCSN)*, Chengdu, China, Jul. 2018, pp. 506–510.
[10] D. Zhang, Y. Xian, J. Li, G. Lei, and Y. Chang, "UAV path planning based on chaos ant colony algorithm," in *Proc. Int. Conf. Comput. Sci. Mech. Autom. (CSMA)*, Hangzhou, China, Oct. 2015, pp. 81–85.
[11] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.
[12] Z. Daoqing and J. Mingyan, "Parallel discrete lion swarm optimization algorithm for solving traveling salesman problem," *J. Syst. Eng. Electron.*, vol. 31, no. 4, pp. 751–760, Aug. 2020.
[13] X. Chen, P. Zhang, G. Du, and F. Li, "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems," *IEEE Access*, vol. 6, pp. 21745–21757, 2018.
[14] F. Yu, X. Fu, H. Li, and G. Dong, "Improved roulette wheel selection-based genetic algorithm for TSP," in *Proc. Int. Conf. Netw. Inf. Syst. for Comput. (ICNISC)*, Wuhan, China, Apr. 2016, pp. 151–154.
[15] S. Goyal and R. Gupta, "Optimization of fidelity with adaptive genetic watermarking algorithm using roulette-wheel," in *Proc. Int. Conf. Comput. Intell. Commun. Netw.*, Bhopal, Madhya Pradesh, Nov. 2010, pp. 591–596.
[16] W. Cheng, H. Shi, X. Yin, and D. Li, "An elitism strategy based genetic algorithm for streaming pattern discovery in wireless sensor networks," *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 419–421, Apr. 2011.
[17] A. Bhateja and S. Kumar, "Genetic algorithm with elitism for cryptanalysis of vigenere cipher," in *Proc. Int. Conf. Issues Challenges Intell. Comput. Techn. (ICICT)*, Ghaziabad, Uttar Pradesh, Feb. 2014, pp. 373–377.

[18] B. Zhao and Z. Xiong, "Research and application of genetic algorithm based on variable crossover probability," in *Proc. Int. Conf. Virtual Reality Visualizat. (ICVRV)*, Zhengzhou, China, Oct. 2017, pp. 156–159.
[19] N.-E. Croitoru, "High probability mutation and error thresholds in genetic algorithms," in *Proc. 17th Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Timisoara, Romania, Sep. 2015, pp. 271–276.
[20] W. Liu, C. Qin, K. Gao, H. Li, Z. Qin, Y. Cao, and W. Si, "Research on medical data feature extraction and intelligent recognition technology based on convolutional neural network," *IEEE Access*, vol. 7, pp. 150157–150167, 2019.
[21] D. Song, Z. Zhen, B. Wang, X. Li, L. Gao, N. Wang, T. Xie, and T. Zhang, "A novel marine oil spillage identification scheme based on convolution neural network feature extraction from fully polarimetric SAR imagery," *IEEE Access*, vol. 8, pp. 59801–59820, 2020.
[22] A. Yang, X. Yang, W. Wu, H. Liu, and Y. Zhuansun, "Research on feature extraction of tumor image based on convolutional neural network," *IEEE Access*, vol. 7, pp. 24204–24213, 2019.
[23] J. Gan, K. Jiang, H. Tan, and G. He, "Facial beauty prediction based on lighted deep convolution neural network with feature extraction strengthened," *Chin. J. Electron.*, vol. 29, no. 2, pp. 312–321, Mar. 2020.
[24] Y. Pang, M. Sun, X. Jiang, and X. Li, "Convolution in convolution for network in network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1587–1597, May 2018.
[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.
[26] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281–4298, Jun. 2019.
[27] Y. Cai, Z. Wei, R. Li, D. W. K. Ng, and J. Yuan, "Joint trajectory and resource allocation design for energy-efficient secure UAV communication systems," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4536–4553, Jul. 2020.
[28] J. Gong, T.-H. Chang, C. Shen, and X. Chen, "Flight time minimization of UAV for data collection over wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1942–1954, Sep. 2018.
[29] S. H. Rubin, T. Bouabana-Tebibel, Y. Hoadjli, and Z. Ghalem, "Reusing the NP-hard traveling-salesman problem to demonstrate that P NP (Invited Paper)," in *Proc. IEEE 17th Int. Conf. Inf. Reuse Integr. (IRI)*, Pittsburgh, PA USA. Jul. 2016, pp. 574–581.
[30] B. Kong, H. Huang, and X. Jia, "Path planning for sensor data collection by using UAVs," in *Proc. 14th Int. Conf. Mobile Ad-Hoc Sensor Netw. (MSN)*, Shenyang, China, Dec. 2018, pp. 199–205.

YUWEN PAN is currently pursuing the master's degree with the School of Information and Communication Engineering. He is also with the University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan, China. His current research interests include deep learning and optimization algorithm.

YUANWANG YANG (Member, IEEE) received the Ph.D. degree in signal and information system from the University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan, China, in 2011. He had twice visiting scholarship experience in USA, once with the University of Illinois, Urbana & Champaign, in 2010, once with the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2016. He is currently with UESTC. His main research interests include intelligent signal identification, wireless RF communication system design, and high-performance frequency synthesis.

WENZAO LI (Member, IEEE) received the Ph.D. degree in communications and information system from SCU, in 2016. He had visiting scholarship experience with Simon Fraser University (SFU), Vancouver, BC, Canada. He is currently an Associate Professor with the Chengdu University of Information Technology (CUIT), Chengdu, Sichuan, China. His current research interests include wireless communication networks and mobile edge computing.