A Study of Wireless Sensing Multi-Mobile Sink Path Planning Based on Improved AOA

Jia DING, Jian HE, Meijuan LI and Xuebo CHEN*

***Abstract*-Wireless Sensor Networks are essential for real-time data collection in disaster response, military surveillance, and environmental monitoring. Each sensor detects environmental parameters and sends data to a base station (BS) or sink. Using multiple mobile sinks (MSs) based on clustering algorithms is more efficient than static sink-based methods, especially in scenarios where the number of sensor nodes (SNs) is large. This scheme can improve path planning. This paper presents a Multi-mobile Sink Path planning algorithm with Dual Clustering (MSPDC) to improve rescue efficiency in earthquake scenarios. The algorithm combines the K-means algorithm, improved Density Peak Clustering (DPC), and an improved Arithmetic Optimization Algorithm (AOA). First, we use the K-means algorithm to pinpoint the cluster centers, which serve as rendezvous points (RPs), and adjust them for better coverage. Secondly, we apply an improved DPC algorithm to cluster these RPs, further increasing coverage. Finally, we use an improved AOA for 3D path planning. By calculating the average Euclidean distance between each cluster center and its RPs, we pick the center with the smallest distance as the best starting spot for our robots. As the algorithm searches for paths, it cleverly adapts its step size based on the density of obstacles in the environment, dynamically adjusting arithmetic operators and setting a safe distance to avoid obstacles. Algorithm simulation analysis shows that the improved AOA, compared with the traditional AOA, the Bat Algorithm, the Improved Grey Wolf Optimization Algorithm, the Genetic Algorithm, and the Whale Optimization Algorithm, can search for the shortest path more quickly. It significantly shortens the total path traversal time, reduces the number of turns, and improves the success rate of obstacle avoidance in complex environments. The MSPDC proposed in this paper demonstrates stronger applicability and reliability in complex scenarios such as earthquake rescue.**

***Index Terms*-Wireless sensor networks, multi-mobile sink, path planning, arithmetic optimization algorithm**

Manuscript received April 18, 2025; revised July 9, 2025.

This work was supported by the National Natural Science Foundation of China, under grant numbers 71571091 and 71771112.

Jia Ding is a postgraduate student of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China (e-mail: dingjia_0107@163.com).

Jian He is a professor of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China (e-mail: lnashj@126.com).

Meijuan Li is a professor of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China (e-mail: 454675273@qq.com).

Xuebo Chen is a professor of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China (corresponding author, phone: +8613614128208; e-mail: xuebochen@126.com).

I. Introduction

The Internet of Things (IoT) exemplifies a sophisticated and comprehensive application of contemporary information technology. Its significance in fostering industrialization, advocating for ecological sustainability, and facilitating intelligent networking is paramount in shaping a sustainable economy and modern society. Wireless Sensor Networks (WSNs), as a critical subset of IoT technology, provide an essential infrastructure for the exponential growth and advancement of IoT [1]. WSNs consist of intricate networks of distributed, autonomous devices designed to sense and monitor diverse environmental parameters cooperatively. These networks find extensive applications in areas such as habitat surveillance, prediction and detection of natural disasters, and medical monitoring, among others [2]. The primary components of WSNs encompass sensing units designed for environmental perception, processing units tasked with the manipulation of raw data, transceiver units responsible for data transmission, and power units that ensure a consistent power supply [3]. In WSNs, sensor nodes (SNs) operate on battery power and utilize wireless communication technology. Their compact size allows for easy deployment in various locations with minimal environmental impact. This versatility significantly reduces deployment and maintenance costs, positioning WSNs as a more cost-effective and efficient alternative to wired sensor networks for sensor data collection [4]. Numerous sensor nodes join together to form WSNs, which are well-suited for gathering data from carefully managed environments. These nodes communicate through wireless connections, sending all the data they pick up to special nodes called sink nodes [5]. The sink node is crucial in data collection, gathering data from sensor nodes, and sending it to a central server or data center. With a continuous power supply, large storage capacity, and strong communication capabilities, it can work statically or dynamically to ensure efficient data transmission and management.

In WSNs, data transfer methodologies predominantly encompass single-hop and multi-hop transmissions. Single-hop transmission involves direct communication between the sensor nodes and the base station (BS), whereas multi-hop transmission relies on intermediary relay nodes for data relay. However, a major challenge is the rapid energy depletion of nodes near the BS, causing uneven network energy distribution. This imbalance creates "energy holes"—areas with severely low energy levels—that harm overall network efficiency and lifespan [6]. The main task of SNs is to collect relevant data from the environment and send the collected data to the BS or sink through multi-hop communication, a method that helps extend the life of the network. Static sink (SS) and mobile sink (MS) are two common data acquisition methods. In scenarios where a single SS is employed, it collects data from all SNs and sends it to the BS. However, this can create an energy hole, mainly because nodes near the SS consume excessive energy. Additionally, there is a significant energy consumption gap: nodes closer to the SS use more energy than those farther away. This discrepancy can readily lead to the formation of energy holes. Furthermore, for the purpose of load balancing, the SS is typically positioned at the centroid of the WSNs. Moreover, if there is an excessive concentration of nodes in the vicinity of the SS, there is a risk of these nodes depleting their energy reserves and failing, thereby disconnecting the SS from the remaining WSNs [7]. To address this challenge, employing multiple SSs for data acquisition is a strategic approach, as it mitigates the energy hole issue and enhances transmission efficiency. However, the fixed nature and security vulnerabilities of SS positions pose challenges in determining optimal locations. To circumvent these limitations, the use of an MS for data collection is advisable. The mobility of the MS allows for extensive traversal within the network, facilitating comprehensive data acquisition from various SNs. In scenarios where SNs establish contact with the MS, they promptly transmit their sensor data to the MS. When direct communication is not feasible, SNs retain the data until they can reconnect with the MS. Alternatively, they relay the data to the MS via intermediate SNs through multi-hop communication. This method facilitates load balancing and equitable energy distribution across the network, thereby optimizing the network's lifespan [8]. The constantly shifting location of the MS makes it difficult for attackers to pinpoint it and enhances security. However, if only one MS is in use and there are many SNs, the MS must visit each SN to gather data, causing delays. The communication between SNs and MS can also cap the network's throughput and lifespan. To boost WSNs' performance, one tactic is to introduce multiple mobile sinks (MSs) [9]. Having several MSs enhances the WSNs' ability to manage faults, balance the load, and ensure data reliability. It also expands the network's reach, making it adaptable to different applications.

The extensive utilization of WSNs in intelligent and environmental monitoring has elevated the significance of MS path planning. With the application of mobile robots, researchers and scholars have extensively studied path planning technology [10]. The goal of 3D path planning is to find a path that avoids obstacles, minimizes path length or time, and considers physical limitations and motion constraints in 3D space [11]. In the context of mobile robotics, path planning involves employing sophisticated algorithms to determine the shortest or quickest routes for robots, drones, or autonomous vehicles while ensuring collision-free navigation. The strategic selection of path planning algorithms is crucial for guaranteeing secure and efficient point-to-point travel [12]. In the realm of autonomous vehicle navigation, path planning algorithms play a pivotal role in facilitating obstacle avoidance and coordination with other mobile robots [13]. The field of path planning is divided into two main categories: global and local strategies. Global strategies are limited due to terrain uncertainties and robustness issues. In contrast, local strategies demonstrate greater adaptability in partially known or unknown scenarios, optimizing route efficiency [14]. Path planning technology is an essential driver of efficiency and safety in diverse applications, including warehouse automation, rescue robotics, and military reconnaissance. It represents a cornerstone of 21st-century technological advancement [15]. For effective data collection and energy conservation, designing path planning algorithms is crucial. Current approaches include overall path planning, ideal for small-scale but complex tasks, and cluster-based path planning, better for large-scale environments. These approaches enhance computational speed and system scalability.

Domestic and international studies have evolved from focusing solely on individual path planning to addressing the collaborative optimization of multiple MSs. This approach efficiently orchestrates the routes of MSs, thereby mitigating issues of uneven load distribution. In large-scale networks, a single MS can consume a significant amount of energy. To address this, a clustering strategy is implemented, alongside multiple MSs, to significantly boost data collection efficiency. The network's nodes are divided into various clusters, each with its own MS. Generally, SNs forward their collected data to specific rendezvous points (RPs), and each MS needs only to visit the RPs within its cluster. This approach reduces the number of positions an MS must visit, shortens its travel path, boosts information collection efficiency, and cuts energy consumption. Additionally, this method bypasses the need to send data to the BS, thereby reducing in-network communication distances, alleviating congestion, and further reducing WSN energy consumption [16]. In practical applications, due to environmental obstacles, considerations such as sensor information and perception capabilities, obstacle representation and map updating, dynamic path planning and re-planning strategies, and security issues must be carefully addressed.

Earthquakes, hurricanes, volcanic eruptions, and tsunamis are common but highly destructive natural disasters. The first 72 hours after a disaster are often called the "golden rescue time". How to implement rescue tasks more efficiently after an earthquake is a problem that we need to study [17]. Because BS is often damaged after a disaster, the traditional rescue means are ineffective. The WSN technology can monitor environmental changes in real time, provide key data for rescue, help rapid decision-making, and improve efficiency and success rate. By pre-deploying SNs, after an earthquake, WSNs collect information about the location and ruins of survivors in the collapsed building from the installed sensor equipment and transmit it to the emergency center, where the rescue robot can obtain information in advance. If there are SNs within a few meters of the survivors, robots can detect the location of the survivors in the rubble [18].

In disaster-stricken areas, where multiple SNs are pre-positioned, clustering technology is being used, along with several MSs, to gather vital information from these SNs, thereby enhancing the rescue efforts for those affected by earthquakes. Data collection efficiency can be improved through rational path planning for each MS. This paper proposes a Multi-mobile Sink Path planning algorithm with Dual Clustering (MSPDC), which combines the K-means algorithm, Density Peak Clustering (DPC) algorithm, and Arithmetic Optimization Algorithm (AOA). First, the K-means and DPC algorithms are combined and optimized to improve node coverage in clusters. Secondly, in the path planning phase, we use an improved AOA to determine the optimal initial deployment positions for each robot, laying the foundation for efficient path planning. Finally, in simulating a complex environment for an earthquake rescue scenario, random obstacles are set during the path planning process. This is combined with obstacle detection technology and optimized obstacle avoidance strategies. It enables robots to quickly perceive environmental changes and flexibly plan the optimal obstacle avoidance path, greatly enhancing rescue efficiency. By leveraging the dual benefits of optimized initial deployment points and intelligent obstacle avoidance strategies, the time required for finding the optimal path and traversal can be significantly reduced. Additionally, it effectively shortens the total path length and minimizes time loss due to frequent turning. At the same time, it can significantly improve the robots' obstacle avoidance success rate in complex and variable environments. This thereby reduces the overall energy loss of WSNs and practically enhances the adaptability and reliability of the system in actual earthquake rescue scenarios.

II. System Model

A. Network Model

In our analysis, we adapt a system model that represents a single WSN, which includes n SNs and m MSs. SNs are scattered randomly across a 3D space, each occupying a distinct location and gathering unique data. We define a set N as the set $N=\{n_i | 1\le i\le n\}$ of SNs, where n_i symbolizes the i-th node. We define a set M as the set $M=\{m_i | 1\le i\le m\}$ of MSs, where m_i symbolizes the i-th MS. In sensor data collection, ensuring complete area coverage is key. Each MS follows a set path to gather data from SNs in its range. This ensures that every spot in the sensing area is monitored by at least s SNs $(s\ge 1 \text{ and } s\in N)$.

When dealing with numerous SNs, identifying RPs becomes essential. These RPs gather data from nearby SNs. Assuming that K RPs are generated within the network and these K RPs constitute the RPs set $Q=\{Q_k | 1\le k\le K\}$, where Q_k represents the k-th RP. We then group these RPs into m clusters, with each cluster having a dedicated MS. The path of each MS is planned so that it can collect data from the RPs in its area, under the assumption that each MS has sufficient energy to traverse its route and complete data collection.

By applying the K-means algorithm, we create eight RPs among the effective nodes—that is, K=8. We use the improved DPC algorithm to divide these RPs into two clusters, m=2. Create two irregularly shaped obstacles in a 3D space. For each obstacle, we randomly generate eight vertices and six faces. We also specify the size of both obstacles. Within each cluster, we use an improved AOA to plot the paths for the MSs, ensuring they navigate around these obstacles efficiently. Fig. 1 shows a schematic of the network model in a 3D environment.

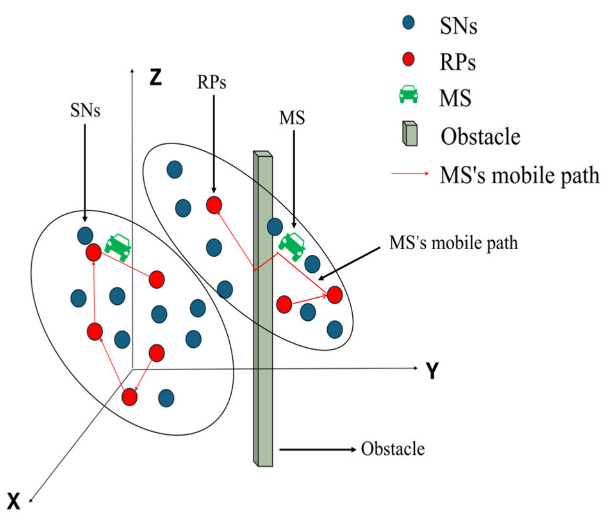


Fig. 1. 3D environmental network model

B. Path Length Model

During path planning, SNs transmit their data packets to the nearest RPs through multi-hop transmission. The MS traverses each RP in each cluster, and the starting and ending points of the MS path are called the starting point and the target point, respectively. We define "shortest distance" as minimizing the path length between the starting point and the target point. In any iteration, if the distance between the point $wp_j(t)$ and the destination point $wp(N)$ is the shortest, then this point is selected as the optimal point. The "shortest distance" between the two points can be represented as [19]:

$$F_1(x,y,z)=d(wp_j(t),wp(N)) \tag{1}$$

Where d_t is the Euclidean distance in 3D space:

$$d_t=\sqrt{\begin{array}{l}(x_{wp_j}(t+1)\text{-}x_{wp_j}(t))^2+(y_{wp_j}(t+1)\text{-}y_{wp_j}(t))^2\\+(z_{wp_j}(t+1)\text{-}z_{wp_j}(t))^2\end{array}} \tag{2}$$

The Shortest Path Length (SPL) is calculated by adding up the distances between the starting point $wp(1)$ and the destination point $wp(N)$ through all intermediate points $(wp_j(2)...wp_j(N-1))$ generated by the path planning algorithm, that is:

$$SPL=\sum_{t=1}^{N-1}d(wp_j(t),\ wp_j(t+1))=\sum_{t=1}^{N-1}d_t \tag{3}$$

Where x, y, and z are the coordinate values of the 3D point, t is the number of iterations, j is the index for the best solution found by our group optimization path planning algorithm, and N is the number of points visited along the MS's path.

C. Time Model

The total time t for MS path planning includes three parts:

(1) The time taken to find the optimal path using the improved AOA is t_1.

(2) The time taken for the MS to traverse all RPs within its cluster is t_2.

(3) The MS stops at each RP for one minute. Assuming the number of RPs traversed within each cluster is $p\,(p\in Q)$, the total stopping time is t_3.

(4) Every time an MS makes a turn, it takes 0.5 minutes.

Assuming the number of turns an MS makes while searching for the optimal path is q, then the total time spent on turning is t_4.

$$t_3 = 1 \times p,\ t_4 = 0.5 \times q,\ t = t_1 + t_2 + t_3 \tag{4}$$

Assuming MS moves at a constant speed v, the shortest path length is:

$$SPL = t_2 \times v \tag{5}$$

Where t_2 is the time required for each MS to traverse all paths from the starting point to the end point within each cluster, an MS starts moving from a node $wp_j(t)$ and reaches its destination through time t_2 [20], $v \approx 0.83 m/s$.

III. ALGORITHM

In this article, we introduce a cluster-based approach to path planning named MSPDC. It consists of three core algorithms:

(1) Look for the optimal location of RPs. In this stage, we employ the K-means algorithm to determine the number of RPs, excluding any invalid ones, in the 3D scene and pinpoint their best locations. This algorithm is simple to understand and implement, with low computational complexity and great scalability.

(2) During the clustering phase, we utilize the improved DPC algorithm to group all RPs, maximizing node coverage. It is adept at identifying clusters of any form and dealing with data that is spread out differently.

(3) During the path planning phase, an MS is allocated within each cluster. The improved AOA is employed to determine the optimal initial positions of each MS, facilitating quick identification of the shortest path. Additionally, we introduce random 3D obstacles to simulate realistic earthquake rescue environments, enabling robots to navigate around obstacles effectively. This approach significantly enhances rescue efficiency and possesses considerable practical value.

A. K-means Algorithm for Finding RPs

The K-means algorithm [21], [22] is a deterministic global optimization method that does not depend on any initial parameter values and requires the number of clusters to be determined in advance. The K-means algorithm clustering problem aims to divide the RPs into C mutually exclusive clusters (C_1, ..., C_M), with the center of each cluster being the RPs. The algorithm consists of two phases:

(1) Randomly select r cluster centers.

(2) Place each node with its closest cluster center, with r set in advance. Measure the distance between the cluster center and the node using the Euclidean distance method. Define the vector coordinates of the current cluster center as $A = (x_a, y_a, z_a)$, and the vector coordinates of the current node as $B = (x_b, y_b, z_b)$. The Euclidean distance in 3D space can be calculated as:

$$d_{AB} = [\sum_{a=1,b=1}^{n} (x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2]^{1/2} \tag{6}$$

The number of RPs is r, define a set $R = \{R_i | 1 \le i \le r\}$ as the set of RPs, where R_i represents the i-th RP. Input the specified number of cluster centers r and the database N consistin g of (1-6%) n SNs, where $N = \{n_i | 1 \le i \le n\}$, output r cluster centers. In the end, use the K-means algorithm to generate eight RPs, through multiple iterations, the optimal positions of the eight RPs can be found.

B. DPC Algorithm for Clustering RPs

Using the located RPs, we employ the DPC algorithm [23], [24] for clustering. This approach focuses on the local density of SNs and their relative distances, without predefining the number of clusters. The DPC algorithm is versatile, capable of identifying clusters of any shape and handling density-varying data. The core idea is that a cluster center should have a higher density than its neighboring points and be relatively far from other points with similar or higher density. Such points are typically regarded as ideal cluster centers. The process involves calculating the density of all RPs, pinpointing the centers of dense areas, and ensuring these centers have a higher density than the surrounding RPs. The algorithm works as follows:

1) Calculate the local density ρ_i for each RP

The local density can be estimated using a simple statistical method. Generally, the ρ_i of a data point reflects the number of points around it. There are two methods to calculate the ρ_i for each point i. One method involves using a Truncated kernel, focusing solely on the count of data points within the cluster's vicinity, which is defined as:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{7}$$

Where $\chi(d_{ij} - d_c) = 1$. d_{ij} is the distance from point i, which is the cluster center, to the nearby point j, and d_c is the threshold value set by the user. An alternative approach involves calculating the Gaussian kernel, in which the density of each RP signifies its "crowding degree" within the spatial context. The formula is as follows:

$$\rho_i = \sum_j \exp[-(\frac{d_{ij}}{d_c})^2] \tag{8}$$

2) Calculate the relative distance Δ_i of each RPs

For each RPs, calculate the distance Δ_i to the point with higher density, which is:

$$\Delta_i = \min_{j:\rho_j > \rho_i} d_{ij} \tag{9}$$

Here, ρ_j represents the density of data point j, and d_{ij} is the distance between point i and point j. The relative distance indicates how far data point i is from the closest data point that has a higher density.

3) Select cluster centers

Define a "clustering goodness" indicator as:

$$\delta_i = \rho_i \cdot \Delta_i \tag{10}$$

Based on the ρ_i and Δ_i, the δ_i of each RP can be calculated. If a point's ρ_i is higher than the density of all its neighboring points and the distance to points with higher density Δ_i is also far, then this point may be a cluster center. Choose RPs with higher δ_i values as cluster centers. Typically, these RPs are in high-density areas and are distant from other such areas.

4) Assign RPs to the nearest cluster center

After determining the clustering centers, the remaining RPs are assigned to the nearest clustering center, which means assigning the remaining RPs to the nearest clustering center with a density greater than themselves.

5) Output the clustering results

Through the above steps, the RPs can eventually be divided into multiple clusters, each cluster containing a set of RPs close to the cluster center. The traditional DPC algorithm uses a Truncated kernel to calculate density, which is easily affected by data distribution and parameters. After improvement, it adopts a Gaussian kernel, assigning different weights according to the distance between data points, which can finely present the density distribution. The output results are continuous, allowing for a more precise determination of density peak points. By leveraging the smoothing characteristics of the Gaussian function, there is no need to precisely set the neighborhood radius, mitigating the impact of parameter settings. Additionally, optimizing the bandwidth parameter enhances the adaptability to data distribution. When detecting invalid nodes overlapping with cluster centers, using the traditional algorithm for clustering and randomly removing invalid nodes may lead to the destruction of the cluster structure. Through iterative optimization, center point allocation can be timely adjusted, enhancing robustness.

C. Improved AOA path planning

AOA [25]-[27] is a metaheuristic optimization algorithm that achieves global optimization based on the distribution characteristics of arithmetic operators. Drawing inspiration from how arithmetic operators are used in math problems, AOA employs basic arithmetic operations like addition, subtraction, multiplication, and uses them to identify the optimal solution from a set of candidate solutions. AOA is divided into two phases: exploration and exploitation. In the exploration phase, the algorithm employs multiplication and division to conduct a comprehensive global search. This approach not only broadens the distribution of solutions but also significantly boosts the algorithm's capability to uncover a multitude of promising solutions within an extensive search landscape. In the exploitation phase, the algorithm utilizes addition and subtraction strategies to narrow down the focus. This helps consolidate the solutions within a specific area, allowing for a more detailed search and enhancing the speed at which the optimal solution is identified and reached within a set of promising solutions.

In AOA, the optimization process starts with a set of randomly generated candidate solutions, as shown in matrix (11), where the best candidate solution in each iteration is the optimal or near-optimal solution obtained so far.

$$x=\begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & x_{N-1,j} & \cdots & x_{N-1,n} \\ x_{N,1} & \cdots & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{bmatrix} \tag{11}$$

First, AOA uses coefficients calculated by equation (12) to select the search phase through the Math Optimizer Accelerated (MOA), with the formula as follows:

$$\mathrm{MOA}(t)=Min+t\times(\frac{Max-Min}{T}) \tag{12}$$

Another coefficient defined by equation (13) is the Math Optimizer Probability (MOP), which is used to control the range of candidate solutions during the exploration or exploitation phase, namely:

$$\mathrm{MOP}(t)=1-(\frac{t}{T})^{\frac{1}{\alpha}} \tag{13}$$

In which MOP (t) and MOA (t) are the values at the t-th iteration, t and T represent the current iteration and the maximum number of iterations, respectively. *Max* and *Min* are the highest and lowest values of MOA, and α is a key parameter that defines the local exploitation precision in the iteration process. Define r_1 as a random number between 0 and 1. When r_1 > MOA, AOA performs global exploration; when r_1 < MOA, AOA enters the local development phase.

During the exploration phase, the division operator and the multiplication operator have a high degree of discreteness, which may lead to divergence and make it difficult to approach the target. However, after several iterations, communication between operators is increased to support the search during the exploration phase, adopting the simplest rules to simulate the behavior of arithmetic operators. The position update during the exploration phase is defined as:

$$x_{i,j}(t+1)=\begin{cases} best(x_j)\div(MOP+\varepsilon)\times\left[(UB_j-LB_j)\right. \\ \quad \left.\times\mu+LB_j\right], \quad r_2<0.5, \\ best(x_j)\times MOP\times\left[(UB_j-LB_j)\right. \\ \quad \left.\times\mu+LB_j\right], \quad otherwise \end{cases} \tag{14}$$

Among them, $r_2 \in [0,1]$, $x_{i,j}(t)$ represents the j-th position of the i-th solution in the current iteration. Best (x_j) is the best solution obtained so far at the j-th position, and ε is a very small value. UB$_j$ and LB$_j$ represent the upper and lower bounds at the j-th position, respectively, and μ is the control parameter to adjust the search process, with a value of 0.499. When r_2 < 0.5, the division search strategy is executed, and when r_2 > 0.5, the multiplication search strategy is executed.

During the development phase, the subtraction operator or addition operator has lower discreteness, which helps them approach the optimal solution. The position update formula is as follows:

$$x_{i,j}(t+1)\begin{cases} best(x_j)-MOP\times\left[(UB_j-LB_j)\times\mu+LB_j\right], \\ \quad r_3<0.5, \\ best(x_j)+MOP\times\left[(UB_j-LB_j)\times\mu+LB_j\right], \\ \quad otherwise \end{cases} \tag{15}$$

Among which, r_3 is a random number between 0 and 1.

In AOA, getting the best starting point is crucial for balancing exploration and exploitation. Picking a good initial point can speed up the algorithm's convergence and improve the quality of the final solution. The paper introduces a function to find the optimal starting point for robot path planning. It does this by calculating the average distance between each potential center point and all the data points, and then choosing the center point with the shortest average distance. First, initialize the average distance array by creating a zero vector of length *a*, where *a* is the number of RPs in each cluster. For each centroid *i*, calculate the norm with other node vectors—that is, calculate the Euclidean distance in three dimensions for each centroid. After calculating the distance between the current center point and all data points, find the average of these distances. Store the average distance of each center point to all RPs in the cluster in the average distance array. This array represents the average distance between each center point and all RPs in the cluster. Finally, locate the minimum value in the average distance array and select the corresponding center point as the initial point, thereby identifying the optimal starting point.

In real-world rescue missions, unexpected obstacles are common, and predicting their locations is impossible. This study simulates this scenario by placing two irregularly shaped obstacles and creating a special function to detect collisions with these obstacles. Each obstacle is defined by its eight corners, which are like the points on a polygon. To see if our robot is in danger of hitting these obstacles, we use something called an Axis-Aligned Bounding Box (AABB) to check if the robot's current spot is inside the obstacle's space. First, compute the minimum and maximum coordinates of the bounding box to find the six edges of the AABB in 3D space: the top and bottom of the *x-axis*, the left and right of the *y-axis*, and the front and back of the *z-axis*. Then, check if the point's coordinates are inside this box. If all the point's coordinates fit within the box's limits, it means the point is inside the obstacle's space. If a collision is detected, the obstacle avoidance function is activated. The goal of the obstacle avoidance function is to tweak the path points so that the robot does not crash into any obstacles. The strategy is to find the direction that is furthest from where we are trying to go and avoid obstacles in that way. If we get too close to an obstacle, we figure out a new spot to steer the robot, making sure it avoids the obstacle while still heading toward the target. We start by calculating the direction vector from our current location to our destination. Next, we use the cross product to find a direction that is perpendicular to our target direction, which we use to dodge the obstacle. Usually, we do this by swerving around the *z-axis*, but if that does not work (if the cross-product is zero, it means we are parallel to the *z-axis*), we switch to swerving around the *y-axis* instead.

The core of AOA in obstacle avoidance lies in its flexible use of basic arithmetic operations to achieve environmental adaptation, multi-objective coordination, and efficient computation. The arithmetic operations for two-dimensional path planning cannot be straightforwardly applied to 3D space. In a 3D environment, AOA extends the search space through multi-dimensional arithmetic operations, with path points represented as $P=(x,y,z)$. Candidate paths are generated through addition and subtraction:

$$P_{new}=P_{current}\pm\Delta\cdot(v_x,v_y,v_z) \tag{16}$$

where Δ represents the dynamic step size and $v=(v_x,v_y,v_z)$ represents the direction vector. We utilize the vector dot product to assess the alignment between the path direction and the target direction and the cross product to avoid the normal vectors of obstacle surfaces, thereby enhancing path safety.

AOA dynamically modulates the step size in response to obstacle density, thereby facilitating adaptive environmental responsiveness through the strategic manipulation of arithmetic operators. In regions with high obstacle concentration, division operations are implemented to decrement the step size, thereby enhancing path safety. Conversely, in less obstructed areas, multiplication operations are utilized to expedite global search processes, thereby augmenting operational efficiency. By dynamically selecting arithmetic operators and density-related parameters, flexible scaling of the step size is realized. Define the obstacle density d as the proportion of the local area around the current path point occupied by obstacles:

$$d=\frac{N_{obs}}{N_{total}} \tag{17}$$

Where N_{obs} is the area occupied by the obstacle in the localized area and N_{total} is the total area of the localized area.

The step size s is adaptively adjusted according to the obstacle density *d*, combining multiplication (open area) and division (dense area) operations:

$$s=s_{base}\times\frac{1+\alpha(1-d)}{1+\beta d} \tag{18}$$

Where s_{base} is the base step size, α is the open region amplification factor ($\alpha>0$, which controls the intensity of the penalization operation), and β is the dense region reduction factor ($\beta>0$, which controls the intensity of the division operation). $\frac{1+\alpha(1-d)}{1+\beta d}$ as a dynamic adjustment factor. When $d\to 0$, the adjustment factor converges to $1+\alpha$, the step size is enlarged by $(1+\alpha)$ times to accelerate the global search; when $d\to 1$, the adjustment factor converges to $1/(1+\beta)$, the step size is reduced to $S_{base}/(1+\beta)$ to improve the safety of obstacle avoidance.

In 3D space, the step direction needs to be corrected in conjunction with the obstacle distribution. Define the modulus of the direction vector as $\|v\|=s$, direction is determined by a combination of target point and obstacle repulsion:

$$v_{new}=v_{\text{target}}\times(1-d)+v_{repel}\times d \tag{19}$$

where v_{target} is the unit vector pointing to the target, v_{repel} is the unit vector in the direction of obstacle repulsion, weights $(1-d)$, d realize the balance between target orientation and obstacle avoidance.

A safety distance penalty term is introduced to ensure that paths are kept away from obstacles:

$$\cos t_{safe}=\sum_{i=1}^{N}\frac{k_{repel}}{(d_i+\varepsilon)^2} \tag{20}$$

where d_i is the normalized distance to the *i*-th obstacle, ε is a very small constant, and k_{repel} is the repulsion coefficient. In comparison to conventional algorithms, AOA exhibits the benefits of streamlined operation and minimal parameterization, effectively harmonizing security and efficiency.

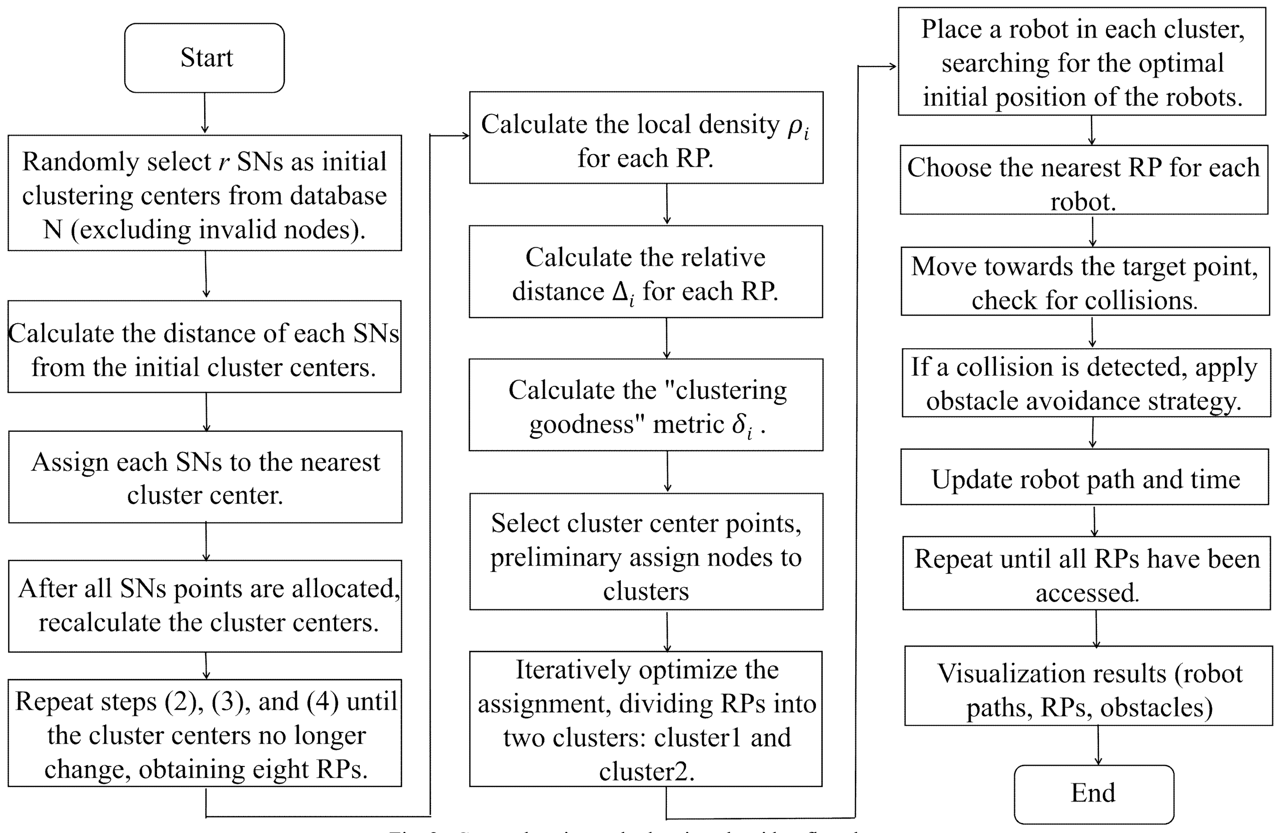

Start
Randomly select r SNs as initial clustering centers from database N (excluding invalid nodes).
Calculate the distance of each SNs from the initial cluster centers.
Assign each SNs to the nearest cluster center.
After all SNs points are allocated, recalculate the cluster centers.
Repeat steps (2), (3), and (4) until the cluster centers no longer change, obtaining eight RPs.
Calculate the local density ρ_i for each RP.
Calculate the relative distance Δ_i for each RP.
Calculate the "clustering goodness" metric δ_i .
Select cluster center points, preliminary assign nodes to clusters
Iteratively optimize the assignment, dividing RPs into two clusters: cluster1 and cluster2.
Place a robot in each cluster, searching for the optimal initial position of the robots.
Choose the nearest RP for each robot.
Move towards the target point, check for collisions.
If a collision is detected, apply obstacle avoidance strategy.
Update robot path and time
Repeat until all RPs have been accessed.
Visualization results (robot paths, RPs, obstacles)
End

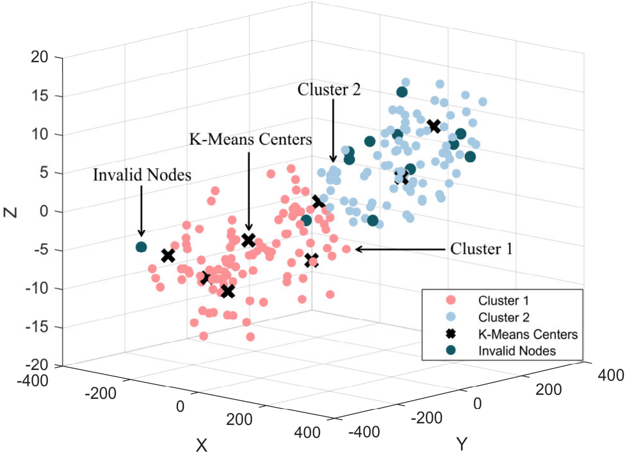
Fig. 2. Comprehensive path planning algorithm flowchart

The algorithm flowchart of MSPDC is shown in Fig. 2. First, use the K-means algorithm to find eight RPs. Randomly select valid nodes as initial points, that is, cluster centroids, calculate the distance between SNs and these initial points, and assign each SN to different clusters based on the distance. After the assignment, recalculate the cluster centroids and repeat the above steps until the centroids no longer change, thus finding the RPs. Later, the DPC algorithm is refined by calculating the sum of ρ_i and Δ_i for each RP. The "cluster goodness" index δ_i was calculated, and the cluster center was selected, the RPs were initially divided into two clusters. The allocation was optimized iteratively to improve the cluster stability. Finally, place one MS in each cluster and use the improved AOA for path planning. Select the optimal initial position for each MS through the optimal initial point function. Choose the nearest initial point for each MS, then move towards the target point. During this process, MSs use the obstacle avoidance function to navigate around obstacles until all RPs have been traversed. Visualize the path of each MS and calculate the total time taken to find the optimal path and traverse all RPs, the time consumed for turning, the length of the traversed path, and the obstacle avoidance success rate.

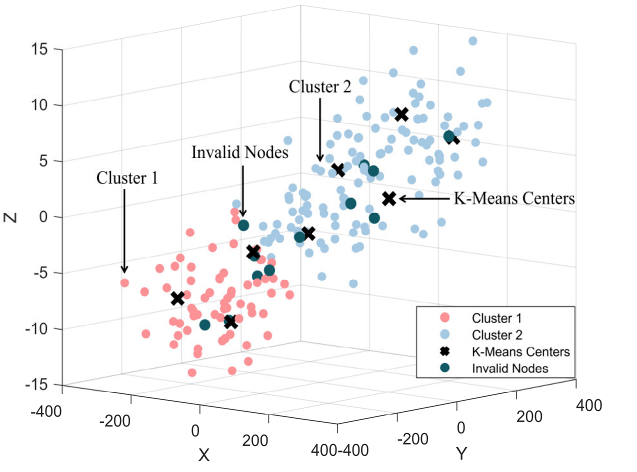
IV. Performance Comparison

In a MATLAB-based simulation of a wireless sensor network spanning 400 m × 400 m × 30 m, we randomly placed 200 SNs, $(n = 200)$, each with its own (x_i, y_i, z_i), $(1 \le i \le 200)$ coordinates and a communication radius of nine meters. The data for these nodes is organized into a 3D array. To mimic real-world network failures, we designated 6% of these nodes as non-functional. The K-means algorithm excels in rapidly processing extensive datasets, promptly yielding a limited number of cluster centroids, which substantially diminishes the volume of data for subsequent analysis. The improved DPC algorithm further leverages this efficiency by directly utilizing the sparse set of RPs generated by the K-means algorithm, thereby significantly reducing computational complexity. This enhancement renders the algorithm particularly adept in big data contexts, greatly amplifying its processing capabilities. In real-world applications, where node distributions often assume complex, irregular forms, the refined DPC algorithm demonstrates its versatility. It adeptly partitions these RPs based on density reachability, thereby not only identifying clusters of arbitrary geometries but also ensuring the robustness of cluster centroids. An iterative optimization process allows for continual refinement of these partitions, enhancing accuracy. By merging the K-means algorithm's quick generation of approximate solutions with the refined adjustments of the improved DPC algorithm, we achieve a notable improvement in clustering accuracy. This approach also maintains processing efficiency, offering an effective and precise solution for node clustering in complex scenarios. Initially, we used the K-means algorithm to find the best locations for eight RPs, followed by the DPC algorithm for network clustering.

To assess the performance of our improved algorithm, we compared it with traditional methods. The traditional approach, which initializes the DPC algorithm with the K-means algorithm, achieved cluster coverages of 94.23% for Cluster 1 and 93.75% for Cluster 2. Our improved algorithm, which ensures that cluster centers are always functional nodes and uses a Gaussian kernel for local density calculation with optimized bandwidth parameters, slightly outperformed the traditional method, increasing coverages to 94.29% for Cluster 1 and 93.85% for Cluster 2. This represents a 0.07% and 0.1% improvement, respectively. A visual comparison of the clustering results in a 3D space is shown in Fig. 3, with (a) showing the traditional method's clustering and (b) showing the improved algorithm's superior precision and coverage in complex 3D environments.



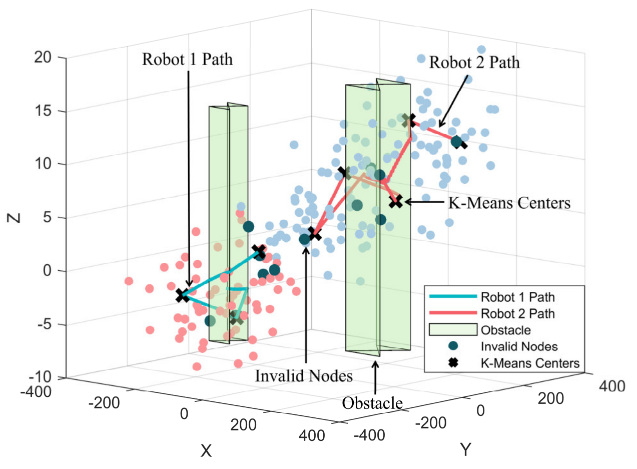
(a) DPC algorithm clustering with original K-means initialization



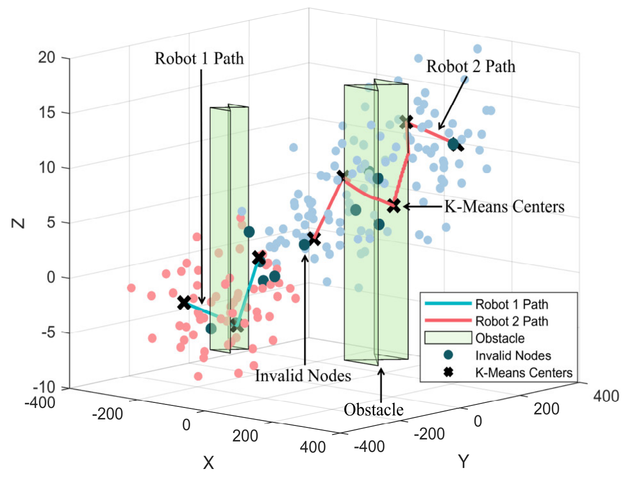
(b) DPC algorithm clustering with improved K-means initialization

Fig. 3. Comparison of the clustering results in the 3 D environment

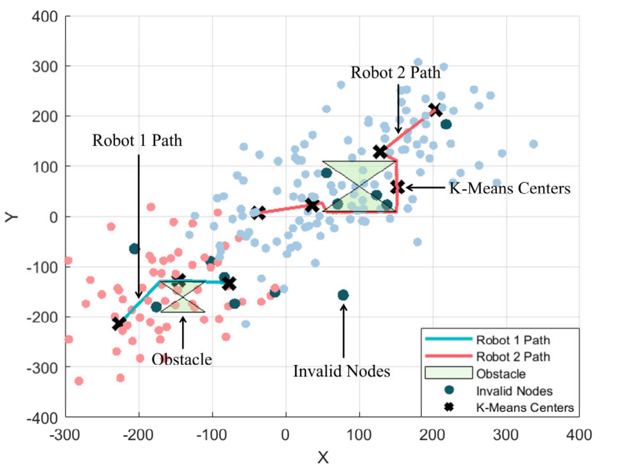
To address the limitations of most path planning methods being confined to two-dimensional spaces using MS for data collection and path planning, this study develops a 3D model for post-earthquake rescue operations and introduces MSPDC. In this 3D setting, we first applied a DPC algorithm, initialized with the K-means algorithm, to categorize effective nodes into two clusters. Each cluster has a dedicated mobile robot for navigating through its RPs. For path planning, we utilize an improved AOA approach to create the most efficient routes. These routes link path points via line segments, with the total MS travel distance being the combined length of these segments. The time used for MS path planning includes the time for path optimization, the time spent stopping at RPs, the time consumed for turning, and the time taken to traverse the path. MS stays at each RP for one minute to collect node data within the coverage of the RPs. The improved AOA efficiently maps out the shortest path from the starting node to the destination. A visual comparison of the simulation outcomes is shown in Fig. 4. In this figure, (a) shows the result of using traditional AOA to find the optimal path, (b) presents the result of using improved AOA to find the optimal path, and (c) provides a top view of the path planned by the improved algorithm. These visual representations provide a clear validation of the improved algorithm's proficiency in optimizing paths within intricate 3D settings.



(a) Multi-obstacle path planning based on traditional AOA



(b) Multi-obstacle path planning based on improved AOA



(c) Top view of multi-obstacle path planning based on improved AOA

Fig. 4. Comparison of path planning results between traditional AOA and improved AOA

In path planning, the choice of the initial point significantly affects the quality of the solution, easily leading to the solution space getting stuck in local optima. Traditional AOA is usually applied in simple or ideal two-dimensional environments. Using this method for path planning takes approximately 47.2 min, with a path length of 1560 m. This paper uses the improved AOA to optimize the initial position of the robot for complex 3D environments. It calculates the average Euclidean distance between each RP and other RPs, selecting the point with the smallest average distance as the optimal initial point. Considering the presence of 3D obstacles in space, we employ the AABB detection method to ascertain the location of obstacles and introduce an obstacle avoidance mechanism. By adjusting the step size through adaptive arithmetic operations, we achieve obstacle avoidance, target coordination, and efficient computation, balancing safety with efficiency. Upon detecting an impending collision, the algorithm automatically selects a direction to evade the obstacle and promptly adjusts the position of the MS to shift until avoiding collision with static obstacles. It can also swiftly and accurately locate the most optimal solution currently available. Experiments show that the improved AOA reduces the path planning time to about 29.6 min and the path length to 930 m, saving about 17.6 min and 630 m compared to traditional methods. This significantly improves convergence speed and obstacle avoidance effectiveness. It also rapidly generates initial solutions in complex scenarios like earthquake rescue. Additionally, it optimizes the shortest paths, effectively enhancing the efficiency of actual rescue operations.

By iterating 100 times, the improved AOA is compared with other MS path planning algorithms to find the optimal solution. The algorithms compared include the Bat Algorithm (BAT) [28], the Improved Grey Wolf Optimization Algorithm (CGWO) [29], the Genetic Algorithm (GA) [30], and the Whale Optimization Algorithm (WOA) [31]. The simulation route results of each algorithm are shown in Fig. 5-Fig. 8.

Fig. 5 shows the simulation outcomes of the BAT's path planning capabilities. This algorithm employs an echolocation-inspired approach to optimize paths, capitalizing on its robust global search proficiency to identify relatively optimal solutions. Nevertheless, in three-dimensional environments, it exhibits vulnerability to local optima, particularly during obstacle avoidance phases where the algorithm does not fully account for path optimality. This can result in repeated traversal of RPs and an escalation in path traversal time. Comparative performance analysis reveals that BAT exhibits the longest execution time, approximately 49.19 min, which exceeds the improved AOA by approximately 19.59 min. Furthermore, its path traversal length is approximately 1709.65 m, representing an increment of approximately 779.65 m relative to the improved AOA.

Fig. 6 shows the path planning simulation results of the CGWO, an improved version of the Grey Wolf Optimizer (GWO). CGWO incorporates chaotic mapping to enhance its global search capabilities and significantly improves search efficiency through the integration of multiple strategies. However, this also increases the number of computational steps, thereby extending the overall computation time. Even with its advanced global search capabilities, AOA faces inefficiencies during traversal in intricate 3D settings with a high density of SNs. This inefficiency can lead to suboptimal path optimization and a challenge in striking the right balance between optimizing path length and meeting other objectives. During traversal, CGWO takes approximately 49.51 min, with a path length of about 1675.72 m, which is around 19.91 min longer than the improved AOA and an increase of about 745.72 m in path length.

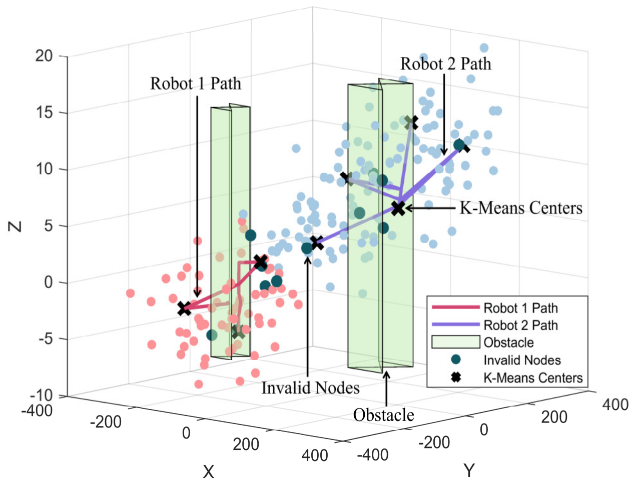


Fig. 5. Multi-obstacle path planning based on BAT

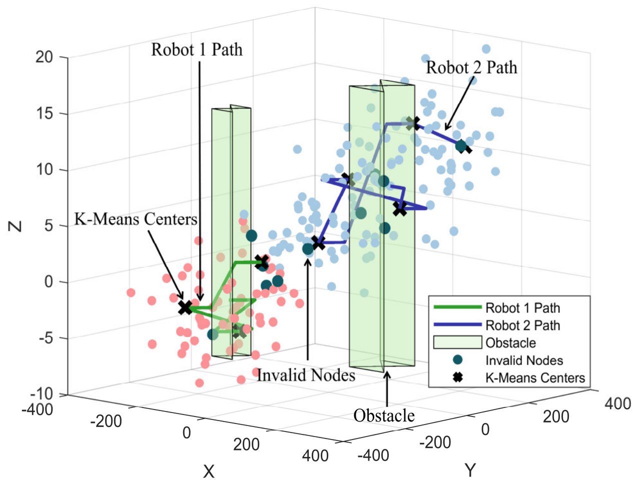


Fig. 6. Multi-obstacle path planning based on CGWO

Fig. 7 shows the path planning simulation results of GA. GA optimizes paths by simulating the process of natural selection and explores global optimal solutions through population evolution mechanisms. In 3D path planning problems, the population size and number of iterations directly affect computation time, leading to longer computational times. During obstacle avoidance, this algorithm selects to evade obstacles in a vertical manner, which considerably raises the count of paths traversed and is not apt for real-life earthquake rescue situations. In terms of performance, GA's performance resembles that of the basic AOA, with a longer duration dedicated to finding the optimal path. Its traversal time is approximately 45.86 min, and the path length is about 1643.05 m. Compared to the improved AOA, GA reduces time by about 16.26 min but increases the path length by approximately 713.05 m.

Fig. 8 shows the simulation results of the WOA for path planning. WOA effectively avoids local optima by mimicking the behavior of whales encircling prey and attacking with bubble nets, rendering it suitable for path optimization in complex obstacle-filled environments. It prioritizes finding the nearest point to the current position during path search. However, simulating whale behavior escalates computational complexity and imposes real-time constraints, potentially causing repeated traversals of RPs during path optimization. Consequently, utilizing WOA for optimal path search expends more time, approximately 40.13 min, and explores longer paths, with a traversal length of about 1356.53 m. Compared to the improved AOA, WOA extends path planning time by about 10.53 min and adds approximately 426.53 m to the path length.

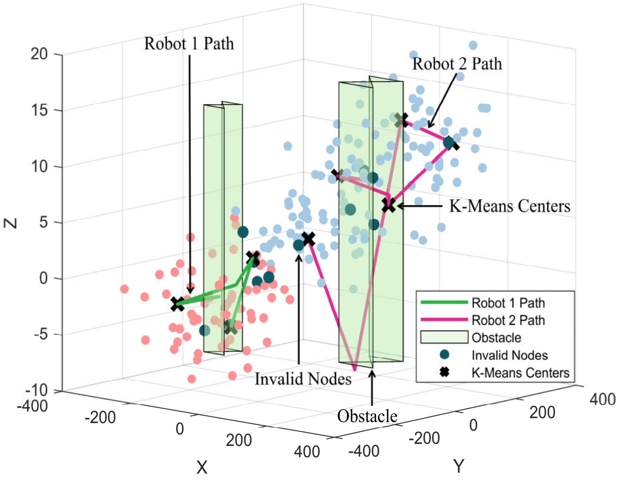


Fig. 7. Multi-obstacle path planning based on GA

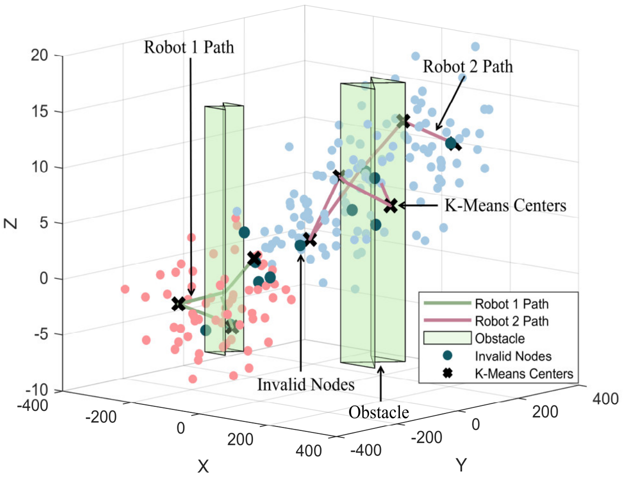


Fig. 8. Multi-obstacle path planning based on WOA

To comprehensively evaluate the performance advantages of the improved AOA, a comparative experiment was designed under unified conditions. Analyzing five key dimensions with the control variable of 100 iterations, the experiment recorded the performance of each algorithm in terms of optimization time, path length, traversal time, number of turns, and success rate of obstacle avoidance in complex environments. In the comparative experiments, five sets of standardized tables clearly present the performance differences between the improved AOA and traditional AOA, BAT, CGWO, GA, and WOA. Table I shows the shortest path optimization time. The improved AOA has an extremely short pathfinding time. R1, R2, and total pathfinding times are only 0.00094 s, 0.00214 s, and 0.00308 s, respectively. This significantly outperforms other algorithms. It strongly demonstrates its efficient path search capability. Table II shows the shortest path lengths. The R1, R2, and total paths of the improved AOA planning are about 276 m, 654 m, and 930 m, respectively. These are the shortest among all algorithms. This confirms its precise ability to plan concise paths. Table III highlights the total time taken to traverse the path. The improved AOA traverses the total distance in 29.6 min. This is significantly shorter than the 47.2 min for AOA and 49.19 min for BAT. It clearly shows how effective time optimization really is. Table IV reveals the number of turns in the travel path. The improved AOA needs just 6 turns to get from start to finish. On the other hand, the traditional AOA and CGWO require 16 turns to cover the entire route. Reducing turns helps lower robot energy consumption and control complexity. Table V shows the success rate of obstacle avoidance in complex environments. The improved AOA shows a prominent advantage in R1 and R2 obstacle avoidance success rates. These rates are 100% and 84%, respectively. In contrast, AOA, BAT, etc. perform poorly. For example, the BAT's R2 obstacle avoidance success rate is only 38%. This reflects its stronger adaptation and avoidance capabilities in complex environments. Overall, the experimental results intuitively reveal the comprehensive advantages of the improved AOA in multi-objective optimization, surpassing traditional algorithms in pathfinding efficiency, path simplification, time control, motion complexity, and environmental adaptability, demonstrating the core advantages of optimized path planning, and laying a solid foundation for the application of this algorithm in actual earthquake rescue scenarios.

A. Path -finding Time

Fig. 9 shows the fitness function curves for path optimization of two robots, where (a) and (b) record the time variation trends for R1 and R2 in finding the optimal path, respectively. From the characteristics of the curve, it is evident that the traditional AOA has significant drawbacks: R1 requires approximately 50 iterations to converge, with a slow convergence speed, a lengthy path optimization process, and a high susceptibility to getting trapped in local optima. To address these issues, the introduction of an adaptive step size mechanism effectively improves the algorithm's insufficient exploration in the initial stages of iteration. Simultaneously, by incorporating an optimal initial point selection function, the path exploration time is significantly reduced. The improved AOA exhibits exceptional performance with the fastest convergence speed, completing the optimization in approximately 0.003 s, significantly reducing path optimization time, and substantially enhancing global search capabilities. Other comparative algorithms also exhibit different performance characteristics: BAT takes a reasonable amount of time to search paths but lacks global search ability, taking more than 30 iterations to find the best solution, fitting only for quick responses in certain situations. CGWO is a bit slower in convergence than the improved AOA, with its fitness curve showing big swings initially and

TABLE I
PATH FINDING MINIMUM TIME COMPARISON

Algorithm	Improved AOA	AOA	BAT	CGWO	GA	WOA
R1 path-finding time (s)	0.00094	0.07797	0.01603	0.01953	0.14980	0.00529
R2 path-finding time (s)	0.00214	0.21483	0.01972	0.01642	0.16035	0.00648
Total path-finding time (s)	0.00308	0.29280	0.03575	0.03596	0.31015	0.01177

TABLE II
SHORTEST PATH LENGTH COMPARISON

Algorithm	Improved AOA	AOA	BAT	CGWO	GA	WOA
R1 path length (m)	276.1608	501.0122	526.7394	642.2150	454.4200	433.6760
R2 path length (m)	653.9160	1059.0000	1182.9134	1033.5001	1188.6281	922.8580
Total path length (m)	930.0768	1560.0122	1709.6528	1675.7151	1643.0481	1356.5340

TABLE III
SHORTEST TRAVERSAL TIME COMPARISON

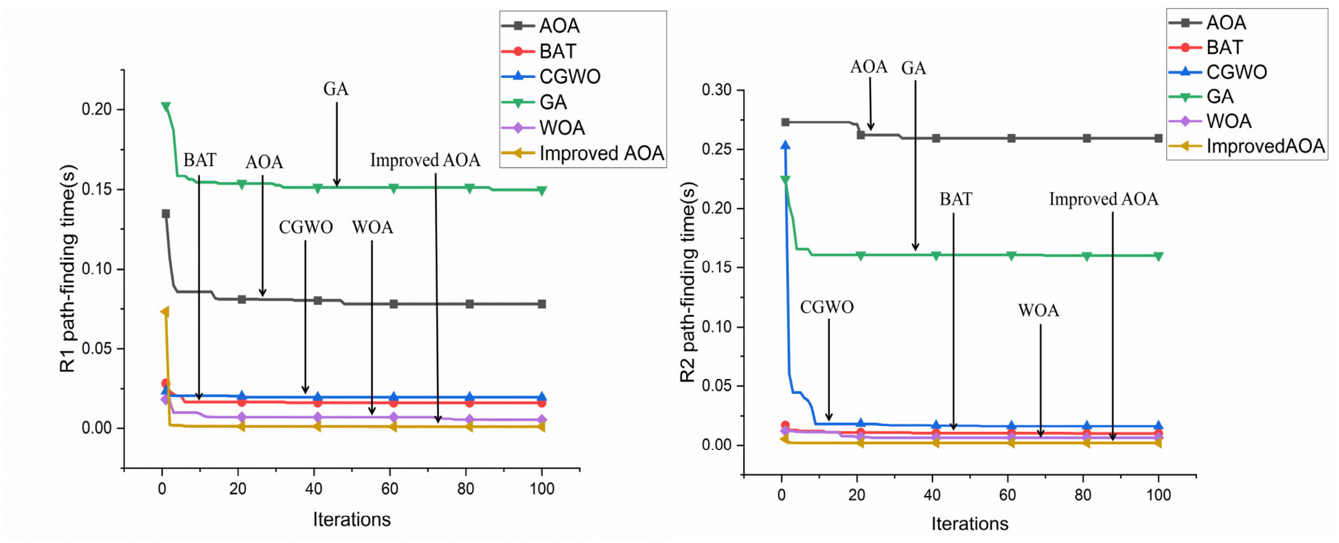
Algorithm	Improved AOA	AOA	BAT	CGWO	GA	WOA
R1 traversal time (min)	9.0232	16.0200	16.0348	19.3443	13.5884	13.6735
R2 traversal time (min)	20.5783	31.1800	33.1583	30.1700	32.2726	26.4572
Total traversal time (min)	29.6015	47.2000	49.1931	49.5143	45.8610	40.1307

TABLE IV
COMPARISON OF THE MINIMUM NUMBER OF TURNS

Algorithm	Improved AOA	AOA	BAT	CGWO	GA	WOA
R1 number of turns	1	6	5	7	3	4
R2 number of turns	5	10	9	9	7	6
Total number of turns	6	16	14	16	10	10

TABLE V
COMPARISON OF OBSTACLE AVOIDANCE SUCCESS RATES

Algorithm	Improved AOA	AOA	BAT	CGWO	GA	WOA
R1's obstacle avoidance success rate	100%	59%	34%	67%	33%	66%
R2's obstacle avoidance success rate	84%	61%	38%	40%	40%	57%



(a) Time fitness curve of R1 optimization (b) Time fitness curve of R2 optimization

Fig. 9. The fitness function curves for path optimization

a higher optimization time in the end. The traditional GA is inefficient, requiring over 80 iterations to converge, leading to a significant increase in optimization time for both robots within their respective clusters. WOA may have a shorter path optimization time, but R1 requires almost 80 iterations to reach the optimal value, indicating poor convergence efficiency.

Overall, the improved AOA really shines with its minimal iterations needed for full convergence and virtually zero optimization time. It outperforms other algorithms, cuts down on computational resources, and demonstrates significant practicality and advantage.

B. *Path Length*

Fig. 10 shows the fitness function curves for the path lengths of two robots, with (a) and (b) recording the variation process of the path lengths traversed by R1 and R2, respectively. By observing the trend of the curve, it is evident that the traditional AOA has significant shortcomings in optimizing path length. It takes approximately 20 iterations to converge to the optimal value, with a moderate convergence speed. However, the final obtained path length is relatively long, revealing the limitations of its global search capability. The improved AOA excels in path length optimization through innovative mechanisms. It not only converges fastest but also achieves the optimal final solution, skillfully balancing the relationship between global exploration and local exploitation. It effectively avoids premature convergence, achieving the shortest path length of approximately 32 m. Other comparative algorithms also exhibit different performance characteristics: BAT shows a relatively fast convergence speed in the initial stages of iteration, but it easily falls into local optima in the later stages. R2 requires nearly 80 iterations to find the shortest path, indicating low efficiency. CGWO demonstrates stability in global search with a moderate convergence speed, but its ability to refine local paths is weak, leading to a longer overall traversal path. GA, highly dependent on population diversity, converges slowly in the early iterations and is more prone to stagnation in the later stages, making it difficult to find the optimal path solution. The overall path length is relatively large. WOA performs well in exploration and converges quickly but fails to find optimal paths effectively.

Overall, the improved AOA achieves the best balance between convergence speed and solution quality, accurately identifying the shortest path. In practical application scenarios, especially in fields like earthquake rescue where path planning requires high timeliness and accuracy, the improved AOA showcases unparalleled technical advantages and potential applications.

C. Total Traversal Time

The total time for the robot to traverse includes the time spent on path optimization, the time taken to traverse the path length, the time consumed for turning, and the duration of stay at each RP. In the total time iteration curves for the two robots shown in Fig. 11, (a) and (b) record the total time changes for R1 and R2, respectively. During the actual path traversal, the traditional AOA tends to produce subpar solutions because it repeatedly visits RPs. This not only slows down the convergence but also affects the stability of the algorithm during iterations. The improved AOA records traversed RPs in real time by creating a path set. Whenever the algorithm discovers new RPs, it adds them to the set. When planning the next node, it carefully selects from nodes outside the set, completely avoiding the issue of revisiting. Other comparative algorithms also exhibit different performance characteristics: BAT converges slowly, and in complex settings, R2 needs approximately 80 iterations to identify the optimal solution, which tends to be high in value but less effective. CGWO shows relatively better convergence, yet the solution quality is mediocre, with R1's optimal solution taking roughly 16 min, and it underperforms in simple environment optimizations. GA's convergence speed is limited, exhibiting a gradual decline in the time curve during iterations, necessitating numerous iterations for optimization, leading to prolonged application time and low efficiency. While WOA can locate better optimal solutions, it converges too slowly and easily gets trapped in local optima.

Overall, the improved AOA shines in cutting down the total time, clocking in at just around 29.6 min. It nails the shortest time and fastest convergence, and it seriously ramps up computational efficiency.

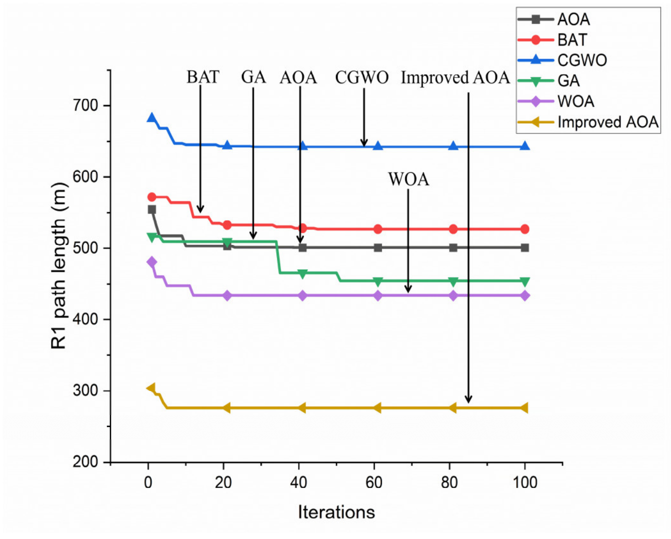
D. Number of turns

In the process of robots traversing paths, a turn is counted when the turning angle is greater than 10 degrees. Fig. 12 shows the fitness function curves of the number of turns made by two robots, where (a) and (b) record the changes in the number of turns during the traversal of R1 and R2, respectively. The traversal path of R1 is relatively simple, while that of R2 is more complex. The traditional AOA has poor global search capability and inefficient exploration in the early stages. It takes approximately 60 iterations to find the minimum number of turns. The traditional AOA is only suitable for simple obstacle environments, and the paths this algorithm generates often contain redundant inflection points. The improved AOA adaptively adjusts step sizes according to obstacle density and achieves obstacle avoidance via dynamic arithmetic operations. This improvement significantly enhances path quality, effectively avoids invalid inflection points, and greatly reduces the number of turns. The two robots require only six turns to complete the traversal. Other comparative algorithms also exhibit different performance characteristics: BAT has poor optimization efficiency, leading to frequent turns during path traversal and slow convergence. It is only suitable for scenarios with low real-time requirements. CGWO can demonstrate efficient optimization capabilities in the early stages of iteration, but it often requires multiple turns to traverse complex paths, resulting in low practicality. GA demonstrates strong initial optimization performance but easily converges with suboptimal solutions, requiring multiple iterations to minimize the number of turns. WOA suits long-term iterative tasks and gradually approaches the optimal solution, but its initial solution quality is suboptimal, and it tends to get trapped in local optima.

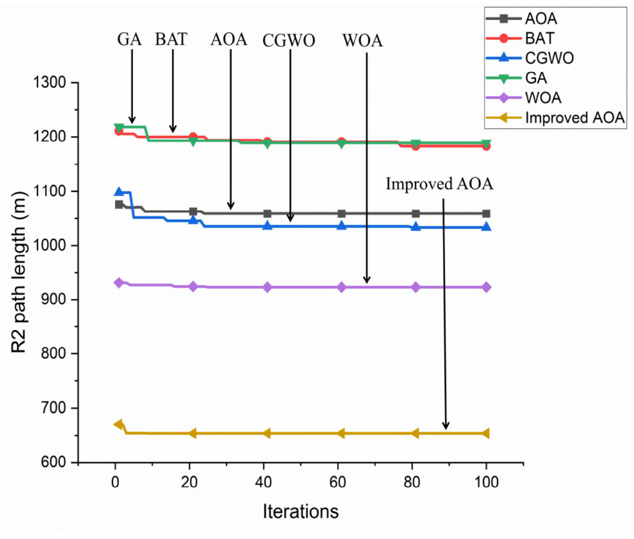
Overall, the improved AOA has significant advantages in path planning performance, requiring only about 20 iterations to achieve the minimum number of turns, greatly improving the efficiency of path planning, and making it more suitable for applications in earthquake rescue scenarios.

E. Success rate of obstacle avoidance

Obstacle avoidance success rate is an important indicator of robot performance in complex environments, which is defined as the ratio of the number of successful obstacle avoidance attempts to the total number of obstacle avoidance attempts by the robot during traversal. The success rates of the traditional AOA are 59% and 61%, indicating that it has basic obstacle avoidance capability, but it is limited by the lack of global search efficiency and is prone to fail in complex obstacle scenarios due to path redundancy. By systematically optimizing the strategy, the improved AOA shows excellent performance in the obstacle avoidance task: the success rate of R1 reaches 100%, and R2 increases to 84%. Other comparative algorithms also exhibit different performance characteristics: BAT has the lowest success rates of 34% and 38% in obstacle avoidance. It relies on the acoustic pulse simulation mechanism, which has weak anti-jamming ability in complex obstacle environments. Consequently, it fails to effectively handle the demands of complex path planning. CGWO has success rates of 67% and

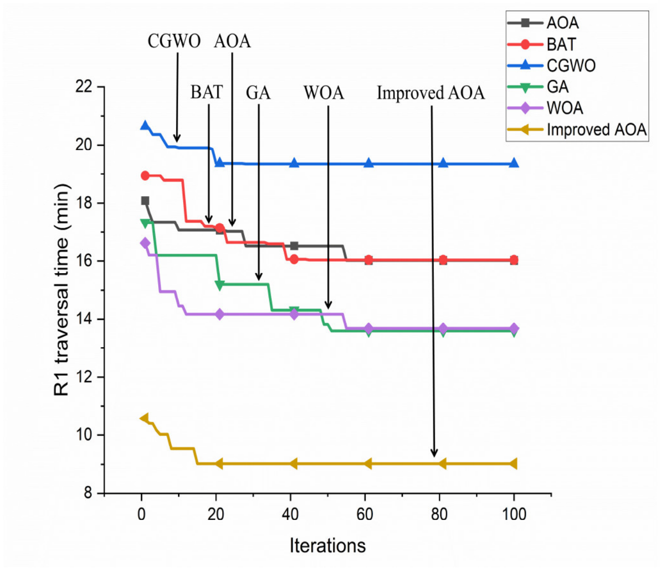


(a) The path length fitness curve of R1

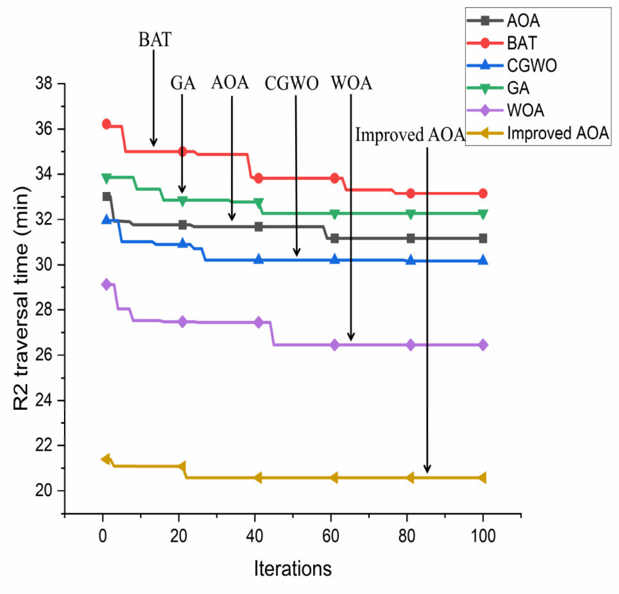


(b) The path length fitness curve of R2

Fig. 10. The fitness function curves for path length

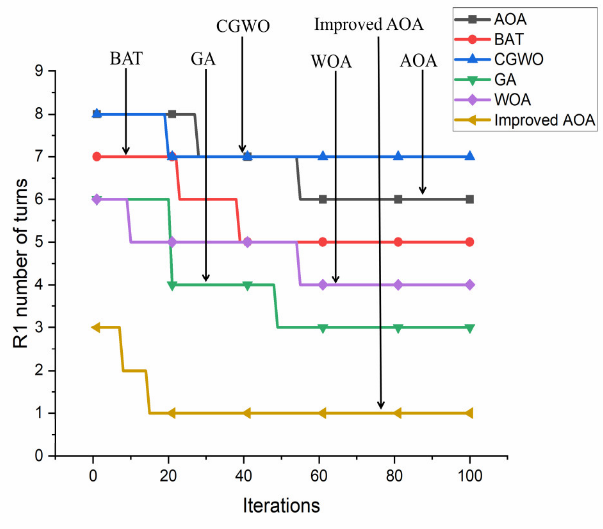


(a) Total time fitness curve for R1

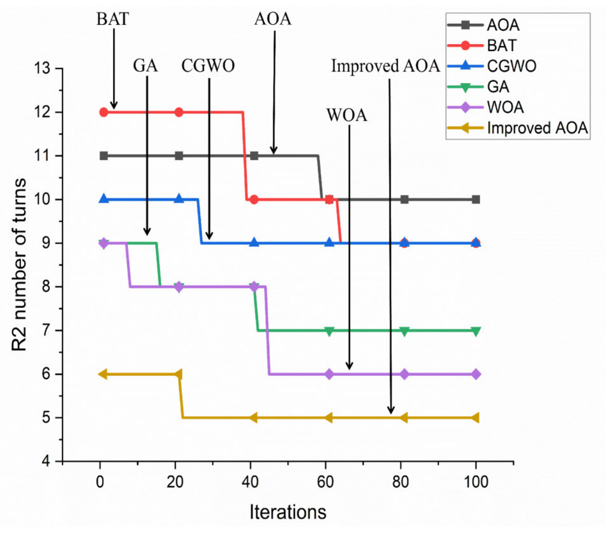


(b) Total time fitness curve for R2

Fig. 11. The fitness function curve of the total traversal time



(a) The number of turns of the fitness curve for R1



(b) The number of turns of the fitness curve for R2

Fig. 12. The fitness function curves of the number of turns

40%, which are adapted to simpler obstacle environments. And the success rates of 33% and 40% for GA reflect that the genetic operation is not able to effectively balance the exploration and optimization of path optimization. Path optimization fails to effectively balance exploration and exploitation. The success rates of WOA are 66% and 57%, and its simulated whale spiral search mechanism shows stability in most scenarios, but the algorithm relies on long-term iterations to gradually approach the optimal solution, and the overall efficiency is low.

Overall, the improved AOA performs excellently in obstacle avoidance efficiency, with remarkable optimization results, and can meet the task requirements in various complex environments.

From a multi-dimensional experimental perspective, the improved AOA demonstrates significant advantages in path planning performance. Firstly, in the 3D environment simulation experiments, visually comparing the planned paths of various algorithms reveals that the improved AOA generates more concise and fluent paths. It accurately avoids various obstacles, demonstrating strong environmental adaptability. Secondly, a quantitative comparative analysis of core indicators is conducted. These indicators include the shortest optimization time, shortest path length, shortest traversal time, minimum number of turns, and obstacle avoidance success rate. The analysis reveals that the improved AOA outperforms other algorithms in all key metrics. This fully validates its superior comprehensive performance. Lastly, a detailed observation of the fitness function curves of each indicator further highlights the performance advantages of the improved AOA. It not only converges faster and has higher solving efficiency but also demonstrates outstanding stability and reliability in complex environments. These characteristics make the improved AOA highly practical and promising for complex scenarios such as earthquake rescue, effectively meeting the stringent requirements of high-risk, high-complexity tasks for path planning.

V. Conclusion

In WSNs, numerous SNs are spread across the 3D space. The current research focus is on using MSs to gather data from these SNs. The challenge lies in optimizing the paths for multiple MSs, ensuring a balance between the time taken for algorithm optimization, total traversal time, path length, number of turns, and the success rate of obstacle avoidance. To enhance rescue efficiency in earthquake-stricken areas, this paper introduces the MSPDC. It comprises three steps: pre-deploying several SNs in the disaster zone, utilizing the K-means algorithm for its efficiency in handling large data sets to identify the cluster centers of intact nodes, and then using these as RPs to establish the best possible locations. Subsequently, an improved DPC algorithm is employed to segment all RPs into distinct clusters. This algorithm excels in automatically identifying cluster centroids based on local density and relative distance. It iteratively refines the allocation of boundary points to enhance node coverage and clustering stability, thereby maximizing the reception of information from the disaster area. Finally, path planning is performed using an improved AOA, which is simple and easy to implement with its global search capability. Each cluster is assigned a robot equipped with an MS responsible for collecting data from the nodes within its cluster and transmitting it to the emergency command center. This coordinated approach ensures that multiple MSs operate efficiently, avoiding the redundancy of collecting data from the same RPs. The algorithm has been further enhanced to determine optimal initial positions for robots within each cluster in a 3D space. Additionally, to navigate potential unknown obstacles in actual rescue operations, an AABB collision detection mechanism is implemented to ascertain the robot's position relative to obstacle boundaries. The step size is adjusted adaptively according to the density of obstacles, and flexible obstacle avoidance is achieved by dynamically adjusting arithmetic operators. The algorithm undergoes further optimization for effective evasion of randomly appearing obstacles in earthquake rescue scenarios. It strikes a balance between computational efficiency and real-time performance. Additionally, it avoids excessive computational complexity to ensure it does not hinder real-time response and network performance. In a 3D environment, the improved AOA is compared with traditional AOA, BAT, CGWO, GA, and WOA. The improved algorithm shows significant advantages in finding the optimal path, reducing the total path length and time, decreasing the number of turns, and increasing the success rate of obstacle avoidance. These features make it suitable for practical scenarios like earthquake rescue operations.

This paper assumes that obstacles are randomly generated and fixed, without considering the issue of moving obstacles. In the actual earthquake rescue process, the real-time movement of victims within the disaster area needs to be considered. Although the MSPDC proposed in this paper improves node coverage, quickly finds the shortest path in a 3D environment, reduces the number of turns, and enhances the success rate of obstacle avoidance, it does not address the issue of network energy consumption. Future research will focus on parameter optimization of the MSPDC and explore effective strategies to reduce network energy consumption. This is not only a further improvement of the algorithm's performance but also a key step in promoting its widespread application in complex earthquake rescue environments.

References

[1] J. Wang, J. Cao, R. S. Sherratt, and J. H. Park, "An Improved Ant Colony Optimization-Based Approach with Mobile Sink for Wireless Sensor Networks," The Journal of Supercomputing, vol. 74, no. 12, pp6633–6645, 2018.

[2] R. V. Kulkarni, A. Förster, and G. K. Venayagamoorthy, "Computational Intelligence in Wireless Sensor Networks: A Survey," IEEE Communications Surveys & Tutorials, vol. 13, no. 1, pp68-96, 2010.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," Computer Networks, vol. 38, no. 4, pp393-422, 2002.

[4] F. Wang, and J. Liu, "Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches," IEEE Communications Surveys & Tutorials, vol. 13, no. 4, pp673-687, 2010.

[5] R. Xie, and X. Jia, "Transmission-Efficient Clustering Method for Wireless Sensor Networks Using Compressive Sensing," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp806-815, 2013.

[6] N. Sharmin, A. Karmaker, W. L. Lambert, M. S. Alam, and M. S. A. Shawkat, "Minimizing the Energy Hole Problem in Wireless Sensor

Networks: A Wedge Merging Approach，" Sensors， vol. 20, no. 1, pp277, 2020.

[7] M. I. Khan, W. N. Gansterer, and G. Haring, "Static vs. Mobile Sink: The Influence of Basic Parameters on Energy Efficiency in Wireless Sensor Networks， "Computer Communications, vol. 36, no. 9, pp965-978, 2013.

[8] S. Yang, U. Adeel, Y. Tahir, and J. A. McCann, "Practical Opportunistic Data Collection in Wireless Sensor Networks with Mobile Sinks," IEEE Transactions on Mobile Computing, vol. 16, no. 5, pp1420-1433, 2016.

[9] W. Liu, K. Lu, J. Wang, G. Xing, and L. Huang, "Performance Analysis of Wireless Sensor Networks with Mobile Sinks," IEEE Transactions on Vehicular Technology, vol. 61, no. 6, pp2777-2788, 2012.

[10] H. B. Zhang, and B. Huang, "Research on Mobile Robot Path Planning Based on Multi-Strategy Improved Ant Colony Optimization Algorithm," Engineering Letters, vol. 33, no. 3, pp688-703, 2025.

[11] Y. B. Wang, J. S. Wang, and X. F. Sui, " Improved Particle Swarm Optimization Algorithm with Logistic Function and Trigonometric Function for Three-dimensional Path Planning Problems," Engineering Letters, vol. 33, no. 2, pp442-459, 2025.

[12] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," Vehicles, vol. 3, no. 3, pp448-468, 2021.

[13] M. M. Costa, and M. F. Silva, "A Survey on Path Planning Algorithms for Mobile Robots," 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, pp1-7, 2019.

[14] M. N. Zafar, and J. C. Mohanta, "Methodology for Path Planning and Optimization of Mobile Robots: A Review," Procedia Computer Science, vol. 133, pp141-152, 2018.

[15] L. Liu, X. Wang, X. Yang, H. Liu, and J. Li, "Path Planning Techniques for Mobile Robots: Review and Prospect," Expert Systems with Applications, vol. 227, pp120254, 2023.

[16] A. Ben Yagouta, B. Ben Gouissem, S. Mnasri, M. Alghamdi, M. Alrashidi, M. A. Alrowaily, I. Alkhazi, R. Gantassi, and S. Hasnaoui, "Multiple Mobile Sinks for Quality of Service Improvement in Large-Scale Wireless Sensor Networks," Sensors, vol. 23, no. 20, pp8534, 2023.

[17] S. F. Ochoa, and R. Santos, "Human-Centric Wireless Sensor Networks to Improve Information Availability During Urban Search and Rescue Activities," Information Fusion, vol. 22, pp71-84, 2015.

[18] M. T. Lazarescu, "Design of A WSN Platform for Long-Term Environmental Monitoring for Iot Applications," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 3, no. 1, pp45-54, 2013.

[19] F. H. Ajeil, I. K. Ibraheem, M. A. Sahib, and A. J. Humaidi, "Multi-Objective Path Planning of An Autonomous Mobile Robot Using Hybrid PSO-MFB Optimization Algorithm," Applied Soft Computing, vol. 89, pp106076, 2020.

[20] H. Salarian, K. W. Chin, and F. Naghdy, "An Energy-Efficient Mobile-Sink Path Selection Strategy for Wireless Sensor Networks," IEEE Transactions on Vehicular Technology, vol. 63, no. 5, pp2407-2419, 2013.

[21] B. Altintas, and T. Serif, "Improving RSS-Based Indoor Positioning Algorithm Via K-Means Clustering," 17th European Wireless 2011-Sustainable Wireless Technologies, VDE, pp1-5, 2011.

[22] S. Tiwari, and T. Solanki, "An Optimized Approach for K-Means Clustering," International Journal of Computer Applications, vol. 975, pp8887, 2013.

[23] Y. Wang, J. Qian, M. Hassan, X. Zhang, T. Zhang, C. Yang, and F. Jia, "Density Peak Clustering Algorithms: A Review on The Decade 2014–2023," Expert Systems with Applications, vol. 238, pp121860, 2024.

[24] H. Wang, B. Zhou, J. Zhang, and R. Cheng, "A Novel Density Peaks Clustering Algorithm Based on Local Reachability Density," International Journal of Computational Intelligence Systems, vol. 13, no. 1, pp690-697, 2020.

[25] R. B. Wang, W. F. Wang, L. Xu, J. S. Pan, and S. C. Chu, "An Adaptive Parallel Arithmetic Optimization Algorithm for Robot Path Planning," Journal of Advanced Transportation, vol. 2021, no. 1, pp3606895, 2021.

[26] Z. Wang, H. Sun, H. Li, and T. Lai, "AOA Positioning and Path Optimization of UAV Swarm Based on A-Optimality," IEEE Access, vol. 10, pp14946-14958, 2022.

[27] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The Arithmetic Optimization Algorithm," Computer Methods in Applied Mechanics and Engineering, vol. 376, pp113609, 2021.

[28] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Berlin, Heidelberg: Springer Berlin Heidelberg, vol. 284, pp65-74, 2010.

[29] H. Yu, Y. Yu, Y. Liu, Y. Wang, and S. Gao, "Chaotic Grey Wolf Optimization," 2016 International Conference on Progress in Informatics and Computing (PIC), IEEE, pp103-113, 2016.

[30] S. Mirjalili, and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp51-67, 2016.

[31] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology," Control, and Artificial Intelligence, MIT Press, 1992.