Received November 29, 2019, accepted December 13, 2019, date of publication December 19, 2019, date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960859

Intelligent Path Planning for AUVs in Dynamic Environments: An EDA-Based Learning Fixed Height Histogram Approach

RUN-DONG LIU[1], (Student Member, IEEE), ZONG-GAN CHEN[1], (Student Member, IEEE), ZI-JIA WANG[2], (Student Member, IEEE), AND ZHI-HUI ZHAN[1], (Senior Member, IEEE)
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China

Corresponding authors: Zi-Jia Wang (wangzjia@mail2.sysu.edu.cn) and Zhi-Hui Zhan (zhanapollo@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102, in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006.

ABSTRACT Autonomous underwater vehicles (AUVs) are robots that require path planning to complete missions in different kinds of underwater environments. The goal of path planning is to find a feasible path from the start-point to the target-point in a given environment. In most practical applications, environments have dynamic factors, such as ocean flows and moving obstacles, which make the AUV path planning more challenging. This paper proposes an estimation of distribution algorithm (EDA) based approach, termed as learning fixed-height histogram (LFHH) to solve path planning problems for AUVs in dynamic environment. The LFHH uses a learning transformation strategy (LTS) to improve its accuracy and convergence speed. Besides, a smooth method is employed to accelerate the speed of finding feasible paths. Moreover, a planning window is adopted to help handle dynamic factors. LFHH is tested in both complex 2-D and 3-D environments with time-variant dynamic factors, and experimental results validate the effectiveness of LFHH.

INDEX TERMS Autonomous underwater vehicles (AUVs), learning fixed height histogram (LFHH), estimation of distribution algorithm (EDA), dynamic environments, path planning.

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) are robots with autonomous ability to work under the water [1]. Due to their autonomous ability, AUVs have been widely adopted to complete missions as a substitute for humans in many extreme environments, such as submarine cable routing [2], tracking a targeted isothermal layer [3], tracing chemical flume [4], collecting data [5], monitoring missions [6], following mobile marine organisms [7], and inspecting subsea cables [8]. Usually, AUVs should follow the path planned in advance to preferably complete different kinds of missions. However, the path may be changed in some complex environments with dynamic factors. Therefore, to stably adapt to different kinds of underwater environments, AUVs should possess the ability of path planning, i.e., finding a feasible and suitable path in a given environment with the start-point and target-point. In fact, when planning a path for AUV, there are many constraints, such as safety, energy consumption, traveling time, and dynamic factors in the environment [9]. These constraints make path planning more difficult.

In order to deal with the path planning problem for AUV, many path planning approaches have been proposed in recent years, which can be categorized into graph search-based approaches [10]–[14], tree search-based approaches [16], [17], artificial potential field (APF)-based approaches [18]–[23], and evolutionary algorithms (EAs)-based approaches [24]–[28]. In the graph search-based approaches, Wang *et al.* [10] propose a hybrid A* algorithm based path planner to handle path planning problems for AUV in static 2-D environment. Kularatne *et al.* [11] apply a graph search-based path planning algorithm in both static and time-variant flow fields. Soulignac [12] proposes a Dijkstra-like algorithm to address the path planning problem

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

in the 2-D environment with ocean flows. The probabilistic road map (PRM) proposed by Kavraki *et al.* [13] can also be regarded as graph-search based algorithm. Later, Saoud *et al.* [14] combine the PRM algorithm and Dijkstra algorithm to plan path for sailboat. Based on exist works, Karaman and Frazzoli [15] develop an improved PRM named PRM* and obtain more promising performance. Besides the graph search-based approaches, tree search-based approaches have also been employed to handle path planning problems for AUV. Karaman and Frazzoli [15] not only develop the PRM* algorithm in their work, but also develop an improved rapidly exploring random trees star (RRT*) algorithm, which can also be regarded as tree search-based algorithm. Later, Cui *et al.* [16] adopt a multidimensional RRT* algorithm to plan paths for multiple AUVs in the 3-D environment. However, the ocean flows and obstacles are not considered. Petres *et al.* [17] present a fast marching based approach to plan paths for AUV, which also consider the ocean flows. In the APF-based approaches, the original APF algorithm is proposed by Khatib [18], and its variants have been widely used to plan path for AUV [19]–[23]. Cheng *et al.* [19] use the APF method to effectively avoid obstacles for AUV in the 2-D environment. Saravanakumar and Asokan [20] propose a multipoint potential filed method to plan path for AUV in the 3-D environment, and it is applicable for real-time implementation. Kelasidi *et al.* [21] propose a two-stage strategy based on APF to plan path for underwater snake robots in the 2-D environment. Saoud *et al.* [22] use the APF method to plan path for autonomous ailing boats in the 2-D environment. Besides, they have considered the ocean flows. Chen *et al.* [23] propose an improved APF method to help construct the intelligent mobile system for unmanned ships in the 2-D environment.

In recent years, EAs-based approaches are widely used to handle path planning problems for AUV in both 2-D and 3-D environments. In [24], an improved particle swarm optimization (PSO) based path planner is proposed for multi-AUV cooperative problem in 2-D environment with ocean flows. Cheng *et al.* [25] combine genetic algorithm (GA) and dynamic programming to plan paths for AUV in static 3-D environment, which can satisfy navigation requirement. Zhang *et al.* [26] propose an adaptive differential evolution (DE) algorithm to plan paths for AUV in static 3-D environment, and the obstacles are taken into account. Mahmoudzadeh *et al.* [27] also adopt DE algorithm to plan paths for AUV in dynamic 3-D environment. For the dynamic factors, they take ocean flows and moving obstacles into account. Zhou *et al.* [28] propose a hybrid PSO algorithm to handle path planning problems for AUV in dynamic 3-D environment. Also, they have taken ocean flows into consideration.

The above researches focus on only one kind of environment (i.e., either the 2-D environment or 3-D environment). However, the 2-D environment (i.e., path planning on the surface of the sea) and the 3-D environment (i.e., path planning in the deep sea) are both necessary for practical application. Therefore, different from existing works that only study in 2-D or 3-D environment, this paper solves the path planning problem for AUV in both 2-D and 3-D environments. More significantly, this paper takes the dynamic factors into considerations in both 2-D and 3-D environments to make the problem models closer to practical application. These considerations make the path planning problem more complex and more difficult. Therefore, an efficient path planning approach that has a good ability for global search and dealing with dynamic factors is in great need.

Being a kind of EAs, the estimation of distribution algorithm (EDA) [29], [30] has been extensively studied in various optimization problems in real-world applications, such as insurance investment planning [31], processing uncertain capacitated arc routing problems [32], processing the uncertain scheduling problem in the steelmaking industry [33], and planning paths in driving system [34]. Planning paths for AUV is also a kind of optimization problems and EDA may be a promising solver. Different from GA or DE that uses pairs of individuals to generate offspring via crossover, and also different from PSO that individual updates by learning from only itself and another individual, EDA uses a set of promising individuals to construct a probabilistic model and generates new individuals according to the model. Therefore, EDA has a good diversity. Meanwhile, EDA can utilize more global information from more different promising individuals of the whole population so that it has a good convergence speed. For the optimization problem in dynamic environment, it is important to find optimal solution fast and keep the diversity while the environment changes. This way, EDA may be more effective and promising for dealing with the dynamic environments in the AUV path planning because different individuals may compensate for each other to eliminate the dynamic noise. Therefore, we consider the EDA-based approach in this paper. Among the EDA family, a variant named fixed-height histogram (FHH) algorithm has shown good global search ability [41], being helpful to handle path planning problems for AUVs. Therefore, this paper proposes an improved FHH variant, termed as learning FHH (LFHH) algorithm for AUV path planning in dynamic environments. Different from FHH algorithm, the LFHH algorithm uses the learning transformation strategy (LTS) to make individuals learn from the best individual in the current population to improve the accuracy of solutions and convergence speed. Besides, a smooth method is employed into LFHH to find feasible paths fast. During the evolutionary process, if a path swerves sharply, that means the path is not feasible. The smooth method can correct the path and this step can accelerate the speed of finding feasible paths. In order to handle the dynamic factors, a planning window is also employed into LFHH. Specially, when the environment change occurs, only paths inside the planning window are re-planned. Besides, the planning window size can be dynamically changed if the algorithm cannot find a feasible path with current planning window size. The LFHH algorithm is tested in both 2-D and 3-D environments with time-variant dynamic factors, i.e., the moving obstacles and the ocean flows. The experimental results show that the proposed LFHH has an effective performance while handling path planning for AUV in dynamic environment. The contributions of this paper are shown as follows.

1) We propose the LTS to make individuals learn from the best individual in the current population. The LTS can improve the accuracy of solutions and accelerate the convergence speed.
2) A smooth method is employed into LFHH. Specifically, during the evolutionary process, if a path swerves sharply, which means the path is not feasible. The smooth method can correct the path and this step can accelerate the speed of finding feasible paths.
3) A planning window with dynamic window size is employed into LFHH to deal with the dynamic environments. When the environment change occurs, only paths inside the planning window are re-planned. Besides, the planning window size can be dynamically changed if the algorithm cannot find a feasible path with current planning window size. This procedure can cut down the time consumption while planning paths in dynamic environment.

The rest of this paper is organized as follows. In Section II, EDAs and its variants will be introduced. In Section III, the problem formulation in both 2-D and 3-D environments will be presented. In Section IV, the LFHH algorithm plans path for AUV in dynamic environments will be presented in detail. In Section V, a series of experiments will be implemented to measure LFHH's performance in both 2-D and 3-D environments. Conclusions will be given in Section VI.

II. EDA AND ITS VARIANTS

A. FRAMEWORK OF EDA

EDA is a member of EAs. Compared with other EAs (e.g., GA [36] and DE [37], [38]), EDA does not have crossover and mutation operations. Instead, EDA uses a set of promising individuals to build a probabilistic model and samples new individuals according to the model. The procedures of EDA are as follows. First, randomly generate *NP* individuals within the search space to form the population, where *NP* is the population size. Then sort the individuals in the population from good to poor (i.e., in an ascending order for the minimum problem) according to their fitness values. After that, EDA uses a set of promising individuals to build a probabilistic model and samples new individuals according to the model. Then fitness values of the new individuals are calculated. After that, combine the new individuals and old individuals to select top *NP* individuals with smaller fitness values (i.e., for minimum problem) to form a new population. Because of its evolutionary mechanism, EDA can make good use of global information from various individuals (i.e., a set of promising individuals) of the current population. The procedure of basic EDA is shown in **Algorithm 1**.

Algorithm 1 Procedure of Basic EDA

```
1:  Begin
2:   Initialize the population with size $NP$;
3:   While (Terminate criteria is not satisfied)
4:      /* Model building phase */
5:      Select $S$ good individuals to build probabilistic model;
6:      /* Sampling phase */
7:      Use the probabilistic model to sample $NP$ individuals and calculate their fitness values;
8:      /* Forming a new population */
9:      Form a new population by selecting top $NP$ individuals from the combination set of $NP$ new individuals and $NP$ old individuals;
10:  End of While
11: End
```

B. FHH

Tsutsui *et al.* [40] develop two kinds of histogram-based EDAs. One is the FHH, whose histogram has the same height and varies in width. The other one is fixed-width histogram EDA, whose histogram has the same width and varies in height. The two kinds of EDA variants are shown in Fig. 1.

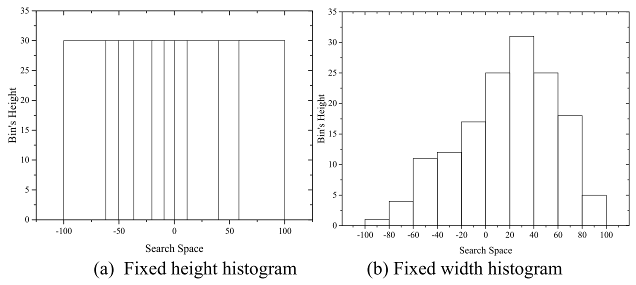


FIGURE 1. Two kinds of EDA variants.

As this paper develops an improved FHH for the path planning in AUV, we briefly describe the FHH algorithm herein. First, FHH randomly generates *NP* individuals to form the initial population. Then the main steps of FHH during the evolutionary process are described as follows:

Step 1) Model building phase. First, select top S individuals from the population according to their fitness values. Then construct histograms for each dimension according to these S individuals. Without loss of generality, we take the jth dimension as an example. In order to construct the histogram, the jth dimension values of the top S individuals will be copied into an array $arrayX$. After that, these S values are sorted in an ascending order, so that $arrayX = [item_1, item_2, \ldots, item_S]$ with $item_1 \leq item_2 \leq \ldots \leq item_S$. Then the histogram for the jth dimension is constructed by using the information in $arrayX$ according to Eqs. (1)-(2) [41]. As shown in Fig. 1(a), the histogram is formed by $binNum$ bins. The problem is how to determine the lower bound and upper bound of each bin. The basic idea of FHH is that each bin contains almost the same number of individuals. In order to implement this idea, the sorted *arrayX* can help to divide the S individuals into *binNum* sets. Therefore, the lower bound $lbound_k^j$ and upper bound $ubound_k^j$ of the kth bin are calculated as Eqs. (1)-(2), respectively. Lower bound of the first bin and upper bound of the last bin are set as the boundary $xmin^j$ and $xmax^j$ of the jth dimension, respectively.

$$lbound_k^j = \begin{cases} xmin^j, & \text{if } k=1 \\ (arrayX_{\lfloor (k-1)\cdot S/binNum \rfloor -1} \\ \quad + arrayX_{\lfloor (k-1)\cdot S/binNum \rfloor})/2, & \text{otherwise} \end{cases} \tag{1}$$

$$ubound_k^j = \begin{cases} xmax^j, & \text{if } k = binNum \\ lbound_{k+1}^j, & \text{otherwise} \end{cases} \tag{2}$$

Step 2) Sampling phase. In order to generate a new individual, all its dimensions have to be sampled according to their corresponding histograms. Take the jth dimension as an example, a bin will be randomly selected from its histogram, and a random number obeys the uniform distribution within the lower bound and upper bound of this bin is generated as the new dimension's value. When all dimensions' values are generated, the individual is formed. *NP* new individuals will be sampled according to the histogram model in this way. After new individuals are sampled, their fitness values will be calculated.

Step 3) Forming a new population. NP new individuals will be merged with those *NP* individuals in the old population. According to their fitness values, only top *NP* individuals are selected from these combined 2 *NP* individuals to be the final *NP* new individuals to form the new population.

III. PROBLEM FORMULATION

In some situations, AUV may work on the surface of the sea. In other situations, it may also work in the deep sea. We consider the path planning problem of AUV in both of the above two kinds of environments in this paper. For these two kinds of environments, the one on the surface of the sea can be regarded as a 2-D environment, while the other one in the deep sea can be regarded as a 3-D environment. Moreover, dynamic factors are considered in both the two kinds of environments to make our model more practical. In each environment, a cost function is designed to evaluate the quality of the path. The cost function is constructed by considering the path's length, safety, curvature, and the degree of coping with ocean flows.

A. PROBLEM FORMULATION FOR 2-D ENVIRONMENT

In the 2-D environment, AUV works on the surface of the sea. The environment on the surface of the sea is simpler than that in the deep sea. In order to facilitate the description for problem formulation, we name the horizontal axis and vertical axis with label Y and label X, respectively, as shown in Fig. 2. In the figure, the vortices represent ocean flows and the circled areas represent obstacles. The AUV should avoid colliding with obstacles. The positions of obstacles and the center of vortices will change as time goes by. When the change occurs, obstacles' positions can be detected by AUV's sensors. Ocean flows are simulated by Eqs. (3)-(5), similar to [39], as:

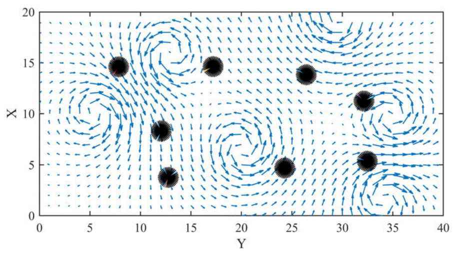


FIGURE 2. The 2-D environment with ocean flows and obstacles.

$$u(R) = -\eta \frac{y - y_0}{2\pi (R - R_0)^2} \left[1 - e^{-\left(\frac{(R-R_0)^2}{\zeta^2}\right)} \right] \tag{3}$$

$$v(R) = \eta \frac{x - x_0}{2\pi (R - R_0)^2} \left[1 - e^{-\left(\frac{(R-R_0)^2}{\zeta^2}\right)} \right] \tag{4}$$

$$V_c = (u, v) \tag{5}$$

where V_c represents the velocity of current field, $R = (x, y)$ represents the 2-D spatial domain, $R_0 = (x_0, y_0)$ represents the center of current vertex, η and ζ are fixed parameters which are used to control the strength and radius of the current vertex. In this paper, η and ζ are set as 2.0 and 1.2, respectively.

In order to find a feasible path, a set of waypoints have to be determined. Each waypoint is formed by X and Y positions. In our problem formulation, the 2-D environment is split into a number of bins by lines along the Y-axis (i.e., the horizontal direction), and therefore only the X values in different Y lines are needed to be optimized. As shown in Fig. 3, the environment is split into 40 sections by 40 lines, each line's length is the same as the maximal value of the X-axis, and it is also the same as the width of the 2-D environment. The overlaps between obstacles and lines are unsafe areas. Every waypoint is located on a safe position of one and only one line. When all waypoints are found (i.e., the solid red circles in all the lines), then a path is formed.

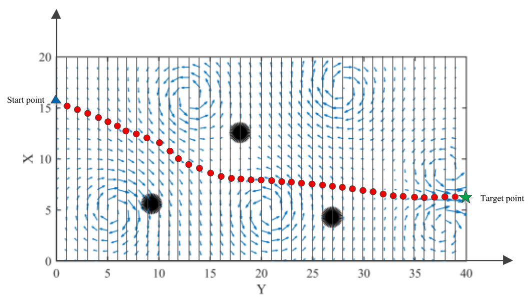


FIGURE 3. Split the 2-D spatial domain.

The fitness function of a path is defined as

$$F_{cost} = C_{length} + C_{curvature} + C_{block} + C_{current} \tag{6}$$

where C_{length} penalizes the path which is too long, $C_{curvature}$ penalizes the path which swerves (i.e., change the moving direction) sharply, C_{block} penalizes the path which collides with obstacles, and $C_{current}$ penalizes the path which cannot adapt with ocean flows well. The optimization objective is to minimize the fitness of the path as small as possible. The items of this fitness function are illustrated by Eqs. (7)-(10) and are described as follows.

C_{length} is defined as:

$$C_{length} = 1 - \frac{L_{start\_target}}{L_{path}} \tag{7}$$

where L_{start_target} represents the Euclidean distance between start-point and target-point, and L_{path} represents the length of path which is found by path planning algorithm. C_{length} ranges in [0, 1).

$C_{curvature}$ is defined as:

$$C_{curvature} = \begin{cases} \frac{N_{unable}}{N_{all}} + 2, & \text{if } N_{unable} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where N_{unable} is the number of path segments which beyond AUV's motion ability, and N_{all} represents the number of path segments in one path. As shown in Fig. 4, if the angle θ between two neighboring path segments is smaller than T_theta, it is difficult for AUV to change the direction smoothly. In this case, the path will be regarded as unsmooth, and the number of N_{unable} will increases 1. T_theta is set as $\pi/6$ in this paper, and its value may be different according to different AUVs. $C_{curvature}$ ranges in $0 \cup (2, 3)$.

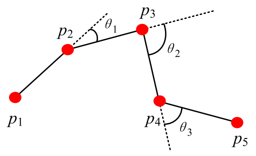


FIGURE 4. Angle between path segments.

C_{block} is defined as:

$$C_{block} = \begin{cases} \frac{N_{block}}{N_{all}} + 2, & \text{if } N_{block} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where N_{block} represents the number of path segments which collide with obstacles. C_{block} ranges in $0 \cup (2, 3)$.

$C_{current}$ is defined as:

$$C_{current} = \frac{1}{N_{all}} \sum_{n=1}^{N_{all}} \frac{\alpha_n - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \tag{10}$$

where α_n is the angle between nth path segment and ocean flows as illustrated in Fig. 5. The α_{min} and α_{max} represent the minimum and maximum angles, respectively. $C_{current}$ ranges in (0, 1).

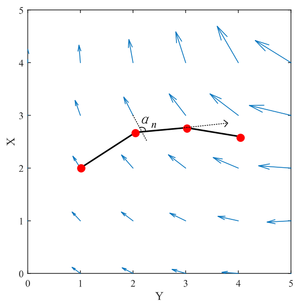


FIGURE 5. The angles between path segments and ocean flows.

In the fitness function, $C_{curvature}$ and C_{block} are critical penal items. When $C_{curvature}$ and C_{block} are equal to 0, that means the path is feasible. C_{length} and $C_{current}$ are used to improve the quality of paths for path planning algorithms. The smaller they are, the better path is. Herein, if F_{cost} is less than 2.0, that means the path is feasible.

B. PROBLEM FORMULATION FOR 3-D ENVIRONMENT

When dealing with the 3D environment, we only handle ocean flows in the horizontal plane. Because of the motion of earth, ocean flows are almost bi-dimensional [35]. As shown in Fig. 6, spheres represent obstacles, and the curved surfaces represent seabed. Ocean flows are similar to that in the 2-D environment. The seabed of the 3-D environment is simulated by a mathematical function as

$$Z = h \cdot \exp\left(-\frac{(Y - y_0)^2 \cdot a^2 + (X - x_0)^2 \cdot b^2}{2\sigma^2}\right) \tag{11}$$

where X, Y, Z represent the 3-D spatial domain. Parameters x_0 and y_0 represent peaks' positions in the seabed. Parameters σ, a, and b control peaks' shapes in the seabed. h represents the height of peaks in the seabed.

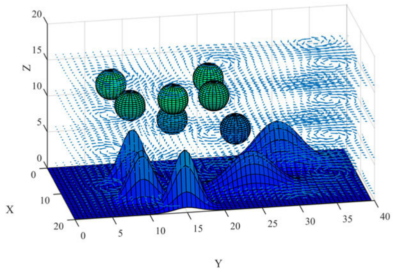
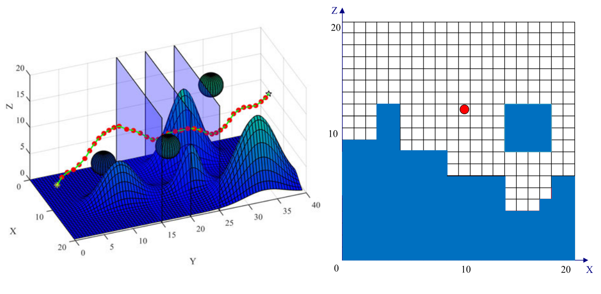


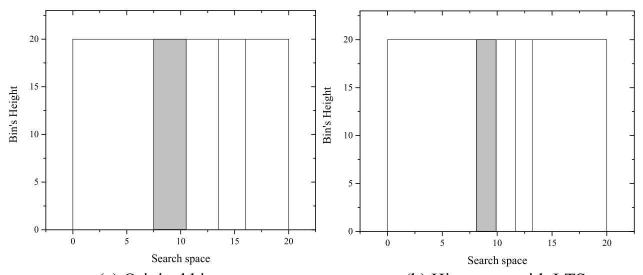
FIGURE 6. The 3-D environments with obstacles and ocean flows.

In order to extract information from the 3-D environment, the X-Y-Z cube is split into a number of sections along the Y-axis. As shown in Fig. 7(a), the 3-D environment of Fig. 6 is split into 40 sections, then a path in the 3-D environment consists of a set of successive waypoints in all sections along the Y-axis. In order to clearly present this procedure, only three sections are shown in Fig. 7(a). The shape of the section



(a) Split the X-Y-Z cube (b) Fit the section with grids

FIGURE 7. The profile section of the 3-D environment.



(a) Original histogram (b) Histogram with LTS

FIGURE 8. Perform LTS in the histogram.

is formed by grids in X-axis and Z-axis, as shown in Fig. 7(b). For each waypoint, the value of Y-axis is the index of the section, while the values of X-axis and Z-axis are needed to be optimized. The blue grids represent obstacles, and the others represent safe areas. Besides, the obstacle areas corresponding grids will be filled with 1, and the others will be filled with 0. In each section, a waypoint will be found in the safe areas, as the solid red circle shown in Fig. 7(b). When all waypoints are found, a path is formed.

In this 3-D environment, we also use Eq. (6) as the fitness function to evaluate the quality of a path the same as that in the 2-D environment.

IV. LFHH FOR AUVs PATH PLANNING

A. SOLUTION ENCODING

A feasible path is made up by a set of waypoints with each waypoint representing the positions of AUV along with the path. For the LFHH algorithm, each individual represents a solution (i.e., a path). Therefore, each dimension of the solution represents a waypoint of the path (i.e., the X value in the 2-D environment, or the X and Z values in the 3-D environment).

The solution expressions in the 2-D environment and 3-D environment are shown as Eq. (12) and Eq. (13), respectively, where Dim represents the number of waypoints. No matter in 2-D spatial domain or 3-D spatial domain, all solutions' Y-axis values are initialized as $\{1, 2, \ldots, Dim\}$. The other axis values (i.e., the X value in the 2-D environment, or the X and Z values in the 3-D environment) are to be optimized by the LFHH algorithm.

$$solution_{2\mathrm{D}} = \begin{bmatrix} x^1, & x^2, & x^{Dim} \\ y^1, & y^2, & y^{Dim} \end{bmatrix} = \begin{bmatrix} x^1, & x^2, & x^{Dim} \\ 1, & 2, & Dim \end{bmatrix} \tag{12}$$

$$solution_{3\mathrm{D}} = \begin{bmatrix} x^1, & x^2, & x^{Dim} \\ y^1, & y^2, & y^{Dim} \\ z^1, & z^2, & z^{Dim} \end{bmatrix} = \begin{bmatrix} x^1, & x^2, & x^{Dim} \\ 1, & 2, & Dim \\ z^1, & z^2, & z^{Dim} \end{bmatrix} \tag{13}$$

In the initialization, values of each dimension of a solution will be generated with the uniform distribution in their corresponding spatial domains, except for the first and last dimensions. The first and last dimensions of all solutions are set as the start-point and target-point, respectively. Then all solutions' fitness function values are calculated by using Eq. (6). After that, solutions will be sorted from small to large (i.e., from best to worst) according to their fitness function values.

B. MODEL BUILDING OF LFHH

In the 2-D environment, histogram will be constructed for each dimension in X-axis values. The model building phase of LFHH algorithm is similar to FHH. The difference is that we adopt the LTS to improve the solution accuracy and convergence speed in LFHH algorithm. As aforementioned, top S solutions will be selected from the current population according to their fitness function values. Take the j^{th} dimension as an example, values in j^{th} dimension of these S good solutions (i.e., X-axis values in individuals) will be copied into $arrayX$.

Before constructing the histogram, the LTS is applied to make the selected values close to the best solution's j^{th} dimension value. That means other solutions learn from the best solution. The LTS is performed as

$$arrayX = item_i + \lambda \cdot \left(x_{bestj} - item_i\right) \tag{14}$$

where $1 \le i \le S$, λ is a learning factor, x_{bestj} represents the best solution's j^{th} dimension value, $item_i$ represents the i^{th} value in $arrayX$, and the size of $arrayX$ is S. After completing this learning transformation step, the array $arrayX$ will be sorted in an ascending order (i.e., from small to large). Then the histogram of the j^{th} dimension will be constructed by using Eqs. (1)-(2), which is the same as FHH. As shown in Fig. 8, the left histogram is the original one, the right histogram is the one that employs LTS, and the gray bin is related to x_{best}. After performing LTS, we can see that the individuals will converge to the best individual to some degree from Fig. 8.

In the 3-D environment, histograms in X-axis and Z-axis of each dimension are constructed by following the same procedure in the 2-D environment. Take one dimension as an example, as shown in Fig. 9, the histogram in red color corresponds to X-axis, and histogram in blue color corresponds to Z-axis. The histogram in the 3-D environment is formed by these two histograms.

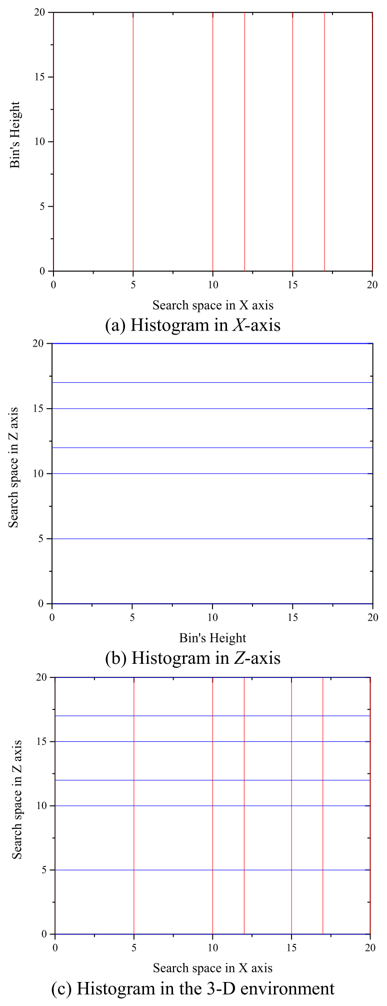


FIGURE 9. Histogram of one dimension in the 3-D environment.

C. SAMPLING NEW SOLUTIONS

In the 2-D environment, each dimension of a solution randomly selects a bin from its histogram distribution constructed in the model building phase. Then LFHH uses the uniform distribution to generate new values of this dimension with the limitation of bin's boundaries. When all dimensions are generated, the new solution is formed. Then use Eq. (6) to calculate its fitness value. These steps are the same as that in FHH.

It should be noted that, in the every T^{th} generation, LFHH does not use the probabilistic model to sample new solutions but uses a smooth method to generate new solutions. The smooth method is performed on each old solution to try to smooth the path. As illustrated in the left of Fig. 10, for the waypoint p_j, if its left path segment and right path segment have significantly different lengths, the curvature of the two path segments may be poor. In this case, if we can modify the position of waypoint p_j, the length difference of the two path segments may be reduced, which is benefit for finding a feasible and good path fast.

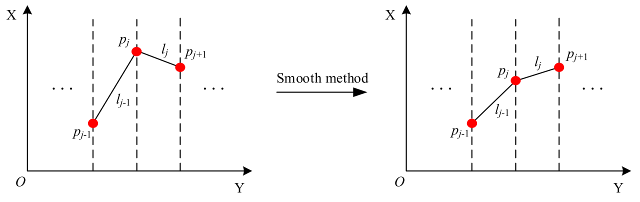


FIGURE 10. Procedure of smooth method.

Therefore, in the smooth method, for the j^{th} waypoints of the old solution (p_j), if its left path segment is longer than its right path segment, then p_j will be set as the average of p_{j-1} and p_j. Otherwise, the p_j will be set as the average of p_j and p_{j+1}. The smooth process is shown as

$$\begin{cases} p_j = \left(p_{j-1} + p_j\right)/2, & \text{if } l_{j1} > l_j \\ p_j = \left(p_j + p_{j+1}\right)/2, & \text{otherwise} \end{cases} \tag{15}$$

where l_{j-1} and l_j are the lengths of two left and right path segments of p_j, respectively.

In the 3-D environment, first, a grid is randomly selected from the histogram as shown in Fig. 11(a). Then we judge whether the grid is safe with the map information. As shown in Fig. 11(b), if the grid is intersect with safe area, then it will be regarded as a safe grid. Otherwise, a new grid will be randomly selected until found the safe one. After that, we use the boundary in X-axis and Z-axis of the select grid to generate new values in X-axis and Z-axis, respectively. When all the new dimensions are sampled, a new solution is formed. Similar to that in the 2-D environment, every T^{th} generation, LFHH uses the smooth method to generate new solutions. Then use Eq. (6) to calculate their fitness values. This sampling method can make a good use of map information and is helpful for finding safe waypoint in complex 3-D environment.

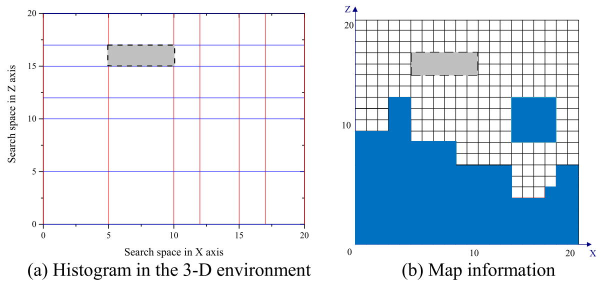


FIGURE 11. Sampling from histogram in 3-D environment.

D. GENERATION OF NEW POPULATION

After the generation of the new solutions, we firstly combine new solutions with old solutions. Then sort them from best to worst based on fitness values. After that, we select the top *NP* good solutions to form a new population. The procedure of LFHH is shown in **Algorithm 2**.

Algorithm 2 Procedure of LFHH

```
1:  Begin
2:     Initialize the population, and set up the start-point and
       target-point; $gen$=1;
3:     While (Termination criteria is not satisfied)
4:       /* Sampling phase by smooth method */
5:       If $gen\%T == 0$
6:          Use smooth method to generate $NP$ new solutions;
7:          Go to Step 22;
8:       End of If
9:       /* Model building */
10:      Select top $S$ good solutions based on fitness values;
11:      For each dimension, use Eq. (14) to construct
         $arrayX$;
12:      Sort the $arrayX$ from small to large;
13:      Use Eq. (1) and Eq. (2) to construct histograms;
14:      /* Sampling phase by probabilistic model */
15:      If environment is 2D spatial domain
16:         Randomly select bins from histograms;
17:         Sample new values to form new solutions;
18:      Else
19:         From the histogram grids select safe grids;
20:         Sample new values from bins corresponding to safe
         histogram grids to form new solutions;
21:      End of If
22:      Use Eq. (6) to calculate fitness value of solutions;
23:      /* Generating new population */
24:      Combine the old and new solutions to select top $NP$
         solutions as new population;
25:      $gen=gen+1$;
26:    End of While
27: End
```

E. HANDLING METHOD FOR ENVIRONMENT CHANGES

When the environment change occurs, AUV can use its sensors to detect the positions of obstacles. Besides, the ocean flows can be inferred with the help of onboard horizontal acoustic Doppler current profiler. Then the environment map can be updated according to the information acquired by sensors. Besides, the current position of AUV is recorded as the new start-point. The new target-point is set according to the planning window size, as shown in Fig. 12. The region with red boundary represents the planning window. That is, only the paths in planning window are re-planned. The waypoint of the global path which drops on the right boundary of the planning window is recorded as the new target-point.

Based on the updated environment map, a new path is planned by LFHH for AUV with the given planning window size. First, a new population is randomly initialized with the limitation of the planning window. Then LFHH performs its evolutionary procedures to find a feasible path. If LFHH does not find a feasible path in finite generations, then increase the planning window size by one and use LFHH to re-plan the path until finding a feasible path or the size of planning window cannot be increased. It should be pointed that the right boundary of the planning window should be less than or equal to the maximum size of environments. After LFHH completes path re-planning, the global path is updated by using the re-planned path in the planning window. Changing the planning window size can make the path planner more flexible and stable. It is helpful for providing more safe areas for LFHH to find feasible paths.

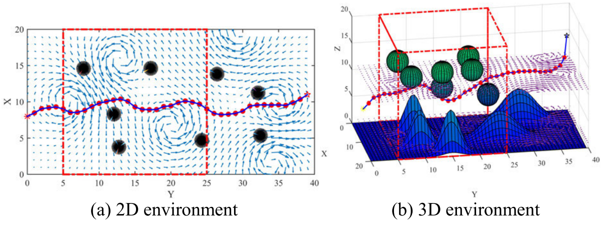


FIGURE 12. Planning window in different spatial domains.

F. PROCEDURES OF PATH PLANNING BY LFHH

Combining with the components mentioned above, the whole procedures of path planning by LFHH are shown as follows. First, use LFHH to plan a global path in the current environment. AUV will navigate along with the global path. When AUV detects the change of environment, LFHH will be used to plan a new path with the limitation of the planning window in the current environment. After that, the global path will be updated. Then AUV continues to navigate along with the global path. These steps will be looped until AUV arrives at the target-point. These procedures are shown as follows.

Step (1) Load the environment map.

Step (2) Use LFHH in Algorithm 2 to plan a global path.

Step (3) If there are no changes occur in the environment, AUV will still navigate along the global path. Otherwise, go to *Step (4)*.

Step (4) If the environment changes, Use LFHH to re-plan the path in the planning window. After that, use the new path to update the global path.

Step (5) If AUV arrives at the target-point, then the algorithm will stop. Otherwise, go to *Step(3)*.

V. EXPERIMENTS AND COMPARISONS

In this paper, both 2-D and 3-D environments are simulated in MATLAB. The 2-D environment is with the size of 20 km $\times$ 40 km, and the 3-D environment is with the size of 20 km $\times$ 40 km $\times$ 20 km. The 3-D environment has three scenarios with different shapes of seabed. Ocean flows are also taken into consideration in both 2-D and 3-D environments. All algorithms are implemented in C and executed in the computers of the same configurations, i.e., Intel(R) Core(TM) i7-7700 CPU 3.6GHz, 8GB RAM.

Dimensions of the two environments are respectively set as 40 corresponding to 40 waypoints of a path (as shown in Fig. 3 and Fig. 7(a)). The planning window size is initialized as 20. We assume that the environment will change when AUV navigates for per 6 km. In our experiment, the environment changes five times, resulting in six phases. In each phase, the maximum generation is set as 100 and 300 for all algorithms in the 2-D environment and 3-D environment, respectively. The average of cost function values corresponding to 5 changes are calculated as the final results. Besides, the values of each item in the fitness function and the time consumption are also recorded. Positions of obstacles follow the Gaussian distribution in 2-D environment, and in 3-D environment follow the uniform distribution. Vortices' center positions of ocean flows follow Gaussian distribution in both two spatial domains.

The proposed LFHH algorithm compared with six algorithms, i.e., FHH [40], FHH with LTS, EDA with Gaussian distribution termed as EDA-G [42], an adaptive DE algorithm [26], RRT* [15], and PRM* [15]. All evolutionary algorithms use the same population size $NP = 200$. For the LFHH algorithm, the learning factor λ and parameter T are set as 0.08 and 5, respectively. The good solutions used to construct histograms are selected with quantity $S = 100$. Bins' quantity *binNum* is set as 100. For the EDA-G, top S good solutions are selected to construct the Gaussian distribution model, where $S = 100$. Parameters α, β, and F_0 of the adaptive DE algorithm based path planner are set as 0, 0.4, and 0.6, respectively. For the details, please refer to [26]. For the RRT*, the maximum iteration is set as 300, and the radius centered in one tree node is set as 15. For the PRM*, the number of nodes *Nodes_Num* used to construct load map is set as 300, and the *rPRM* is set as 15, which is used to control the connection radius of PRM*. Besides, PRM* uses Dijkstra algorithm to find feasible path in the query phase in this paper. Parameters of these seven algorithms are shown in Table 1.

TABLE 1. **Parameter settings of the seven algorithms.**

Algorithms	Parameter settings
LFHH	$\lambda = 0.08$, $T = 5$, $binNum = 100$, $S = 100$, $NP = 200$
FHH	$binNum = 100$, $S = 100$, $NP = 200$
FHH with LTS	$\lambda = 0.08$, $binNum = 100$, $S = 100$, $NP = 200$
EDA-G	$S = 100$, $NP = 200$
Adaptive DE	$\alpha = 0$, $\beta = 0.4$, $F_0 = 0.6$, $NP = 200$
RRT*	*iteration*=300, *radius*=15
PRM*	*Nodes_Num*=300, *rPRM*=15

All the algorithms independently run 30 times, and the mean results are compared. The results in the bracket represent the standard variance, and the best results are marked in **boldface**. Besides, we also make the Wilcoxon's rank sum tests [43] with a significant level $\alpha = 0.05$. The symbols of '$+$', '$\approx$', and '$-$' means the LFHH performs significantly better than, similar to, and significantly worse than other algorithms.

A. EXPERIMENTAL RESULTS IN 2-D ENVIRONMENT

As shown in Table 2, LFHH algorithm can get feasible paths (as F_{cost} is less than 2.0) in all tests. Besides, it has lower time consumption in most tests with different start-points and target-points. FHH, FHH with LTS, EDA-G, adaptive DE and RRT* cannot find feasible paths in most tests. The PRM* can also find feasible paths in all scenarios. However, LFHH has a lower time consumption compared with PRM*. Besides, LFHH performs better than PRM*. Lower time consumption is important for path planner when planning paths in dynamic environments. From the results of FHH and FHH with LTS, we can see that the LTS can make FHH get smaller F_{cost}. The LTS can make individuals learn from the best individual during the evolutionary process. Therefore, FHH with LTS has a better performance compared with FHH. Besides, FHH with LTS has a lower time consumption compared with FHH, which indicates LTS can accelerate the convergence speed of FHH. From the results of LFHH and FHH with LTS, we can see that LFHH has a better accuracy and lower time consumption. That is because a smooth method is employed into LFHH to periodically correct paths which swerve sharply. This method can accelerate the speed of finding feasible paths. Therefore, LFHH can find a feasible path within the finite generations, and there is no need to change the planning window size frequently. If the size of planning window is changed frequently, the time consumption will be too large to be practical.

We select a group of results which is found by LFHH and plot the path in a 2-D environment. As shown in Fig. 13, Path in phase 1 is the global path. Paths in other phases with green color are the re-planned paths, and paths with blue color are the global paths. When the environment changes, the path can adaptively change as the changing of environment. Besides, only a part of old path is re-planned because of the planning window. If we re-plan the whole old path, LFHH will need more time to get new feasible path.

B. EXPERIMENTAL RESULTS IN 3-D ENVIRONMENT

For the 3-D environment, three different scenarios with different seabed shapes and obstacle positions are simulated. In each scenario, two groups of tests are implemented with different start-points and target-points. As shown in Table 3, LFHH can find feasible paths in all scenarios. FHH, FHH with LTS, EDA-G, adaptive DE, RRT*, and PRM* cannot find feasible paths in all scenarios. PRM* can find feasible paths in the 2-D environment. However, in the 3-D environment, its performance becomes worse compared that in the 2-D environment. LFHH has a better performance in both 2-D and 3-D environments. It is shown that the PRM* is less flexible than LFHH. From the results of FHH and FHH with LTS, we can see that the LTS makes FHH run faster. Meanwhile, the accuracy of FHH is also improved by LTS. This is because the LTS can make individuals in current generation learn from the best individual. From the results of LFHH and FHH with LTS, we can see that LFHH has a lower time consumption and higher accuracy. That means the smooth method employed by LFHH can accelerate the speed of finding feasible paths. Meanwhile, the time consumption is also cut down. As shown in Table 3, LFHH has the lowest time consumption in most cases, since LFHH can find

TABLE 2. Fitness function values of paths in the 2-d environment.

Start/Target		LFHH	FHH	FHH with LTS	EDA-G	Adaptive DE	RRT*	PRM*
S:(13,0)	F_{cost}	**6.60E-01(1.71E-01)**	4.56E+00(4.14E-01) (+)	3.69E+00(5.33E-01) (+)	2.07E+00(4.02E-01) (+)	5.66E+00(4.12E-01) (+)	2.04E+00(5.92E-01) (+)	1.04E+00(4.48E-01) (+)
T:(6,39)	Time	**1.907s**	11.038s	10.236s	25.114s	20.195s	2.752s	**1.482s**
S:(6,0)	F_{cost}	**6.92E-01(1.44E-01)**	4.12E+00(2.72E-01) (+)	3.71E+00(5.13E-01) (+)	1.77E+00(3.83E-01) (+)	5.56E+00(4.40E-01) (+)	2.43E+00(4.67E-01) (+)	8.99E-01(4.44E-01) (≈)
T:(8,39)	Time	**2.474s**	11.002s	10.247s	21.128s	19.913s	2.803s	**1.322s**
S:(5,0)	F_{cost}	**7.34E-01(1.75E-01)**	4.12E+00(2.99E-01) (+)	3.80E+00(5.46E-01) (+)	1.99E+00(2.36E-01) (+)	5.52E+00(4.30E-01) (+)	2.57E+00(5.30E-01) (+)	1.12E+00(4.41E-01) (+)
T:(12,39)	Time	**2.942s**	11.076s	10.195s	22.801s	20.441s	2.678s	**1.311s**

'+', '≈' and '–' mean the LFHH performs significantly better than, similar to, and significantly worse than other algorithms

TABLE 3. Fitness function values of paths in the 3-D environments.

Scenario	Start/Target		LFHH	FHH	FHH with LTS	EDA-G	Adaptive DE	RRT*	PRM*
1	S:(12,0,7)	F_{cost}	**6.62E-01(3.15E-02)**	3.85E+00(3.96E-01) (+)	3.22E+00(2.21E-01) (+)	2.89E+00(1.26E-01) (+)	6.27E+00(2.06E-01) (+)	3.07E+00(3.07E-01) (+)	2.59E+00(6.58E-01) (+)
	T:(3,39,15)	Time	**0.585s**	14.260s	12.306s	10.970s	17.332s	1.444s	1.269s
	S:(14,0,8)	F_{cost}	**6.56E-01(2.27E-02)**	3.86E+00(5.16E-01) (+)	3.29E+00(2.80E-01) (+)	2.84E+00(2.06E-01) (+)	6.22E+00(2.24E-01) (+)	3.35E+00(3.19E-01) (+)	2.78E+00(4.27E-01) (+)
	T:(5,39,13)	Time	**0.563s**	14.312s	12.329s	10.770s	17.446s	1.324s	1.089s
2	S:(13,0,7)	F_{cost}	**6.27E-01(8.30E-02)**	4.90E+00(4.37E-01) (+)	4.31E+00(4.77E-01) (+)	2.97E+00(2.70E-01) (+)	6.24E+00(2.52E-02) (+)	3.39E+00(2.87E-01) (+)	2.72E+00(5.95E-01) (+)
	T:(7,39,5)	Time	**2.572s**	14.019s	12.213s	11.223s	16.134s	1.486s	**1.061s**
	S:(12,0,7)	F_{cost}	**6.53E-01(8.27E-02)**	4.96E+00(3.57E-01) (+)	4.53E+00(4.49E-01) (+)	3.10E+00(3.22E-01) (+)	6.24E+00(2.56E-02) (+)	3.39E+00(3.54E-01) (+)	2.76E+00(5.82E-01) (+)
	T:(13,39,5)	Time	**1.090s**	13.954s	12.314s	11.286s	16.078s	1.539s	1.110s
3	S:(9,0,6)	F_{cost}	**7.10E-01(1.39E-01)**	4.62E+00(6.15E-01) (+)	3.46E+00(3.85E-01) (+)	2.89E+00(9.82E-02) (+)	6.37E+00(2.23E-02) (+)	3.16E+00(3.92E-01) (+)	2.66E+00(4.94E-01) (+)
	T:(15,39,4)	Time	**1.657s**	14.085s	12.407s	11.150s	16.765s	1.377s	**1.097s**
	S:(12,0,5)	F_{cost}	**7.34E-01(1.77E-01)**	4.51E+00(4.14E-01)	3.41E+00(4.18E-01) (+)	2.83E+00(9.67E-02) (+)	6.36E+00(8.33E-02) (+)	3.56E+00(2.62E-01) (+)	2.56E+00(5.93E-01) (+)
	T:(5,39,8)	Time	**1.355s**	14.090s	12.329s	11.082s	16.116s	1.341s	**1.153s**

'+', '≈' and '–' mean the LFHH performs significantly better than, similar to, and significantly worse than other algorithms

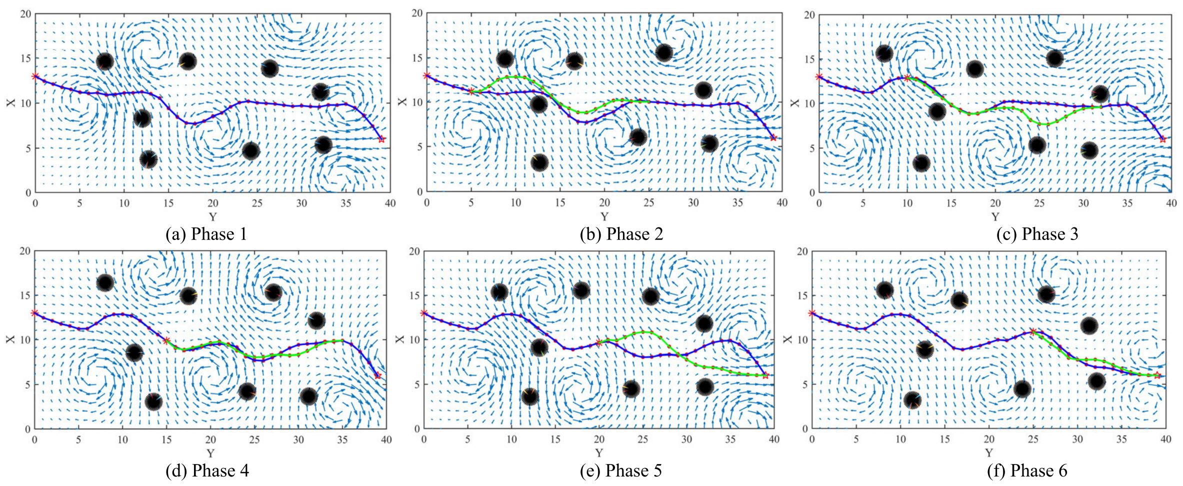


FIGURE 13. Path found by LFHH in the 2-D environment.

feasible paths within the finite iterations. Therefore, the size of planning window is changed only few times and this is helpful to cut down time consumption.

Similar to that in the 2-D environment, we also select a path found by LFHH and plot it in the 3-D environment, shown in Fig. 14. When the environment changes, the path can adaptively change as the changing of environment, since a part of path is re-planned due to the planning window.

C. EFFECTS OF DIFFERENT COMPONENTS IN LFHH

In LFHH, the LTS, smooth method, and planning window are the main components to plan path for AUV in dynamic environments. Herein, we make comparisons to test the influence of each component on the performance of LFHH.

1) THE EFFECTS OF LTS

In the 2-D environment, as shown in Table 4, LFHH and LFHH without LTS have the similar performance, however, LFHH has the lower time consumption in most cases. The LTS can make individuals learn from the best individual in the current population. This strategy can improve the convergence speed of algorithm. Therefore, the LFHH has a lower time consumption. In the 3-D environment, as shown in Table 5, the performance of LFHH without LTS becomes worse, especially in scenario 1 and scenario 3. Besides, LFHH has a lower time consumption compared with LFHH without LTS. Therefore, LFHH can also effectively handle path planning problem even the environment becomes more complex.

2) THE EFFECTS OF SMOOTH METHOD

In the 2-D environment, from the results of LFHH and LFHH without smooth method in Table 4, we can see that the LFHH has a better performance and lower time consumption. During the evolutionary process, the smooth method can periodically

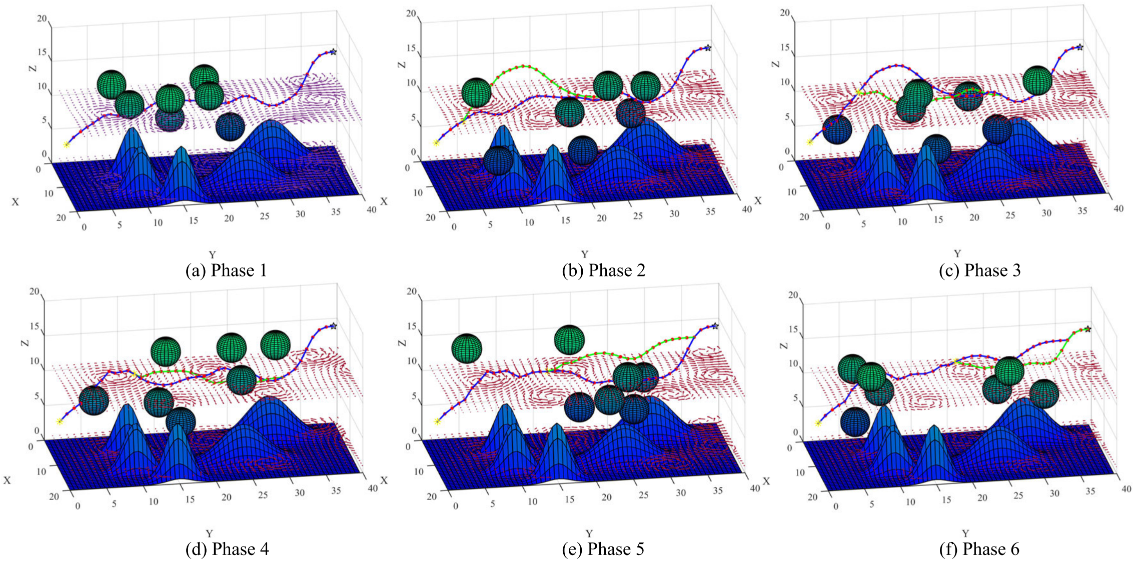


FIGURE 14. **Path found by LFHH in the 3-D environment.**

TABLE 4. **Components comparisons of LFHH in the 2-D environments.**

Start/Target		LFHH	LFHH without LTS	LFHH without smooth method	LFHH without planning window
S:(13,0)	F_{cost}	6.60E-01(1.71E-01)	6.30E-01(7.37E-02) **(≈)**	3.09E+00(2.41E-01) (+)	**5.78E-01(7.62E-02)** (-)
T:(6,39)	Time	1.907s	**1.812s**	10.391s	2.368s
S:(6,0)	F_{cost}	6.92E-01(1.44E-01)	7.04E-01(1.68E-01) (≈)	3.12E+00(2.22E-01) (+)	**6.74E-01(2.05E-01)** (≈)
T:(8,39)	Time	**2.474s**	2.929s	10.443s	3.337s
S:(5,0)	F_{cost}	7.34E-01(1.75E-01)	**6.56E-01(7.58E-02)** (≈)	3.06E+00(1.30E-01) (+)	7.25E-01(1.66E-01) (≈)
T:(12,39)	Time	**2.942s**	3.243s	10.406s	3.149s

TABLE 5. **Components comparisons of LFHH in the 3-D environments.**

Scenario	Start/Target		LFHH	LFHH without LTS	LFHH without smooth method	LFHH without planning window
1	S:(12,0,7)	F_{cost}	6.62E-01(3.15E-02)	6.52E-01(3.14E-02) (≈)	2.95E+00(9.20E-02) (+)	**6.45E-01(3.26E-02)** (≈)
	T:(3,39,15)	Time	0.585s	**0.583s**	14.320s	0.852s
	S:(14,0,8)	F_{cost}	6.56E-01(2.27E-02)	6.84E-01(3.09E-02) (+)	2.93E+00(4.51E-02) (+)	**6.04E-01(2.37E-02)** (-)
	T:(5,39,13)	Time	**0.563s**	0.571s	14.362s	0.900s
2	S:(13,0,7)	F_{cost}	6.27E-01(8.30E-02)	6.54E-01(1.08E-01) (≈)	2.94E+00(1.42E-01) (+)	**6.17E-01(1.09E-01)** (-)
	T:(7,39,5)	Time	2.572s	**2.256s**	14.242s	2.321s
	S:(12,0,7)	F_{cost}	**6.53E-01(8.27E-02)**	6.54E-01(3.57E-02) (≈)	3.14E+00(2.90E-01) (+)	7.14E-01(1.72E-01) (≈)
	T:(13,39,5)	Time	1.090s	**0.893s**	14.289s	2.725s
3	S:(9,0,6)	F_{cost}	**7.10E-01(1.39E-01)**	7.81E-01(1.89E-01) (≈)	2.96E+00(1.30E-01) (+)	9.60E-01(2.28E-01) (+)
	T:(15,39,4)	Time	**1.657s**	1.939s	14.203s	3.951s
	S:(12,0,5)	F_{cost}	7.34E-01(1.77E-01)	8.42E-01(2.14E-01) (+)	2.91E+00(1.26E-01) (+)	**7.28E-01(1.76E-01)** (≈)
	T:(5,39,8)	Time	**1.355s**	1.951s	14.287s	2.861s

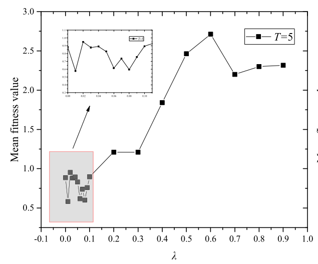
correct paths which swerve sharply. This procedure can accelerate the speed of finding feasible paths. In the 3-D environment, as shown in Table 5, LFHH has a better performance and lower time consumption compared with LFHH without smooth method. The smooth method can also correct paths in the 3-D environment likewise. Therefore, the smooth method can accelerate the speed of finding feasible paths in both 2-D and 3-D environments. The LFHH without smooth method has a larger time consumption compared with LFHH. That is because without smooth method LFHH cannot find feasible path in current planning window, therefore, LFHH will frequently change the planning window in order to find feasible paths. If the planning window is frequently changed, the time consumption will become higher.

3) THE EFFECTS OF PLANNING WINDOW

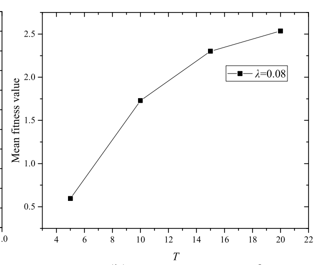
From the results in Table 4 and Table 5, we can see that LFHH and LFHH without planning window have similar performances in both 2-D and 3-D environments. However, LFHH has a lower time consumption in both 2-D and 3-D environments. That is because when the environment changes, LFHH without planning window will re-plan the rest of global path. Conversely, LFHH just needs to re-plan a part of global path which is in the planning window. Therefore, LFHH needs less time to re-plan path and it has a lower time consumption compared with LFHH without planning window. Lower time consumption is important for planning path for AUV in dynamic environment and can guarantee the real-time ability in some degree.

D. PARAMETERS TEST OF LFHH

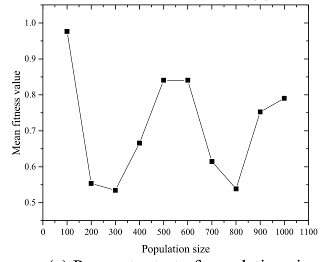
In LFHH, parameter λ is the learning rate of LTS. Smaller λ means individuals have smaller steps of moving toward the best individual, while larger λ means individuals have larger steps of moving toward the best individual. Parameter T can control the rate of performing the smooth method. In this



(a) Parameter test of λ

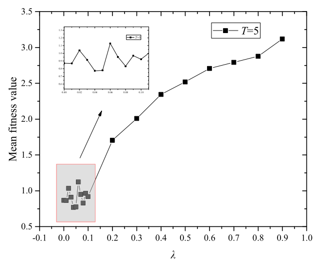


(b) Parameter test of T

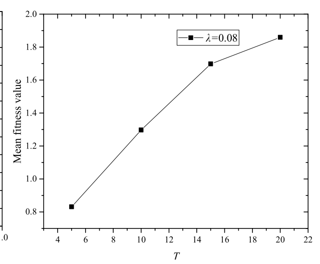


(c) Parameter test of population size

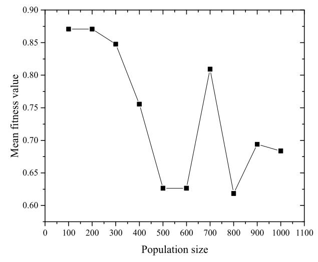
FIGURE 15. Parameters test in 2-D spatial domain.



(a) Parameter test of λ



(b) Parameter test of T



(c) Parameter test of population size

FIGURE 16. Parameters test in 3-D spatial domain.

situation, we take an investigation to the parameters λ and T. We implement this investigation in both 2-D and 3-D spatial domains. Besides, we also investigate the effects of population size. We set the parameter λ from 0 to 0.09 with a step length of 0.01, and from 0.1 to 1 with a step length of 0.1. For parameter T, we set it from 5 to 20 with a step length of 5. For the population size, we set it from 100 to 1000 with a step length of 100. For each pair of parameters, we independently run the algorithm for 30 times to get the results.

1) PARAMETERS IN 2-D SPATIAL DOMAIN

As shown in Fig. 15(a), when λ increases from 0.1 to 0.9 and $T = 5$, the curve has an upward tendency, measuring the performance being worse When λ increases from 0.01 to 0.09 and T=5, the curve has a fluctuant tendency, measuring the performance being fluctuant. When λ = 0.08, the LFHH has the best performance. As shown in Fig. 15(b), when T increases from 5 to 20 and λ = 0.08, the curve has an upward tendency and the performance of LFHH becomes worse. Therefore, when λ = 0.08 and $T = 5$ LFHH has the best performance. As shown in Fig. 15(c), when the population size is set as 200, 300, and 800, LFHH can get a better performance. However, larger population size will involve heavy time consumption. Therefore, the population size is set as 200 in this paper.

2) PARAMETERS IN 3-D SPATIAL DOMAIN

As shown in Fig. 16(a), when λ increases from 0.1 to 0.9 and $T = 5$, the curve has an upward tendency and the performance of LFHH becomes worse. When λ drops in [0.0, 0.1], LFHH has a relatively better performance. Besides, when λ increases in [0.0, 0.1], the curve has a fluctuant tendency. When λ = 0.08, the LFHH has the best performance. As shown in Fig. 16(b), when T increases from 5 to 20 and λ = 0.08, the curve has an upward tendency. The larger the T is, the worse the LFHH is. Therefore, $T = 5$ and λ = 0.08 is the best parameter combination of the LFHH. As shown in Fig. 16(c), when the population size becomes larger, the performance of LFHH becomes better. However, larger population size is not good for cutting down the time consumption. In this paper, to make the LFHH more convenient to be adopted for both 2-D and 3-D environments, the population size is set as 200.

VI. CONCLUSION

In this paper, we propose an improved histogram-based EDA variant termed as LFHH to handle path planning problems for AUV in 2-D and 3-D environments with dynamic factors. With the help of learning transformation method, the accuracy and convergence speed of LFHH is highly improved. Besides, the smooth method makes LFHH find feasible paths fast. Moreover, we use a planning window to handle the dynamic factors. We compare LFHH with other algorithms in both 2-D and 3-D environments, and make sure they can find similar number of waypoints of a path. As the experimental results shown, LFHH has a stable performance to complete path planning for AUV in both 2-D and 3-D environments. It is expected that LFHH can adopt distributed technology or parallel technology to make further efforts to cut down the time consumption, and applications of LFHH can be implemented in other optimization problems. Besides, it is also expected that adopting LFHH in a real AUV to test its performance in real world environments. Firstly, we plan to implement our algorithm on a smaller AUV and then test it in a pool in laboratory. Moving obstacles will be set on the floor of the pool. Ocean flows can be simulated by the wave generator. Then we can observe the performance of the AUV to check out the availability of our algorithm. At last, we plan to implement our algorithm on a normal AUV then test it in seas.

REFERENCES

[1] J. J. Leonard and A. Bahr, "Autonomous underwater vehicle navigation," in *Springer Handbook of Ocean Engineering*. 2016, pp. 341–358.

[2] S.-W. Huang, E. Chen, and J. Guo, "Efficient seafloor classification and submarine cable route design using an autonomous underwater vehicle," *IEEE J. Ocean. Eng.*, vol. 43, no. 1, pp. 7–18, Jan. 2018.

[3] Y. Zhang, B. Kieft, M. J. Stanway, R. S. McEwen, B. W. Hobson, J. G. Bellingham, J. P. Ryan, T. C. O. Reilly, B. Y. Raanan, and M. Messie, "Isotherm tracking by an autonomous underwater vehicle in drift mode," *IEEE J. Ocean. Eng.*, vol. 42, no. 4, pp. 808–817, Oct. 2017.

[4] W. Li, J. A. Farrell, S. Pang, and R. M. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 292–307, Apr. 2006.

[5] H. Zheng, N. Wang, and J. Wu, "Minimizing deep sea data collection delay with autonomous underwater vehicles," *J. Parallel Distrib. Comput.*, vol. 104, pp. 99–113, Jul. 2017.

[6] G. Best, W. Martens, and R. Fitch, "Path planning with spatiotemporal optimal stopping for stochastic mission monitoring," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 629–646, Jun. 2017.

[7] Y. Lin, J. Hsiung, R. Piersall, C. White, C. G. Lowe, and C. M. Clark, "A multi-autonomous underwater vehicle system for autonomous tracking of marine life," *J. Field Robot.*, vol. 34, no. 4, pp. 757–774, 2017.

[8] C. Yu, X. Xiang, L. Lapierre, and Q. Zhang, "Robust magnetic tracking of subsea cable by AUV in the presence of sensor noise and ocean currents," *IEEE J. Ocean. Eng.*, vol. 43, no. 2, pp. 311–322, Apr. 2018.

[9] Z. Zeng, L. Lian, K. Sammut, F. He, Y. Tang, and A. Lammas, "A survey on path planning for persistent autonomy of autonomous underwater vehicles," *Ocean Eng.*, vol. 110, pp. 303–313, Dec. 2015.

[10] Z. Wang, X. Xiang, J. Yang, and S. Yang, "Composite Astar and B-spline algorithm for path planning of autonomous underwater vehicle," in *Proc. IEEE 7th Int. Conf. Underwater Syst. Technol. Theory Appl. (USYS)*, Dec. 2017, pp. 1–6.

[11] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Going with the flow: A graph based approach to optimal path planning in general flows," *Auto. Robots*, vol. 42, no. 7, pp. 1369–1387, 2018.

[12] M. Soulignac, "Feasible and optimal path planning in strong current fields," *IEEE Trans. Robot.*, vol. 27, no. 1, pp. 89–98, Feb. 2011.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[14] H. Saoud, M. Hua, F. Plumet, and F. B. Amar, "Routing and course control of an autonomous sailboat," in *Proc. Eur. Conf. Mobile Robots*, Sep. 2015, pp. 1–6.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[16] R. Cui, Y. Li, and W. Yan, "Mutual information-based multi-AUV path planning for scalar field sampling using multidimensional RRT," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 7, pp. 993–1004, Jul. 2016.

[17] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 331–341, Apr. 2007.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, pp. 500–505.

[19] C. Cheng, D. Zhu, B. Sun, Z. Chu, J. Nie, and S. Zhang, "Path planning for autonomous underwater vehicle based on artificial potential field and velocity synthesis," in *Proc. IEEE Can. Conf. Electr. Comput. Eng.*, May 2015, pp. 717–721.

[20] S. Saravanakumar and T. Asokan, "Multipoint potential field method for path planning of autonomous underwater vehicles in 3D space," *Intell. Service Robot.*, vol. 6, no. 4, pp. 211–224, 2013.

[21] E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "A waypoint guidance strategy for underwater snake robots," in *Proc. Medit. Conf. Control Autom.*, 2014, pp. 1512–1519.

[22] H. Saoud, F. Plumet, and F. Ben Amar, "Adaptive sampling with a fleet of autonomous sailing boats using artificial potential fields," in *Marine Robotics and Applications*. 2018, pp. 15–27.

[23] W. Chen, B. Xin, and Z. Zhu, "Real-time collision avoidance for unmanned ships under constraints of rules," in *Proc. IEEE 16th Int. Conf. Netw. Sens. Control (ICNSC)*, May 2019, pp. 80–85.

[24] Z. Yan, J. He, and J. Li, "An improved multi-AUV patrol path planning method," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Aug. 2017, pp. 1930–1936.

[25] C.-T. Cheng, K. Fallahi, H. Leung, and K. T. Chi, "A genetic algorithm-inspired UUV path planner based on dynamic programming," *IEEE Trans. Syst. Man, Cybern. C, (Appl. Rev.)*, vol. 42, no. 6, pp. 1128–1134, Nov. 2012.

[26] C.-B. Zhang, Y.-J. Gong, J.-J. Li, and Y. Lin, "Automatic path planning for autonomous underwater vehicles based on an adaptive differential evolution," in *Proc. Genetic Evol. Comput. Conf.*, 2014, pp. 89–96.

[27] S. Mahmoudzadeh, D. W. Powers, A. Yazdani, K. Sammut, and A. Atyabi, "Efficient AUV path planning in time-variant underwater environment using differential evolution algorithm," *J. Marine Sci. Appl.*, vol. 17, pp. 1–7, Sep. 2018.

[28] W. Zhou, Z. Xing, B. Wenbin, D. Chengchen, Y. Xie, and X. Wu, "Route planning algorithm for autonomous underwater vehicles based on the hybrid of particle swarm optimization algorithm and radial basis function," *Trans. Inst. Meas. Control*, vol. 41, no. 4, pp. 942–953, 2019.

[29] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Proc. Int. Conf. Parallel Problem Solving From Nature*, 1996, pp. 178–187.

[30] E. Bengoetxea, P. Larrañaga, I. Bloch, and A. Perchant, "Estimation of distribution algorithms," *Genetic Algorithms Evol. Comput.*, vol. 64, pp. 1140–1148, 2001.

[31] W. Shi, W.-N. Chen, Y. Lin, T. Gu, S. Kwong, and J. Zhang, "An adaptive estimation of distribution algorithm for multi-policy insurance investment planning," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 1–14, Feb. 2019.

[32] J. Wang, K. Tang, J. A. Lozano, and X. Yao, "Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 96–109, Feb. 2016.

[33] S. Jiang, M. Liu, and J. Hao, "A two-phase soft optimization method for the uncertain scheduling problem in the steelmaking industry," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 416–431, Mar. 2017.

[34] H. Lim, C. C. Mi, and W. Su, "A distance-based two-stage ecological driving system using an estimation of distribution algorithm and model predictive control," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6663–6675, Aug. 2017.

[35] B. Garau, A. Alvarez, and G. Oliver, "AUV navigation through turbulent ocean environments supported by onboard H-ADCP," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2006, pp. 3556–3561.

[36] Z. J. Wang, Z. H. Zhan, and J. Zhang, "Solving the energy efficient coverage problem in wireless sensor networks: A distributed genetic algorithm approach with hierarchical fitness evaluation," *Energies*, vol. 11, no. 12, p. 3526, 2018.

[37] Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H.-Q. Yuan, T.-L. Gu, S. Kwong, and J. Zhang, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[38] Z. J. Wang, Z. H. Zhan, Y. Lin, W. J. Yu, H. Wang, S. Kwong, and J. Zhang, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2019.2910721.2019.

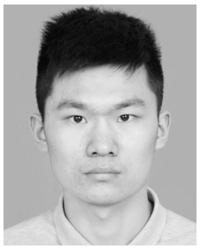
[39] Z. Zeng, A. Lammas, K. Sammut, and F. He, "Optimal path planning based on annular space decomposition for AUVs operating in a variable environment," in *Proc. IEEE/OES Auton. Underwater Vehicles (AUV)*, Sep. 2012, pp. 1–9.

[40] S. Tsutsui, M. Pelikan, and D. E. Goldberg, "Evolutionary algorithm using marginal histogram models in continuous domain," *IlliGAL Rep.*, vol. 2001019, no. 999, p. 1050, 2001.

[41] J.-H. Zhong, J. Zhang, and Z. Fan, "MP-EDA: A robust estimation of distribution algorithm with multiple probabilistic models for global continuous optimization," in *Proc. Int. Conf. Simulated Evol. Learn.*, 2010, pp. 85–94.

[42] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Proc. Int. Conf. Parallel Problem Solving From Nature*, 1998, pp. 418–427.

[43] W. Haynes, *Wilcoxon Rank Sum Test*, New York, NY, USA: Springer, 2013.



RUN-DONG LIU (S'17) received the B.S. degree in electronic and information engineering from Xinjiang University, Urumqi, China, in 2016. He is currently pursuing the master's degree with the School of Computer Science and Engineering, South China University of Technology.

His research interests include artificial intelligence, evolutionary computation algorithms, swarm intelligence, and their applications in real-world problems.

ZONG-GAN CHEN (S'17) received the B.S. degree from Sun Yat-Sen University, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in computer science and technology with the South China University of Technology, Guangzhou.

His current research interests include ant colony optimization, differential evolution, and their applications in real-world optimization problems.

ZI-JIA WANG (S'15) received the B.S. degree in automation from Sun Yat-Sen University, Guangzhou, China, in 2015, where he is currently pursuing the Ph.D. degree.

His current research interests include evolutionary computation algorithms like differential evolution, particle swarm optimization, and their applications in design and optimization, such as cloud computing resources scheduling.

ZHI-HUI ZHAN (M'130–SM'18) received the bachelor's degree and the Ph.D. degree in computer science from Sun Yat-Sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-Sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundations of China (NSFC), in 2018, and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence, in 2017. His doctoral dissertation received the China Computer Federation (CCF) Outstanding Ph.D. Dissertation and the IEEE Computational Intelligence Society (CIS) Outstanding Ph.D. Dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of *Neurocomputing* and *International Journal of Swarm Intelligence Research*.