Estimation of Distribution Algorithm for Autonomous Underwater Vehicles Path Planning

Run-Dong Liu, Zhi-Hui Zhan(✉), Wei-Neng Chen, Zhiwen Yu, and Jun Zhang(✉)

Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
zhanapollo@163.com

Abstract. Path planning is that given the start point and target point, finding out a shortest or smallest cost path. In recent years, more and more researchers use evolutionary algorithms (EAs) to solve path planning problems, such as genetic algorithms (GAs) and particle swarm optimization (PSO). Estimation of distribution algorithms (EDAs) belong to a kind of EAs that can make good use of global statistic information of the population. However, EDAs are seldom used to solve path planning problems. In this paper, we propose an EDA variant named adaptive fixed-height histogram (AFHH) algorithm to make path planning for autonomous underwater vehicles (AUVs). The proposed AFHH algorithm can adaptively shrink its search space to make good use of computational resource. We use a regenerate approach to avoid getting stuck in local optimum. We also measure the ability of fixed-height histogram (FHH) algorithm for path planning. We simulate a 3-D environment to measure the ability of the proposed AFHH algorithm. The results show that AFHH has a good convergence rate and can also get better performance.

Keywords: Estimation of distribution algorithm (EDA) · Autonomous underwater vehicles (AUVs) · Path planning · Adaptive fixed-height histogram (AFHH)

1 Introduction

Path planning is one of the most significant methods to make some robots more intelligent such as unmanned vehicles. Autonomous underwater vehicles (AUVs) are a kind of underwater robots [1]. They can work underwater according to humans' predefined commands. During the working period, they don't need to communicate with humans. Therefore, in some dangerous underwater environment such as contaminated areas, they can complete missions instead of humans [2]. Due to the widely application of AUVs, path planning for AUVs has become a very significant research topic. Although path planning problems (PPP) have been widely studied in the literatures, many of them don't need to consider too many conditions. For example, the traveling salesman problem (TSP) which is also a kind of PPPs, just needs to find a shortest path without considering complex environment [3].

According to study in [4], path planning algorithm can be divided into two groups. One is the resolution complete algorithms, and the other one is the probabilistic resolution complete algorithms. The A* approach is resolution complete algorithm. Garau *et al*. [5] used A* approach to make path planning for AUVs in complex environment. EAs are a kind of probabilistic resolution complete algorithms. In recent years, many researchers use EAs to solve some kinds of PPPs. Shih *et al*. [6] proposed a genetic-based effective approach to solve the PPP for AUVs. Zhang *et al*. [7] used an adaptive differential evolution (ADE) algorithm to make path planning for AUVs.

In this paper, we propose an adaptive fixed-height histogram (AFHH) algorithm to make path planning for AUVs in 3-D environment. The fixed-height histogram (FHH) algorithm is a kind of EDAs. It is also a kind of EAs. The proposed AFHH algorithm can adaptively shrink search space in the course of evolution. We randomly generate new individuals with the current search space every T generations to make the algorithm avoid getting stuck in local optimum. Furthermore, we use a 3-D environment model to check out whether AFHH has a good path planning ability.

The rest of this paper is organized as follows. Section 2 introduces the background knowledge of EDAs and environment modeling of AUVs path planning. Section 3 describes the proposed AFHH algorithm for path planning in detail. Section 4 presents results of experiments. Section 5 makes a conclusion of this paper.

2 Background Knowledge

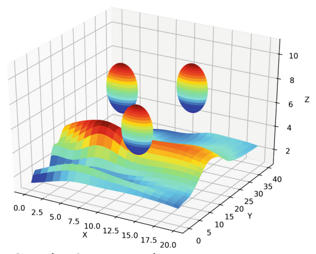
2.1 Environment Modeling for AUVs Path Planning

To make path planning for AUVs, the first step is constructing an environment model. Environment underwater in some area is very complex. The shape of seabed is not smooth. It maybe changes sharply in different areas. Besides, some floating obstacles make the environment more complex. In this paper, we use a 3-D static environment which is constructed in [7] to model the PPP for AUVs.

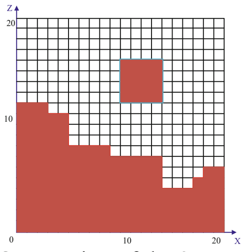
Herein, we consider the path planning in 3-D space with length, width, and depth. As shown in Fig. 1(a), the X axis, Y axis, and Z axis represent the environment's length, width and depth respectively. In Fig. 1(a), spheres represent floating obstacles, and the others represent seabed.

The 3-D environment model is split to 40 parallel sections along the y axis. The section is a profile of the 3-D environment. The section is shown in Fig. 1(b). The red areas represent obstacles. Each section is formed with a 20×20 grid. Besides, a $40 \times 20 \times 20$ array is used to save all sections. If there are obstacles in some areas, then the corresponding grids will be set as 1. Otherwise, they will be set as 0.

In each section, a feasible waypoint will be found. The trajectory is formed with 40 waypoints which are respectively generated in 40 sections. After that, we use a cost function to evaluate the path's quality according to its length, variation of depth, turning radius, and safety. The cost function is defined as:



(a) Static 3-D environment



(b) One section of the 3-D environment

Fig. 1. The environment models of path planning for AUVs.

$$F_{cost} = C_{length} + C_{depth} + C_{smooth} + C_{block} \tag{1}$$

where C_{length} represents the length of a path. C_{depth} represents the variation of a path's depth. C_{smooth} represents a path's smooth degree. C_{block} represents the safety of a path. C_{length} is defined as:

$$C_{length} = 1 - \frac{L_{PsPt}}{L_{path}}, C_{length} \in (0, 1) \tag{2}$$

where L_{PsPt} is the Euclidean distance between start point and stop point. L_{path} is the length of the whole path. C_{depth} is defined as:

$$C_{depth} = \frac{D_{max} - D_{min}}{D}, C_{depth} \in (0, 1) \tag{3}$$

where D_{max} is the max depth of the path, and D_{min} is the min depth of the path. D is the max depth of the 3-D environment model. C_{smooth} is defined as:

$$C_{smooth} = \begin{cases} 2 + \frac{N_{unable}}{N_{segment}}, & N_{unable} > 0 \\ 0, & N_{unable} = 0 \end{cases}, C_{smooth} \in 0 \cup (2, 3) \tag{4}$$

where N_{unable} is the number of path segments whose turning radius beyond the motion ability of AUVs. If the length of one path segment larger than a predefined value θ, then N_{unable} will increase. We set θ as 1.75. $N_{segment}$ is the number of all path segments. C_{block} is defined as:

$$C_{block} = \begin{cases} 2 + \frac{N_{block}}{N_{space}}, & N_{block} > 0 \\ 0, & N_{block} = 0 \end{cases}, C_{block} \in 0 \cup (2, 3) \tag{5}$$

where N_{block} is the number of unfeasible waypoints. N_{space} is the number of sections.

2.2 Estimation of Distribution Algorithm

EDA is a kind of stochastic algorithms which belongs to EAs. It performs better than GA in most case because EDA makes a good use of the global statistics. EDA's procedure is similar to GA. However, EDA doesn't have mutation and crossover procedures. Tsutsui *et al.* [8] proposed two histogram based EDAs, one is the fixed-width histogram (FWH) algorithm, and the other one is the FHH algorithm. Xiao *et al.* [10] proposed a histogram-based EDA which is used for processing continuous optimization problems. Yang *et al.* [11] proposed an improved EDA which can process multimodal problems.

3 Proposed AFHH Algorithm

In this section, we describe the proposed AFHH algorithm for path planning in detail. We first choose the organization of individual's structure, which is proposed in [7]. As shown in Eq. (6), the individual's structure is made of waypoints. Each waypoint is a tuple which contains 3-D positional information. The X axis and Z axis values are generated in each section. The Y axis values are known. That is, $y_j = j$, where j is the index of sections. The x and z values are updated with the same procedure.

$$individual = \begin{bmatrix} x_1, x_2, \ldots, x_{Dim} \\ y_1, y_2, \ldots, y_{Dim} \\ z_1, z_2, \ldots, z_{Dim} \end{bmatrix} \tag{6}$$

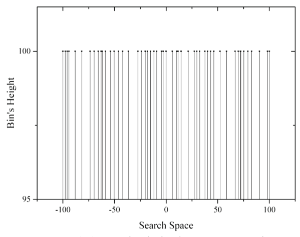
In traditional FHH algorithm's model building step, search space of each variable is divided into K bins. All bins have the same height. Each bin's width will change in each generation. Let $X_i = \{x_1, x_2, \ldots, x_{Dim}\}$, where Dim is the number of variables, i is the index of individuals. First, select S best individuals and sort them according to the value of the j^{th} variable. Then, use Eqs. (7) and (8) to update bins.

$$Bin\_lower_m = \begin{cases} lower, & \text{if } m = 0 \\ (x_{m \cdot S/K} + x_{m \cdot S/K-1})/2, & \text{otherwise} \end{cases} \tag{7}$$

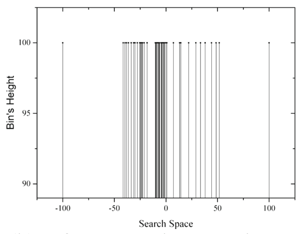
$$Bin\_upper_m = \begin{cases} Bin\_lower_{m+1}, & \text{if } m < K-1 \\ upper, & \text{otherwise} \end{cases} \tag{8}$$

where Bin_lower_m is the lower value of the m^{th} bin, Bin_upper_m is the upper value of the m^{th} bin, *lower* and *upper* represent search space's lower bound and upper bound respectively.

When sampling new individuals, each variable will select a bin randomly. Then use the bin's lower bound and upper bound to generate a value randomly. As shown in Fig. 2, during evolution, the bin's width is continuously changing. Figure 2(b) shows that the first bin and the last bin have larger width than others. It is little help for finding global optimum when using them to generate new individuals.



(a) Ininitial generation



(b) After several generations

Fig. 2. Bin's distribution in search space of one variable in different generations

In AFHH's model building step, all individuals are sorted according to the value of the j^{th} variable. Then record the smallest value and the largest value of the j^{th} variable with *intervalLower*[*j*] and *intervalUpper*[*j*] respectively, where *intervalLower* and *intervalUpper* are arrays whose length is *Dim*. After that, if AFHH finds a feasible path, which means the cost function's value is less than 2.0, parameter p_m which is defined as Eq. (9) will be used to control whether to change search space's size of the j^{th} variable [9]. Then, AFHH will use Eqs. (7) and (8) to update current bins.

$$p_m = 0.01 + 0.99 \cdot \frac{\exp\left(\frac{10 \cdot g}{G}\right) - 1}{\exp(10) - 1} \in [0.01, 1.00] \tag{9}$$

where g is the current generation, G is the maximum iteration. p_m is a nonlinear parameter. When change the search space's size, we use the mean of *intervalLower* [*j*] and Bin_lower_0 as search space's lower bound, and use the mean of *intervalUpper* [*j*] and Bin_upper_{k-1} as search space's upper bound. The procedure of updating bins and sampling are the same as traditional FHH algorithm. It is illustrated in Fig. 3.

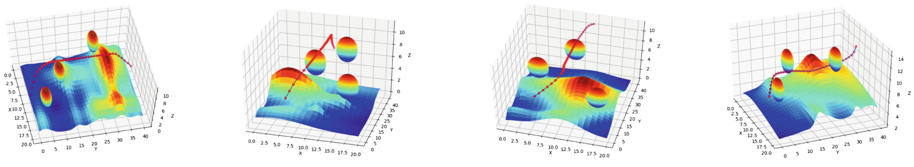
4 Experiments and Results

4.1 Experiments Setup

The real underwater environment is simulated like the ones in Fig. 4. The area is 20 m long, 40 m width and 20 m depth. Four scenarios are used to measure AFHH's performance. AFHH, FHH, PSO, enhance differential evolution with random walk (RWDE) [12], and ADE [7] are used to make a comparison of their ability on path planning [7]. For the FHH and AFHH algorithms, bins size K is set as 50, and S is set as 100. The PSO is a conventional global topology PSO variant, whose w is set as 0.5. c_1 and c_2 are both set as 2.0. For the RWDE algorithm, two pairs of fixed parameters are used (F = 0.5, CR = 0.1; F = 0.1, CR = 0.5). All algorithms use the same maximum function evaluations (FEs) 5000000, and the same population's size 200. All results are got after 30 times independently run. All algorithms are implemented on a PC with Intel core i7 processor running 3.6 GHz, RAM of 8 GB.

```
Procedure of AFHH's Model Building
Begin
 Load current scenario's map;
 Initialize the population;
 While (Not Stop)
   //model building
   While ($j < Dim$)
    Find the biggest and smallest values from $j^{th}$ variables;
    $intervalUpper[j]$ = biggest value;   $intervalLower[j]$ = smallest value;
    Sort $S$ best individuals' $j^{th}$ variable from small to large;
    //Decide whether to shrink the search space of $j^{th}$ variable
    If ((current best fitness value < 2.0) and (random(0,1)<$p_m$))
       $lower[j] = intervalLower[j]$;   $upper[j] = intervalUpper[j]$;
    End of If
    Use Eq. (7) and Eq. (8) to update $j^{th}$ variable's bins;
   End of While
   //generate New Individuals
   If (($gen\%T$ == 0) and ($gen$ != 0))
     Use current search space randomly generate new individuals;
   Else
     While ($i < NP$)
        Randomly select a bin for each variable to generate new variable;
        Use new variables to form the $i^{th}$ individual;
        Calculate the $i^{th}$ individual's cost function's value;
     End of While
   End of If
   Select $NP$ best individuals to form new population;
 End of While
End
```

Fig. 3. Procedure of the proposed AFHH algorithm for path planning



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 4. The 3-D static environment models we simulate.

4.2 Results Comparisons

In Table 1, the first column represents different scenario, the second column fills with start points and target points. The mean and standard deviation of 30 independently runs of a path are calculated. Values in brackets are standard deviation. Results in Table 1 show that the proposed AFHH algorithm performs better than other algorithms in most cases. Besides, it is steadier in all independent runs. If a path is feasible, the value of cost function will be less than 2.0, and the smaller the better. As shown in

Table 2, although the AFHH algorithm can obtain high quality paths, it runs slowest in all scenarios. Because AFHH needs more time to construct probability model. PSO runs fastest in all scenarios, but it can't obtain feasible paths.

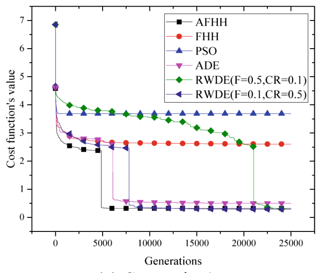
As shown in Fig. 5, AFHH converge faster than other algorithms in all scenarios. It has steady performance in all scenarios. Besides, AFHH not only converges faster, but also gets better results. The FHH can also get feasible paths, but it needs more iterations. Moreover, it is not steady sometimes. PSO performs worst, it can't find feasible path in all scenarios. The ADE can find feasible path. However, it needs more generations. Moreover, it is not steady sometimes. The RWDE with different parameters can also get good results in all scenarios, but it needs more iterations.

Table 1. Experimental results on AFHH, FHH, PSO, RWDE, and ADE with different scenarios

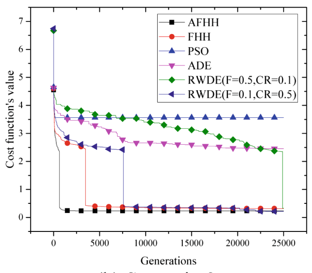
Scenario	Start point Target point	Cost function value					
		AFHH	FHH	PSO	RWDE (F = 0.5, CR = 0.1)	RWDE (F = 0.1, CR = 0.5)	ADE
1	S:(12,0,4) T:(10,39,6)	**0.33(6.18E−02)**	1.00(0.98)	3.56(0.29)	0.56(0.69)	0.39(9.25E−02)	0.94(1.10)
	S:(11,0,4) T:(3,39,6)	**0.32(3.47E−02)**	2.84(0.36)	3.29(0.32)	0.80(0.89)	0.40(0.14)	2.10(0.85)
	S:(9,0,6) T:(13,39,8)	0.25(4.75E−02)	0.74(0.87)	3.19(0.25)	**0.24(4.01E−02)**	0.27(6.42E−02)	0.97(1.01)
2	S:(7,0,7) T:(14,39,10)	**0.19(2.15E−02)**	0.50(0.68)	3.28(0.24)	0.28(3.17E−02)	0.24(6.75E−02)	0.54(0.68)
	S:(8,0,6) T:(17,39,10)	**0.26(1.97E−02)**	1.59(1.06)	3.25(0.31)	0.55(0.65)	0.53(0.71)	1.65(1.06)
	S:(6,0,5) T:(10,39,7)	**0.28(5.28E−02)**	1.02(0.97)	3.23(0.28)	0.79(0.86)	0.37(9.07E−02)	1.18(1.02)
3	S:(6,0,7) T:(12,39,11)	**0.23(1.36E−02)**	0.28(3.46E−02)	3.41(0.21)	0.27(4.21E−02)	0.32(6.54E−02)	0.75(0.87)
	S:(5,0,7) T:(11,39,8)	**0.22(1.43E−02)**	0.46(0.61)	3.05(0.15)	0.29(5.26E−02)	0.28(6.10E−02)	1.02(1.17)
	S:(3,0,8) T:(12,39,9)	0.22(1.42E−02)	0.60(0.83)	3.16(0.29)	0.68(0.83)	**0.20(7.47E−02)**	1.30(1.09)
4	S:(7,0,9) T:(15,39,10)	**0.17(6.97E−03)**	0.64(0.88)	3.22(0.23)	0.27(4.86E−02)	0.26(3.73E−02)	0.47(0.71)
	S:(15,0,8) T:(10,39,12)	**0.32(4.67E−02)**	1.14(1.05)	3.05(0.23)	0.54(0.62)	0.51(0.60)	0.97(0.99)
	S:(12,0,6) T:(13,39,12)	0.44(5.14E−02)	0.81(0.84)	3.33(0.18)	0.61(0.66)	**0.39(5.60E−02)**	1.06(0.96)

Table 2. Computation time of AFHH, FHH, PSO, RWDE, and ADE in different scenarios

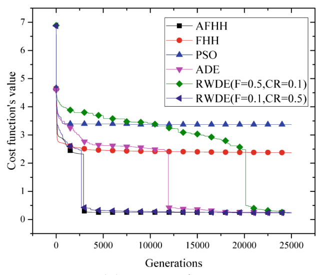
Scenario	Start point Target point	Computation time (Sec)					
		AFHH	FHH	PSO	RWDE (F = 0.5, CR = 0.1)	RWDE (F = 0.1, CR = 0.5)	ADE
1	S:(12,0,4) T:(10,39,6)	72.09	61.02	**36.76**	57.45	45.28	40.54
2	S:(7,0,7) T:(14,39,10)	72.04	60.98	**37.82**	56.46	44.96	37.94
3	S:(6,0,7) T:(12,39,11)	72.02	60.95	**35.89**	56.49	45.00	37.26
4	S:(7,0,9) T:(15,39,10)	72.19	60.79	**35.39**	56.31	44.73	37.55



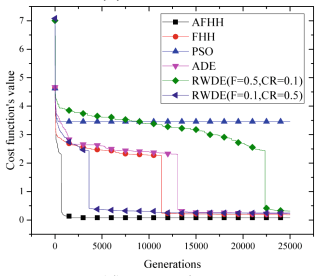
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 5. All algorithms' convergence property in different scenarios

5 Conclusions

In this paper, we propose the AFHH algorithm, which has good convergence rate and better performance. Many earlier works used GA, DE and PSO to make path planning. We find that EDAs can also complete this mission well. Maybe when use EDAs to solve path planning problems, parameters or properties of EDAs should change corresponding to problems. As the results shown in Sect. 4, AFHH needs more computation time to obtain high quality paths. This weakness is to be optimized in future works. Besides, the ability of EDAs to make path planning in dynamic environment should be checked.

Acknowledgments. This work was partially supported by the National Natural Science Foundations of China (NSFC) with Nos. 61772207 and 61332002, the Natural Science Foundations of Guangzhou Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the GDUPS (2016), and the Fundamental Research Funds for the Central Universities.

References

1. Blidberg, D.: The development of autonomous underwater vehicles (AUV); a brief summary. In: Proceedings of the International Conference on Robotics and Automation (ICRA), Seoul, South Korea, May 2001
2. Wang, B., Yu, L., Deng, Z., Fu, M.: A particle filter-based matching algorithm with gravity sample vector for underwater gravity aided navigation. IEEE/ASME Trans. Mechatron. **21** (3), 1399–1408 (2016)
3. Arango, M.D., Serna, C.A.: A memetic algorithm for the traveling salesman problem. IEEE Lat. Am. Trans. **13**(8), 2674–2679 (2015)
4. Zeng, Z., Lian, L., Sammut, K., He, F., Tang, Y., Lammas, A.: A survey on path planning for persistent autonomy of autonomous underwater vehicles. Ocean Eng. **110**, 303–313 (2015)
5. Garau, B., Alvarez, A., Oliver, G.: Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach. In: Proceedings of the IEEE Conference on Robotics and Automation, Barcelona, Spain, pp. 546–556, April 2005
6. Shih, C.-C., Horng, M.-F., Pan, T.-S., Pan, J.-S., Chen, C.-Y.: A genetic-based effective approach to path-planning of autonomous underwater glider with upstream-current avoidance in variable oceans. Soft. Comput. **21**(18), 5369–5386 (2017)
7. Zhang, C., Gong, Y., Li, J.J., Lin, Y.: Automatic path planning for autonomous underwater vehicles based on an adaptive differential evolution. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference, Canada, pp. 89–96, July 2014
8. Tsutsui, S., Pelikan, M., Goldberg, D.E.: Evolutionary algorithm using marginal histogram models in continuous domain. IlliGAL Report, vol. 2001019, no. 999, p. 1050 (2001)
9. Zhan, Z.H., Liu, X.F., Zhang, H., Yu, Z., Weng, J., Li, Y., Gu, T., Zhang, J.: Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version. IEEE Trans. Parallel Distrib. Syst. **28**(3), 704–716 (2017)
10. Xiao, J., Yan, Y., Zhang, J.: HPBILc: a histogram-based EDA for continuous optimization. Appl. Math. Comput. **215**(3), 973–982 (2009)
11. Yang, P., Tang, K., Lu, X.: Improving estimation of distribution algorithm on multimodal problems by detecting promising areas. IEEE Trans. Cybern. **45**(8), 1438–1449 (2015)
12. Zhan, Z.-H., Zhang, J.: Enhance differential evolution with random walk. In: International Conference on Genetic and Evolutionary Computation Conference Companion, pp. 1513–1514. ACM, New York (2012)