Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 246469, 13 pages
http://dx.doi.org/10.1155/2014/246469

Research Article

Path Planning Method for UUV Homing and Docking in Movement Disorders Environment

Zheping Yan,[1] Chao Deng,[1] Dongnan Chi,[2] Tao Chen,[1] and Shuping Hou[3]

[1] *College of Automation, Harbin Engineering University, Harbin 150001, China*
[2] *Beijing Institute of Space Mechanics & Electricity, Beijing 100094, China*
[3] *College of Mechanical and Electrical Engineering, Harbin Engineering University, Harbin 150001, China*

Correspondence should be addressed to Chao Deng; soledc@163.com

Received 23 March 2014; Revised 7 May 2014; Accepted 19 May 2014; Published 22 June 2014

Academic Editor: Giuseppe A. Trunfio

Path planning method for unmanned underwater vehicles (UUV) homing and docking in movement disorders environment is proposed in this paper. Firstly, cost function is proposed for path planning. Then, a novel particle swarm optimization (NPSO) is proposed and applied to find the waypoint with minimum value of cost function. Then, a strategy for UUV enters into the mother vessel with a fixed angle being proposed. Finally, the test function is introduced to analyze the performance of NPSO and compare with basic particle swarm optimization (BPSO), inertia weight particle swarm optimization (LWPSO, EPSO), and time-varying acceleration coefficient (TVAC). It has turned out that, for unimodal functions, NPSO performed better searching accuracy and stability than other algorithms, and, for multimodal functions, the performance of NPSO is similar to TVAC. Then, the simulation of UUV path planning is presented, and it showed that, with the strategy proposed in this paper, UUV can dodge obstacles and threats, and search for the efficiency path.

1. Introduction

Unmanned underwater vehicles (UUV) were first designed for military purposes. Comparing it with an ordinary vehicle, it is more suitable for stealthy and scouting missions. Recent advances in technology have driven the development of unmanned vehicles to a new level. UUV are widely applied in civilian applications such as surveying, landscape mapping, and life rescuing [1]. A major challenge in the development of UUV is the realization of a real-time path planning and obstacle avoidance strategy that can effectively guide the vehicle in unstructured environment [2].

In recent years, many researchers have developed various approaches to solve the path planning problems, such as genetic algorithms (GAs), linear programming, potential fields, probabilistic sampling methods like rapidly exploring random trees (RRTs), and artificial intelligence (AI) methods like A^*, which assures the path optimality. Cheng et al. [1] proposed a GA-inspired UUV path planner that is based on dynamic programming (DP). In the proposed path planner, the random-based crossover operator in GA is replaced with a deterministic crossover operator. The proposed path planner can always provide the best combination of crossover points from the available path segments. Li et al. [3] proposed a novel artificial bee colony (ABC) algorithm for unmanned combat aerial vehicles (UCAVs) path planning. Simulation results confirm that the algorithm is more competent for the UCAV path panning scheme than other ABC algorithms. Mansury et al. [4] have proposed a solution to the problem of path planning using artificial bee colony (ABC) algorithm and cubic Ferguson splines. Firstly, a path for robot movement is described by Ferguson splines and then ABC algorithm is used to optimize the parameter of splines to find an optimal path between the starting and the goal points considering obstacles between the starting and the goal points considering obstacles between them. Khelchandra and Jie [5] proposed a path planning method that is based on random sampling. Their proposed method has shown a high probability to find collision-free paths in short time. However, because of its randomized nature, their method may come up with infeasible solutions. Fernández-Perdomo et al. [6] proposed a novel path planning algorithm

for gliders using ocean currents. It bases on the A^* family of algorithms and incorporates a probabilistic framework. Instead of discretizing the search space, a set of bearing angles is sampled at each surfacing point and the glider trajectory is integrated. Like other exact algorithms, the computational complexity of an A^* algorithm increases with the size of the search domain. Li et al. [7] redefined potential functions to eliminate oscillations and local minima problems and use improved wall-following methods for the robots to escape nonreachable target problems. Meanwhile, they develop a regression search method to optimize the planned path. The optimization path is calculated by connecting the sequential points produced by improved artificial potential field (APF) method. The simulation results confirm that the proposed path planning approach can calculate a shorter or more nearly optimal path than the general APF. Jaillet et al. [8] presented a sampling-based algorithm to compute paths in problems which involve high-dimensional cost spaces. The proposed method combines the exploratory strength of the rapidly exploring random tree (RRT) algorithm, with the efficiency of stochastic optimization methods. It integrates an adaptive mechanism that helps to ensure a good performance for a large set of problems. Due to the nature of path planning, it is a constrained optimization problem, and many optimization algorithms have been proposed to overcome the problem. Brand et al. [9] proposed the application of ant colony optimization and compared two different pheromone reinitialization schemes.

Operating in an unknown semistructured environment, UUV may encounter obstacles which are not described in the electronic chart. In order to ensure global and real-time path planning, Yun et al. [10] divided the path planning algorithm into two levels: firstly, employing global path planning based on geometric method and then carrying out local path planning based on one new artificial field potential function, which uses sector as the parameter. In order to ensure real-time path planning, the information of sonar is combined to the static global map. Dolgov et al. [11] described a practical path-planning algorithm for an autonomous vehicle operating in an unknown semistructured environment, where obstacles are detected online by the robot's sensors. Firstly, they use a variant of A^* search to obtain a kinematically feasible trajectory and then improve the quality of the solution via numeric nonlinear optimization, leading to a local optimum. Further, they extend algorithm to use prior topological knowledge of the environment to guide path planning, leading to faster search and final trajectories better suited to the structure of the environment.

UUV must have homing and docking functions in order to be completely autonomous. Homing is an operation in which UUV return to the vicinity of the launcher after a mission. Path planning and tracking control of the planned path are also included in this operation. Docking is another operation in which UUV are fixed with the launcher exactly when it is close to the launcher. Docking is more difficult than homing since the position of the launcher can be changed easily by ocean current and waves. Hence, much more precise path considered that future position of the launcher is needed [12]. Nowadays, many of the docking methods have been studied. But, most of the developed methods are concerned about towing or underwater structure designed for docking with the docking station [13, 14]. And many articles are concerned about providing accurate measurement of the UUV position and orientation or providing control strategy for UUV docking [15–18]. However, plan homing and docking path is the first step for UUV homing and docking. Sujit et al. [19] present a navigation function based approach for docking onto a moving submarine. The motion planning system is based on potential fields but avoids the problem of local minima around no fly zones by using a Koditschek and Rimon navigation function. The authors represented the no fly zones with circles and ellipses; these zones have tangential potential. Thus, when the AUV reaches them, it gets deflected away depending on the direction of the tangential potentials. Using these directional potentials of the no fly zones, the navigation function based controller must guide the AUV towards the dock. But the resulting path is not an optimal rendezvous.

In this paper, we proposed a strategy to plan UUV homing and docking path in movement disorders environment. And the application of the particle swarm optimization (PSO) algorithm is investigated [20, 21]. PSO is considered as one of the modern heuristic algorithms for optimization first proposed by Kennedy and Eberhart in 1995 [22]. The motivation for the development of this method was based on the simulation of simplified animal social behaviors [23]. The PSO algorithm works on the social behavior of particles in the swarm [24]. In PSO, the population dynamics simulates a bird flock's behavior where social sharing of information takes place and individuals can profit from the discoveries and previous experience of all other companions during the search for food. That is, the global best solution is found by simply adjusting the trajectory of each individual towards its own best location and towards the best particle of the entire swarm at each time step [22, 23, 25]. Owing to its reduction on memory requirement and computational efficiency with convenient implementation, it has gained lots of attention in various optimal control system applications, compared to other evolutionary algorithms [20, 21, 26]. Several researches were carried out so far to analyze the performance of the PSO with different settings; for example, Shi and Eberhart [27] indicated that the optimal solution can be improved by varying the value of ω from 0.9 at the beginning of the search to 0.4 at the end of the search for most problems, and they introduced a method named TVIW with a linearly varying inertia weight over the generations. Guimin et al. [28] introduced exponential inertia weight strategies, which is found to be very effective for TVIW. Ratnaweera et al. [23] propose time-varying acceleration coefficients as a parameter automation strategy for the PSO named TVAC, which reduces the cognitive component and increases the social component by changing the acceleration coefficients.

The contribution of this paper is concluded as follows. Firstly, the avoiding static obstacle cost function, the avoiding dynamic obstacle cost function, and the approaching objective cost function are built to construct the path planning cost function. Secondly, we proposed a new strategy to plan path for UUV homing and docking. Thirdly, we proposed a novel

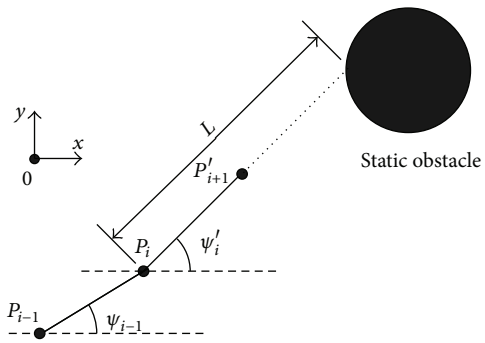


Figure 1: Schematic diagram of static obstacles avoidance for UUV.

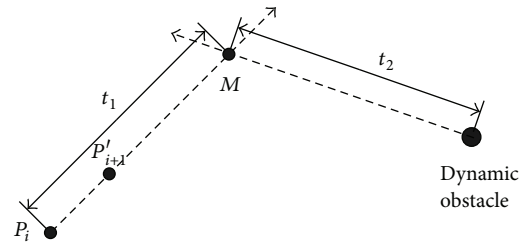


Figure 2: Schematic diagram of dynamic obstacles avoidance for UUV.

particle swarm optimization algorithm and used it to obtain the optimization cost function for path point.

The rest of this paper is organized as follows. In Section 2, the path planning cost function and a new strategy to plan path for UUV homing and docking are introduced. In Section 3, the basic PSO methodology and previous developments are summarized. In Section 4, a novel particle swarm optimization algorithm is proposed. In Section 5, the experimental settings for the benchmarks and simulation strategies are explained, and the conclusion is drawn based on the comparison analysis. In Section 6, the simulation of UUV path planning is explained. In Section 7, we present some concluding remarks.

2. UUV Path Planning for Homing and Docking

In this section, UUV path planning for mother vessel recovery using PSO in dynamic obstacles environment is presented and the reasonable path is studied. Path planning cost function has three parts: cost function for static obstacles, cost function for dynamic obstacles, and cost function for approaching target. To simplify the issue, UUV sail in the mission zones at certain fixed velocity so that UUV sail at certain fixed distance in each step. As each step for path planning has a fixed length of time, if we obtain the heading angle for UUV sailing to next waypoint, then we can obtain the position of the next waypoint. Set each particle of particle swarms as a heading angle for UUV sailing to next waypoint. Using NPSO, we can obtain the heading angle for UUV sailing to the optimal next waypoint with minimum cost function.

2.1. UUV Path Planning Cost Function. Firstly, define the initial position $P_0(x_0, y_0)$, the waypoint $P_i(x_i, y_i)$, $i = 1, 2, \ldots, n$, and the target $P_g(x_g, y_g)$.

2.1.1. Cost Function for Static Obstacles. In the underwater environment, UUV have to avoid static obstacles including submarine ridge, reef rock, and hidden pole. In Figure 1, static obstacles avoidance for UUV is illustrated, and the obstacles are represented by circles.

If the position of UUV is P_i, the next candidate waypoint is P'_{i+1}. As shown in Figure 1, the straight distance L is range from P_i to obstacle at the direction along P'_{i+1}. Cost function for static obstacles can be written as

$$F_1 = \frac{N * V}{L}, \tag{1}$$

where N is an adjusting coefficient, which is constant and given. V is velocity of UUV, which is given. The shorter the time from UUV to obstacle is, the larger the value of F_1 is, and vice versa. Set the candidate waypoint with minimum value of F_1 as the next waypoint.

2.1.2. Cost Function for Dynamic Obstacles. UUV also have to avoid dynamic obstacles existing in the mission zone. In Figure 2, the point M is regarded as the cross point of the line $P_iP'_{i+1}$ and the movement direction for dynamic obstacle. Assume that the time interval of UUV moving from P_i to M is t_1 and the time interval of dynamic obstacle motion to M is t_2. If $t_2 > t_1$, it is said that UUV are faster than the obstacle to arrive at M. When $t_2 - t_1$ is large, UUV will safely sail. Choose the avoiding cost function $e^{-\beta(t_2-t_1)^2}$. Besides, when the time between UUV and M is longer, more safety is obtained, and vice versa. Therefore, the cost function for dynamic obstacles is adjusted to

$$e^{-\alpha t_1^2} * e^{-\beta(t_2-t_1)^2}. \tag{2}$$

If $t_2 < t_1$, it is obvious that the dynamic obstacles will arrive at point M earlier than UUV, and UUV can avoid the dynamic obstacle safely. Above all, cost function for dynamic obstacles F_2 is

$$F_2 = \begin{cases} e^{-[\alpha t_1^2 + \beta(t_2-t_1)^2]} & t_2 > t_1 \\ 0 & \text{others.} \end{cases} \tag{3}$$

Consider the size of UUV and dynamic obstacle, F_2 in (3) can be rewritten as

$$F_2 = \begin{cases} e^{-[\alpha t_1^2 + \beta(t_2-t_1+\tau_1)^2]} & t_2 > t_1 - \tau_2 \\ 0 & \text{others,} \end{cases} \tag{4}$$

where α and β are adjusting coefficients, which are constant and given, τ_1 is the ratio of length to velocity of UUV, and τ_2 is the ratio of length to velocity of dynamic obstacle. If the cross point M vanishes, UUV trajectory and the dynamic obstacle trajectory have no intersection, and $F_2 = 0$.

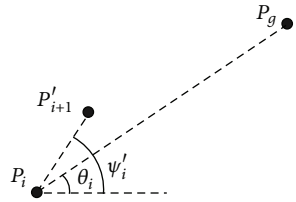


Figure 3: Schematic diagram of approaching target.

2.1.3. Cost Function for Approaching Target. To implement the motion of UUV towards the target, cost function for approaching target is designed. The schematic diagram of approaching target is shown in Figure 3.

In Figure 3, θ_i describes the angle of the connecting line between p_i and P_g. Because p_i and P_g are known, so it is easy to calculate the θ_i. ψ'_i is optimization parameter, which represents the heading angle of candidate waypoint P'_{i+1}. When $\psi'_i = \theta_i$, UUV move to target directly, and cost function for approaching target F_3 is the strongest. So, normal distribution is introduced to represent F_3. Consider

$$F_3 = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\psi-\theta)^2/2\sigma^2}. \quad (5)$$

2.1.4. UUV Path Planning Cost Function. UUV path planning cost function is iterated with three cost functions and described as

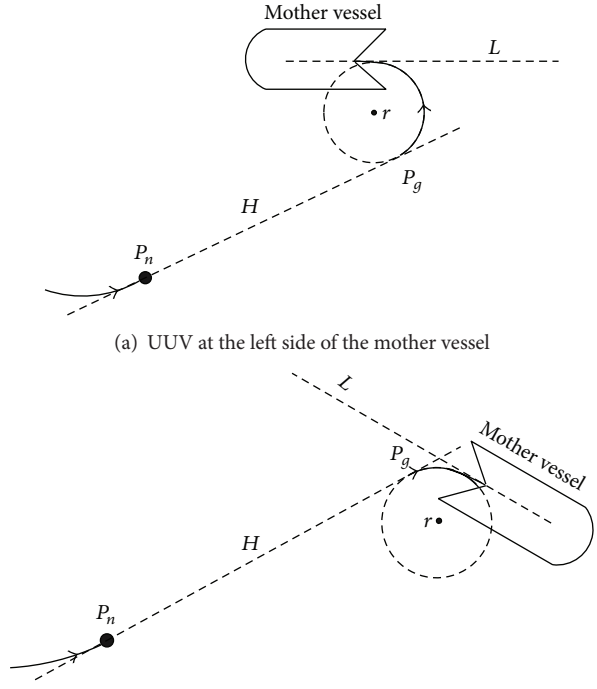
$$F = k_1 F_1 + k_2 F_2 - k_3 F_3, \quad (6)$$

where k_i is weight coefficients ($k_i > 0$), which are constant and given. UUV sail from initial waypoint with fixed velocity, and each step for path planning is fixed length of time. Set each particle of particle swarms as a heading angle for UUV sailing to the next waypoint. Using NPSO, we can obtain the heading angle for UUV sailing to the optimal next waypoint with minimum cost function F.

2.1.5. The Admittance Angle for Homing and Docking. For homing and docking, UUV will enter into the mother vessel (launcher) with a fixed angle (Figure 4). To obtain the effective path at the end of the planning path, let L denote mother vessel hatch center line and a circle is made with the radius r which is tangent to one point with the line L. The range of tangent point to the hatch is larger than the length of vessel (generally choose the length of two UUV), and r is larger than minimum turning radius.

In the path planning for homing and docking, the objective p_g is the tangent point between the circle and the line through the current UUV path point P_n. When $|P_n P_g| < v\tau$ (where v is the UUV velocity and τ is the time slot for one step), the path planning is finished and UUV enter into mother vessel at the end of arc.

Influenced by ocean current and waves, the mother vessel is moving, and UUV are moving and has to avoid obstacles, so UUV may sail change between the right side and the left side of the mother vessel. No matter what happened, we just make sure that the circle is in the same side of the mother vessel with UUV.



(a) UUV at the left side of the mother vessel

(b) UUV at the right side of the mother vessel

Figure 4: The recovery admittance angle.

3. Some Previous Work of PSO

Introduced by Dr. Kennedy and Dr. Eberhart in 1995, PSO has ever since turned out to be a competitor in the field of numerical optimization, and there has been a considerable amount of work done in developing the original version of PSO. In this section, we summarize some entire significant previous developments.

3.1. Basic Particle Swarm Optimization (BPSO). In PSO, each solution called a "particle" flies in the search space searching the optimal position to land. PSO system combines local search method (through individual experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation [29]. Each particle has a position vector $x_i(k)$, a velocity vector $v_i(k)$, the position with the best fitness encountered by the particle, and the index of the best particle in the swarm. The position vector and the velocity vector of the ith particle in the d-dimensional search space can be represented as $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots x_{id})$ and $v_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{id})$, respectively. The best position of each particle (p_{best}) is $p_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{id})$, and the fitness particle found so far at generation k (g_{best}) is $p_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$. In each generation, each particle is updated by the following two equations:

$$v_{id}(k+1) = v_{id}(k) + c_1 \times r_1 \times \left(p_{id}(k) - x_{id}(k)\right) + c_2 \times r_2 \times \left(p_{gd}(k) - x_{id}(k)\right), \quad (7)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1). \quad (8)$$

The parameters c_1 and c_2 are constants known as acceleration coefficients. r_1 and r_2 are random values in the range from 0 to 1, and the value of r_1 and r_2 is not the same for every iteration. Kennedy and Eberhart [22] suggested setting either of the acceleration coefficients at 2, in order to make the mean of both stochastic factors in (7) unity so that particles would over fly only half the time of search. The first equation shows that, in PSO, the search towards the optimum solution is guided by the previous velocity, the cognitive component, and the social component.

Since the introduction of the particle swarm optimization, numerous variations of the algorithm have been developed in the literature. Eberhart and Shi showed that PSO searches for wide areas effectively but tends to lack local search precision. They proposed in that work a solution by introducing ω, an inertia factor. In this paper, we name it as basic particle swarm optimization (BPSO):

$$\begin{aligned} v_{id}(k+1) &= \omega \times v_{id}(k) + c_1 \times r_1 \times \left(p_{id}(k) - x_{id}(k)\right) \\ &\quad + c_2 \times r_2 \times \left(p_{gd}(k) - x_{id}(k)\right), \\ x_{id}(k+1) &= x_{id}(k) + v_{id}(k+1). \end{aligned} \tag{9}$$

3.2. Time-Varying Inertia Weight (TVIW). The role of the inertia weight ω is considered very important in PSO convergence behavior. The inertia weight is applied to control the impact of the previous history of velocities on the current velocity. Large inertia weight facilitates global exploration, while small one tends to facilitate local exploration. In order to assure the particles converge to the best point in the course of the search, Shi and Eberhart [30] have found that time-varying inertia weight (TVIW) has a significant improvement in the performance of PSO and proposed linear decreasing inertia weight PSO (LWPSO) with a linear decreasing value of ω. This modification can increase the exploration of the parameter space during the initial search iterations and increase the exploitation of the parameter space during the final steps of the search [31]. The mathematical representation of inertia weight is given as follows:

$$\omega = (\omega_1 - \omega_2) \times \left(\frac{\text{MAXITER} - k}{\text{MAXITER}}\right) + \omega_2, \tag{10}$$

where ω_1 and ω_2 are the initial and final values of the inertia weight, respectively, k is the current iteration number, and MAXITER is the maximum number of allowable iterations. Shi and Eberhart [27] indicate that the optimal solution can be improved by varying the value of ω from 0.9 at the beginning of the search to 0.4 at the end of the search for most problems.

Guimin et al. [28] proposed natural exponential (base e) inertia weight strategies, named EPSO and expressed as

$$\omega = \omega_2 + (\omega_1 - \omega_2) \times \exp\left[-\left(\frac{K}{(\text{MAXITER}/4)}\right)^2\right]. \tag{11}$$

3.3. Time-Varying Acceleration Coefficient (TVAC). In PSO, the particle updated due to the cognitive component and the social component. Therefore, proper control of these two components is very important to find the optimum solution accurately and efficiently. Ratnaweera et al. [23] introduced a time-varying acceleration coefficient (TVAC), which reduces the cognitive component and increases the social component, by changing the acceleration coefficients c_1 and c_2 with the time evolution. The objective of this development is to enhance the global search in the early part of the optimization and to encourage the particles to converge towards the global optima at the end of the search. The TVAC is represented using the following equations:

$$\begin{aligned} c_1 &= (c_{\max} - c_{\min}) \frac{k}{\text{MAXITR}} + c_{\min}, \\ c_2 &= (c_{\min} - c_{\max}) \frac{k}{\text{MAXITR}} + c_{\max}, \end{aligned} \tag{12}$$

where $c_{\min}$ and $c_{\max}$ are constants, k is the current iteration number, and MAXITR is the maximum number of allowable iterations.

Simulations were carried out with numerical benchmarks to find out the best ranges of values for c_1 and c_2. From the results, it was observed that the best solutions were determined when changing c_1 from 2.5 to 0.5 and changing c_2 from 0.5 to 2.5 over the full search range.

4. Proposed New Developments

In the particle swarm algorithm, the trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of the other particles in the search space [23]. Kennedy and Eberhart [22] indicate that a relatively high value of the cognitive component, compared with the social component, will result in excessive wandering of individuals through the search space. In contrast, a relatively high value of the social component may lead particles to rush prematurely towards a local optimum.

Considering those concerns, a novel and effective approach to PSO algorithms is proposed in this paper. Particles are divided into several groups, and each group contains two particles. One particle in the group is noted as number 1, which is developed according to its own flying experience and the flying experience of swarms, and owns high individual experience acceleration coefficients and low group experience acceleration coefficients. So number 1 is encouraged to wander through the entire search space, without clustering around local optima, while the other particle in the group named number 2 is developed according to the flying experience of swarms and the flying experience of the group (both number 1 and number 2). When the flying experience of the group changed, set the flying experience of the group as position of number 2 and velocity of number 2 which is vanishing. Number 2 is encouraged to converge

Step 1. Initial.
Substep 1. Set the initial parameters: n, M, ω_i^t, $c_{\max}$, $c_{\min}$
Substep 2. Random initial $x_{id}^t(0)$
Substep 3. Random initial $v_{id}^t(1)$
Substep 4. Calculate fitness $(x_i^t(0))$ and set $p_i^t(0) = x_i^t(0)$, $p_u^t(0) = \min\{p_1^t(0), p_2^t(0)\}$
Substep 5. $p_g(0) = \{x_i^t(0) \mid x_i^t(0) \in \min(\text{fitness}(x_i^t(0)))\}$
Step 2. If the criteria are satisfied, output the best solution; otherwise, go to Substep 6.
Substep 6. Update $v_{id}^t(k)$,

$$v_{1d}^t(k+1) = \omega_1^t v_{1d}(k) + c_{\max} r_1^t \left(p_{1d}^t(k) - x_{1d}^t(k)\right) + c_{\min} r_2^t \left(p_{gd}(k) - x_{1d}^t(k)\right);$$

if $p_{ud}^t(k) \geq p_{ud}^t(k-1)$

$$v_{2d}^t(k+1) = \omega_2^t v_{2d}(k) + c_{\min} r_3^t \left(p_{ud}^t(k) - x_{2d}^t(k)\right) + c_{\max} r_4^t \left(p_{gd}(k) - x_{2d}^t(k)\right);$$

else

$$v_{2d}^t(k+1) = 0;$$

Substep 7. Update $x_{id}^t(k)$,

$$x_{1d}^t(k+1) = x_{1d}^t(k) + v_{1d}^t(k+1);$$

if $p_{ud}^t(k) \geq p_{ud}^t(k-1)$

$$x_{2d}^t(k+1) = x_{2d}^t(k) + v_{2d}^t(k+1);$$

else

$$x_{2d}^t(k+1) = p_{ud}^t(k);$$

Substep 8. Calculate fitness $(x_i^t(k))$
Substep 9. if fitness $(x_i^t(k))$ < fitness $(p_i^t(k))$

$$p_i^t(k) = x_i^t(k);$$

if $p_1^t(k) < p_2^t(k)$

$$p_u^t(k) = p_1^t(k);$$

else

$$p_u^t(k) = p_2^t(k);$$

if $\min\{\text{fitness}(x_i^t(k)), i = 1, 2, t = 1, \ldots, n/2\} < \text{fitness}(p_g(k))$

$$p_g(k) = \{x_i^t(k) \mid \min[\text{fitness}(x_i^t(k)), i = 1, 2, t = 1, \ldots, n/2]\};$$

Substep 10. Go back to Step 2.

Algorithm 1

towards the global optima, with a small cognitive component and a large social component. Consider

$$v_{1d}^t(k+1) = \omega_1^t v_{1d}(k) + c_{\max} r_1^t \left(p_{1d}^t(k) - x_{1d}^t(k)\right) + c_{\min} r_2^t \left(p_{gd}(k) - x_{1d}^t(k)\right),$$

$$v_{2d}^t(k+1) = \begin{cases} \omega_2^t v_{2d}(k) + c_{\min} r_3^t \left(p_{ud}^t(k) - x_{2d}^t(k)\right) + c_{\max} r_4^t \left(p_{gd}(k) - x_{2d}^t(k)\right) & p_{ud}^t(k) \geq p_{ud}^t(k-1) \\ 0 & p_{ud}^t(k) < p_{ud}^t(k-1), \end{cases}$$

$$x_{1d}^t(k+1) = x_{1d}^t(k) + v_{1d}^t(k+1),$$

$$x_{2d}^t(k+1) = \begin{cases} x_{2d}^t(k) + v_{2d}^t(k+1) & p_{ud}^t(k) \geq p_{ud}^t(k-1) \\ p_{ud}^t(k) & p_{ud}^t(k) < p_{ud}^t(k-1), \end{cases} \quad (13)$$

where r_1^t, r_2^t, r_3^t, and r_4^t are random values, uniformly distributed between zero and one, and the their value is not same for every iteration, $x_{id}^t(k)$ is the dth dimension position of the subparticle i of the tth group after k time iteration, ω_i^t is the inertia weigh value of the subparticle i of the tth group, $c_{\max}$ is the maximum value of acceleration coefficients, $c_{\min}$ is the minimum value of acceleration coefficients, $v_{id}^t(k)$ is the dth dimension velocity of the subparticle i of the tth group during k time iteration, $p_{id}^t(k)$ is the dth dimension position of the optimal position of the ith subparticle of the tth groupafter k time iteration, $p_{ud}^t(k)$ is the dth dimension position of the optimal position of the tth group after k time iteration, and $p_{gd}(k)$ is the dth dimension position of the swarm optimal position after k time iteration.

Remark 1. In this paper, two subparticles are generated for each group; therefore, $i = 1, 2$.

Define n as the number of particles in the swarm, M as the maximum iteration frequency, and fitness $(x_i^t(k))$ as the value of the cost function for the subparticle i of the tth group after k time iteration.

The detailed steps are shown as in Algorithm 1.

5. Experimental Settings and Simulation for Benchmark Testing

Simulations were carried out to observe the rate of convergence and the quality of the optimum solution of the new methods introduced in this investigation by comparing

Table 1: Parameters for simulation.

Method	Parameters	
NPSO	$\omega_i^t = 0.7$, $c_{\max} = 2.5$, $c_{\min} = 0.5$	$n = 30$, $D = 10$, $x_i \in [-100, 100]$, $v_i \in [-100, 100]$
BPSO	$c_1 = c_2 = 2.0$, $\omega = 0.7$	
LWPSO	$c_1 = c_2 = 2.0$, $\omega_1 = \omega_{\max} = 0.9$, $\omega_2 = \omega_{\min} = 0.4$	
EPSO	$c_1 = c_2 = 2.0$, $\omega_1 = \omega_{\max} = 0.9$, $\omega_2 = \omega_{\min} = 0.4$	
TVAC	$c_{\max} = 2.5$, $c_{\min} = 0.5$, $\omega = 0.7$	

between BPSO, LWPSO, EPSO, and TVAC. From the standard set of benchmark problems available in the literature, there are 6 important functions considered to test the efficacy of the proposed method. All of the test functions reflect different degrees of complexity.

5.1. Functions Introduction. The functions are as follows.

5.1.1. Sphere Function. Consider

$$f_1(x) = \sum_{i=1}^{D} x_i^2. \tag{14}$$

With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [0, \ldots, 0]^D$ with $f(x) = 0$. It is very simple, convex unimodal function with only one local optimum value.

5.1.2. Rotated Hyperellipsoid Function: Schwefel's Problem 1.2. Consider

$$f_2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2. \tag{15}$$

With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [0, \ldots, 0]^D$ with $f(x) = 0$. It is continuous, convex, and unimodal. With respect to the coordinate axes, this function produces rotated hyperellipsoids.

5.1.3. Rosenbrock Function. Consider

$$f_3(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]. \tag{16}$$

With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [1, \ldots, 1]^D$ with $f(x) = 0$. It is a unimodal function and the global optimum is inside a long, narrow, parabolic shaped flat valley. Finding the valley is trivial.

5.1.4. Rastrigin Function. Consider

$$f_4(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos\left(2\pi x_i\right) + 10 \right]. \tag{17}$$

With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [0, \ldots, 0]^D$ with $f(x) = 0$. It is highly multimodal. However, the locations of the minima are regularly distributed.

5.1.5. Griewank Function. Consider

$$f_5(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1. \tag{18}$$

With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [0, \ldots, 0]^D$ with $f(x) = 0$. It is a multimodal function and has many widespread local minima. However, the locations of the minima are regularly distributed.

5.1.6. Sum of Different Powers Function. Consider

$$f_6(x) = \sum_{i=1}^{D} \left| x_i \right|^{(i+1)}. \tag{19}$$

The sum of different powers is a commonly used unimodal test function. With the search space $\{x_i \mid -100 < x_i < 100\}$, the global minimum is located at $x = [0, \ldots, 0]^D$ with $f(x) = 0$.

5.2. The Coefficients Setting. The parameters for simulation are listed in the Table 1.

In this table, $c_{(\cdot)}$ expresses the accelerations coefficients, ω denotes inertia weight, the dimension is D, and the range of the search space and the velocity space are $x(\cdot)$ and $v(\cdot)$. If the current position is out of the search space, the position of the particle is taken to be the value of the boundary and the velocity is taken to be zero. If the velocity of the particle is outside of the boundary, its value is set to be the boundary value. The maximum number of iterations is set to 1000. For each function, 100 trials were carried out and the average optimal value and the standard deviation are presented. To verify the performance of the algorithm at different dimensions, variable dimension D increases from 10 to 100, and the optimal mean and variance of test function with 6 algorithms are calculated.

5.3. The Results Comparing NPSO with the Previous Developments of PSO. The simulation results are given in Table 2. The comparison results elucidate that the searching accuracy and stability arranged from low to high are BPSO, LWPSO, EPSO, TVAC, and NPSO for unimodal function. For multimodal function, the performance of NPSO is the same as TVAC, but, for the highly multimodal Rastrigin function, TVAC is superior to NPSO.

With the increase of test functions dimension, the searching accuracy and stability for each algorithm are decreased.

Table 2: Comparison between BPSO, LWPSO, EPSO, TVAC, and NPSO.

F	D	Average (standard deviation)				
		BPSO	LWPSO	EPSO	TVAC	NPSO
f_1	10	4.7979$e-11$ (1.1343$e-10$)	3.2219$e-24$ (1.0706$e-23$)	8.7747$e-51$ (3.408$e-50$)	1.4826$e-54$ (1.4770$e-53$)	2.2681$e-78$ (2.1874$e-77$)
	30	5.3768 (4.3459)	4.0137$e-05$ (5.926$e-05$)	2.3219$e-12$ (1.1495$e-11$)	1.1062$e-10$ (7.3302$e-10$)	2.0175$e-23$ (2.0172$e-22$)
	50	2.8412$e+01$ (4.5047)	0.2149 (0.1634)	2.2215$e-05$ (3.5696$e-05$)	0.0019 (0.0079)	7.6676$e-09$ (5.7490$e-08$)
	70	4.4685$e+01$ (4.3454)	1.1727$e+01$ (1.2113$e+01$)	0.0192 (0.0238)	0.0609 (0.1033)	0.0010 (0.0082)
f_2	10	0.0013 (0.0018)	4.0624$e-08$ (1.10659$e-07$)	3.1542$e-15$ (1.9686$e-14$)	3.6072$e-27$ (1.6376$e-26$)	3.1897$e-27$ (1.0275$e-26$)
	30	2.0647$e+01$ (7.9831)	3.7596 (2.0255)	0.5809 (0.3373)	0.0091 (0.0197)	0.0012 (0.0101)
	50	7.2982$e+01$ (2.8263$e+01$)	2.6189$e+01$ (1.3125$e+01$)	1.0295$e+01$ (5.1265)	0.4883 (0.3021)	0.1684 (0.7178)
	70	1.4821$e+02$ (4.6281$e+01$)	6.9336$e+01$ (3.3717$e+01$)	2.9336$e+01$ (1.0417$e+01$)	2.7039 (1.0853)	0.8808 (1.3369)
f_3	10	5.6418 (1.092)	4.3037 (1.2401)	3.1701 (1.2637)	0.7262 (0.9490)	0.6790 (1.0522)
	30	1.2322$e+03$ (9.4693$e+02$)	4.2313$e+01$ (2.6596$e+01$)	3.1052$e+01$ (1.7699$e+01$)	2.3370$e+01$ (1.9238)	2.2096$e+01$ (5.8478)
	50	6.5575$e+03$ (1.3919$e+03$)	2.9054$e+02$ (1.6229$e+02$)	7.3894$e+01$ (3.6633$e+01$)	4.6563$e+01$ (2.5861)	4.6325$e+01$ (1.2231$e+01$)
	70	1.1834$e+04$ (1.9184$e+03$)	5.2635$e+03$ (4.0246$e+03$)	2.0414$e+02$ (7.4873$e+01$)	7.5240$e+01$ (1.2063$e+01$)	8.0415$e+01$ (2.9457$e+01$)
f_4	10	5.1090 (3.0667)	3.6806 (1.7685)	3.8703 (1.7882)	0.9750 (0.9693)	2.0911 (1.2360)
	30	1.5696$e+02$ (3.9880$e+01$)	3.2291$e+01$ (1.0059$e+01$)	2.9948$e+01$ (7.1988)	1.7282$e+01$ (5.4892)	2.0219$e+01$ (6.9868)
	50	3.8006$e+02$ (3.5258$e+01$)	7.5142$e+01$ (2.5871$e+01$)	6.1896$e+01$ (1.5107$e+01$)	3.6575$e+01$ (9.4137)	4.3956$e+01$ (1.1176$e+01$)
	70	5.8512$e+02$ (3.6592$e+01$)	1.6259$e+02$ (6.2498$e+01$)	8.8994$e+01$ (1.7553$e+01$)	5.4198$e+01$ (1.1482$e+01$)	7.3153$e+01$ (1.4861$e+01$)
f_5	10	9.8646$e-05$ (0.0009)	0.0041 (0.0096)	0.0017 (0.0055)	0 (0)	0 (0)
	30	0.1366 (0.1090)	0.0008 (0.0045)	0.0013 (0.0071)	1.0210$e-10$ (1.0021$e-09$)	2.3425$e-16$ (8.3751$e-16$)
	50	0.5856 (0.0891)	0.0027 (0.0021)	9.9034$e-05$ (0.0009)	0.0001 (0.0007)	4.3243$e-10$ (2.4063$e-09$)
	70	0.6941 (0.0646)	0.0329 (0.0190)	0.0002 (0.0010)	0.0011 (0.0017)	1.1807$e-05$ (4.5670$e-05$)
f_6	10	5.8516$e-18$ (2.5897$e-17$)	2.2299$e-40$ (1.6366$e-39$)	2.0832$e-84$ (1.6585$e-83$)	2.5229$e-87$ (1.1045$e-86$)	1.6901$e-96$ (1.6901$e-95$)
	30	1.9895$e+02$ (6.2708$e+02$)	1.0418$e-06$ (3.7845$e-06$)	7.3163$e-18$ (2.7837$e-17$)	1.0483$e-39$ (7.2123$e-39$)	4.6820$e-50$ (4.6820$e-49$)
	50	1.9927$e+07$ (7.1603$e+07$)	1.8348$e+01$ (4.8704$e+01$)	0.0017 (0.0067)	6.5668$e-25$ (3.1648$e-24$)	4.3136$e-21$ (4.2561$e-20$)
	70	1.5637$e+13$ (8.3656$e+13$)	2.6886$e+08$ (1.6837$e+09$)	1.4973$e+04$ (1.0807$e+05$)	4.9509$e-19$ (2.6306$e-18$)	7.7603$e-13$ (7.5801$e-12$)

The performance of NPSO is superior to other algorithms. With regard to high multimodal function, the performance of TVAC is superior to NPSO. From Figure 5, it shows that the order of searching speed from high to low is BPSO (green solid curve), LWPSO (red circle), EPSO (blue triangle), TVAC (black dot curve), and NPSO (blue dash-dot). It is obvious that the performance of NPSO is more effective than the other algorithms. This is because, for the NPSO, two-particle coordination evolution is in the same group. One particle of the group is encouraged to wander through the entire search space, without clustering around local optima. The other particle of the group is flying between group experiences and swarm experiences, which is encouraged to converge towards the global optima.

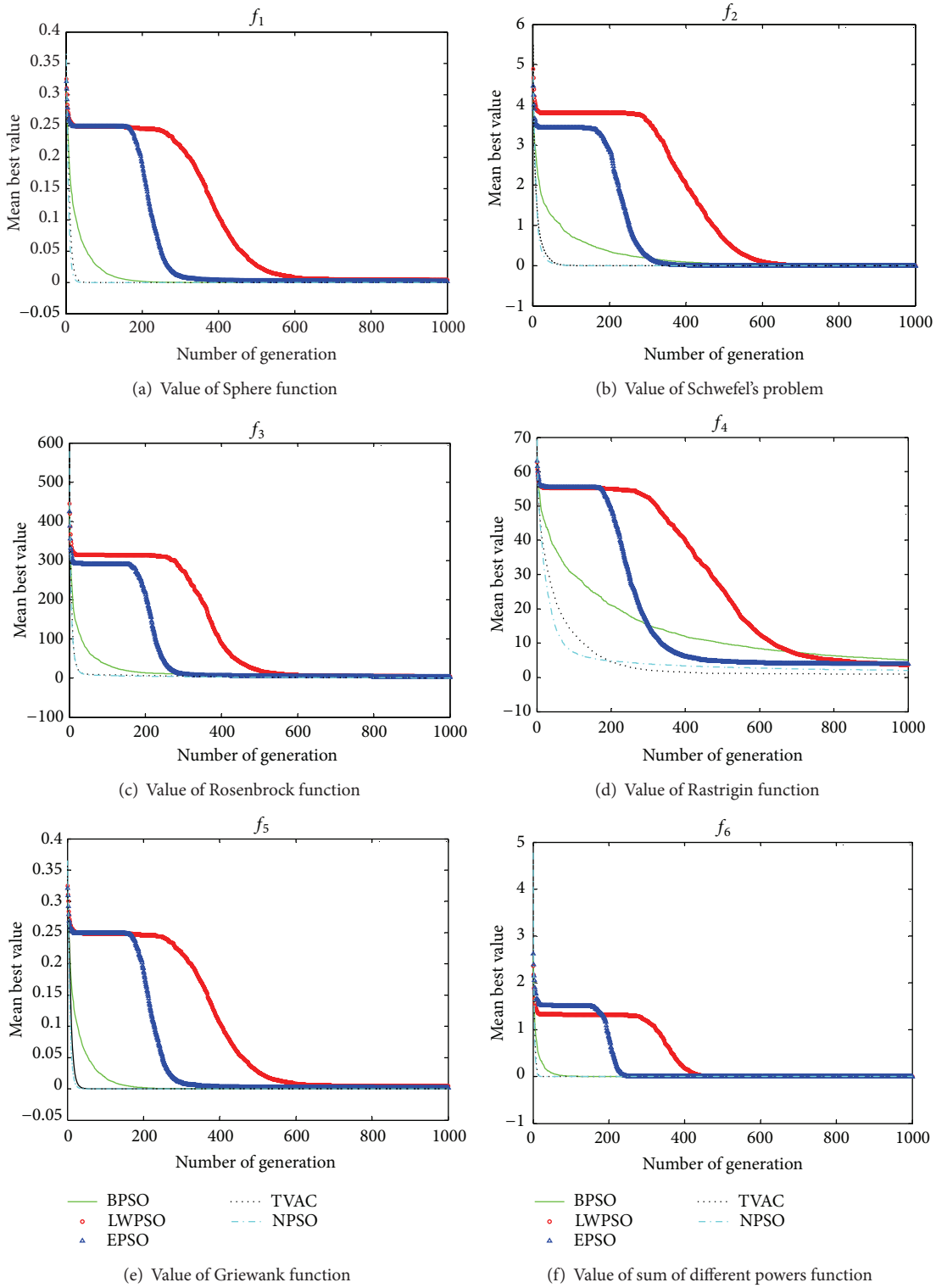


FIGURE 5: Variation of average optimum value with time.

6. The Simulation of UUV Path Planning

The simulation is designed using the proposed algorithm for UUV path planning in dynamic obstacles environment. UUV start at the initial point, and the candidate point at the next step is searched using NPSO in the range of the turning angle limitation. And the next path point is obtained when the cost function reaches minimum value. Simulation is designed in the map with size of 1000 m × 1000 m in Figure 6. The initial point is A (50, 50), and dock hatch point is D (900,

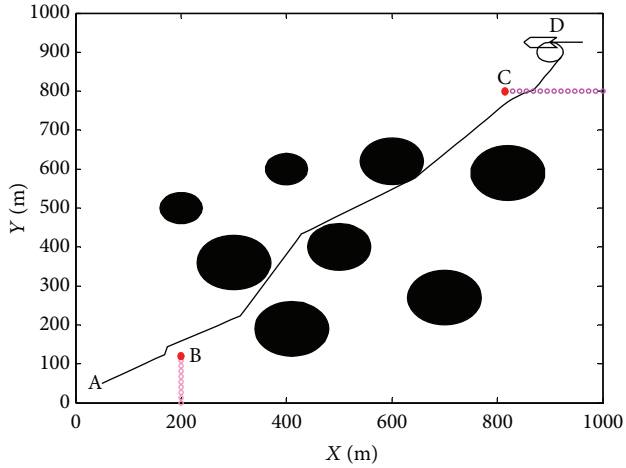


Figure 6: Path planning in dynamic obstacles environment.

Table 3: Parameters for simulation.

Method	Parameters	
NPSO	$\omega_i^t = 0.7$, $c_{\max} = 2.5$, $c_{\min} = 0.5$	
BPSO	$c_1 = c_2 = 2.0$, $\omega = 0.7$	$n = 30$,
LWPSO	$c_1 = c_2 = 2.0$, $\omega_1 = \omega_{\max} = 0.9$, $\omega_2 = \omega_{\min} = 0.4$	$D = 3$,
EPSO	$c_1 = c_2 = 2.0$, $\omega_1 = \omega_{\max} = 0.9$, $\omega_2 = \omega_{\min} = 0.4$	$x_i \in [-\pi/3, \pi/3]$,
TVAC	$c_{\max} = 2.5$, $c_{\min} = 0.5$, $\omega = 0.7$	$v_i \in [-\pi/3, \pi/3]$

900). The maximum turning limitation is $|\Delta\psi| \leq \pi/3$, $k_1 = k_2 = k_3 = 1$, $N = 1$. To simplify the issue, the velocity is set to a constant 4 m/s, and UUV will encounter dynamic obstacle B which is parallel to the y-axis with sway coordinate at 200 m and dynamic obstacle C parallel to the x-axis with surge coordinate at 800 m.

In Figure 6, for obstacle B, UUV avoid it by steering left. For obstacle C, UUV avoid it by changing its course towards its starboard side. The simulation results demonstrate that the proposed algorithm is effective to implement the path planning in dynamic obstacle environment.

To compare the performance of NPSO with LWPSO/EPSO/TVAC for path planning, the simulation is designed for path planning without dynamic obstacles. Table 3 shows the parameters that were used for the different algorithms used, and Table 4 shows the simulation results.

From Table 4, NPSO and TVAC present better performance than the other PSO.

To compare the performance of the algorithm proposed in this paper with traditional APF and A^*, complex environment map has been introduced in Figure 7.

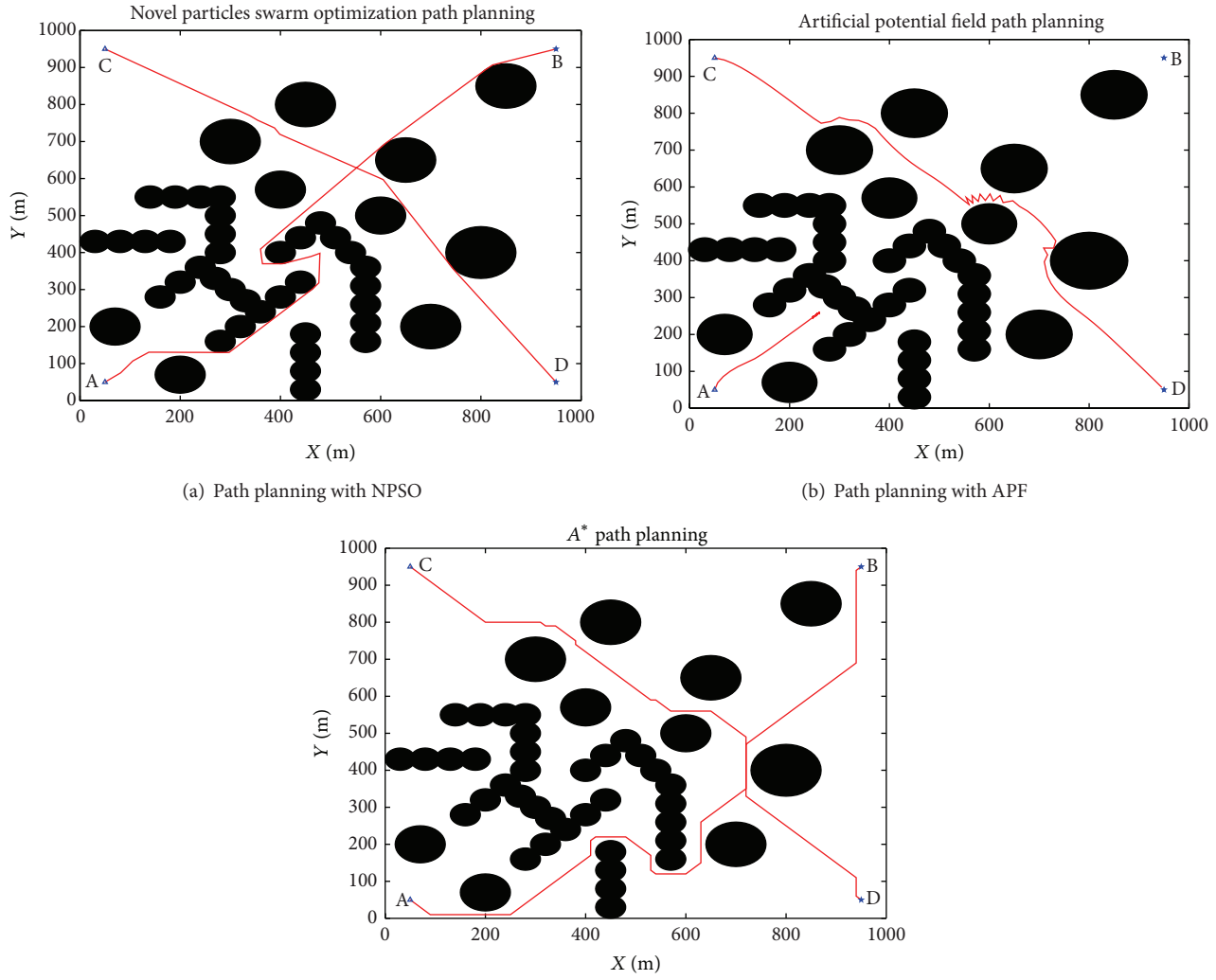
As shown in Figure 7, it turns out that proposed algorithm can obtain a better performance than the traditional APF for UUV path planning in complex environment. If the starting point is A (50, 50), and the target is B (950, 950), the traditional APF is easy to fall into local minima and cannot finish the path planning. To further analyze the performance, we set starting point C (50, 950) and target D (950, 50). From Figure 7, traditional APF is easy to shock in front of the obstacle. The A^* algorithm discretizes the search space with a uniform grid, so the possible bearings are discretized too. As a consequence, the time between consecutive surfacing is nonconstant. From Table 5, the algorithm proposed can obtain shorter path than the other algorithms. The proposed algorithm can obtain a more smooth path and is superior to traditional APF and A^* algorithm. Table 6 shows the computation time of different algorithms. It is clear that traditional APF spent shorter time than other algorithms, and NPSO spent the longest time.

Table 4: Comparison between BPSO, LWPSO, EPSO, TVAC, and NPSO for UUV path planning.

M	Length of the path (m)				
	BPSO	LWPSO	EPSO	TVAC	NPSO
10	1290.2548	1289.9196	1289.4264	1288.8112	1288.0532
50	1287.8672	1287.8556	1287.8548	1287.8544	1287.8544
100	1287.8596	1287.8544	1287.8544	1287.8544	1287.8544
200	1287.8544	1287.8544	1287.8544	1287.8544	1287.8544
500	1287.8544	1287.8544	1287.8544	1285.7200	1284.8792
1000	1287.8544	1287.8544	1287.8544	1285.7200	1284.8792

Table 5: Comparison of different algorithms for path planning.

Path	Length of the path (m)		
	NPSO	APF	A^*
A → B	1593.79	—	1716.81
C → D	1307.15	1547.84	1401.66



(a) Path planning with NPSO

(b) Path planning with APF

(c) Path planning with A^*

Figure 7: UUV path planning under complex environment.

Table 6: Computation time of different algorithms for path planning.

Path	Computation time (s)		
	NPSO	APF	A^*
A → B	94.83	—	11.70
C → D	54.55	0.27	10.17

7. Conclusion

In this paper, UUV path planning for homing and docking in dynamic obstacle environment using NPSO is present and the appropriate strategy is explored at the mother vessel recovery. The simulation results demonstrate the feasibility of NPSO. UUV can avoid the dynamic obstacles and navigate along the reasonable path to implement the recovery.

The simulation of UUV path planning in complex environment shows that the proposed algorithm can get better path than traditional APF and A^* algorithm, and it is not easy to be trapped in local minima. In order to completely escape from local minima or U shaped obstacles, it is necessary to design some escape rules, such as wall-following, random escaping.

NPSO is introduced and tested through a set of 6 benchmark functions and the statistical analyses of the simulated results are compared with BPSO, LWPSO, EPSO, and TVAC. The test of proposed method with the unimodal benchmark functions indicates its superiority over the other methods. However, for highly multimodal Rastrigin function, TVAC shows better performance than NPSO.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the Natural Science Foundation of China (51179038; 51109043; and 51309067/

E091002) and the Foundation of Heilongjiang Province, China (E201123).

References

[1] C.-T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired UUV path planner based on dynamic programming," *IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews*, vol. 42, no. 6, pp. 1128–1134, 2012.

[2] M. H. Kim and J. Sur, "A stream function approach to design obstacle avoidance path planning algorithm for autonomous underwater vehicles," in *Proceedings of the 21st International Offshore and Polar Engineering Conference (ISOPE '2011)*, pp. 285–290, June 2011.

[3] B. Li, L. Gong, and W. Yang, "An improved artificial bee colony algorithm based on balance evolution strategy for unmanned combat aerial vehicle (UCAV) path planning," *The Scientific World Journal*, vol. 2014, Article ID 232704, 10 pages, 2014.

[4] E. Mansury, A. Nikookar, and M. E. Salehi, "Artificial Bee Colony optimization of ferguson splines for soccer robot path planning," in *Proceedings of the 1st RSI/ISM International Conference on Robotics and Mechatronics (ICRoM '13)*, pp. 85–89, February 2013.

[5] T. Khelchandra and H. Jie, "A recursive sampling based method for path planning," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '07)*, pp. 2408–2412, ISIC, October 2007.

[6] E. Fernández-Perdomo, J. Cabrera-Gámez, D. Hernández-Sosa et al., "Adaptive Bearing Sampling for a Constant-Time Surfacing A* path planning algorithm for gliders," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 2350–2355, 2011.

[7] G. Li, Y. Tamura, A. Yamashita et al., "Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning," *International Journal of Mechatronics and Automation*, vol. 3, no. 3, pp. 141–170, 2013.

[8] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.

[9] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," in *Proceedings of the International Conference on Computer Design and Applications (ICCDA '10)*, pp. V3436–V3440, June 2010.

[10] G. Yun, W. Zhiqiang, G. Feixiang et al., "Dynamic path planning for underwater vehicles based on modified artificial potential field method," in *Proceedings of the IEEE 4th International Conference on Digital Manufacturing and Automation (ICDMA '13)*, pp. 518–521, 2013.

[11] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[12] K. H. Oh, J. Y. Kim, I. W. Park et al., "A study on the control of AUV's homing and docking," in *Proceedings of the IEEE Conference on Mechatronics and Machine Vision in Practice*, Chiang Mai, Thailand, 2002.

[13] T. Fukasawa, T. Noguchi, T. Kawasaki, and M. Baino, "'Marine Bird', a new experimental AUV with underwater docking and recharging system," in *Proceedings of the OCEANS 2003*, vol. 4, pp. 2195–2200, IEEE, September 2003.

[14] R. S. McEwen, B. W. Hobson, J. G. Bellingham, and L. McBride, "Docking control system for a 54-cm-diameter (21-in) AUV," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 550–562, 2008.

[15] M. D. Feezor, F. Y. Sorrell, P. R. Blankinship, and J. G. Bellingham, "Autonomous underwater vehicle homing/docking via electromagnetic guidance," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 515–521, 2001.

[16] A. Martins, J. M. Almeida, H. Ferreira et al., "Autonomous surface vehicle docking manoeuvre with visual information," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, pp. 4994–5000, April 2007.

[17] P. Batista, C. Silvestre, and P. Oliveira, "A two-step control strategy for docking of Autonomous Underwater Vehicles," in *Proceedings of the American Control Conference (ACC '12)*, pp. 5395–5400, June 2012.

[18] M. Breivik and J.-E. Loberg, "A virtual target-based underway docking procedure for unmanned surface vehicles B," in *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 13630–13635, September 2011.

[19] P. B. Sujit, A. J. Healey, and J. B. Sousa, "AUV docking on a moving submarine using a K-R navigation function," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems: Celebrating 50 Years of Robotics (IROS '11)*, pp. 3154–3159, September 2011.

[20] M. H. Abdulameer, S. N. H. S. Abdullah, and Z. A. Othman, "Support vector machine based on adaptive acceleration particle swarm optimization," *The Scientific World Journal*, vol. 2014, Article ID 835607, 8 pages, 2014.

[21] H. Li, H. Jiang, and B. Li, "Reflection reduction on DDR3 high-speed bus by improved PSO," *The Scientific World Journal*, vol. 2014, Article ID 257972, 11 pages, 2014.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[23] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[24] Q. Ni and J. Deng, "A new logistic dynamic particle swarm optimization algorithm based on random topology," *The Scientific World Journal*, vol. 2013, Article ID 409167, 8 pages, 2013.

[25] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[26] M. S. Arumugam, M. V. C. Rao, and A. W. C. Tan, "A novel and effective particle swarm optimization like algorithm with extrapolation technique," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 308–320, 2009.

[27] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, 1999.

[28] C. Guimin, H. Xinbo, J. Jianyuan, and M. Zhengfeng, "Natural exponential inertia weight strategy in particle swarm optimization," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, pp. 3672–3675, June 2006.

[29] B. Jiao, Z. Lian, and Q. Chen, "A dynamic global and local combined particle swarm optimization algorithm," *Chaos, Solitons and Fractals*, vol. 42, no. 5, pp. 2688–2695, 2009.

[30] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98), IEEE World Congress on Computational Intelligence*, pp. 69–73, May 1998.

[31] M. Schwaab, E. C. Biscaia Jr., J. L. Monteiro, and J. C. Pinto, "Nonlinear parameter estimation through particle swarm optimization," *Chemical Engineering Science*, vol. 63, no. 6, pp. 1542–1552, 2008.