Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner

Hui Sheng Lim
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
hui.lim@utas.edu.au

Shuangshuang Fan
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
shuangshuang.fan@utas.edu.au

Christopher K.H. Chin
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
c.chin@utas.edu.au

Shuhong Chai
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
shuhong.chai@utas.edu.au

***Abstract*—Over years of development, many optimization techniques have been proposed for the path planning of the Autonomous Underwater Vehicle (AUV). The development in swarm intelligence optimization, particularly the particle swarm optimization (PSO), has significantly improved the performance of the AUV path planner. This study presents 12 variants of particle swarm intelligence (PSI)-based algorithms, which were applied to evaluate their performances in solving the optimal path planning problem of an AUV operating in 2D and 3D ocean environments with obstacles and non-uniform currents. Throughout the structure of the optimization problem, the practicability of the path planning algorithms were considered by taking into account the physical limitations of the AUV actuations. To compare the performances of these PSI-based algorithms, extensive Monte Carlo simulations were conducted to evaluate these algorithms based on their respective solution qualities, stabilities and computational efficiencies. Ultimately, the strengths and weaknesses of these algorithms were comprehensively analyzed, in order to identify the most appropriate optimization algorithm for AUV path planning in dynamic environments.**

Keywords—Autonomous Underwater Vehicle; Path planning; Optimization; Swarm intelligence; Particle Swarm Optimization

I. INTRODUCTION

AUVs are unmanned underwater vehicles which can be remotely programmed to conduct various missions. To date, many efforts have been made to enable the operation of AUVs in more dynamic and constrained environments. The exploration of AUVs in highly dynamic regions possesses several technical issues, particularly for its path planning. An optimum AUV path planner should be able to determine a path that safely guides the AUV from a starting point to a target under rapid-changing dynamic environments, based on either minimum time or energy cost criterion [1].

Planning the path for the AUVs is essentially a multimodal optimization problem. Developing the algorithms for AUV path planning faces several intrinsic difficulties, particularly in balancing the computational requirements and performance of the path planner. For a high-dimensional problem space, the computational requirement of the algorithms could escalate exponentially. A general path planning approach is to simplify the 3D environment into a 2D space, in order to reduce the computational time and the memory requirement [1]. However, this compromises the performance of the path planner due to reduced amount of 3D information available, such as currents field, bathymetry and obstacles in the ocean environment. A recent comparison study in [2] for the existing path optimization techniques proved the superiority of evolutionary algorithms, particularly the evolutionary particle swarm intelligence (PSI)-based algorithms, which were found to be remarkably robust and efficient for solving high-dimensional path planning problem.

Particle Swarm Optimization (PSO) and its most significant variant, the Quantum Particle Swarm Optimization (QPSO) are extensively used in various optimization problems ever since their emergence in 1995 and 2004 respectively due to their fine search abilities and easy implementations [3]. In recent years, many strategies that modified the PSO and QPSO algorithms have been proposed to improve their performances in path planning of various autonomous systems. Each of these variants of the algorithms was claimed to have different extends of improvement over the original PSO and QPSO. Nevertheless, there is lack of a systematic method to evaluate the performances of these algorithms. It is crucial to present a study that compares and reviews these algorithms for the required application.

Therefore, this paper aims to provide a comprehensive evaluation study on these algorithms through the application in AUV path planning. A novel path planner based on an AUV was developed and integrated with different PSI-based algorithms. To evaluate the performance of these algorithms, path planning scenario with multiple obstacles and non-uniform current field was simulated in 2D and 3D domains. Although the actual AUV operates in a 3D ocean field, 3D path planning simulations are not widely discussed in majority of other literature due to the complexity of high dimensional path planning problem. It is critical to apply the path planning algorithms in 3D space to assess the effectiveness of the algorithms under realistic conditions. Extensive Monte Carlo simulations were conducted to analyse the performances of different PSI-based algorithms.

The rest of this paper is arranged as follows. Section II provides literature review on the basic PSO, QPSO and their variants. The path planning problem is formulated in Section III. Section IV presents the simulation setup, results and discussion. Lastly, Section V concludes the paper with future directions.

II. PSO, QPSO AND THEIR VARIANTS

This section presents the overview of various PSI-based algorithms, including the basic PSO, QPSO, and their variants. The variants of PSO and QPSO are classified based on the methods of modification used to improve their performances.

A. PSO Algorithm

PSO is a heuristic optimization algorithm introduced by [4] based on the inspiration from the analogues of cognitive abilities and social interaction in animals. [5, 6] are some examples of pioneering works of PSO application in path planning. PSO consists of particles that move within a multidimensional search space to search for potential solutions, which are represented by the particles' positions. The particles' velocities are updated by the particle's own experience (cognitive behaviour) and the swarm's experience (social behaviour) to vary their positions.

In a standard PSO algorithm consisting of N particles with D number of dimensions for solving a cost evaluation function f, the position vector of the i^{th} particle at t^{th} iteration is denoted as:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ \ldots \ , x_{i,D}^t \right], \quad i \in \{1, 2, ..., N\} \tag{1}$$

Based on its previous best position, *pbest* and global best position in the swarm, *gbest*, the velocity V and the position X of the i^{th} particle at $(t+1)^{\text{th}}$ iteration are updated as follows:

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1^t \cdot (pbest_i^{\,t} - X_i^t) + C_2 \cdot r_2^t \cdot (gbest^{\,t} - X_i^t) \tag{2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{3}$$

$$pbest_i^{\,t} = \begin{cases} pbest_i^{t-1}, & if\ f(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & if\ f(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{4}$$

$$gbest^{\,t} = \arg\min \left[f(pbest_i^{\,t}) \right] \tag{5}$$

In (2), r_1 and r_2 are random positive numbers that are less than 1.0. C_1 and C_2 are the acceleration coefficients for cognitive and social components respectively; they are both set to 2.0 for most applications [3]. w is the inertia weight for balancing the particle global exploration and local exploitation to improve the performance. The common strategy is to set w at an initial value of 0.9, and linearly decrease to 0.4 during the iteration [3].

B. QPSO Algorithm

Inspired by quantum mechanics and PSO, [7] proposed the QPSO algorithm, which assumes the particles to have quantum behaviour. QPSO algorithm is well known to be an improved version of PSO. Its application in path planning was pioneered by [8, 9]. In QPSO, the position of the i^{th} particle is given as:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^{\,t} + (1 - \varphi_i^t) \cdot gbest^{\,t} \\ \quad + \beta \cdot \left| mbest^{\,t} - X_i^t \right| \cdot \ln\left(1/u_i^t\right), & if\ u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^{\,t} + (1 - \varphi_i^t) \cdot gbest^{\,t} \\ \quad - \beta \cdot \left| mbest^{\,t} - X_i^t \right| \cdot \ln\left(1/u_i^t\right), & if\ u < 0.5 \end{cases} \tag{6}$$

$$mbest^{\,t} = \sum_{i=1}^{N} pbest_i^t / N \tag{7}$$

where u and φ are random positive numbers that are less than 1. β is the contraction-expansion (CE) coefficient, and *mbest* is the mean best position which is defined as the average of personal best positions of all particles as shown in (7). When applying the QPSO algorithm, β is the most critical parameter for controlling the algorithm performance. A linearly decreasing β from $\beta_{\max}$ of 1.0 to $\beta_{\min}$ of 0.5 is suggested for most applications [3].

C. Variants of PSO and QPSO

1) Improvement by Controlling Parameters

In PSI-based algorithms, the equation coefficients are the most critical parameters for controlling the performance. In PSO, w, C_1 and C_2 must be controlled to balance the particles' global exploration and local exploitation. Reference [10] proposed Adaptive PSO, which uses an evolutionary factor f as an indicator representing the particles' evolutionary state to control the equation coefficients. Adaptive PSO was applied in solving robotic path planning problem by [11]. To determine the evolutionary factor f, the mean distance d_i of the i^{th} particle to other particles is calculated using (8). f is then given by (9).

$$d_i = \frac{1}{N-1} \sum_{j=1,\, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k \right)^2} \tag{8}$$

$$f = \left(d_g - d_{\min} \right) \Big/ \left(d_{\max} - d_{\min} \right) \quad \in [0, 1] \tag{9}$$

where d_g is the mean distance of the global best particle, $d_{\min}$ and $d_{\max}$ are the minimum and maximum of mean distances respectively. f varies from 1 - 0 as the particles move from global exploration to local exploitation phase. w can be calculated from f using (10), while C_1 and C_2 can be adapted using (11).

$$w = 1 \Big/ \left(1 + 1.5 e^{-2.6 f} \right) \quad \in [0.4, 0.9] \tag{10}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f - 0.5|} \\ C_2 &= 3.2 - 2e^{-|f - 0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{11}$$

The only coefficient that needs to be controlled in QPSO is β. To adapt β with the particle evolution, [12] proposed Dynamic-Weighted QPSO (DWQPSO) to solve an AUV path planning problem. In DWQPSO, β is controlled by classifying the particles based on their individual fitness F_i. The global best fitness is denoted as F_{gbest}, and the average of all particles fitness values is F_{avg}. The mean of fitness that are above average (better than F_{avg}) is denoted as F_{good}. The categories of the particles are:

- $F_i \geq F_{avg}$: For these particles, the global exploration of the particles should be boosted with a higher β using (12).

$$\beta_1 = 1.5 - 1 \Big/ \left(1 + 1.5 \cdot e^{(F_{gbest} - F_{good})} \right) \tag{12}$$

- $F_{good} < F_i < F_{avg}$: Linear decreasing model in (13) is used to balance between global exploration and local exploitation.

$$\beta_2 = \beta_{\max} - (t / MaxIt) \cdot (\beta_{\max} - \beta_{\min}) \tag{13}$$

- $F_i \leq F_{good}$: These particles should focus on local exploitation to find the optimal solution. Thus, β is updated as follows.

$$\beta_3 = (\beta_2 - 0.5) \cdot \left| \left(F_i - F_{gbest} \right) \Big/ \left(F_{good} - F_{gbest} \right) \right| \tag{14}$$

2) Improvement by Novel Update Equation

Some studies introduced strategies to improve the algorithm by modifying the position and velocity update equations. In [13],

Accelerated PSO was proposed by simplifying the update equation in PSO. It was successfully applied in developing a fast and simple path planner by [14]. Accelerated PSO disregards the particles' personal best positions and focuses on the global best position using a simple update equation as shown in (15).

$$X_i^{t+1} = 0.5X_i^t + 0.5gbest^t + e^{-0.97t} \cdot \left(r_i^t - 0.5\right) \quad (15)$$

where r is a random number with a value ranging from 0 to 1.0.

In [15], phase angle-encoded PSO (θ-PSO) is proposed by mapping the position vectors into phase angle vectors through (16), while the increment of phase angle replaces the velocity vectors. The phase angle vector θ of the i^{th} particle at $(t+1)^{th}$ iteration and its increment $\Delta\theta$ are given by (17) and (18).

$$X_i^t = \left[(X_{\max} - X_{\min}) \cdot \sin(\theta_i^t) + X_{\max} + X_{\min}\right]/2 \quad (16)$$

$$\Delta\theta_i^{t+1} = w \cdot \Delta\theta_i^t + C_1 \cdot r_1^t \left(pbest_i^t - \theta_i^t\right) + C_2 \cdot r_2^t \left(gbest^t - \theta_i^t\right) \quad (17)$$

$$\theta_i^{t+1} = \theta_i^t + \Delta\theta_i^{t+1} \quad \in [-\pi/2, \pi/2] \quad (18)$$

Inspired by [15], θ-QPSO was proposed by [16], who applied a similar phase angle mapping in QPSO to develop an Unmanned Aerial Vehicle (UAV) path planner, and proved that θ-QPSO has better performance than θ-PSO. θ-QPSO only computes for the vector θ as shown in (19).

$$\begin{aligned}\theta_i^{t+1} = {} & \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t \\ & \pm \beta \cdot \left|mbest^t - \theta_i^t\right| \cdot \ln\left(1/u_i^t\right) \quad \in [-\pi/2, \pi/2]\end{aligned} \quad (19)$$

3) Improvement by Hybrid Method

Hybridization is used to combine the beneficial feature of other optimization techniques with PSO or QPSO algorithm. In [17], PSO is combined with Differential Evolution (DE) to form DEPSO. DEPSO increases the swarm diversity without altering the original particle swarm dynamics. Based on the inspiration from DEPSO, [18] applied the hybridization concept in QPSO to propose DEQPSO, and successfully used both DEPSO and DEPQSO for UAV path planning. In DEPSO and DEQPSO, the conventional position update operation is carried out, followed by a successive DE operation as described below.

- Mutation: A mutated vector U is generated using (20).

$$U_i^t = gbest^t + \left[(pbest_{i1}^t - pbest_{i2}^t) + (pbest_{i3}^t - pbest_{i4}^t)\right]/2 \quad (20)$$

  where i_1, i_2, i_3 and i_4 are randomly selected particle indices and $i_1 \neq i_2 \neq i_3 \neq i_4 \neq gbest$.

- Crossover: A trial vector T is generated to increase the diversity, by conducting crossover between the mutated vector and the personal best position as shown in (21).

$$\begin{aligned} T_i^t &= \left[t_{i,1}^t, \ldots, t_{i,j}^t, \ldots, t_{i,D}^t\right] \\ t_{i,j}^t &= \begin{cases} u_{i,j}^t, & if\ r_j \le 0.85 \,||\, j = r \\ pbest_{i,j}^t, & if\ r_j > 0.85 \,||\, j \neq r \end{cases} \end{aligned} \quad (21)$$

  where r_j is a random number ranging from 0 to 1.0, and r is a random integer ranging from 1 to D.

- Selection: A greedy selection is used to decide whether the trial vector T should replace the current position X in $(t+1)$th iteration. X will only be replaced if T has better fitness value.

4) Improvement by Combination of Multiple Approaches

Some studies improved the algorithm by applying more than one method. Reference [19] proposed IPSO-SQP algorithm, in which an improved PSO (IPSO) with adaptive inertia weight was combined with Sequential Quadratic Programming (SQP) algorithm. SQP has strong searching ability for the local optimum solution, although its solution quality is highly dependent on the initial solution. SQP was integrated into PSO to accelerate the local exploitation phase. In IPSO-SQP, the inertia weight w is controlled adaptively according to (22).

$$w_i^t = 1/\left[1 + e^{-F(pbest_i^t)/gbest1}\right] \quad (22)$$

When the change in global best fitness between iterations in IPSO-SQP is less than a predefined value, SQP is initialized using the global best solution from the IPSO operation. The final solution is updated using a greedy selection method, which allows the SQP solution to replace the solution from IPSO only if the SQP solution is better. IPSO-SQP was applied in solving the motion planning of a REMUS AUV by [20].

A hybrid QPSO algorithm, LTQPSO was proposed by [21]. LTQPSO use individual particle evolutionary rates and swarm dispersion as the control parameters for its novel local attractor and position update equations as shown in (23) and (24).

$$p_i^t = gbest^t + ip_i^t \cdot r \cdot (pbest_i^t - gbest^t) \quad (23)$$

$$X_i^{t+1} = \begin{cases} p_i^t + \beta \cdot \left|mbest^t - gs^t \cdot X_i^t\right| \cdot \ln(1/u_i^t), & if\ u \ge 0.5 \\ p_i^t - \beta \cdot \left|mbest^t - gs^t \cdot X_i^t\right| \cdot \ln(1/u_i^t), & if\ u < 0.5 \end{cases} \quad (24)$$

where ip_i^t and gs^t are the control parameters for evolutionary rate and swarm dispersion, as given in (25) and (26) respectively.

$$ip_i^t = F(gbest^t)/F(pbest_i^t) \quad \in [0,1] \quad (25)$$

$$gs^t = \left[\frac{\sigma(pbest_{i,1}^t)}{\sigma(X_{i,1}^t)}, \frac{\sigma(pbest_{i,2}^t)}{\sigma(X_{i,2}^t)}, \ldots, \frac{\sigma(pbest_{i,D}^t)}{\sigma(X_{i,D}^t)}\right] \quad (26)$$

For each iteration in LTQPSO, natural selection is conducted after the standard QPSO operation. Natural selection sorts the particles according to their personal best fitness, and replaces those of the worst fitness with those of the best. The natural selection operator increases the evolutionary rate of the entire swarm by eliminating the least desirable solutions, leading to a faster global convergence. LTQPSO was proven by [1] to have good performance for robotic path planning in 2D environment.

III. Problem Formulation for AUV Path Planning

A. Path Formulation

In an AUV path planner, the optimal path among a group of potential paths for the AUV to travel toward a target location is required to be determined. Each potential path comprises a series of nodes from the start point to the endpoint. Optimizing the coordinates of path nodes will produce the optimal path. The start and end points are not involved in the optimization because all the potential paths share the same start and end locations.

Each potential path solution for the problem is modelled as an individual particle in the swarm. The swarm population is denoted by a matrix $X = [X_1, X_2, \ldots, X_N]^T$, where X is the particle's

position vector and N is the total number of particles. The entries of the position vector represent the coordinates of the path nodes. Assuming a path consists of n+2 nodes including the start and end points, the number of nodes involved in the optimization is n. To record the coordinates of n nodes, the position vector of a particle in 2D problem has 2n dimensions, while a particle in 3D has 3n dimensions. The position vector of the i^{th} particle at t^{th} iteration for 3D can be given as follows:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, ..., x_{i,n}^t, x_{i,n+1}^t, ..., x_{i,3n}^t \right], \quad i \in \{1,2,...,N\} \quad (27)$$

Based on the path nodes including the start and end points, B-spline geometry is used to construct the AUV path. The path nodes act as the control points for the B-spline curve according to the curve function in (28), which gives output vector $P(u)$ representing a B-spline curve with k+1 order in the form of discretised waypoints. Given the total number of control points is n+2, the total number of piecewise polynomials in B-spline is one less than the number of control points, which is n+1.

$$P(u) = \sum_{i=0}^{n+1} x_i \, B_{i,k}(u), \qquad i \in \{0,1,2,...,n+1\} \quad (28)$$

where x_i are the control points, u is the non-decreasing knot sequence contained in a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$, and $B_{i,k}(u)$ are the piecewise polynomial basis functions of k degree defined by Cox de Boor recursion [22] as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (29)$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \quad (30)$$

B. Evaluation Function

Suitable cost evaluation functions are required for PSI-based algorithms to measure the fitness of the particles. Due to the high computational efficiency of PSI-based algorithms, fitness evaluation usually contribute to the majority of computational time [3]. For path planning, a lower cost/fitness indicates a better solution. The main criteria for evaluating the AUV path are: the travel time required to reach the target, the exposure to threats, and the compliance with AUV's physical motion limitations. Since it is almost impossible to achieve all criteria at the same time, a trade-off between these criteria can be established using a weighting scheme with multiple evaluation functions. The fitness of a particle/path X_i can be given by the summation of fitness from multiple evaluation functions F_k for different criteria, with each criterion weighted by a cost factor f_k.

$$F\left(X_i^t\right) = \sum_{k=1}^{K} f_k F_k\left(X_i^t\right), \quad k \in \{1,2,...,K\} \quad (31)$$

where k refers to different evaluation functions and K is the total number of functions for the problem.

1) Path Travel Time Cost

The main evaluation function for path planning problem is to measure the path cost based on its time to travel on the path. A given path X_i can be represented in the form of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is the output from B-spline function and m is the number of discretised waypoints. The travel time cost F_1 of a path can be determined using (32).

$$F_1(X_i) = \sum_{j=1}^{m-1} \left[\left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \Big/ \left| V_g \right| \right], \quad j \in \{1,2,...,m-1\} \quad (32)$$

where V_g is the resultant ground reference velocity of the AUV, The contribution of current on the AUV can be obtained by projecting the current velocity V_c in the direction of the AUV water reference velocity V_a. Thus, V_g is given as shown in (33).

$$V_g = V_a + \left(V_c \bullet \overrightarrow{p_{i,j}\, p_{i,j+1}} \right) \Big/ \left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \quad (33)$$

2) Threat Cost

The obstacles avoidance of path planner relies on the threat cost evaluation, which measures the path's exposure to threats. All threats in the problem space are modelled as eclipses in 2D, and as ellipsoids in 3D. The common method to measure the threat cost is by calculating the distance of the discretised waypoints to the centre of threat using (34), and penalising the cost if the distance is smaller than the semi-major axis of threat.

$$d_{threat} = \left\| \overrightarrow{p_{i,j}\, O_{c,h}} \right\|, \qquad h \in \{1,2,...,H\} \quad (34)$$

where O_c is the threat centre, h refers to different threats and H is the total number of threats in the problem space. However, the accuracy of this method depends on the path fineness, i.e. the number of discretised waypoints on the path. The threat cost is inaccurate when the distance between two consecutive waypoints is greater than the minor axis of the threat. Therefore, a threat evaluation method based on the intersection between the path and the threats is used. The intersection-based method has fineness-independent accuracy, meaning that the computational requirement can be lowered without affecting the cost accuracy.

Assuming a threat h in 3D problem space with centre $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and semi principal axes $O_{r,h} = (O_{rx}, O_{ry}, O_{rz})$, its parametric equation can be expressed in (35). The equation of a path segment that connects two consecutive waypoints $p_{i,j} = (x_1, y_1, z_1)$ and $p_{i,j+1} = (x_2, y_2, z_2)$ can be written as (36).

$$\left(\frac{x - O_{cx}}{O_{rx}}\right)^2 + \left(\frac{y - O_{cy}}{O_{ry}}\right)^2 + \left(\frac{z - O_{cz}}{O_{rz}}\right)^2 = 1 \quad (35)$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + s \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \quad (36)$$

Substituting (36) into (35) yields the following quadratic equation, which is expressed in term of s.

$$A s^2 + B s + C = 0 \quad (37)$$

$$A = \frac{O_{ry}^2 O_{rz}^2 (x_2 - x_1) + O_{rx}^2 O_{rz}^2 (y_2 - y_1) + O_{rx}^2 O_{ry}^2 (z_2 - z_1)}{O_{rx}^2 O_{ry}^2 O_{rz}^2} \quad (38)$$

$$B = \frac{(O_{cx} - x_1)(x_1 - x_2)}{0.5 O_{rx}^2} + \frac{(O_{cy} - y_1)(y_1 - y_2)}{0.5 O_{ry}^2} + \frac{(O_{cz} - z_1)(z_1 - z_2)}{0.5 O_{rz}^2} \quad (39)$$

$$C = \frac{(O_{cx} - x)^2}{O_{rx}^2} + \frac{(O_{cy} - y)^2}{O_{ry}^2} + \frac{(O_{cz} - z)^2}{O_{rz}^2} - 1 \quad (40)$$

The intersection of the path with the threat can be evaluated by obtaining the discriminant D of (37) according to (41).

$$D = B^2 - 4AC \quad (41)$$

A safety margin is added to the principal axes of all threats regions so that the AUV will not conflict with the threat when $D = 0$. When $D > 0$, the path will conflict with the threat, and the threat cost is proportional to the length of segment containing within the threat region. If the path intersects with the threat, the intersection points can be found by solving (37) using (42). The threat cost of a path X_i can then be obtained using (43).

$$S_1,\ S_2 = \left(-B \pm \sqrt{D}\right) / 2A \tag{42}$$

$$F_2(X_i) = \sum_{h=1}^{H} \sum_{j=1}^{m-1} \left\| \overrightarrow{S_1\ S_2} \right\| \Big/ \left[2 \times \max(O_{r,h}) \right] \tag{43}$$

3) Physical Motion Limitations

The considerations for physical motion limitations of AUV should include its yaw (turning) and pitch motions. To check the path compliance with the yaw limitation, the turning angle of the path in the *x*-*y* plane is measured and compared against the maximum allowable turning angle $\psi_{\max}$. Considering two consecutive path segments that consist of three waypoints $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j+2}$ (refer to Fig. 1), the turning angle ψ can be obtained from the cosine function as shown in (44).

$$\psi_j = \cos^{-1}\left[\frac{\left(\overrightarrow{p'_{i,j}\ p'_{i,j+1}} \bullet \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}} \right) \Big/ \left\| \overrightarrow{p'_{i,j}\ p'_{i,j+1}} \right\|}{\left\| \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}} \right\|} \right] \tag{44}$$

The cost F_3 for violating the yaw limitation can be obtained from the calculated turning angle as shown in (45).

$$F_3(X_i) = \sum_{j=1}^{m-1} \begin{cases} 0 & , \text{if } |\psi_j| \le \psi_{\max} \\ \left(180 - |\psi_j|\right) / \left(180 - \psi_{\max}\right) & , \text{if } |\psi_j| > \psi_{\max} \end{cases} \tag{45}$$

For the pitch motion, the instantaneous pitch angle θ and the change in pitch $\Delta\theta$ of the AUV at any point should not exceed their respective maximum values ($\theta_{\max}$ & $\Delta\theta_{\max}$). Referring to Fig. 1, θ can be determined using basic tangent function as shown in (46). Next, $\Delta\theta$ can be calculated using (47).

$$\theta_j = \tan^{-1}\left[\left\| p_{i,j+2}\ p''_{i,j+2} \right\| \Big/ \left\| p_{i,j+1}\ p''_{i,j+2} \right\| \right] \tag{46}$$

$$\Delta\theta_j = \theta_{j+1} - \theta_j \tag{47}$$

From the calculated pitch, the cost F_4 for violating $\theta_{\max}$ and the cost F_5 for $\Delta\theta_{\max}$ can be obtained as follows:

$$F_4(X_i) = \sum_{j=1}^{m-1} \begin{cases} 0 & , \text{if } |\theta_j| \le \theta_{\max} \\ \left(90 - |\theta_j|\right) / \left(90 - \theta_{\max}\right) & , \text{if } |\theta_j| > \theta_{\max} \end{cases} \tag{48}$$

$$F_5(X_i) = \sum_{j=1}^{m-2} \begin{cases} 0 & , \text{if } |\Delta\theta_j| \le \Delta\theta_{\max} \\ \left(90 - |\Delta\theta_j|\right) / \left(90 - \Delta\theta_{\max}\right) & , \text{if } |\Delta\theta_j| > \Delta\theta_{\max} \end{cases} \tag{49}$$

IV. Simulations

A. Simulation Setup

The AUV path planning was conducted in a 1000-run Monte Carlo simulation under 2D scenario, followed by 3D scenario. The problem space was a current field that consists of 50×50 square grids for 2D, and 50×50×50 cube grids for 3D, with each side of the grid equivalent to 1 metre. Non-uniform ocean current and static obstacles of different sizes are present in the problem space. The AUV is required to travel with a pre-set water reference velocity of 1.5m/s. The safety margin used in the threat computation is set to 1 metre, while the angles $\psi_{\max}$, $\theta_{\max}$ and $\Delta\theta_{\max}$ are set to 30°, 45° and 10° respectively. The cost factor for the path travel time, f_1 was set to be 1.0, and the other cost factors $f_2 - f_5$ were all set to be 0.25, and thus all costs except the travel time cost have similar impact on the solutions. In each simulation run, the maximum number of iterations was set to 100. The population size was 150 particles, with each particle consists of 4 path nodes, meaning each particle has 8 dimensions for 2D problem and 12 dimensions for 3D. The algorithm parameters were set to be the values suggested in Section II.

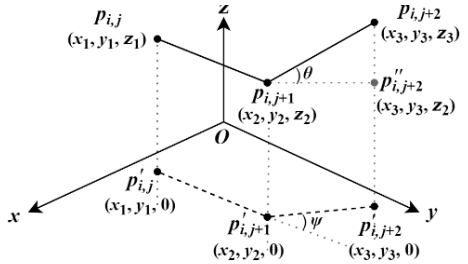

Fig. 1. Yaw angle and pitch angle of a path

B. Simulation Results

The optimal path solutions obtained from the Monte Carlo simulation under 2D and 3D scenarios are shown in Fig. 2 and Fig. 4 respectively. The AUV is required to travel from the starting point (green square) to the target (pink star) without running into the obstacles, while trying to take advantage of the favourable current to assist the AUV motion. In 2D (Fig. 2), the blue-coloured zones indicate the favourable current while the red-coloured zones denote the less favourable current. In both domains, the solid sections of the AUV paths indicate that the favourable current has positive effect on the AUV motion while the dotted sections suggest otherwise. It can be seen that most of the generated paths are able to follow the favourable zone and avoid the less favourable zone to achieve a shorter travel time.

The performances of the algorithms are compared based on their solution qualities, stabilities, convergence behaviours, and computational requirements; these properties can be evaluated by studying the fitness values of the solutions obtained and the computational time required to obtain the solutions. The fitness values are simply the time required for the AUV to reach the endpoint from the start point by travelling on the path. Thus, a lower fitness value indicates a higher solution quality.

The convergence behaviours of the algorithms under 2D and 3D scenarios are compared in Fig. 3 and Fig. 5. The convergence speed of the algorithm can be given by the minimum number of iterations required for the algorithm to converge at an optimal or sub-optimal solution. It can be observed in the graphs that the convergence speeds of all algorithms significantly decrease when the dimensionality of the problem increases from 2D to 3D. DEPSO and DEQPSO are found to be outperforming other algorithms with similar performance under both scenarios; the two algorithms achieve the fastest convergence and the global convergence with lowest fitness. Adaptive PSO and IPSO-SQP are also able to offer faster and better convergence than

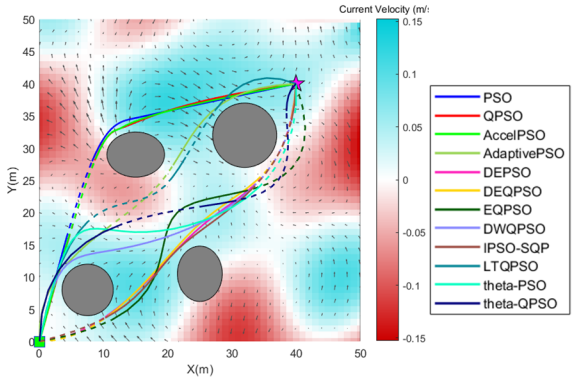

Fig. 2. Path planning solutions in 2D scenario

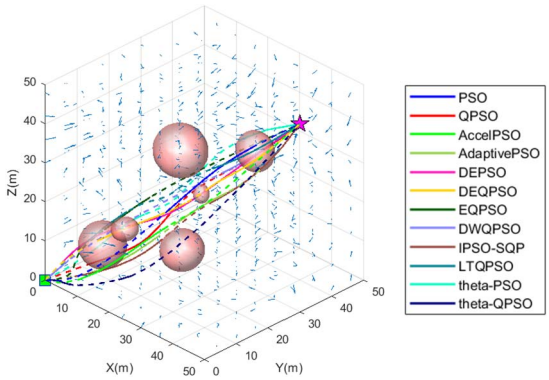

Fig. 4. Path planning solutions in 3D scenario

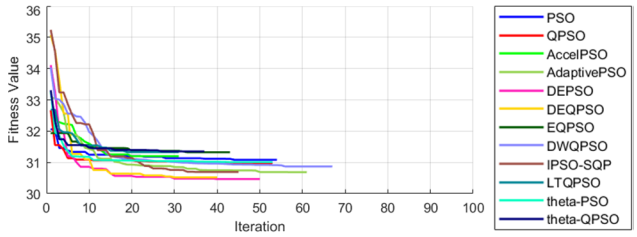

Fig. 3. Convergence curves of fitness values in 2D scenario

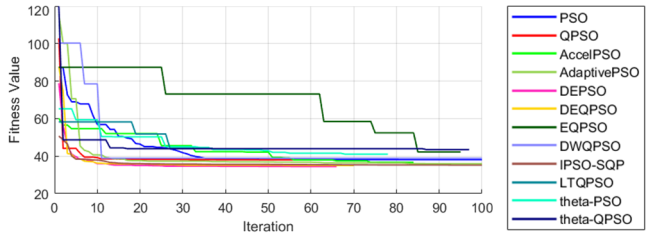

Fig. 5. Convergence curves of fitness values in 3D scenario

conventional PSO and QPSO under both scenarios. Conversely, EQPSO, θ-PSO and θ-QPSO performed poorly especially in 3D scenario. EQPSO is observed to be highly intolerant to local minimum when the dimensionality of the problem increases.

The simulation results of 2D and 3D scenarios are graphed in boxplots as shown in Fig. 6, Fig. 7, Fig. 8 and Fig. 9. In the boxplots, the mean of data is represented by the blue plus sign, the median by the red horizontal line, and the blue box on the plot indicates the range of 25th to 75th percentile. The acceptable data range is indicated by the black whisker, and the outliers are represented by red dots. In the fitness value plots, the extreme lowest end of each whisker gives the individual best fitness obtained by each algorithm over the 1000-run simulation, and the green cross sign represents the best known (lowest) fitness value among all algorithms in the simulations. The acceptable data range, percentile range and the outliers are indicators for the standard deviations or the stabilities of the performances, while the means and medians give information about the solution qualities and search abilities of the algorithms.

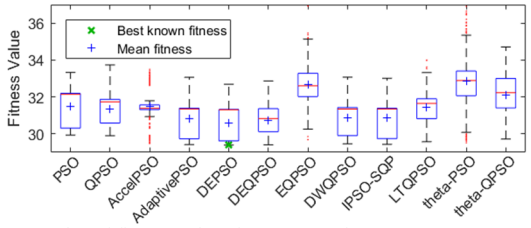

Fig. 6. Boxplot of fitness values in 2D scenario

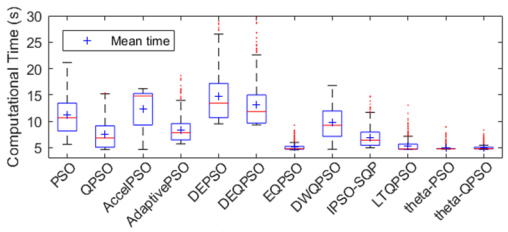

Fig. 7. Boxplot of computational time in 2D scenario

It can be seen on the boxplots that DEPSO outperformed other algorithms by achieving the lowest mean fitness value in both 2D and 3D, with its individual best fitness being the best known fitness value in 2D, and the second best known fitness value in 3D. Following closely the performance of DEPSO, DEQPSO has the second top mean fitness and second best known fitness in 2D; while for 3D, DEQPSO achieved the fourth top mean fitness and the best known fitness. It is also worth noting that DEQPSO achieved the lowest standard deviation for the fitness values in 2D, while DEPSO has the lowest standard deviation in 3D. These observations indicates that the hybridization of DE operation into the PSI- based algorithm offers great improvement to the searching ability and stability of the algorithms. However, as shown in Fig. 7 and Fig. 9, DEPSO and DEQPSO require significantly higher computational time compared to other algorithms, and the increase in computational time is even more obvious when the dimensionality increases to 3D. This is because greedy selection is used in the DE operation, and requires the fitness values of the particles to be evaluated

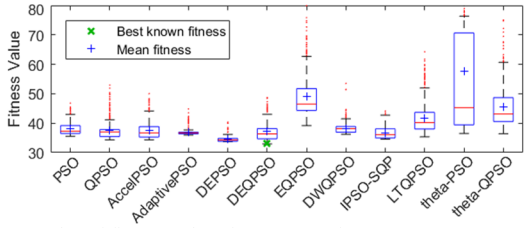

Fig. 8. Boxplot of fitness values in 3D scenario

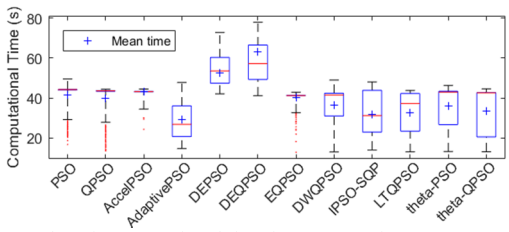

Fig. 9. Boxplot of computational time in 3D scenario

twice for comparison purpose, meaning an additional fitness evaluation for every particle in every iteration. Since the fitness evaluation usually contributes to the majority of the computational time, the greedy selection operator drastically increases the computational requirements of the algorithms.

Adaptive PSO and IPSO-SQP are also able to offer excellent performances; both generate higher solution quality than the conventional PSO and QPSO. In fact, the two algorithms show good balance between solution quality and computational time. In 2D, Adaptive PSO is ranked as the third in terms of mean fitness value, and IPSO-SQP is ranked as the fourth. While for 3D, IPSO-SQP has the second top mean fitness, and Adaptive PSO scored third. More importantly, both algorithms require less computational time than most algorithms, indicating their high efficiency in solving the path planning problem.

Although DWQPSO achieved a comparable mean fitness, its computational time is significantly higher than the average. Accelerated PSO and LTQPSO do not offer significant performance improvement over PSO and QPSO in terms of solution quality, despite that LTQPSO requires less computation time. EQPSO, θ-PSO and θ-QPSO are found to be performing poorly based on their poorer mean fitness. The extremely low computational time of the three algorithms in 2D indicates that they are prone to be trapped by local minimum. In 3D, the three algorithms have significantly high mean fitness values and high standard deviations, indicating a poor and unstable performance.

V. Conclusion

This paper presents a performance evaluation study to compare various PSI-based algorithms through the application in a novel AUV path planner. Based on the Monte Carlo simulation, both DEPSO and DEQPSO were identified to be outperforming the other algorithms with equivalently excellent performance in terms of solution quality, stability and convergence behaviour, thus proving that the DE hybridization offers significant improvement on the particles' searching ability. However, the computational requirement of the DE-hybridized algorithms was observed to be higher due to the greedy selection operator. Adaptive PSO and IPSO-SQP were also found to have excellent performances by achieving a balance between computational requirement and solution quality, with their solution qualities slightly lower than the DE-hybridized algorithms. Most importantly, DEPSO, DEQPSO, Adaptive PSO and IPSO-SQP are proven to be capable of generating high quality AUV paths.

The future works of this study can be extended in several directions. Firstly, the DE-hybridized algorithms could be improved by modifying the greedy selection operator to reduce their computational requirements, while maintaining similar excellent search abilities. Next, the possibility of integrating an adaptive mechanism, such as those employed in Adaptive PSO and IPSO-SQP, into the DE-hybridized algorithms can be considered. Lastly, it should be noted that this study considered only static obstacle and non-time-varying current in the problem space. The enormous potential of these high performance stochastic algorithms for the application in AUV path planning under realistic environmental conditions should be exploited.

References

[1] T. Xue, R. Li, M. Tokgo, J. Ri, and G. Han, "Trajectory planning for autonomous mobile robot using a hybrid improved QPSO algorithm," *Soft Computing,* vol. 21, no. 9, pp. 2421-2437, 2017.

[2] Z. Zeng, K. Sammut, L. Lian, F. He, A. Lammas, and Y. Tang, "A comparison of optimization techniques for AUV path planning in environments with ocean currents," *Robotics and Autonomous Systems,* vol. 82, pp. 61-72, 2016.

[3] J. Sun, C. H. Lai, and X. J. Wu, *Particle Swarm Optimisation Classical and Quantum Perspectives*. Boca Raton, FL: CRC Press, 2012.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. Proceedings of the 6th International Symposium on*, 1995, pp. 39-43: IEEE.

[5] Y. Qin, D. Sun, N. Li, and Q. Ma, "Path planning for mobile robot based on particle swarm optimization," *Robot,* vol. 26, no. 3, pp. 222-225, 2004.

[6] B. Sun, W. D. Chen, and Y. G. Xi, "Particle swarm optimization based global path planning for mobile robots," *Control and Decision,* vol. 20, no. 9, p. 1052, 2005.

[7] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Evolutionary Computation, 2004. CEC2004. Congress on*, 2004, vol. 1, pp. 325-331: IEEE.

[8] Y. Fu, M. Ding, C. Zhou, C. Cai, and Y. Sun, "Path planning for UAV based on quantum-behaved particle swarm optimization," in *Proceedings of SPIE - The International Society for Optical Engineering*, 2009.

[9] Z. B. Shi, Y. Li, X. Wang, T. Wang, and W. M. Li, "Path planning for mobile robot based on quantum-behaved particle swarm optimization," *Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology,* Article vol. 42, no. SUPPL. 2, pp. 33-37, 2010.

[10] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 39, no. 6, pp. 1362-1381, 2009.

[11] B. Song, Z. Wang, L. Zou, L. Xu, and F. E. Alsaadi, "A new approach to smooth global path planning of mobile robots with kinematic constraints," *International Journal of Machine Learning and Cybernetics,* pp. 1-13, 2017.

[12] H. Wang, H. Zhou, and H. Yao, "Research on autonomous planning method based on improved quantum Particle Swarm Optimization for Autonomous Underwater Vehicle," in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1-7: IEEE.

[13] X. S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *International Conference on Networked Digital Technologies*, 2011, pp. 53-66: Springer.

[14] A. Z. Mohamed, S. H. Lee, H. Y. Hsu, and N. Nath, "A faster path planner using accelerated particle swarm optimization," *Artificial Life and Robotics,* vol. 17, no. 2, pp. 233-240, 2012.

[15] W. M. Zhong, S. J. Li, and F. Qian, "θ-PSO: a new strategy of particle swarm optimization," *Journal of Zhejiang University-SCIENCE A,* vol. 9, no. 6, pp. 786-790, 2008.

[16] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 42, pp. 511-526, 2012.

[17] W. J. Zhang and X. F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003, vol. 4, pp. 3816-3821: IEEE.

[18] Y. Fu, M. Ding, C. Zhou, and H. Hu, "Route Planning for Unmanned Aerial Vehicle (UAV) on the Sea Using Hybrid Differential Evolution and Quantum-Behaved Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 43, 2013.

[19] H. Modares and M.-B. N. Sistani, "Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 3, pp. 476-484, 2011.

[20] T. Taleshian and S. Minagar, "Motion planning for an autonomous underwater vehicle," in *Knowledge-Based Engineering and Innovation (KBEI), 2015 2nd International Conference on*, 2015, pp. 285-290: IEEE.

[21] Q. Qian, M. Tokgo, C. Kim, C. Han, J. Ri, and K. Song, "A hybrid improved quantum-behaved particle swarm optimization algorithm using adaptive coefficients and natural selection method," in *Advanced Computational Intelligence (ICACI), 2015 Seventh International Conference on*, 2015, pp. 312-317: IEEE.

[22] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978.