Constrained path planning of autonomous underwater vehicle using selectively-hybridized particle swarm optimization algorithms

Hui Sheng Lim*, Shuangshuang Fan*, Christopher K.H. Chin*, Shuhong Chai*, Neil Bose, Eonjoo Kim***

** National Centre for Maritime Engineering and Hydrodynamics, Australian Maritime College, University of Tasmania, Launceston, TAS, 7250, Australia*
(e-mail: hui.lim; shuangshuang.fan; c.chin; shuhong.chai; eonjoo.kim@utas.edu.au).
*** Department of Ocean and Naval Architectural Engineering, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada*
(e-mail: nbose@mun.ca)

Abstract: This paper presents an autonomous underwater vehicle (AUV) path planning scenario as an optimization problem constrained by the combination of hard constraints and soft constraints. The path planner aims to generate the optimum path that safely guides an AUV through an ocean environment with priori known obstacles and non-uniform currents in both 2D and 3D. The path planner uses 2 variants of particle swarm optimization (PSO) algorithms, which are the selectively Differential Evolution (DE)-hybridized Quantum PSO (SDEQPSO) and Adaptive PSO (SDEAPSO). The performances of the path planners using different constraints are analyzed in a series of extensive Monte Carlo simulations and ANOVA (analysis of variance) procedures based on their respective solution qualities, stabilities and computational efficiencies. Based on the simulation results, the SDEQPSO path planner with the setting of hard constraint for boundary condition and soft constraint for obstacle avoidance was found to be able to generate smooth and feasible AUV path with higher efficiency than other algorithms, as indicated by its relatively low computational requirement and excellent solution quality.

Keywords: path planning, optimization problems, constraints, Monte Carlo simulation, autonomous vehicle

1. INTRODUCTION

To date, numerous efforts have been made in the attempt to enable the operation of AUVs in more dynamic and constrained environments. The exploration of AUVs in highly dynamic regions is challenging and possesses several technical issues, particularly for the path planning of the AUVs. An optimum AUV path planner should be able to determine a path that safely guides the AUV from a starting position to a destination in an ocean environment, based on either a minimum time or energy cost criterion.

Planning the path for the AUVs is essentially a multimodal and multi-objective optimization problem; numerous techniques have been proposed to solve this problem effectively and efficiently. Nonetheless, developing the algorithms for AUV path planning still faces several intrinsic difficulties, particularly in balancing the computational requirement and the performance of the path planner. Recently, Zeng et al. (2016), and Youakim and Ridao (2018) compared and classified various path planning techniques including Artificial Potential Field APF, search-based methods, sampling-based methods and optimization methods. APF method (Kruger et al., 2007) is fast and efficient, but very susceptible to local minima. Search heuristic-based planners such as Field D* (Ferguson and Stentz, 2006) and Fast Marching* FM* (Petres et al., 2007) are capable of generating optimal and robust path, but their computational efficiencies are limited to less complex and lower dimensional problems. Sampling-based methods like Rapidly-exploring Random Trees RRT (Rao and Williams, 2009) and its variants (Hernández et al., 2019) are effective for high-dimensional and highly time-constraint scenario, at the cost of the path optimality. Optimization methods such as the evolutionary algorithms (Alvarez et al., 2004, Witt and Dunbabin, 2008) show excellent performance in terms of solution optimality. They are effective for high-dimensional complex problems, but their practicality for implementation depends highly on the complexity of their mathematical functions. Among the existing evolutionary algorithms, Zeng et al. (2016) further pointed out that the particle swarm optimization (PSO)-based algorithms are remarkably robust and efficient for solving high-dimensional path planning problem. Lim et al. (2018) compared various PSO-based algorithms for AUV path optimization to identify their strengths and weaknesses. Inspired by these studies, Lim et al. (2019) proposed the selectively Differential Evolution (DE)-hybridized Quantum PSO (SDEQPSO) and Adaptive PSO (SDEAPSO), which were developed by hybridizing the PSO algorithm with DE operation based on a selective scheme. They were found to be capable of generating high quality path while maintaining a low computational requirement.

Since the implementation of PSO-based algorithms for path optimization is highly dependent on the mathematical model, it is critical to develop the path planner by formulating the appropriate cost functions and types of constraint. To ensure a smooth, feasible and collision-free path for the AUV, there are many conflicting criteria that need to be considered to achieve

an optimal control decision. These criteria involve trade-offs between the following objectives: 1) Determine the path with minimum travel time or energy cost; 2) Avoid collision and keep a safe distance with obstacles; 3) Ensure sufficient path control points are placed to generate the path; 4) Ensure the path satisfies the minimum turning radius and the pitch control limitation of the AUV. These criteria render the path planning scenario into a multi-objective optimization problem which can contain two classes of constraints: the hard constraints which must be satisfied by all solutions, and the soft constraints which may or may not be satisfied with different relative weightages (Jiang et al., 1995). The benefits of using a soft constraint over a hard one is that the soft constraint does not need to be satisfied in every iteration, instead, they can be optimized over the iterations; this reduces the solution generation time in every iteration during the optimization (Dariani et al., 2014). If a solution exceeds the soft constraints of the problem, penalty functions with predefined relative weightages can be applied to penalise the fitness of the solution. Choosing a right class of constraint for the path planning problem requires a balance between the computational efforts and the feasibilities of the solutions.

This paper presents a comprehensive comparison between different classes of constraints used for defining the AUV path planning problem, which is solved by using the SDEAPSO and SDEQPSO algorithms. The effect of the types of constraints on the performance of these stochastic PSO-based algorithms will be thoroughly analysed. For each test case, the path planning scenario with multiple obstacles and non-uniform current field was simulated in both 2-dimensional (2D) domain and 3-dimensional (3D) domain. Extensive Monte Carlo simulations were conducted for all test cases and the simulation results were analysed based on their respective solution qualities and stabilities.

The rest of this paper is arranged as follows. In Section 2, the overview of the algorithms used are provided. The formulation of the path planning problem is described in Section 3. Lastly, Section 4 presents the simulation setup, results and discussion.

2. OVERVIEW OF ALGORITHMS

2.1 APSO and QPSO Algorithms

Particle swarm optimization (PSO) is a heuristic population-based optimization algorithm introduced by Eberhart and Kennedy (1995). This algorithm 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 iteratively updated by the particle's own experience (cognitive behaviour) and the entire swarm's experience (social behaviour) to vary the particles' positions. In a standard PSO that consists 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:

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

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:

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

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

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

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

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, while w is the inertia weight for balancing the particle global exploration and local exploitation to improve the performance. Zhan et al. (2009) proposed Adaptive PSO (APSO), which uses an evolutionary factor f as an indicator representing the particles' evolutionary state to control these equation coefficients. To determine the evolutionary factor f, the mean distance d_i of the i^{th} particle to other particles is calculated using (6). f is then given by (7).

$$d_i = \frac{1}{N-1} \sum_{j=1,\, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k \right)^2} \tag{6}$$

$$f = \left(d_g - d_{\min} \right) \Big/ \left(d_{\max} - d_{\min} \right) \quad \in [0,1] \tag{7}$$

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 is calculated from f using (8), while C_1 and C_2 can be adapted using (9).

$$w = 1 \Big/ \left(1 + 1.5 e^{-2.6 f} \right) \quad \in [0.4, 0.9] \tag{8}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f-0.5|} \\ C_2 &= 3.2 - 2e^{-|f-0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{9}$$

Inspired by quantum mechanics and PSO, Sun et al. (2004) proposed the QPSO algorithm, which assumes the particles to have quantum behaviour. QPSO algorithm is well known to be an improved version of PSO. In QPSO, the position of the i^{th} particle is given as:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^{\,t} + \left(1 - \varphi_i^t\right) \cdot gbest^{\,t} \\ \quad + \beta \cdot \left| mbest^{\,t} - X_i^{\,t} \right| \cdot \ln\left(1/u_i^t\right), & \text{if } u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^{\,t} + \left(1 - \varphi_i^t\right) \cdot gbest^{\,t} \\ \quad - \beta \cdot \left| mbest^{\,t} - X_i^{\,t} \right| \cdot \ln\left(1/u_i^t\right), & \text{if } u < 0.5 \end{cases} \tag{10}$$

$$mbest^{\,t} = \sum_{i=1}^{N} pbest_i^t \big/ N \tag{11}$$

where u and φ are random positive numbers that are smaller 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 (11). 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 according to (12) is suggested for most applications (Sun et al., 2012).

$$\beta = \beta_{\max} - (t/t_{\max})(\beta_{\max} - \beta_{\min}) \tag{12}$$

2.2 SDEAPSO and SDEQPSO Algorithms

A selective hybridization of differential evolution (DE) operator with APSO and QPSO was proposed by Lim et al. (2019) to present the SDEAPSO and SDEQPSO algorithms, which were successfully applied to solve an unconstrained AUV path planning problem. Using the selective scheme, these proposed algorithms apply the DE operation to a selected number of particles only, instead of the entire swarm. The number of particles selected for DE operation, N_S, is controlled by a selective factor S as shown in (13). S is recommended to be 0.3 for AUV path planning problem by Lim et al. (2019) as this setting helps to promote swarm diversity while retaining an adequate group of potentially optimum particles.

$$N_S = N \times S, \quad S \in [0,1] \tag{13}$$

In SDEAPSO and SDEQPSO, the DE operation initiates by sorting all the particles in the entire swarm according to their personal best positions. Next, a number of selected particles with the best fitness undergo the mutation using (14) to generate the same number of mutated vectors U.

$$U_i^t = gbest^t + \left[(pbest_{i1}^t - pbest_{i2}^t) + (pbest_{i3}^t - pbest_{i4}^t) \right] / 2 \tag{14}$$

where i_1, i_2, i_3 and i_4 are randomly selected particle indices and $i_1 \neq i_2 \neq i_3 \neq i_4 \neq gbest$. The mutated vectors will then crossover with the personal best positions to generate the same number of trial vectors according to (15).

$$T_i^t = \left[t_{i,1}^t, ..., t_{i,j}^t, ..., t_{i,D}^t \right]$$
$$t_{i,j}^t = \begin{cases} u_{i,j}^t, & if\ r_j \le 0.85 \,||\, j = r \\ pbest_{i,j}^t, & if\ r_j > 0.85 \,||\, j \neq r \end{cases} \tag{15}$$

where r_j is a random number ranging from 0 to 1.0, and r is a random integer ranging from 1 to D. The trial vectors are then subjected to a natural selection operator, in which the same number of particles with the worst fitness are replaced by the trial vectors. Since only the worst particles are replaced in this process, all potentially best solutions will never deteriorate. Furthermore, the computational requirement of the algorithms will not be significantly affected because the natural selection operator does not involve fitness comparison between the particles, which requires additional particle fitness evaluation in every iteration. The DE operation with natural selection increases the diversity and the evolutionary rate of the entire swarm by eliminating the least desirable solutions, hence leading to a faster and better global convergence.

The implementation of SDEAPSO and SDEQPSO algorithms in AUV path planning can be conducted as described in the following pseudo code after selecting the appropriate parameters for the algorithm, i.e. the population size N, the number of particle dimensions D and the maximum number of iterations t_{max}.

Step 1. **Input** the algorithm parameters and environmental information of the ocean field.

Step 2. **Initialize** particles with random positions in (1) to represent an initial group of candidate paths. Set *pbest* to be the current particle positions.

Step 3. **While** the stop criteria is not met,

For $t = 1, 2, \ldots, t_{max}$,

SDEAPSO	***SDEQPSO***
Evaluate the cost function $f(X_i^t)$.	Compute *mbest* according to (11).
Update *pbest* and *gbest* according to (4) and (5) respectively.	Evaluate the cost function $f(X_i^t)$.
Update w, C_1 and C_2 according to (8) and (9) respectively.	Update *pbest* and *gbest* according to (4) and (5) respectively.
	Update β according to (12).

For each particle $i = 1, 2, \ldots, N$,

SDEAPSO	***SDEQPSO***
Update particle velocity and position according to (2) and (3) respectively.	Update particle position according to (10).

End

Sort all particles according to the fitness of their personal best positions.

For $k = 1, 2, \ldots, N_S^{th}$ best performing particle,

Mutation: Generate mutated vector U_k^t using (14)

Crossover: Generate trial vector T_k^t using (15).

Natural selection: Replace k^{th} worst performing particle with trial vector T_k^t.

End

End

Step 4. **Output** gbest that holds the optimal path when the stop criteria is met or when t_{max} is reached.

3. PROBLEM FORMULATION

3.1 Path Formulation

An AUV path planner is required to determine the optimal path among a group of potential paths for the AUV to travel toward a target location. Each potential path comprises a series of nodes from the start point to the endpoint. Optimizing the coordinates of path nodes will yield 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. In this paper, the entries of the position vector represent the polar/spherical 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 polar coordinates of n nodes in 2D, the position vector of a particle has 2n dimensions, including n dimensions for radial coordinate r and n dimensions for azimuthal angular coordinate φ. For the spherical coordinates of n nodes in 3D, a particle has 3n dimensions, including an additional n

dimensions for polar angular coordinate θ. The position vector of the i^{th} particle at t^{th} iteration for 3D can be given as follows:

$$X_i^t = \left[r_{i,1}^t, r_{i,2}^t, ..., \varphi_{i,n+1}^t, \varphi_{i,n+2}^t, ..., \theta_{i,3n-1}^t, \theta_{i,3n}^t \right] \quad (16)$$

The polar coordinates of a path node in 2D can be converted to Cartesian coordinates using (17), while spherical coordinates in 3D can be converted using (18).

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \end{aligned} \quad (17)$$

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ z_{i,1} &= r_{i,1} \cos \theta_{i,2n+1} \end{aligned} \quad (18)$$

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 (19), 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 $n+1$.

$$P(u) = \sum_{i=0}^{n+1} x_i \, B_{i,k}(u), \qquad i \in \{0, 1, 2, ..., n+1\} \quad (19)$$

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 (De Boor et al., 1978) as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \quad (21)$$

3.2 Path Fitness Evaluation

Suitable fitness evaluation function is required for PSO-based algorithms to measure the fitness of the particles accurately. Due to the high computational efficiency of PSO-based algorithms, fitness evaluation usually contribute to the majority of computational time (Sun et al., 2012). For path planning, which is a minimization problem, a lower cost/fitness indicates a better solution. In this paper, the main evaluation function is to measure the path fitness 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 (22).

$$F_1(X_i) = \sum_{j=1}^{m-1} \left[\left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \Big/ \left| V_g \right| \right], \quad j \in \{1, 2, ..., m-1\} \quad (22)$$

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 water reference velocity V_a. Thus, V_g is given as (23).

$$V_g = V_a + \left(V_c \cdot \overrightarrow{p_{i,j}\, p_{i,j+1}} \right) \Big/ \left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \quad (23)$$

3.3 Boundaries and Constraints

Two classes of constraints, namely hard constraint and soft constraint, are used in the AUV path planning problem in order to produce a smooth, feasible and collision-free path that satisfies the boundaries and the objectives, which include:

- *Obstacle avoidance:* Avoid collision and keep a safe distance from obstacles.
- *Radial boundary:* Ensure sufficient path nodes are placed.
- *Azimuthal boundary:* Ensure the path satisfies the minimum turning radius.
- *Polar boundary:* Ensure the path satisfies the pitch control limitation.

Different combinations of hard and soft constraints are applied to achieve these objectives in this paper. The test cases investigated are summarised in Table 1.

Table 1: Simulation test cases

Objectives	**Test cases**			
	HBHO	HBSO	SBHO	SBSO
Radial, azimuthal & polar boundaries	Hard	Hard	Soft	Soft
Obstacle avoidance	Hard	Soft	Hard	Soft

The hard constraints must be satisfied by all feasible solutions; while the soft constraints of different relative weightage may or may not be satisfied by the solution. If the hard constraints are violated by a solution, the particular solution will be regenerated. Meanwhile, if the soft constraints are violated, a penalty function with predefined relative weightage will be applied to penalise the fitness of the particle.

To achieve the obstacle avoidance, the path's exposure to threats/obstacles is required to be measured regardless of the class of constraint used. All obstacles in the problem space are modelled as eclipses in 2D, and as ellipsoids in 3D. The threat exposure is evaluated based on the intersection between the path and the obstacles. Assuming an obstacle 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 (24). 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 (25).

$$\left(\frac{x - O_{cx}}{O_{rx}} \right)^2 + \left(\frac{y - O_{cy}}{O_{ry}} \right)^2 + \left(\frac{z - O_{cz}}{O_{rz}} \right)^2 = 1 \quad (24)$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + s \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \quad (25)$$

Substituting (25) into (24) yields the following quadratic equations, which is expressed in term of s.

$$As^2 + Bs + C = 0 \quad (26)$$

$$A=\frac{O_{ry}{}^2O_{rz}{}^2(x_2-x_1)+O_{rx}{}^2O_{rz}{}^2(y_2-y_1)+O_{rx}{}^2O_{ry}{}^2(z_2-z_1)}{O_{rx}{}^2O_{ry}{}^2O_{rz}{}^2} \quad (27)$$

$$B=\frac{(O_{cx}-x_1)(x_1-x_2)}{0.5O_{rx}{}^2}+\frac{(O_{cy}-y_1)(y_1-y_2)}{0.5O_{ry}{}^2}+\frac{(O_{cz}-z_1)(z_1-z_2)}{0.5O_{rz}{}^2} \quad (28)$$

$$C=\frac{(O_{cx}-x)^2}{O_{rx}{}^2}+\frac{(O_{cy}-y)^2}{O_{ry}{}^2}+\frac{(O_{cz}-z)^2}{O_{rz}{}^2}-1 \quad (29)$$

The intersection of the path with the obstacle can be evaluated by obtaining the discriminant D of (26) according to (30).

$$D=B^2-4AC \quad (30)$$

A safe distance is added to the principal axes of all obstacle regions so that the AUV will keep a safe distance from the obstacles and collision will not occur when $D = 0$. If $D > 0$, the collision can be checked by determining (31).

$$s_1,\ s_2=\left(-B\pm\sqrt{D}\right)/2A \quad (31)$$

If $s_1<0$ and $s_2>1$, the path will not intersect with the obstacles, i.e. no collision, and hence the hard constraint is satisfied; otherwise, the path solution will be regenerated. For soft constraint, if the path intersect with the obstacles, the intersection points can be found by solving (26) with (31). The penalty for violating the soft constraint will be proportional to the length of segment containing within the obstacle region as shown in (32). When the soft constraint setting is used for obstacle avoidance, the global best solution of each iteration will still be hard-constrained (meaning the iteration will always continue until the global best solution is not penalised), in order to ensure the final solution is collision-free.

$$F_2(X_i)=\sum_{h=1}^{H}\sum_{j=1}^{m-1}\left\|\overrightarrow{S_1\ S_2}\right\|\Big/\left[2\times\max(O_{r,h})\right] \quad (32)$$

To ensure sufficient path nodes are placed to generate the path, each node is constrained to lie within a concentric annulus. The annuli are the regions bounded by every pair of adjacent concentric circles with predefined radii. To achieve this, the radial coordinates of the path nodes are constrained to a lower boundary $R_{\min}$ and an upper boundary $R_{\max}$.

$$\begin{aligned}R_{\min}&=\left[0, r_d, 2r_d, \ \ldots\ , r_{target}\right]\\ R_{\max}&=\left[r_d, 2r_d, 3r_d, \ \ldots\ , r_{target}\right]\end{aligned} \quad (33)$$

where r_d is the distance between two concentric circles and r_{target} is the radial coordinate of the target location. The number of path nodes, n is decided by r_d as defined by (34).

$$n=ceil\left[r_{target}/r_d\right] \quad (34)$$

where *ceil* is the rounding function toward positive infinity. The hard constraint will be satisfied if the path solution falls between the boundaries $R_{\min}$ and $R_{\max}$. If soft constraint is used, the following penalty function F_3 will be applied.

$$F_3(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ R_{\min,j}\ge r_{i,j}\ge R_{\max,j}\\ R_{\min,j}-r_{i,j} & ,\ if\ r_{i,j}<R_{\min,j}\\ r_{i,j}-R_{\max,j} & ,\ if\ r_{i,j}>R_{\max,j}\end{cases} \quad (35)$$

In order to ensure the minimum turning radius and the pitch limitation are satisfied, the search domain of azimuthal angular coordinate and polar angular coordinate are also constrained within the boundaries φ_{max} and θ_{max}. The path solution will satisfy the hard constraints if $|\varphi_{i,j}| < \varphi_{max}$ and $|\theta_{i,j}| < \theta_{max}$. For soft constraints, the penalty costs follow (36) and (37).

$$F_4(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ |\varphi_{i,j}|\le\varphi_{\max}\\ |\varphi_{i,j}|-\varphi_{\max} & ,\ if\ |\varphi_{i,j}|>\varphi_{\max}\end{cases} \quad (36)$$

$$F_5(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ |\theta_{i,j}|\le\theta_{\max}\\ |\theta_{i,j}|-\theta_{\max} & ,\ if\ |\theta_{i,j}|>\theta_{\max}\end{cases} \quad (37)$$

Using these optimization functions, the test cases in Table 1 are combined with the QPSO, SDEAPSO and SDEQPSO algorithms to solve the path planning problem. The path solutions generated by the path planner will then be validated by setting as the reference trajectory for a dynamic model of REMUS 100, which is an under-actuated AUV with path following controller. Based on Fossen's vectorial representation (Fossen, 1999) and SNAME (Society of Naval Architects and Marine Engineers) standard formulation, the 6 DOF equations of motion for a typical AUV can be modelled as shown in (38) and (39).

$$\dot{\eta}=\begin{bmatrix}R(\eta_2) & 0_{3\times3}\\ 0_{3\times3} & T(\eta_2)\end{bmatrix}\nu \quad (38)$$

$$M\dot{\nu}+C(\nu)\nu+D(\nu)\nu+g(\eta)=\tau \quad (39)$$

where R (η_2) and T (η_2) are the rotation matrices between inertial and body-fixed reference frames for the translational velocities and angular velocities respectively. η in (38) represents the position η_1 and the orientation η_2 of the vehicle with respect to the inertial reference frame, while ν includes the translational velocities ν_1 and the rotational velocities ν_2 of the vehicle with respect to the body-fixed reference frame as described in the vectors in (40).

$$\begin{aligned}\eta&=\begin{bmatrix}\eta_1 & \eta_2\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}x & y & z & \phi & \theta & \psi\end{bmatrix}^{\mathrm{T}},\\ \nu&=\begin{bmatrix}\nu_1 & \nu_2\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}u & v & w & p & q & r\end{bmatrix}^{\mathrm{T}}\end{aligned} \quad (40)$$

In (39), M and $C(\nu)$ describe the inertial and Coriolis matrices (including rigid body and added mass) respectively, while $D(\nu)$ is the hydrodynamics damping matrix, $g(\eta)$ is the hydrostatics restoring forces, and τ describes the control forces from the actuators. This study uses the REMUS 100 model derived from equations (38) – (40) by Prestero (2001). The AUV is controlled with a line-of-sight (LOS) guidance controller to follow the trajectory generated by the path planner. The controller uses the lookahead-based steering law described by Breivik and Fossen (2009), which is deemed suitable because of its lower computational requirement and validity for all cross-track errors. The desired yaw angle (heading) ψ_d is given by the control law in (41). A similar control law is also used for pitch control of the vehicle.

$$\psi_d(e)=\alpha_k+\arctan\left(-K_p e-K_i\int_0^t e(\tau)\,d\tau\right) \quad (41)$$

where α_k is the path-tangential angle, e is the cross-track error, and K_p and K_i are the proportional gain and the integral gain respectively. The integral action in (41) allows an under-actuated vehicle, such as the REMUS 100, to follow a path regardless of ocean current and non-zero sideslip angles.

4. SIMULATIONS

4.1 Simulation Setup

The AUV path planning was conducted in a 1000-run Monte Carlo simulation under 2D and 3D scenarios. 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 and priori known in the problem space. The AUV is required to travel with a pre-set water reference velocity of 1.5m/s. The safe distance for obstacle avoidance is set to 1 metre. r_d is set to 20 metres, while the angles $\psi_{\max}$ and $\theta_{\max}$ are set to 60° and 15° respectively. The population size was 150 particles. The algorithm parameters were set to be the values suggested in Section 2.

In addition to all the test cases described in Table 1, test cases with unconstrained path planning problem (uncon.) are also included for comparison purposes. It was discovered that the computational requirement of SDEAPSO path planner with hard-constrained obstacle avoidance is too high due to the nature of SDEAPSO's position and velocity update equations. Unlike QPSO and SDEQPSO which use the mean best position in their update equations, SDEAPSO has a stronger cognitive component in its equation, making it impossible to satisfy the hard constraint in a single iteration within a reasonable time frame, if the constraint is violated initially. Thus, the HBHO and SBHO cases for SDEAPSO were excluded.

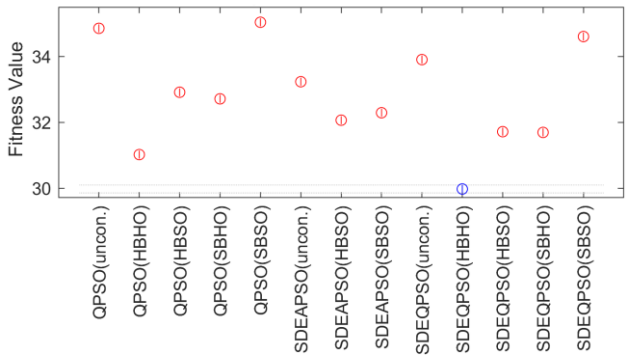

Fig. 1. ANOVA means of fitness values in 2D scenario

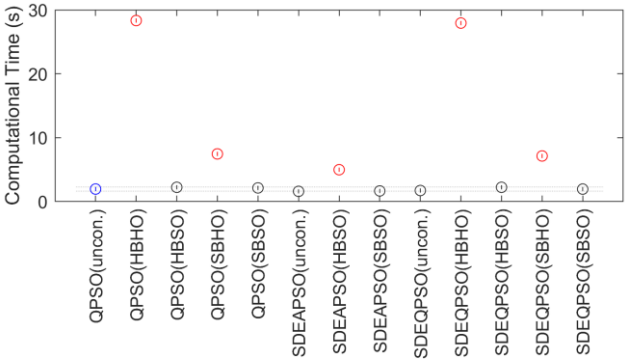

Fig. 2. ANOVA means of computational time in 2D scenario

4.2 Results and Discussion

The performances of the path planners in different test cases are compared based on their solution qualities, stabilities 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 target by travelling on the path. Thus, a lower fitness value indicates a higher solution quality. To comprehensively compare the test cases and the significance of the differences between their performances, a multiple comparison procedure, ANOVA (analysis of variance), was used in this study with a level of significance of 0.05. This procedure uses a 'stepdown' approach, which considers that all but one of the comparisons are less different than the range; such an approach is best suitable for all pairwise comparisons when the confidence intervals are not needed and sample sizes are equal (Sun et al., 2012). The ANOVA results of 2D and 3D scenarios are graphed in Fig. 1, Fig. 2, Fig. 3 and Fig. 4. The best performing results are in blue in the graphs, and those with statistically similar performance to the best performing one are coloured black.

In 2D scenario, it can be seen that SDEQPSO's HBHO case achieved the best (lowest) fitness value, and this is followed closely by QPSO's HBHO case. Although the HBHO case for SDEAPSO is inadequate for comparison, the HBHO setting is observed to have the best performance in terms of fitness value compared to other settings. However, by comparing the computational time, it was found that the HBHO setting has the highest computational requirement, roughly 10 times of computational time compared to others in 2D. The second-best fitness value was achieved by SDEQPSO's HBSO and SBHO cases. Despite similar performance, the SBHO case has much

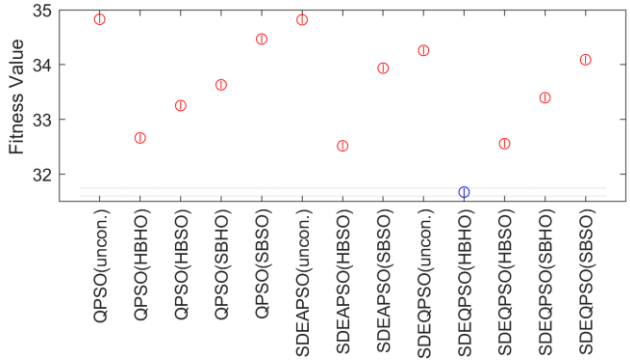

Fig. 3. ANOVA means of fitness values in 3D scenario

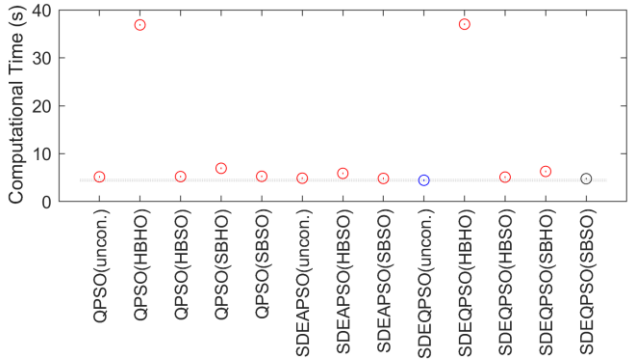

Fig. 4. ANOVA means of computational time in 3D scenario

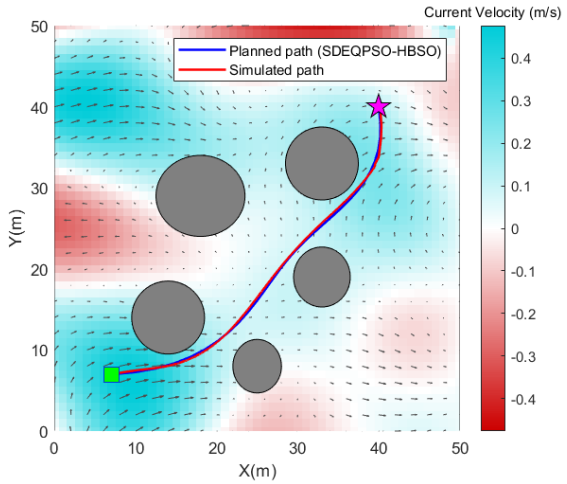

Fig. 5. Validation of path planning solution in 2D scenario

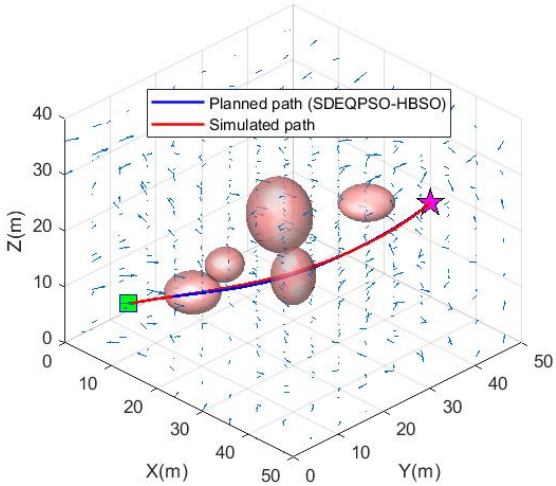

Fig. 8. Validation of path planning solution in 3D scenario

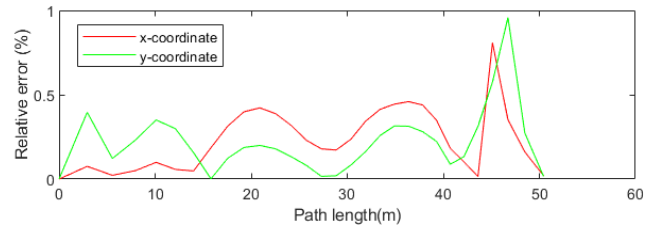

Fig. 6. Relative error of planned and simulated 2D path w.r.t. total path length

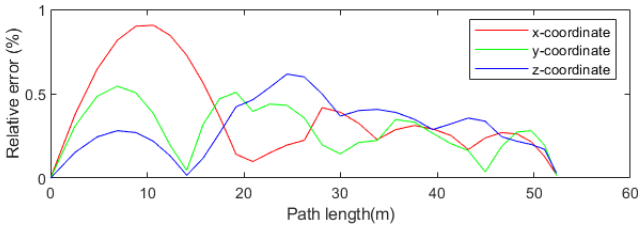

Fig. 9. Relative error of planned and simulated 3D path w.r.t. total path length

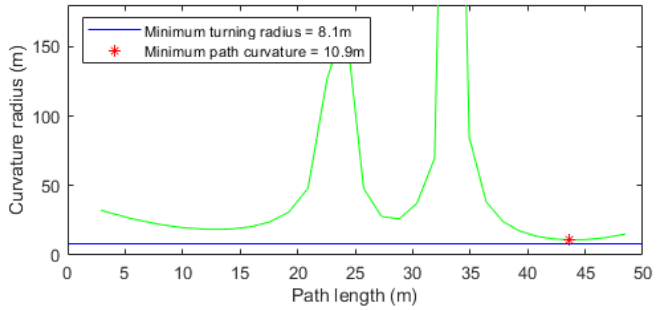

Fig. 7. Curvature radius of 2D simulated path

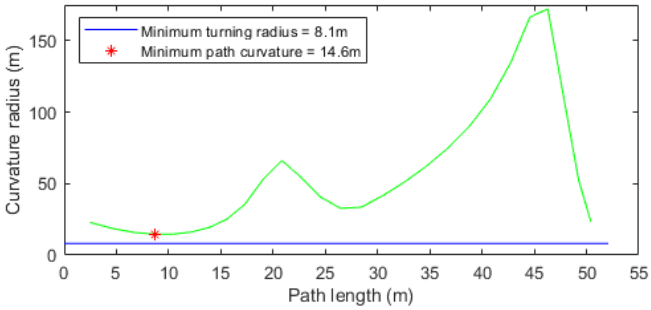

Fig. 10. Curvature radius of 3D simulated path

higher computational requirement in 2D. Based on this observation, it can be deduced that the hard constraint setting for obstacle avoidance is the main reason for the undesirable increase in computational requirement. When algorithm-wise comparison is made, it was found that the SDEQPSO has the best overall performance, although SDEAPSO has better performance in the unconstrained case and SBSO case. SDEAPSO was found to have lower performance whenever hard constraint is involved; this can be explained by its update equation which heavily relies on the cognitive component.

Similar performance trends are observed in 3D scenario. The HBHO cases achieved the best fitness value but at the cost of much higher computational requirement. SDEQPSO's HBSO case displays the second-best fitness value, while maintaining a relatively low computational requirement. SDEAPSO was again only able to outperform other algorithms when hard constraint is not involved. Hence, it can be concluded that the most suitable setting for AUV path planning is the HBSO setting (hard-constrained boundary conditions and soft-constrained obstacle avoidance), which is able to achieve an excellent performance in terms of fitness value without high computational requirement.

The 2D and 3D solutions generated by SDEQPSO with HBSO setting were validated by comparing against the simulated paths in Fig. 5 and Fig. 8. The AUV is required to travel from the starting point (green square) to the target (pink star) while keeping a safe distance from obstacles and trying to take advantage of the favourable current to assist the AUV motion. In 2D (Fig. 5), the blue-coloured zones indicate the favourable current while the red-coloured zones denote the less favourable current. Their respective relative errors in each of the x, y and z domains with respect to the total path length are graphed in Fig. 6 and Fig. 9. It can be observed that the simulated paths closely resemble the planned paths, with relative errors of well below 1% for both 2D and 3D scenarios. The feasibility of the path solutions is further checked by comparing against the minimum turning radius of REMUS 100, which has a minimum turning radius of 8.1 metres in the worst case scenario (Eng et al., 2015). The curvature radius must be higher than the minimum turning radius to satisfy the AUV motion limitation, which can be shown in Fig. 7 and Fig. 10 for the paths in 2D and 3D respectively. Therefore, the simulation results show that the path solutions generated by the proposed algorithm are smooth and feasible for the path planning application.

CONCLUSIONS

This paper evaluates the performance of an AUV path planner under different types of constraint settings. The SDEQPSO path planner with the setting of hard constraint for boundary condition and soft constraint for obstacle avoidance produced the best performance as shown by its high solution quality and computational efficiency. The path planners with hard constrained obstacle avoidance were found to have significantly higher computational requirement. Therefore, the soft constraint setting is recommended for obstacle avoidance of the path planner, with the safety and validity of the path guaranteed by having a hard constrained obstacle avoidance on the final solution of each iteration. The proposed path planner successfully generated a feasible and safe path for a REMUS 100 AUV, which was validated through the simulation of the AUV dynamic model.

Although the simulation assumed a priori known environment to represent the minimum capability of path planner, this path planner can be adapted to the realistic operational condition in future work due to the demonstrably high computational efficiency of this stochastic algorithm, which is suitable for solving compute-intensive problems such as path re-planning in highly dynamic environment. The future extension of this work will be explored by developing a path re-planning algorithm for a priori unknown environment with dynamic obstacles and spatiotemporal ocean current.

REFERENCES

Alvarez, A., Caiti, A. & Onken, R. 2004. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering,* 29**,** 418-429.

Breivik, M. & Fossen, T. I. 2009. Guidance laws for autonomous underwater vehicles. *Underwater vehicles.* InTech.

Dariani, R., Schmidt, S. & Kasper, R. 2014. Optimization based obstacle avoidance. *Optimization,* 1**,** 9999297.

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

Eberhart, R. & Kennedy, J. 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science. IEEE, 39-43.

Eng, Y., Teo, K. M., Chitre, M. & Ming Ng, K. 2015. *Online System Identification of an Autonomous Underwater Vehicle Via In-Field Experiments.*

Ferguson, D. & Stentz, A. 2006. Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics,* 23**,** 79-101.

Fossen, T. I. 1999. *Guidance and Control of Ocean Vehicles*, Norway, John Wiley & Sons.

Hernández, J. D., Vidal, E., Moll, M., Palomeras, N., Carreras, M. & Kavraki, L. E. 2019. Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics,* 36**,** 370-396.

Jiang, Y., Kautz, H. & Selman, B. 1995. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. 1st International Joint Workshop on Artificial Intelligence and Operations Research. 20.

Kruger, D., Stolkin, R., Blum, A. & Briganti, J. 2007. Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 4265-4270.

Lim, H. S., Fan, S., Chin, C. K. & Chai, S. 2018. Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner. 2018 IEEE OES Autonomous Underwater Vehicle Symposium. 1-7.

Lim, H. S., Fan, S., Chin, C. K. H., Chai, S. & Neil, B. 2019. Particle swarm optimization algorithms with selective differential evolution for AUV path planning (unpublished).

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J. & Lane, D. 2007. Path Planning for Autonomous Underwater Vehicles. *IEEE Transactions on Robotics,* 23**,** 331-341.

Prestero, T. T. J. 2001. *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle.* Massachusetts institute of technology.

Rao, D. & Williams, S. B. 2009. Large-scale path planning for underwater gliders in ocean currents. Australasian Conference on Robotics and Automation (ACRA).

Sun, J., Feng, B. & Xu, W. 2004. Particle swarm optimization with particles having quantum behavior. Proceedings of the 2004 congress on evolutionary computation. IEEE, 325-331.

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

Witt, J. & Dunbabin, M. 2008. Go with the flow: Optimal AUV path planning in coastal environments. Australian Conference on Robotics and Automation.

Youakim, D. & Ridao, P. 2018. Motion planning survey for autonomous mobile manipulators underwater manipulator case study. *Robotics and Autonomous Systems,* 107**,** 20-44.

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

Zhan, Z. H., Zhang, J., Li, Y. & Chung, H. S. H. 2009. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* 39**,** 1362-1381.