International Journal of Robotics and Automation (IJRA)
Vol.9, No.2, June 2020, pp. 94~112
ISSN: 2089-4856, DOI: 10.11591/ijra.v9i2.pp94-112

Particle swarm optimization algorithms with selective differential evolution for AUV path planning

Hui Sheng Lim[1], Shuangshuang Fan[2], Christopher K.H. Chin[3], Shuhong Chai[4], Neil Bose[5]
[1,2,3,4,5]National Centre for Maritime Engineering and Hydrodynamics,
Australian Maritime College, University of Tasmania, Australia
[5]Department of Ocean and Naval Architectural Engineering, Memorial University of Newfoundland, Canada

Article Info

Article history:

Received Jan 13, 2020
Revised Feb 17, 2020
Accepted Mar 4, 2020

Keywords:

Autonomous underwater vehicle
Hybridization
Monte Carlo methods
Particle swarm optimization
Path planning

ABSTRACT

Particle swarm optimization (PSO)-based algorithms are suitable for path planning of the Autonomous Underwater Vehicle (AUV) due to their high computational efficiency. However, such algorithms may produce sub-optimal paths or require higher computational load to produce an optimal path. This paper proposed a new approach that improves the ability of PSO-based algorithms to search for the optimal path while maintaining a low computational requirement. By hybridizing with differential evolution (DE), the proposed algorithms carry out the DE operator selectively to improve the search ability. The algorithms were applied in an offline AUV path planner to generate a near-optimal path that safely guides the AUV through an environment with a priori known obstacles and time-invariant non-uniform currents. The algorithm performances were benchmarked against other algorithms in an offline path planner because if the proposed algorithms can provide better computational efficiency to demonstrate the minimum capability of a path planner, then they will outperform the tested algorithms in a realistic scenario. Through Monte Carlo simulations and Kruskal-Wallis test, SDEAPSO (selective DE-hybridized PSO with adaptive factor) and SDEQPSO (selective DE-hybridized Quantum-behaved PSO) were found to be capable of generating feasible AUV path with higher efficiency than other algorithms tested, as indicated by their lower computational requirement and excellent path quality.

Corresponding Author:

Hui Sheng Lim,
National Centre for Maritime Engineering and Hydrodynamics,
Australian Maritime College, University of Tasmania,
Launceston, TAS, 7250, Australia.
Email: hui.lim@utas.edu.au

1. INTRODUCTION

AUVs are unmanned underwater vehicles that can be remotely programmed to conduct various missions, ranging from seabed survey, coastal mapping and environmental monitoring for scientific research purposes, to anti-submarine warfare for defence purposes. To date, numerous efforts have been made in the attempt to enable the operation of AUVs in more dynamic and constrained environments, such as shallow coastal areas, deep ocean regions and regions underneath ice shelves. The operation of AUVs in highly dynamic regions is challenging and it poses several technical issues, particularly for the path planning of the AUVs.

Planning the path for an AUV is essentially a multimodal optimization problem; numerous optimization 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 requirements and the performance of the path planner. The high computational requirements for planning the path in a realistic 3D environment may lead to excessive energy drain in an AUV. A common way to keep the computational requirements of path planner feasible is to reduce the problem to a 2D space [1]. This however compromises the performance of the path planner due to reduced amount of 3D information available for the path planner, such as currents field, bathymetry and obstacles in the ocean environment. Thus, a high computational-efficient algorithm is required for effective AUV path planning in realistic ocean environment.

Recently, Zeng, Sammut [2], and Youakim and Ridao [3] compared and classified various path planning techniques including Artificial Potential Field APF, search-based methods, sampling-based methods and optimization methods. The APF method [4] is fast and efficient, but very susceptible to local minima. Search heuristic-based planners such as Field D* [5] and Fast Marching* (FM*) [6] are capable of generating optimal and robust paths, but their computational efficiencies are limited to less complex and lower dimensional problems. Sampling-based methods such as Rapidly-exploring Random Trees RRT [7] and its variants RRT* [8] are effective for high-dimensional and highly time-constraint scenarios at the cost of the path optimality, and the resultant paths often require further refinement. Meta-heuristic optimization methods such as the evolutionary algorithms [9, 10] show excellent performance in terms of solution optimality. Evolutionary algorithms are effective for high-dimensional complex problems but they may converge to local minima within finite time. Among the existing evolutionary algorithms, Zeng, Sammut [2] further pointed out that the particle swarm optimization (PSO)-based algorithms are remarkably robust and efficient for solving high-dimensional path planning problems.

PSO algorithm and its most significant variant, the quantum-behaved PSO (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 [11]. Some pioneering examples of their applications in path planning can be found in [12-14]. PSO-based path planners are suitable for dynamic environments where online path planning is required because they maintain a large pool of solutions, which is available at any time during the mission. These solutions can serve as the initial solutions whenever path replanning is required, thus significantly improving the computational efficiency. Some successful applications of PSO-based algorithm in online path planning of AUV can be found in [15, 16]. Nonetheless, PSO-based algorithms are susceptible to convergence at local minimum solutions if the time allowed for path planning is limited, which is often the case in real AUV operations.

In recent years, many strategies that modified the PSO and QPSO algorithms have been proposed in order to improve their performances in path planning of various autonomous systems. Each of these variants of the algorithms claimed to have different improvements over the original PSO and QPSO algorithms. To benchmark the PSO and QPSO variants in the application of AUV path planning, a recent comparison study [17] classified and evaluated the algorithms based on their solution qualities, stabilities and computational efficiency. It was concluded from the results of [17] that the hybridization of differential evolution (DE) in PSO and QPSO, which were proposed by [18], are able to produce path planning solution with the highest quality due to their stronger resistance to local minima, but at the cost of higher computational requirements. Moreover, the findings of [17] suggested that having an adaptive mechanism in the evolution of particles in the PSO algorithm can produce solution quality that is second only to DE-hybridized algorithms, but with a relatively low computational requirement; the adaptive PSO (APSO) proposed by [19] was able to generate a path planning solution that achieves a balance between solution quality and computational requirements.

Inspired by the DE hybridization, a number of algorithms, namely SDEPSO (PSO with selective DE hybridization), SDEAPSO (PSO with adaptive factor and selective DE hybridization), and SDEQPSO (QPSO with selective DE hybridization), are proposed in this paper. These algorithms explore the strengths of DE-hybridized algorithms, and minimize their weaknesses in order to improve the algorithm performance. The proposed algorithms were implemented in an offline AUV path planner and their performance were benchmarked against other meta-heuristic algorithms because if the proposed algorithms can provide better computational efficiency to demonstrate the minimum capability of a path planner, then they will outperform the tested algorithms in a realistic online path planner. The objective of the AUV path planner is defined as finding a near-optimal path that safely guides the AUV from a starting position to a destination based on a minimum time criterion. The path planning scenario with a priori known obstacles and non-uniform current field was first simulated in a 2-dimensional (2D) domain, followed by the simulation in a 3-dimensional (3D) domain. Extensive Monte Carlo simulations were conducted on all algorithms 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 basic PSO, QPSO and their variants, including DEPSO, DEQPSO and APSO are provided. Section 3 describes the novel algorithms proposed in this paper. The formulation of the path planning problem is outlined in Section 4. Section 5 presents the simulation setup, results and discussions. The generated path solutions were then validated using an AUV simulator of REMUS 100 in Section 6. Finally, Section 7 concludes the paper along with the future research directions.

2. REVIEW ON PSO AND ITS VARIANTS

This section presents the overview of various particle swarm intelligence based algorithms used for developing the novel algorithms, which include the basic PSO, basic QPSO and their variants.

2.1. PSO algorithm

Introduced by Eberhart and Kennedy [20], PSO algorithm is a heuristic population-based optimization algorithm inspired by the analogues of cognitive abilities and social interaction in animals. The algorithm consists of particles that move within a multidimensional search space to find the 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 algorithm 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 can be denoted as:

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

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 $(t+1)$th iteration are updated by (2) and (3) respectively. *pbest* and *gbest* are determined based on the particle's fitness *f(X)* and its previous best fitness *f(pbest)* as shown in (4) and (5).

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

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

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

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

In (2), r_1 and r_2 are uniform distributed random positive numbers that are less than 1.0. C_1 and C_2 denote the acceleration coefficients for cognitive and social components respectively; they are both set to 2.0 for most applications [21]. Parameter w is the inertia weight introduced by [22] for balancing the global exploration and local exploitation of the particles. A common strategy is to set the inertia weight at an initial $w_{\max}$ value of 0.9, and linearly decreasing to a $w_{\min}$ value of 0.4 according to (6) as the iteration progresses [23, 24].

$$w = w_{\max} - \frac{t}{t_{\max}}(w_{\max} - w_{\min}) \tag{}$$

where $t_{\max}$ is the maximum number of iterations defined for the algorithm.

To confine the particles within the search space, the particle velocity denoted by *V* is usually bound to an interval of $[-V_{max}, V_{max}]$, where the maximum velocity V_{max} is recommended to be 10% to 20% of the dynamic range of the variables [24, 25].

2.2. QPSO algorithm

Inspired by the mechanics of quantum system and dynamical analysis of the PSO algorithm, Sun, Feng [26] proposed the QPSO algorithm. In QPSO, the position of the *i*th particle can be updated using the following stochastic equation:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^t + (1-\varphi_i^t) \cdot gbest^t + \beta \cdot |mbest^t - X_i^t| \cdot \ln(1/u_i^t), if\ u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^t + (1-\varphi_i^t) \cdot gbest^t - \beta \cdot |mbest^t - X_i^t| \cdot \ln(1/u_i^t), if\ u \geq 0.5 \end{cases} \tag{7}$$

$$mbest^t = \sum_{i=1}^{N} pbest_i^t \Big/ N \tag{8}$$

where u is a uniform distributed random positive number that is less than 1.0, φ is a uniform distributed random positive number that is less than 1.0 , and *mbest* is the mean best position which is defined as the average of personal best positions of all particles in the swarm as shown in (8). β is known as the contraction-expansion (CE) coefficient, which is the most critical parameter for tuning the convergence behaviour of QPSO. As suggested by the empirical study of parameter selection in [11], a linearly decreasing β from a maximum value $\beta_{\max}$ of 1.0 to a minimum value $\beta_{\min}$ of 0.5 according to (9) is suitable for most applications.

$$\beta = \beta_{\max} - \frac{t}{t_{\max}}(\beta_{\max} - \beta_{\min}) \tag{9}$$

2.3. DEPSO and DEQPSO algorithms

One of the most effective method used for improving the PSO-based algorithm is by hybridization, in which the beneficial features of other optimization techniques is combined with PSO or QPSO algorithm. In [27], the basic PSO was combined with Differential Evolution (DE), resulting in a hybrid algorithm known as DEPSO. Based on the inspiration from DEPSO, [18] applied a similar hybridization concept in QPSO to propose DEQPSO. In both DEPSO and DEQPSO, the particles undergo the usual position update operations, followed by a successive three-step DE operation, which is the mutation, crossover and selection as described below.

- Mutation: A mutated donor vector U is first generated using (10):

$$U_i^t = gbest^t + \frac{(pbest_{r1}^t - pbest_{r2}^t) + (pbest_{r3}^t - pbest_{r4}^t)}{2} \tag{10}$$

  where r_1, r_2, r_3 and r_4 are randomly selected particle indices that are mutually different, and different from the current index i and the particle index of global best position, i.e. $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \neq gbest$.
- Crossover: A trial vector T is generated to increase the diversity, by conducting crossover between the donor vector and personal best position as shown in (11).

$$\begin{gathered} T_i^t = [\tau_{i1}^t, \dots, \tau_{ij}^t, \dots, \tau_{iD}^t] \\ \tau_i^t = \begin{cases} u_{ij}^t \ , & if\ r_j \leq CR \ \| \ j = r \\ pbest_{ij}^t \ , & if\ r_j > CR \ \| \ j \neq r \end{cases} \end{gathered} \tag{11}$$

  where CR is the crossover probability which is suggested to be 0.85, r_j is a uniform distributed random number ranging from 0 to 1.0, and r is a random positive integer ranging from 1 to the number of particle dimensions D.
- Selection: A greedy selection is used to decide whether the trial vector T should replace the current position X in the $(t+1)$th iteration. The fitness of T will be evaluated and compared with X. X will only be replaced if T has better fitness value; otherwise X will be retained. This means the hybridization of the DE operation will never deteriorate the solution, but only make it better or remain unchanged.

DEPSO and DEQPSO algorithms were applied to solve the path planning problem of Unmanned Aerial Vehicle (UAV) in [18], and has proven to be capable of generating significantly higher solution quality than basic PSO and QPSO algorithms.

2.4. APSO algorithm

In basic PSO, the acceleration coefficients C_1 and C_2, and inertia weight w in the update equation are important for maintaining the balance between the global exploration and local exploitation of the particles. Zhan, Zhang [19] proposed an adaptive PSO (APSO), in which an evolutionary factor is used as an indicator representing the evolutionary state of the particles to control the equation coefficients adaptively. To determine the evolutionary factor, the mean particle distance d_i of the ith particle to other particles has to be calculated using (12). The evolutionary factor f is then computed according to (13).

$$d_i = \frac{1}{N-1}\sum_{j=1, j\neq 1}^{N}\sqrt{\sum_{k=1}^{D}(x_i^k - x_j^k)^2} \tag{12}$$

$$f = (d_g - d_{\min})/(d_{\max} - d_{\min}) \quad \in [0,1] \tag{13}$$

where d_g is the mean particle distance of the global best particle, $d_{\min}$ and $d_{\max}$ are the minimum and maximum of the mean particle distances respectively. The inertia weight w can be calculated from evolutionary factor f using (14). The adaptation of the acceleration coefficients C_1 and C_2 can also be achieved using the evolutionary factor as shown in (15).

$$w = 1/(1 + 1.5e^{-2.6f}) \quad \in [0.4,0.9] \tag{14}$$

$$C_1 = 0.8 + 2e^{-|f-0.5|}$$
$$C_2 = 3.2 + 2e^{-|f-0.5|}, \text{ where } C_1 + C_2 = 4 \tag{15}$$

3. METHODOLOGY: SELECTIVE DE HYBRIDIZATION

Although DEPSO and DEQPSO algorithms are able to generate excellent solution qualities for AUV path planning, the hybridization of DE significantly increases the computational requirements of the algorithm due to the greedy selection operator used in the DE operation [17]. The greedy selection operator requires the fitness of the particles to be evaluated twice for comparison purposes, meaning an additional fitness evaluation for every particle in every iteration. As the fitness evaluation process usually contributes to the majority of the computational time [11], the greedy operator drastically increases the computational requirements of the algorithms. The increase in computational requirements due to the addition of greedy selection operator will be even more obvious when the complexity and the dimensionality of the problem increase [17]. In order to minimize the downside of DE operator in PSO-based algorithm, a selective hybridization scheme is proposed in this paper to present the following algorithms:
- SDEPSO (PSO with selective DE hybridization)
- SDEAPSO (PSO with adaptive factor and selective DE hybridization)
- SDEQPSO (QPSO with selective DE hybridization)

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 (16).

$$N_S = N \times S, \quad S \in [0,1] \tag{16}$$

The DE operation in the proposed algorithms was modified by replacing the greedy selection operator with a natural selection operator. The DE operation proposed in this paper 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 and crossover operators, similar to those in DEPSO and DEQPSO, to generate the same number of trial vectors. The trial vectors are then subjected to a natural selection operator, in which the same number of particles with the worst fitness is replaced by the trial vectors.

As only the worst particles are replaced in this process, all potentially best solutions will never deteriorate. Furthermore, the computational requirements 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 theoretically.

The selective DE hybridization was applied to PSO and QPSO algorithms to develop the SDEPSO and SDEQPSO algorithms in this paper. In addition, another algorithm, namely SDEAPSO, was developed by adding an adaptive mechanism to the control of inertia weight and acceleration coefficients in PSO algorithm, similarly to the APSO algorithm. The implementation of SDEPSO, SDEAPSO and SDEQPSO algorithms in AUV path planning can be conducted as described in the following pseudocode.

```
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,
   Step 3.1 For $t$ = 1, 2, …, $t_{max}$,
```

SDEPSO	*SDEAPSO*	*SDEQPSO*
Evaluate the cost function $f(X_i^t)$. Update *pbest* and *gbest* according to (4) and (5) respectively. Update w according to (6).	Evaluate the cost function $f(X_i^t)$. Update *pbest* and *gbest* according to (4) and (5) respectively. Update w, C_1 and C_2 according to (14) and (15) respectively.	Compute *mbest* according to (8). Evaluate the cost function $f(X_i^t)$. Update *pbest* and *gbest* according to (4) and (5) respectively. Update β according to (9).

```
    Step 3.2 For each particle $i$ = 1, 2, …, $N$,
```

SDEPSO	SDEAPSO	SDEQPSO
Update particle velocity and position according to (2) and (3) respectively.	Update particle velocity and position according to (2) and (3) respectively.	Update particle position according to (7).

```
             End
    Step 3.3 Sort all particles according to the fitness of their personal best positions.
    Step 3.4 For $k$ = 1, 2,…, $N_S$th best performing particle,
                   Mutation: Generate mutated vector $U_k^t$ according to (10).
                   Crossover: Generate trial vector $T_k^t$ according to (11).
                   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.
```

3.1. Complexity Analysis

The time complexity of the proposed algorithms can be measured by counting the number of primitive operations in the algorithm. By referring to the pseudocode of the proposed algorithms, the number of operations can be counted as follows:

- In Step 2, initialization contributes one operation for N times.
- In Step 3.1, cost function evaluation contributes one operation for N times; finding *pbest* requires $N\cdot \log(N)$ operations; finding *gbest* requires $\log(N)$ operations; updating coefficients contributes one operation; SDEQPSO requires N additional operations to calculate *mbest.*
- In Step 3.2, SDEPSO and SDEAPSO perform N loops with 14 operations; SDEQPSO perform N loops with 12 operations.
- Step 3.3 contributes $\log(N)$ operations.
- Step 3.4 performs N_S loops with 8 operations.

Steps 1 – 3.2 are the standard operations in basic PSO, APSO and QPSO, whereas Step 3.3 and 3.4 are the additional operations introduced by the selective DE operator. O-notation is used in this work to denote the asymptotic upper bound of time complexity, which indicates the computational time of the algorithm in the worst case scenario. When computing the O-notation, the lower order terms in the number of operations is negligible because their impact on computational time are relatively insignificant for large input [28]. The highest order term in the operation is $N\cdot \log(N)$ in Step 3.1, and it performs $t_{\max}$ times to check the termination condition. The operations added by the selective DE operator (Step 3.3 and 3.4) are of lower order and do not have significant impact on the time complexity. Thus, the complexity of the proposed algorithms in linear form is $O(N\cdot \log(N)\cdot t_{\max})$, similar to the standard PSO algorithm. PSO-based algorithms have two inner loops when going through the population of N particles, and one outer loop for $t_{\max}$ iterations; this renders the time complexity to be $O(N^2\cdot t_{\max})$ in the extreme case. The spatial complexity of the algorithms is $O(N^2)$, which depends on the population size.

3.2. Benchmark Functions

Metaheuristic algorithms such as the PSO-based algorithms can be evaluated empirically by comparing their performance in solving a set of objective function problems. In addition to the AUV path planning problem, a number of non-linear continuous function problems were used to study and benchmark the characteristics of the proposed algorithms. According to the “no free lunch” (NFL) theorem [29], the development and evaluation of an algorithm for a specific problem should be based on the benchmark function problems of similar class and properties, because the algorithm performance will not be consistent for every kind of problem. Thus, these benchmark functions were selected based on their resemblances to the properties of path planning problem. The selected benchmark functions should have the following properties:

- Multimodal with deceptive local minima and one global minimum, because the path planning problem usually consists of multiple suboptimal paths and an optimal path.
- Multi-dimensional, because the dimensionality of the path planning problem is dependent on the number of control waypoints along the path.

Four test functions were chosen for benchmarking in this study. These minimization problem functions, which are commonly used to evaluate the characteristics of optimization algorithms, were found to exhibit the abovementioned properties. The information on the selected benchmark functions are given in Table 1. The dimensions of all functions are set to 20 in this study.

Table 1. Benchmark functions

Notation	Name	Function formulation	Boundary interval	Global minimum
*F*1	Griewank [30]	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i==1}^{d}\cos\left(\frac{x_1}{\sqrt{i}}\right)+1$	[-600, 600]	$f(x)=0$, at $x=(0,\ldots,0)$
*F*2	Rastrigin [31]	$f(x)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$	[-5.12, 5.12]	$f(x)=0$, at $x=(0,\ldots,0)$
*F*3	Ackley [32]	$f(x)=-20e^{\frac{1}{d}\sqrt{\sum_{i=1}^{d}x_1^2}}-e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}+20+e$	[-32, 32]	$f(x)=0$, at $x=(0,\ldots,0)$
*F*4	Schwefel [33]	$f(x)=418.9829d\sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]	$f(x)=0$, at $x=(420.9687,\ldots,420.9687)$

3.3. Empirical Study on Parameter Selection

In SDEPSO, SDEAPSO and SDEQPSO, the number of best performing particles that undergo the DE operation and the number of worst performing particles that will be replaced during the natural selection are determined by the selective factor *S*. Thus, *S* can be manipulated to control the diversity of the population. In order to study the effects of *S* on the algorithm performance, an empirical study is conducted on SDEPSO by using a range of *S*. The selective factor *S* is a positive number that is less than 1.0. Note that when *S* = 0, the algorithm will not be hybridized with DE at all; while *S* = 1 means the DE operation will be conducted on the entire swarm, and the entire swarm will be replaced during the natural selection, meaning all the solutions generated from the PSO operation will be discarded, which is undesirable. Therefore, the empirical study includes *S* values ranging from 0 to 0.9, meaning that 0%–90% of the particles will undergo the DE operation; the results for *S* = 0 are included for comparison purposes.

Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a population size of 150 particles, the performance of SDEPSO under different *S* settings is evaluated by solving the optimization problems of the benchmark functions and the path planning problem in 2D and 3D scenarios; the formulation of the path planning problem is described in Section 4.

Prior to evaluating the algorithm performance, Shapiro-Wilk test was performed to examine the normality of the obtained simulation data. The normality test revealed that the data was not normally distributed. Hence, the median was used as the indicator for solution quality. The median of fitness obtained (*Med.*) and the best known fitness (*Best*) for each setting of *S* were obtained for all problems and tabulated in Table 2. A lower fitness value indicates a higher solution quality and hence a stronger search ability.

Table 2. Empirical study results

Selective factor, *S*	*F*1		*F*2		*F*3		*F*4		2D path planning ($\times 10^2$)		3D Path planning ($\times 10^2$)	
	Med	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*
0	0.86	0.25	1.28	0.41	0.19	0.06	2.61	1.54	3.07	2.97	3.36	3.30
0.10	0.58	**0.13**	1.22	0.42	**0.15**	0.06	2.22	1.20	3.06	2.99	3.20	3.14
0.20	**0.56**	**0.13**	1.20	0.50	**0.15**	0.07	2.08	0.69	3.01	2.97	3.34	**3.13**
0.30	0.63	0.19	**1.15**	**0.21**	0.17	**0.05**	1.89	0.85	**2.98**	**2.91**	**3.18**	3.14
0.40	0.68	0.34	1.29	0.51	0.23	0.08	1.90	0.81	3.06	2.96	3.30	3.15
0.50	0.66	0.30	1.27	0.40	0.26	0.12	1.71	0.65	3.12	3.02	3.44	3.15
0.60	0.73	0.14	1.41	0.52	0.32	0.11	1.70	0.63	3.05	2.98	3.42	3.18
0.70	0.80	0.34	1.61	0.50	0.47	0.10	1.71	0.60	3.00	2.98	3.33	3.19
0.80	0.87	0.43	1.59	0.84	0.74	0.26	1.51	0.57	3.05	2.97	3.22	3.19
0.90	1.00	0.85	1.77	0.61	1.67	0.43	**1.27**	**0.48**	3.08	2.97	3.35	3.25

The best-performing results for each of the problems are in bold in Table 2. It can be observed from the results that the behaviour of the algorithms varies greatly as *S* increases, and the variations are not consistent for all problems. The best results for the majority of the problems are identified to be in the range of *S* = 0.1 – 0.3, except for problem *F*4. Such results can be explained by the geometry of the Schwefel function *F*4, which has all its local minima and the global minimum spread far apart from one another. Effective optimization of this function requires an algorithm that promotes larger solution diversity (higher *S*), so that it provides a jumping-out ability to prevent trapping in deceptive local minima. This actually complies with the NFL theorem, which suggests that no single algorithm can generate better performance than any other algorithms for every problem. In fact, the improved algorithm performance in one class of problem is not necessarily consistent in all kinds of problems; instead, it is exactly traded with performance in another class of problem [29]. Although all the function problems selected for benchmarking purposes have similar properties (they are all multimodal and multi-dimensional), the geometry of the problems are different. Therefore, the setting of S should be adjusted accordingly for different optimization problems. Based on this empirical study, it can be deduced that the optimal setting of S for the majority of the tested problems is in the range of 0.1 – 0.3. More specifically for the path planning problem, the setting of S = 0.3 was found to be appropriate and effective.

3.4. Benchmark Study

The benchmark functions were used to evaluate and benchmark the proposed algorithms in this study. Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a population size of 150 particles, the performances of the proposed algorithms in solving the optimization problems of the four benchmark functions were compared with other existing PSO-based algorithms. At each run, the initial particle positions for all problems were randomly generated based on the uniform distribution within the boundary intervals given in Table 3.

As the data was not normally distributed according to the Shapiro-Wilk test, the Kruskal-Wallis test [34], which is a non-parametric ANOVA (analysis of variance), was used with a significance level of 0.05 to rank the algorithm performance based on the solution qualities (fitness obtained). The ranking procedure used the Holm–Bonferroni 'stepdown' approach [35], which is best suited for all pairwise comparisons when the confidence intervals are not needed and sample sizes are equal [11]. The algorithms are given the same rank if they are not statistically different from one another. The medians (*Med.*) of fitness obtained, the ANOVA ranks (*#R*) and the medians of computational time required were tabulated in Table 3. The medians of the top two best-performing results for each problem are in bold. The overall performances of the algorithms are given by their total ranks, which are calculated from the summation of the ranks of the algorithm for all problems.

Based on the results, it can be seen that there is no single algorithm that can achieve the best results for all problems; this observation agrees with the NFL theory. For the Griewank function (F_1), DEQPSO produced the best result. In fact, APSO, SDEAPSO, QPSO, DEQPSO, and SDEQPSO algorithms were found to be producing satisfactory results, indicating that the adaptive mechanism and quantum behaviour of the particles are beneficial for solving this problem. DEPSO and SDEPSO algorithms produced equally good performance for the Rastrigin function (F_2). For the Ackley function (F_3), the QPSO-based algorithms, i.e. QPSO, DEQPSO and SDEQPSO produced the best performance, followed by the adaptive PSO-based algorithms, i.e. APSO and SDEAPSO. As far as the Schwefel function (F_4) is concerned, only DEPSO, SDEPSO and SDEAPSO are able to generate satisfactory results, while all the other algorithms seem to have inferior performances.

The total ranking of the algorithms reveal that DEQPSO achieved better overall performance than other algorithms. The second-best performing algorithms are found to be DEPSO and SDEAPSO. Most importantly, the results for all problems show that the fully DE-hybridized algorithms, i.e. DEPSO and DEQPSO required significantly higher computational time to obtain the solutions, while the selectively DE-hybridized algorithms are able to maintain a reasonably similar computational requirement as the PSO, QPSO and APSO algorithms.

Table 3. Benchmark study results

Algorithm	*F1*			*F2*			*F3*			*F4*			Total
	Med	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	Rank
PSO	0.658	8	0.102	1.372	5	0.123	0.453	8	0.104	3.617	5	0.125	26
QPSO	0.089	3	0.160	1.791	6	0.150	0.005	1	0.166	4.555	8	0.187	18
APSO	0.100	4	0.155	1.219	4	0.162	0.041	5	0.177	3.606	5	0.202	18
DEPSO	0.634	6	0.427	**1.140**	1	0.548	0.166	6	0.419	**1.781**	1	0.470	14
DEQPSO	**0.064**	1	0.510	2.092	7	0.502	**0.002**	1	0.490	3.023	4	0.555	13
SDEPSO	0.629	6	0.108	**1.149**	1	0.135	0.172	6	0.177	**1.891**	2	0.199	15
SDEAPSO	0.098	4	0.161	1.196	3	0.157	0.035	4	0.181	2.031	3	0.273	14
SDEQPSO	**0.072**	2	0.177	2.125	7	0.181	**0.002**	1	0.191	3.594	5	0.271	15

4. PROBLEM FORMULATION FOR PATH PLANNING

The AUV path planning problem is formulated in this section. Throughout the formulation, the AUV is assumed to have constant thrust, and hence constant water reference velocity.

4.1. Path Formulation

In this paper, the primary objective of the AUV path planner is to solve a multimodal non-linear optimization problem, in which the optimal path among a group of potential paths for the AUV to travel towards a target location through the ocean environment is required to be determined. Each potential path of the AUV comprises a series of nodes along the path from the start point to the target (end) point. Controlling and optimizing the coordinates of the path nodes will yield the optimized path for the AUV. The start point and the endpoint of the path are not involved in the optimization because all the potential paths share the same start and end locations.

In PSO-based algorithm, each potential path solution for the problem is modelled as an individual particle in the swarm population. The swarm population is denoted by a matrix $X = [X_1, X_2, \ldots, X_N]^T$, where X is the position vector of the particles and N is the number of particles in the swarm. The entries of the position vectors for the particles represent the coordinates of the path nodes. Assuming every path consists of n+2 nodes including the start point and endpoint, the number of nodes involved in the optimization is n. In order to record the coordinates of n nodes, the entries of the position vector for a particle in 2D problem space will have 2n dimensions, while a particle in 3D will have 3n dimensions. Thus, the respective position vectors of the i^{th} particle at t^{th} iteration for 2D and 3D problems are:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t, x_{i,n+1}^t, \ldots, x_{i,2n}^t\right], \quad i \in \{1,2,\ldots,N\} \tag{17}$$

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t, x_{i,n+1}^t, \ldots, x_{i,3n}^t\right], \quad i \in \{1,2,\ldots,N\} \tag{18}$$

Based on the path nodes including the start and end points, B-spline geometry is used to construct the AUV path. B-spline are parametric curves generated from a series of connected piecewise polynomials [36], which are suitable for modelling the AUV path because of its continuity for smooth path and locality for path alteration without loss of continuity. The path nodes act as the control points for the B-spline curve according to the following curve function, which gives the 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 is one less than the number of control points, which is n+1.

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

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 [37] as follows.

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

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

The continuity of the spline is fully dependent on the basis functions. Hence, it can be noted from (19) that the control points, i.e. path nodes can be adjusted during the path optimization process without affecting the spline continuity.

4.2. Evaluation Functions

When implementing PSO-based algorithms in an optimization problem, it is critical to develop the suitable cost evaluation functions to measure the fitness of the particles based on their respective solutions. Due to the high computational efficiency of PSO-based algorithms, the evaluation functions usually contribute to the majority of the computational time [11]. The functions are developed based on the optimization criteria of the problem. They must closely resemble the physical conditions of the problem space to provide an accurate cost representation model for finding the optimal solution. For path planning, which is a minimization problem, a lower cost/fitness indicates a better solution. The main criteria for evaluating the AUV path are:

- Minimum length or travel time required to reach the target
- Minimum exposure to the threats
- Compliance with physical motion limitations of AUV

As the optimum of all criteria does not necessarily coincide, a trade-off between these criteria can be established using a weighting scheme with multiple evaluation functions, which include a main evaluation function to measure the path length/time cost, a function to measure the threat cost along the path, and functions to measure the compliance of the path with respect to the AUV motion limitations. Thus, the fitness of a particle/path X_i can be given by a combination of several evaluation functions F_k for different criteria, with each criterion weighted by a cost factor f_k.

$$F(X_i^t) = \sum_{k=1}^{K} f_k F_k\,(X_i^t), \quad k \in \{1,2,\dots,K\} \tag{22}$$

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

4.3. Path Travel Time Cost

The main evaluation function for path planning problem is to measure the path cost based on its length or time to travel on the path. This study focuses on finding an optimal path that is capable of taking advantage of favourable current to assist the AUV motion, while avoiding the less favourable current to achieve a shorter travel time. For this purpose, a travel-time-based evaluation function is developed in this study.

Based on previous formulation, a given path X_i can be represented as a series of path nodes or alternatively in the form of discretised waypoints $P = [p_{i,1}, p_{i,2}, \dots, p_{i,m}]$, where P is the output from B-spline function and m is the total number of discretised waypoints. The travel time cost F_1 along a path can be determined by finding the sum of discretised time required to travel on each small path segment that connects the consecutive discretised waypoints in P.

$$F_1(X_i) = \sum_{j=1}^{m-1} \frac{\left\|\overrightarrow{p_{\iota,J}\,p_{\iota,J+1}}\right\|}{\left|V_g\right|}, \quad j \in \{1,2,\dots,m-1\} \tag{23}$$

where V_g is the ground reference velocity of the AUV, which is the resultant AUV velocity under the effect of surrounding ocean current. 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, which is essentially the direction of the path vector. Thus, V_g is given by the sum of V_a and the contribution of V_c as shown in (24).

$$V_g = V_a + \frac{V_c \cdot \overrightarrow{p_{\iota,J}\,p_{\iota,J+1}}}{\left\|\overrightarrow{p_{\iota,J}\,p_{\iota,J+1}}\right\|} \tag{24}$$

4.4. Threat Cost

The obstacles avoidance ability of the path planner relies on the threat cost evaluation function, in which the exposure of the path to threats/obstacles is measured. All threats in the problem space are modelled as ellipses (or circles if the major axis and minor axis are equal) under 2D condition, and as ellipsoids (or spheres if all the principal axes are equal) under 3D condition with their principal axes aligned with the coordinate axes. A threat cost evaluation method based on the intersection between the path and the threats is employed in this study.

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 (25). The parametric 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 (26). The cost evaluation in 2D takes a similar approach, except that the dimension reduction in 2D reduces the number of variables and hence simplifies the computation.

$$\left(\frac{x - O_{cx}}{O_{rx}}\right)^2 + \left(\frac{y - O_{cy}}{O_{ry}}\right)^2 + \left(\frac{z - O_{cz}}{O_{rz}}\right)^2 = 1 \tag{25}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + s \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{26}$$

Substituting (26) into (25) yields the following equations, which are expressed in terms of s. The intersection of the path with the threat can be evaluated by obtaining the discriminant ξ of (27) according to (31).

$$As^2 + Bs + C = 0 \tag{27}$$

$$A = \frac{{O_{ry}}^2{O_{rz}}^2(x_1 - x_2) + {O_{rx}}^2{O_{rz}}^2(y_1 - y_2) + {O_{rx}}^2{O_{ry}}^2(z_1 - z_2)}{{O_{rx}}^2{O_{ry}}^2{O_{rz}}^2} \tag{28}$$

$$B = \frac{(O_{cx} - x_1)(x_1 - x_2)}{0.5{O_{rx}}^2} + \frac{(O_{cy} - y_1)(y_1 - y_2)}{0.5{O_{ry}}^2} + \frac{(O_{cz} - z_1)(z_1 - z_2)}{0.5{O_{rz}}^2} \tag{29}$$

$$C = \frac{(O_{cx} - x)^2}{{O_{rx}}^2} + \frac{(O_{cy} - x)^2}{{O_{ry}}^2} + \frac{(O_{cz} - x)^2}{{O_{rz}}^2} - 1 \tag{30}$$

$$\xi = B^2 - 4AC \tag{31}$$

A safety margin is added to the principal axes of all threat regions so that the AUV will not conflict with the threat when $\xi = 0$, i.e. the path is tangent to the threat region. When $\xi > 0$, the path will conflict with the threat if the roots s_1 and s_2 given by (32) are within the range of $0 \leq s_1, s_2 \leq 1$.

$$s_1, s_2 = \frac{-B \pm \sqrt{\xi}}{2A} \tag{32}$$

If the path conflicts with the threat, the threat cost will be proportional to the length of the path segment contained in the threat region as given in (33). The intersection points, S_1 and S_2 can be determined by solving (27) using the obtained s_1 and s_2, and substituting them back into (26).

$$F_2(X_i) = -\sum_{h=1}^{H}\sum_{j=1}^{m-1} \frac{\|\overrightarrow{S_1S_2}\|}{2 \times \max(O_{r,h})} \tag{33}$$

4.5. Physical Motion Limitations

The considerations for physical motion limitations of AUV should include its yaw (turning) and pitch motions at a given forward speed. Evaluation functions are developed to check the compliance of the path with respect to these limitations, and to penalise the cost if any of the limitations is violated. 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 Figure 1), the turning angle ψ can be obtained from the cosine function as shown in (34). The first part of the function is the scalar projection of the second path segment on the first segment in the *x-y* plane, while the second part is the length of the second path segment in the *x-y* plane.

$$\psi_j = \cos^{-1}\left[\frac{\left\|\overrightarrow{p'_{i,j}\,p'_{i,j+1}}\right\| \cdot \left\|\overrightarrow{p'_{i,j+1}\,p'_{i,j+2}}\right\|}{\left\|\overrightarrow{p'_{i,j}\,p'_{i,j+1}}\right\|} \times \frac{1}{\left\|\overrightarrow{p'_{i,j+1}\,p'_{i,j+2}}\right\|}\right] \tag{34}$$

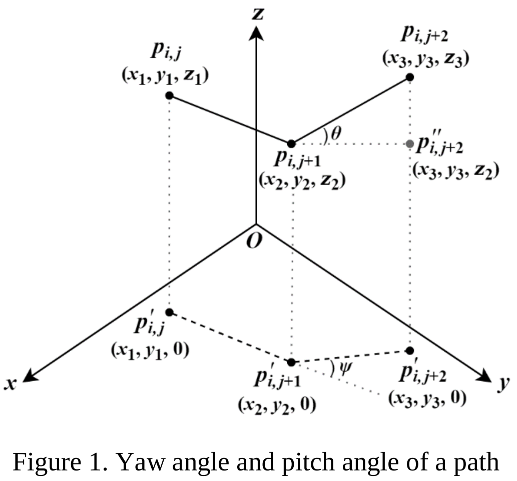

Figure 1. Yaw angle and pitch angle of a path

The cost F_3 for violating the yaw limitation is obtained from the calculated turning angle as shown in (35).

$$F_3(X_i) = -\sum_{j=1}^{m-1} F_3(p_{i,j})$$
$$F_3(p_{i,j}) = \begin{cases} 0 & , if\ |\psi_j| \leq \psi_{max} \\ (90 - |\psi_j|)/(90 - \psi_{max}) & , if\ |\psi_j| > \psi_{max} \end{cases} \tag{35}$$

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 Figure 1, θ can be determined using the basic tangent function as shown in (36). Next, $\Delta\theta$ can be calculated using (37).

$$\theta_j = \tan^{-1}\left[\frac{\left\|\overrightarrow{p_{i,j+2}\, p''_{i,j+2}}\right\|}{\left\|\overrightarrow{p_{i,j+1}\, p''_{i,j+2}}\right\|}\right] \tag{36}$$

$$\Delta\theta_j = \theta_{j+1} - \theta_j \tag{37}$$

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 shown in (38) and (39) respectively.

$$F_4(X_i) = -\sum_{j=1}^{m-1} F_4(p_{i,j})$$
$$F_4(p_{i,j}) = \begin{cases} 0 & , if\ |\theta_j| \leq \theta_{max} \\ (90 - |\theta_j|)/(90 - \theta_{max}) & , if\ |\theta_j| > \theta_{max} \end{cases} \tag{38}$$

$$F_5(X_i) = -\sum_{j=1}^{m-2} F_5(p_{i,j})$$
$$F_5(p_{i,j}) = \begin{cases} 0 & , if\ |\theta\Delta_j| \leq \Delta\theta_{max} \\ (90 - |\Delta\theta_j|)/(90 - \Delta\theta_{max}) & , if\ |\Delta\theta_j| > \Delta\theta_{max} \end{cases} \tag{39}$$

5. SIMULATIONS

The performance of the proposed algorithms is evaluated in the AUV path planning problem under different scenarios in this section.

5.1. Simulation Setup

The path planning of the AUV was conducted in a 1000-run basis Monte Carlo simulation under a 2D scenario, followed by a 3D scenario. The machine used has Intel Core i5-6300U CPU @ 2.4GHz with 8GB RAM. The problem spaces of the simulations were assumed to be a current field that consists of 500×500 square grids for 2D, and 500×500×500 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 current field was generated based on the data obtained from the field experiment conducted at Beauty Point, Tasmania, Australia.

The AUV is required to travel from a starting point to a target with a pre-set water reference velocity of 1.5m/s. Based on the properties of REMUS 100 AUV, 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, so that all costs except the travel time cost have similar impact on the solutions. Hence, when the path solution is not violating the threat exposure (f_2) and the physical motion limitations ($f_3 - f_5$), the fitness value of the solution directly represents the time required for the AUV to travel on the corresponding path.

In each simulation run, the maximum number of iterations for the algorithm was set to 100 with a pre-defined stopping threshold. This means the algorithm will be iterated up to a maximum number of 100, but will be stopped whenever the solution difference between iterations is less than the pre-set threshold.

The population size of all algorithms was set to 150 particles, with each particle consisting of 5 path nodes, meaning each particle has 10 dimensions for the 2D problem and 15 dimensions for the 3D problem. All algorithm parameters were set to be their respective suggested values as discussed in Section 2. For comparison purposes, another path planning technique, RRT* and other metaheuristic algorithms, including Ant Colony Optimization (ACO) [38], Firefly Algorithm (FA) [16], Differential Evolution (DE) and Genetic Algorithm (GA) [9], are also tested in this study.

5.2. Simulation Results

The performances of the algorithms are compared based on the following properties: 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 value of a solution is simply the time required (cost) for the AUV to reach the endpoint from the starting point by travelling on the path corresponding to the solution. Therefore, a lower fitness value indicates a higher solution quality and hence a stronger search ability.

The Monte Carlo simulation results of the 2D and 3D scenarios are graphed and compared in boxplots as shown in Figure 2 and Figure 3. The data was not normally distributed based on the Shapiro-Wilk normality test. In the boxplots, the medians of the data are represented by the red horizontal line; the blue boxes indicate the range of 25th to 75th percentile; the black whiskers indicate the acceptable data range. For the boxplots of fitness values, 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 ranges and percentile ranges are indicators for the stabilities of the algorithms performances, while the medians give information about the solution qualities and search abilities of the algorithms.

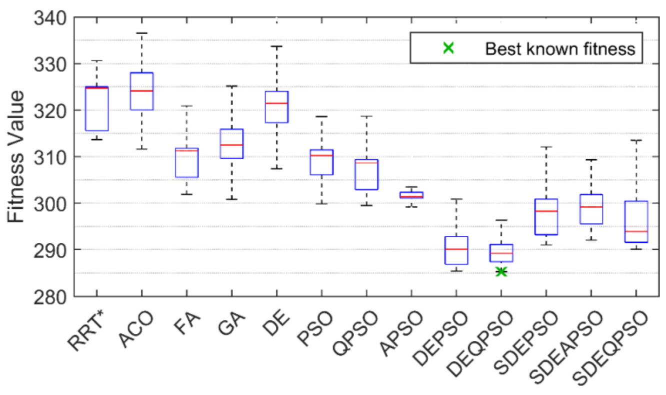

Figure 2. Boxplot of fitness values in 2D scenario

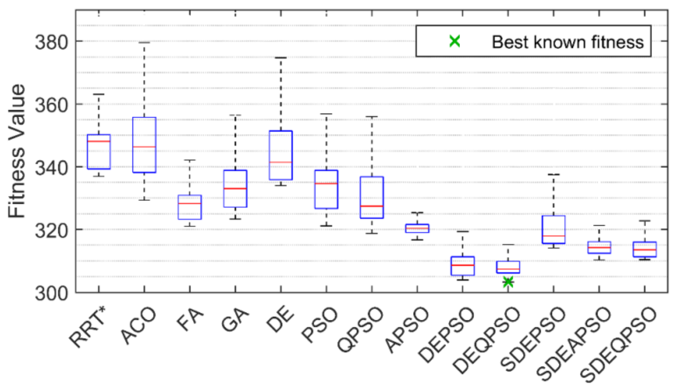

Figure 3. Boxplot of fitness values in 3D scenario

The Kruskal-Wallis ANOVA procedure with a significance level of 0.05 was used to rank the solution qualities (fitness values) based on the Holm–Bonferroni stepdown method. The algorithms are given the same rank if they are not statistically different from one another. Detailed results of the path planning simulation, including the median of fitness obtained (*Med.*), the best known fitness (*Best*), the interquartile range (*IQR*), the ANOVA rank (*#R*), the median of computational time (*T*) and the total ranks, are tabulated in Table 4. The total ranks are calculated from the summation of the ranks for the 2D and 3D scenarios. The ranking of the algorithms does not consider the impact of computational time.

Based on Figure 2, Figure 3 and Table 4, almost all the PSO-based algorithms have better solution quality than RRT* and other metaheuristic algorithms, with the exception of standard PSO being outperformed by FA. Despite having lower solution quality, RRT* has the shortest computational time in both 2D and 3D scenarios. It can also be seen that all variants of PSO and QPSO produced better solution qualities than the standard PSO and QPSO. DEPSO and DEQPSO outperformed all other algorithms by achieving the lowest medians for fitness value in both 2D and 3D. The total ranks of DEPSO and DEQPSO suggest that the two fully DE-hybridized algorithms are able to produce the top two best solution qualities for path planning problem. However, the computational time of DEPSO and DEQPSO are significantly higher than all the other algorithms due to the high computational requirements of the greedy selection operator.

Table 4. Path planning simulation results

Algorithm	2D					3D					Total Rank
	Med. ($\times10^2$)	*Best* ($\times10^2$)	*IQR*	*#R*	*T(s)*	*Med.* ($\times10^2$)	*Best* ($\times10^2$)	*IQR*	*#R*	*T(s)*	
RRT*	3.25	3.14	9.4	11	4.8	3.48	3.37	10.9	11	14.3	22
ACO	3.24	3.12	8.0	13	9.4	3.46	3.29	17.6	11	41.3	24
FA	3.11	3.02	6.2	8	9.2	3.28	3.21	7.7	7	41.2	15
GA	3.13	2.98	6.3	10	12.3	3.33	3.23	11.7	9	48.3	19
DE	3.21	3.05	6.7	11	12.8	3.41	3.34	15.5	11	53.6	22
PSO	3.10	3.00	5.4	8	10.7	3.35	3.21	12.1	9	34.6	17
QPSO	3.09	3.00	6.4	7	9.9	3.27	3.19	13.2	7	30.9	14
APSO	3.01	2.92	1.3	5	10.8	3.20	3.17	2.6	5	37.7	10
DEPSO	2.90	2.85	5.9	1	22.4	3.09	3.04	5.9	1	69.0	2
DEQPSO	2.89	2.85	3.7	1	20.8	3.07	3.03	3.7	1	76.7	2
SDEPSO	2.98	2.91	7.7	6	12.8	3.18	3.14	8.8	5	35.7	11
SDEAPSO	2.99	2.92	6.2	3	14.9	3.14	3.10	3.7	4	38.8	7
SDEQPSO	2.94	2.90	7.8	3	13.7	3.13	3.10	4.7	3	37.6	6

The solution qualities of SDEAPSO and SDEQPSO are second to the fully DE-hybridized algorithms; they are ranked similarly in 2D based on the ANOVA ranking. APSO has better solution quality than SDEPSO in 2D. It is worth noting that APSO has the lowest interquartile range is both 2D and 3D, indicating the highest stability among all the algorithms. In the 3D scenario, SDEQPSO is ranked slightly higher than SDEAPSO, while SDEPSO is ranked similar to APSO. The total ranks of the overall performance in both 2D and 3D reveal that SDEQPSO and SDEAPSO are ranked as the third and the fourth respectively. More importantly, the computational times of the two selectively DE-hybridized algorithms are significantly lower than the fully DE-hybridized algorithms and very close to other PSO-based algorithms. These indicate the higher computational efficiency of SDEQPSO and SDEAPSO in solving the path planning problem because they are able to produce solution quality that is very close to DEPSO and DEQPSO, while having a significantly lower computational requirement. In terms of problem size, the computational time required by the path planner is considered short, particularly in comparison to the computational time required for estimating the ocean environment based on the AUV sensory measurements.

6. VEHICLE PATH VALIDATION

For validation purposes, the path solutions generated by the AUV path planner were used as a reference trajectory for a dynamic model of REMUS 100. This section briefly explains the dynamic model and the path following controller used.

6.1. Dynamic Model

Based on Fossen’s vectorial representation [39] and SNAME (Society of Naval Architects and Marine Engineers) standard formulation, the 6 DOF equation of motion for a typical AUV can be modelled as shown in (40) and (41).

$$\dot{\eta} = \begin{bmatrix} R(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & T(\eta_2) \end{bmatrix} v \tag{40}$$

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{41}$$

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 includes the position η_1 and the orientation η_2 of the vehicle with respect to the inertial reference frame, while the derivative of η in (40) represents its rate of change. v includes the translational velocities v_1 and the rotational velocities v_2 of the vehicle with respect to the body-fixed reference frame as described in the vectors in (42).

$$\begin{aligned} \eta &= [\eta_1 \quad \eta_2]^T = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T, \\ v &= [v_1 \quad v_2]^T = [u \quad v \quad w \quad p \quad q \quad r]^T \end{aligned} \tag{42}$$

In (41), M and $C(v)$ describe the inertial and Coriolis matrices (including rigid body and added mass) respectively, $D(v)$ 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 (40)–(42) by [40]. The hydrodynamics coefficients calculated in [40] are used in the vehicle model.

6.2. Path Following Controller

The path following controller of the AUV model used the integral line-of-sight (iLOS) guidance law to set the yaw and pitch angles for following the trajectory generated by the path planner. The iLOS guidance law described by [41] allows the AUV to shape the convergence towards the path in the presence of ocean current and environmental disturbance. The desired iLOS yaw angle (heading) ψ_d and pitch angle θ_d can be given as follows:

$$\begin{aligned} \psi_d(e) &\triangleq \arctan\left(\frac{e + K_{i,y}e_{int}}{\Delta_y}\right), \quad K_{i,y}, \Delta_y > 0 \\ \dot{e}_{int} &= \frac{\Delta_y e}{\left(e + K_{i,y}e_{int}\right)^2 + \Delta_y^2} \end{aligned} \tag{43}$$

$$\begin{aligned} \theta_d(h) &\triangleq \arctan\left(\frac{h + K_{i,z}h_{int}}{\Delta_z}\right), \quad K_{i,z}, \Delta_z > 0 \\ \dot{h}_{int} &= \frac{\Delta_z h}{\left(h + K_{i,z}h_{int}\right)^2 + \Delta_z^2} \end{aligned} \tag{44}$$

where e is the cross-track error, h is the vertical-track error, $K_{i,y}$ and $K_{i,z}$ are the integral gains, and Δ_y and Δ_z represent the look-ahead distances for iLOS heading and pitch respectively. The integral terms of cross-track error e_{int} and vertical-track error h_{int} will produce non-zero ψ_d and θ_d even when the AUV is on the planned path, allowing the vehicle to counteract any effects of ocean current with the necessary side-slip and pitch angles. The rates of integral terms $\dot{e}_{int}$ and $\dot{h}_{int}$ will reduce the integral action with large cross-track and vertical-track errors (i.e. vehicle is far from the planned path), in order to minimize the risk of integrator wind-up.

6.3. Validation Results

The feasibility of the path solutions is first checked by comparing against the motion limitation of REMUS 100, which has a minimum turning radius of 8.1 metres in the worst case scenario [42]. The curvature radius of a feasible path must be higher than the minimum turning radius. The paths generated by SDEQPSO satisfy the AUV motion limitation as shown in Figure 4.

Next, the 2D and 3D solutions generated by SDEQPSO were validated by comparing against the simulated paths in Figure 5. The AUV is required to travel from the starting point (green square) to the target (pink star) without running into obstacles, while trying to take advantage of the favourable current to assist the AUV motion. In the 2D results, the blue-coloured regions indicate the favourable current while the red-coloured regions denote the less favourable current. In both results, the solid sections of the planned paths indicate that the favourable ocean current has a positive effect on the AUV motion while the dashed sections suggest otherwise. It can be observed that the paths are able to follow the favourable current and avoid the less favourable current to achieve a shorter travel time. The simulated paths closely resemble the planned paths in both scenarios.

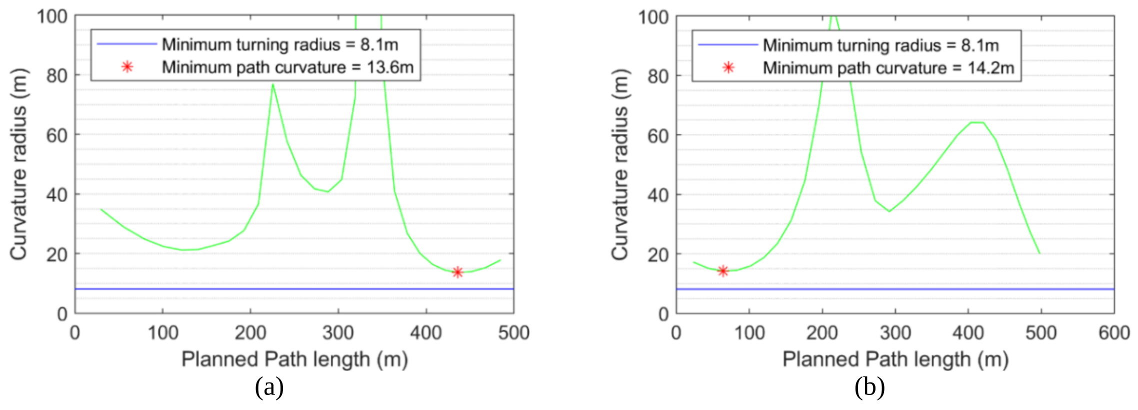

Figure 4. Curvature radius of planned path for (a) 2D and (b)3D

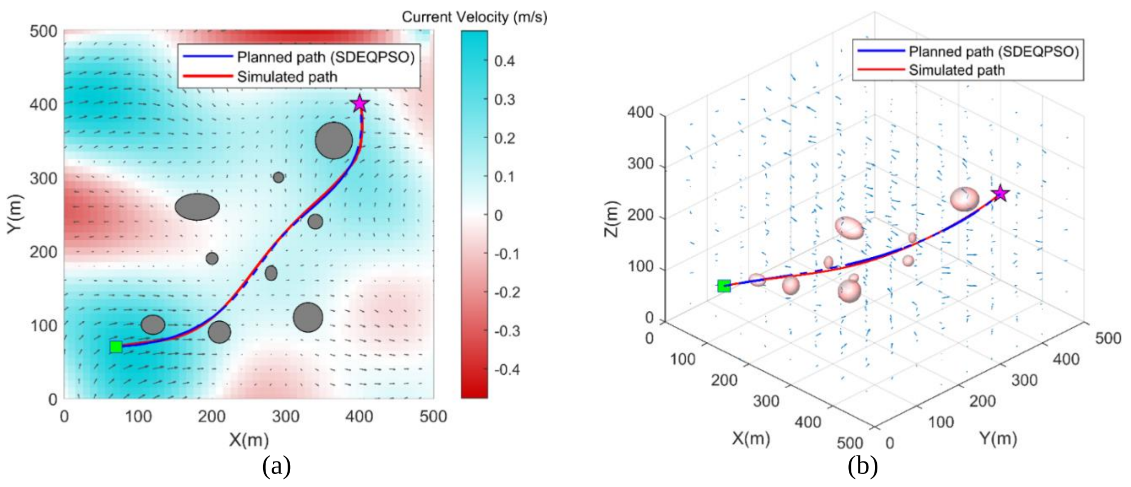

Figure 5. Validation of path solution in (a) 2D scenario and (b) 3D scenario

The cross-track errors of the simulated paths relative to the planned paths for the 2D and 3D scenarios are graphed in Figure 6. The errors for both scenarios are well below 1 metre, proving that the AUV was able to follow the planned paths closely. Hence, the simulation results showed that the path solutions generated by the proposed algorithm are smooth and feasible for the path planning application.

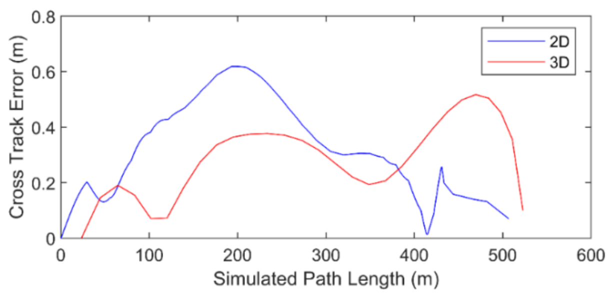

Figure 6. Cross-track error of simulated path relative to planned path

7. CONCLUSION

By selectively hybridizing with differential evolution, this paper presents new variants of PSO algorithm with improved search ability for the global minimum path of an AUV without increasing the computational requirements. The proposed algorithms were benchmarked against other algorithms in an offline AUV path planner because if the proposed algorithms can provide better computational efficiency to demonstrate the minimum capability of a path planner, then they will outperform the tested algorithms in the online path planner. Based on the Monte Carlo simulations and ANOVA procedures, the SDEAPSO and SDEQPSO algorithms were able to achieve a similar performance to DEPSO and DEQPSO algorithms in terms of solution quality and stability, while having a significantly lower computational requirement. Most importantly, the simulation results showed that the planned paths in both the 2D and 3D scenarios are smooth, feasible and able to account for a priori known environment.

The PSO-based algorithms proposed in this study are most efficient for solving nondeterministic polynomial time (NP)-hard problem, such as the path planning problem. Although the simulation assumed a priori known environment to represent the minimum capability of a path planner, the algorithms can be adapted to a more realistic operational condition in future work due to the demonstrably high computational efficiency, which is suitable for solving compute-intensive problems such as path re-planning in highly dynamic environments. The future extension of this work will include developing a path re-planning algorithm that can deal with a priori unknown environment.

ACKNOWLEDGEMENTS

The present work is supported by the Tasmania Graduate Research Scholarship provided by Australia Maritime College (AMC). The authors acknowledge the AUV team in AMC for providing the data from their field experiment at Beauty Point, Tasmania, Australia.

REFERENCES

[1] T. Xue, *et al*., “Trajectory planning for autonomous mobile robot using a hybrid improved QPSO algorithm,” *Soft Computing*, vol. 21, no. 9, pp. 2421-37, 2017.

[2] Z. Zeng, *et al*., “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] D. Youakim and P. Ridao, “Motion planning survey for autonomous mobile manipulators underwater manipulator case study,” *Robotics and Autonomous Systems*, vol. 107, pp. 20-44, 2018.

[4] D. Kruger, *et al*., “Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007.

[5] D. Ferguson and A. Stentz, “Using interpolation to improve path planning: The Field D* algorithm,” *Journal of Field Robotics*, vol. 23, no. 2, pp. 79-101, 2006.

[6] C. Petres, *et al*., “Path Planning for Autonomous Underwater Vehicles,” *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331-41, 2007.

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

[8] J. D. Hernández, *et al*., Online motion planning for unexplored underwater environments using autonomous underwater vehicles,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 370-96, 2019.

[9] A. Alvarez, A. Caiti, and R. Onken, “Evolutionary path planning for autonomous underwater vehicles in a variable ocean,” *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418-29, 2004.

[10] J. Witt and M. Dunbabin, “Go with the flow: Optimal AUV path planning in coastal environments,” in *Australian Conference on Robotics and Automation*, 2008.

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

[12] Y. Qin, *et al*., “Path planning for mobile robot based on particle swarm optimization,” *Robot*., vol. 26, no. 3, pp. 222-5, 2004.

[13] Y. Fu, *et al*., “Path planning for UAV based on quantum-behaved particle swarm optimization,” in *Proceedings of SPIE - The International Society for Optical Engineering*, 2009.

[14] Z. B. Shi, *et al*., “Path planning for mobile robot based on quantum-behaved particle swarm optimization,” *Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology*, vol. 42(SUPPL. 2), pp. 33-7, 2010.

[15] Z. Zeng, *et al*., “Efficient Path Re-planning for AUVs Operating in Spatiotemporal Currents,” *Journal of Intelligent & Robotic Systems*, vol. 79, no. 1, pp. 135-53, 2015.

[16] S. MahmoudZadeh, *et al*., “Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms,” *Applied Soft Computing*, vol. 70, pp. 929-45, 2018.

[17] H. S. Lim, *et al*., “Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner,” in *2018 IEEE OES Autonomous Underwater Vehicle Symposium*, 2018.

[18] Y. Fu, *et al*., “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] Z. H. Zhan, *et al*., “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362-81, 2009.

[20] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995.

[21] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *CEC 99 Proceedings of the 1999 congress*, 1999.
[22] Y. Shi and R. C. Eberhart, “A modified particle swarm optimizer” in *Evolutionary Computation Proceedings, 1998 IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference*, 1998.
[23] Y. Shi and R. C. Eberhart, “Parameter selection in particle swarm optimization,” *International conference on evolutionary programming*, 1998.
[24] Y. Shi and R. C. Eberhart, “Comparing inertia weights and constriction factors in particle swarm optimization. Evolutionary Computation,” in *2000 Proceedings of the 2000 Congress*, 2000.
[25] Y. Shi, “Particle swarm optimization: developments, applications and resources. evolutionary computation,” in *2001 Proceedings of the 2001 Congress*, 2001.
[26] J. Sun, B. Feng, and W. Xu, “Particle swarm optimization with particles having quantum behavior,” in *Proceedings of the 2004 congress on evolutionary computation*, 2004.
[27] W. J. Zhang and X. F. Xie, “DEPSO: hybrid particle swarm with differential evolution operator,” *Systems, Man and Cybernetics*, 2003 IEEE International Conference, 2003.
[28] B. Raphael and I. F. Smith, *Fundamentals of computer-aided engineering*. John wiley & sons, 2003.
[29] D. H. Wolpert and W. G. Macready. “No free lunch theorems for optimization,” *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67-82, 1997.
[30] M. Locatelli, “A note on the Griewank test function,” *Journal of global optimization*, vol. 25, no. 2, pp. 169-74, 2003.
[31] H. Mühlenbein, M. Schomisch, and J. Born, “The parallel genetic algorithm as function optimizer,” *Parallel computing*, vol. 17, no. 6-7, pp. 619-32, 1991.
[32] D. H. Ackley, “The model,” in *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Springer, 1987, pp. 29-70.
[33] T. Bäck, H.-P. “Schwefel An overview of evolutionary algorithms for parameter optimization,” *Evolutionary computation*, vol. 1, no. 1, pp. 1-23, 1993.
[34] P. E. McKight and J. Najab, “Kruskal-wallis test,” in *The corsini encyclopedia of psychology*, 2010.
[35] J. Hochberg and A. C. Tamhane, Multiple comparison procedures. John Wiley & Sons, 1987.
[36] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.
[37] C. de Boor, *A practical guide to splines*. Verlag New York: Springer, 1978.
[38] S. Mirjalili, J. S. Dong, A. Lewis, “Ant colony optimizer: Theory, literature review, and application in AUV path planning,” *Studies in Computational Intelligence*, pp. 7-21, 2020.
[39] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. Norway: John Wiley & Sons, 1999.
[40] T. T. J. Prestero, “Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle,” Massachusetts Institute of Technology, 2001.
[41] W. Caharija, *et al*., “Path following of underactuated autonomous underwater vehicles in the presence of ocean currents,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012.
[42] Y. H. Eng, *et al*., “Online System Identification of an Autonomous Underwater Vehicle Via In-Field Experiments,” IEEE Journal of Oceanic Engineering, vol. 41, no. 1, 2015.

BIOGRAPHIES OF AUTHORS

Hui Sheng Lim received the Bachelor’s degree in Marine and Offshore Engineering in 2016 from University of Tasmania, Australia, where he is currently working toward the Ph.D. degree. His current research interests include optimal guidance, navigation and control system of autonomous underwater vehicle (AUV) using swarm intelligence.

Shuangshuang Fan received a B.E. in Mechanical Engineering from Shandong University, Jinan, China, in 2008, and a PhD in Mechatronic Engineering from Zhejiang University, Hangzhou, China, in 2013. From 2013 to 2014, she was a Research Engineer with the Institute of Shanghai Aerospace Control Technology. She was with the Acoustic Signal Processing Lab, Zhejiang University, as a Postdoctoral Researcher from 2014 to 2017. She was a lecturer at Australian Maritime College, University of Tasmania. Her research interests include the navigation, control and path planning of underwater vehicles in dynamic environments.

Christopher Chin commenced his academic career as a lecturer at RMIT University whilst undertaking his doctoral studies in Mathematics in 2003. He is currently a mathematician and a senior lecturer at the University of Tasmania, Australia. He works within the National Centre of Maritime Engineering and Hydrodynamics at the Australian Maritime College. He works closely with the Maritime Engineers focusing on Maritime Education and alternative energies. He is currently focusing on investigating the potential use of alternative energies in the maritime offshore sector in order to address the depletion of fossil fuels.

Shuhong Chai completed her Masters Degree in Naval Architecture at the Dalian University of Technology in China. In 2006 she obtained her doctoral degree from Universities of Strathclyde and Glasgow. She was then a senior hydrodynamics consultant and project manager with Oceanic Consulting Corporation in Canada. In 2008, Dr Chai joined AMC as a Senior Lecturer. She has since undertaken senior administrative roles including Director of the National Centre for Maritime Engineering and Hydrodynamics, and Associate Dean-Learning and Teaching for the AMC, Associate Dean Global of College of Sciences and Engineering and AMC Principal. She is very active in the subsea and underwater technology field. She is recently appointed as a member of the Committee of Loads of International Ship and Offshore Structures Congress (ISSC) from 2018 to 2021. She has been a specialist member of Committee of Subsea Technology of ISSC from 2015 to 2018 and Committee of Risers and Pipelines of ISSC from 2012 to 2015.

Neil Bose is the Vice President (Research) at Memorial University, Newfoundland and Labrador's University. Previously he was Principal of the Australian Maritime College (AMC), a specialist institute of the University of Tasmania, and a Professor of Maritime Hydrodynamics. Neil obtained his B.Sc. in Naval Architecture and Ocean Engineering from the University of Glasgow in 1978 and his Ph.D. also from Glasgow in 1982. He came to AMC in Tasmania in May 2007 as the Manager of the Australian Maritime Hydrodynamics Research Centre. His personal research interests are in marine propulsion, autonomous underwater vehicles, ocean environmental monitoring, ocean renewable energy, ice/propeller interaction and aspects of offshore design. Neil Bose is an ocean engineer and naval architect with an international reputation in marine propulsion built up through close collaboration with international industry.