frontiers | Frontiers in Neurorobotics

TYPE Original Research
PUBLISHED 06 December 2022
DOI 10.3389/fnbot.2022.1076455

OPEN ACCESS

EDITED BY
Namkyun I. M.,
Mokpo National Maritime University, South Korea

REVIEWED BY
Tianrui Zhou,
Shanghai Maritime University, China
Zhihui Hu,
Jimei University, China

*CORRESPONDENCE
Yuehong Gong
gongyuehonghit@126.com

RECEIVED 21 October 2022
ACCEPTED 17 November 2022
PUBLISHED 06 December 2022

CITATION
Gong Y, Zhang S, Luo M and Ma S (2022) A mutation operator self-adaptive differential evolution particle swarm optimization algorithm for USV navigation.
Front. Neurorobot. 16:1076455.
doi: 10.3389/fnbot.2022.1076455

A mutation operator self-adaptive differential evolution particle swarm optimization algorithm for USV navigation

Yuehong Gong[1]*, Shaojun Zhang[1], Min Luo[2] and Sainan Ma[3]

[1]School of Navigation and Shipping, Shandong Jiaotong University, Weihai, China, [2]School of Information Science and Engineering, Harbin Institute of Technology at Weihai, Weihai, China, [3]Zhejiang Jialan Ocean Electronics Co., Ltd., Zhoushan, China

To keep the global search capability and robustness for unmanned surface vessel (USV) path planning, an improved differential evolution particle swarm optimization algorithm (DePSO) is proposed in this paper. In the optimization process, approach to optimal value in particle swarm optimization algorithm (PSO) and mutation, hybridization, selection operation in differential evolution algorithm (DE) are combined, and the mutation factor is self-adjusted. First, the particle population is initialized and the optimization objective is determined, the individual and global optimal values are updated. Then differential variation is conducted to produces new variables and cross over with the current individual, the scaling factor is adjusted adaptively with the number of iterations in the mutation process, particle population is updated according to the hybridization results. Finally, the convergence of the algorithm is determined according to the decision standard. Numerical simulation results show that, compared with conventional PSO and DE, the proposed algorithm can effectively reduce the path intersection points, and thus greatly shorten the overall path length.

KEYWORDS

differential evolution algorithm, hybridization, mutation, particle swarm optimization, unmanned surface vessel path planning, scaling factor

1. Introduction

With the increasing demands of marine monitoring and exploration, the unmanned surface vessel (USV) becomes widely used in water area environmental inspection. The main research direction in the application of the USV is flight path planning. The purpose of path planning is to obtain a scientific, safe, and concise path in a specific environment. As the USV navigation environment deteriorates, the problems needed to be solved in USV path planning are becoming harsher, and the amount of calculation is also increasing, so the path planning problem gets trickier. In recent years, many studies has have been done for USV flight path planning and swarm intelligence algorithm has been popularly used, including particle swarm optimization (PSO) (Xin et al., 2019;

Krell et al., 2022), artificial fish swarm algorithm (Zhao et al., 2022), ant colony algorithm (Wang et al., 2022), genetic algorithm (GA) (Park et al., 2021), and etc. Among these intelligent algorithms, particle swarm optimization is the most widely used in the field of automatic control because of its simple principle, fewer parameters, fast optimization speed, small amount of calculation and other advantages (Khayati et al., 2019). At present, the application of particle swarm optimization algorithm in robot path planning is very active (Chen and Sun, 2021; Wu et al., 2022; Xiao et al., 2022), but the research on USV flight path planning is still a frontier field. In practical application, due to the irregular, complex and uncertain USV navigation environment, the algorithm is easy to fall into local extreme values, resulting in poor quality of generated paths, waste of time and resources. Therefore, improving the global search ability and robustness of the algorithm becomes a key content in USV flight path planning. The most popular use of PSO for global path planning is genetic particle swarm optimization (GaPSO), which combines the advantages of the genetic algorithm (GA) and the particle swarm optimization algorithm. In Pehlivanoglu (2012), an improved PSO for robot path planning is presented. In Zhang and Xing (2019), GA is combined with the voronoi diagram to generate the optimal path for autonomous unmanned aerial vehicles (UAVs). The applying of GaPSO can reduce the probability of falling into local optimal solutions, however, the global search capability needs to be enhanced. Differential evolution algorithm (DE) has been used in many fields such as intelligent machines, robots, equipments and so on Yang et al. (2011) and Yildiz (2013). Because of its strong global searching ability and high robustness, DE can be used to solve multiobjective, constrained and complex optimization problems.

Compared with genetic algorithms, the global exploration ability of DE is more obvious. However, the accuracy of the optimal solution and convergence speed are affected by the parameter settings and the mutation operations. To get a compromise between the optimal solution accuracy and optimization speed, many studies have been done on the values of the control parameters, including the population size *NP*, the crossover probability *CR*, and the scaling factor *F*. It is well known that the size of the population increases, the probability of finding the optimal solution is higher, however the amount of calculation and computation time may also increase. The value of *F* has a great influence on the speed of optimization. For a larger *F* value, the population is more abundant, but the amount of calculation is larger and the computation time is longer. With a smaller *F*, the optimal solution can be found faster, but the diversity of the population can't be guaranteed, so the algorithm will easily fall into a local optimum value. To obtain appropriate parameters, a lot of studies have been done. In Yildiz (2013), the authors pointed out that *F* should be set to 0.5. In Li et al. (2019), the authors suggested that *F* should between [0.4, 0.95]. The values of these parameters are difficult to select to obtain a better performance.

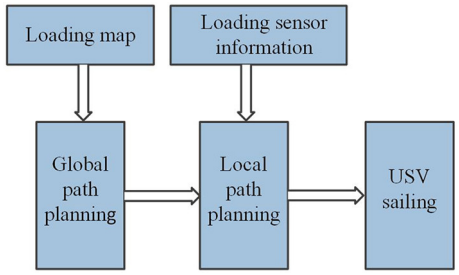

FIGURE 1
USV path planning process diagram.

In order to get a better flight path with fast optimization speed and less computation, this paper proposed a discrete particle swarm optimization algorithm with differential evolution algorithm (DeDSO) for USV flight path planning. This algorithm combines the advantages of the conventional particle swarm optimization algorithm and the differential evolution algorithm, so as to enhance the global search ability and improve the path planning efficiency and stability. In order to ensure both high search speed and high precision solution, the scaling factor *F* in DE is self-adaptive with the iteration.

The remainder of this paper is organized as follows. Section 2 gives the mathematical model of USV automatic inspection problem. Section 3 introduces the working principle of conventional PSO, DE, and the improved DePSO. Section 4 gives the framework and implementation procedure of the DePSO. Section 5 describes the numerical simulation results and comparison. Finally, section 6 gives a short conclusion.

2. Mathematical model of USV automatic inspection problem

Flight path planning is a global optimization problem aims to search the optimal flight path for USVs. When USV is applied in water environment inspection, it is necessary to generate a sailing path. The path planning process of the USV is shown in Figure 1. Firstly, a feasible global path is drawn according to the regional map information by identifying the surrounding environment and the status information of the USVs. Then the ship navigates according to the specified path. During the navigation, the sensors monitor the surrounding environment, and the local path planning is used to avoid obstacles, and then the ship continues to travel according to the original path.

The problem to be solved in this paper is to plan a global path for USVs applied in water area automatic inspection.

In a certain water area, several sampling points are spread, which will be automatically inspected and sampled by the USVs. The purpose of flight path planning is to generate a route between the starting point and the destination, which ensures that all sampling points are inspected only once except the starting point, and the path formed should meet specific optimization objectives, which is equivalent to the traveling salesman problem (TSP) (Laporte, 2010; Zheng et al., 2022). The TSP model has been mainly used for single vehicle operating situations as described in this paper, and the main idea is to search the optimal path to visit all sampling points.

$$\min f = \sum_{i=1}^{n-1} D(i, i+1) + D(1, n) \tag{1}$$

The objective function is the key point of an optimization problem. In our design, the optimization objective is demonstrated as function f, and the mathematical model corresponding to n sampling points is defined as Equation (1). Where D(i, i + 1) respects the distance between sampling point "i" and sampling point "i + 1." In this paper, the USV sampling problem is corresponding to symmetry TSP problem, which satisfies D(i, j) = D(j, i), where i, j $\in$ (1, 2, …, n).

3. Improved particle swarm optimization model

3.1. Particle swarm optimization algorithm

In particle swarm optimization, the updating of population tends to be closer to the optimal value. The optimization process of PSO is shown in Figure 2. Particles adjust their speed and direction of motion according to their own historical optimal value P_{best} and global optimal value G_{best}, and measure the merits of particles by optimizing the fitness value determined by the target, so as to drive the particles to the optimal value (Wu et al., 2017; Rauf et al., 2020).

The next generation particle position $X_{\text{id}}(t+1)$ and velocity $V_{\text{id}}(t)$ are calculated through Equations (2) and (3) respectively. In which, $X_{\text{id}}(t)$ is the tth generation particle position; ω is the inertia weight, which determines the optimization speed of the algorithm; c_1 and c_2 decide the speed for the particle approaching P_{best} and G_{pest} respectively; R_1 and R_2 are random numbers, and their values range from 0 to 1; In this paper, the fitness represents the length of the USV navigation route, the calculation formula is (4), in which (x_{i}, y_{i}) are the coordinates of the i th and the $i+1$ th sampling point, respectively. The pseudo code of PSO is presented in Table 1.

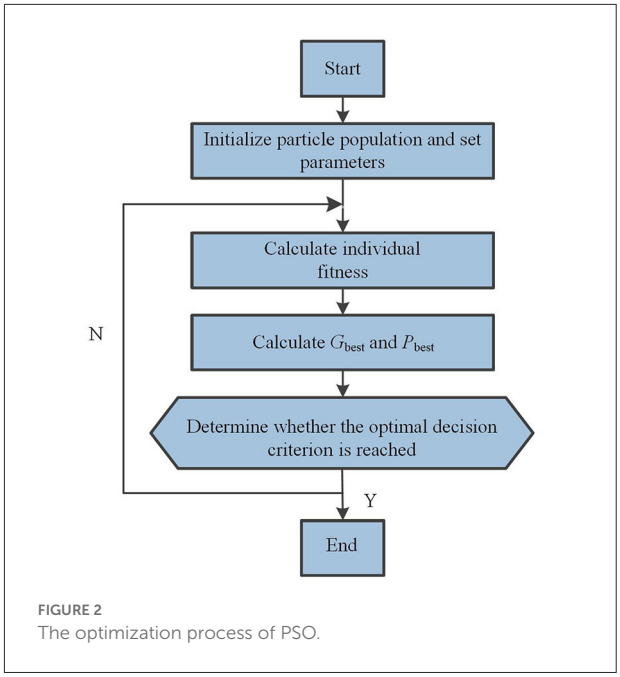

FIGURE 2
The optimization process of PSO.

TABLE 1 Algorithm I: Conventional PSO.

Conventional particle swarm optimization

1. Set fitness function f, set population size N, acceleration coefficients c_1, c_2, R_1, R_2, inertia weight ω, and maximum iteration number $T_{\max}$
2. Generate the initial population
3. **For** t from 1 to $T_{\max}$ **do**
4. **For** i from 1 to N **do**
5. Initialize velocity and position
6. Evaluate initial fitness value
7. Record initial P_{pest} and G_{pest}
8. **end**
9. Refresh velocity and position from Equations (1) and (2)
10. Refresh the fitness of each particle
11. Update P_{pest} and G_{pest}
12. **If** $t = T_{\max}$ or minimum error criteria is achieved **do**
13. Output the particle with best fitness value
14. **end if**
15. **end**

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \tag{2}$$

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 R_1 \left(P_{best} - X_{id}\right) + c_2 R_2 \left(G_{best} - X_{id}\right) \tag{3}$$

$$f(x) = \sum_{i=1}^{n-1} \sqrt{\left(y_{i+1} - y_i\right)^2 + \left(x_{i+1} - x_i\right)^2} \tag{4}$$

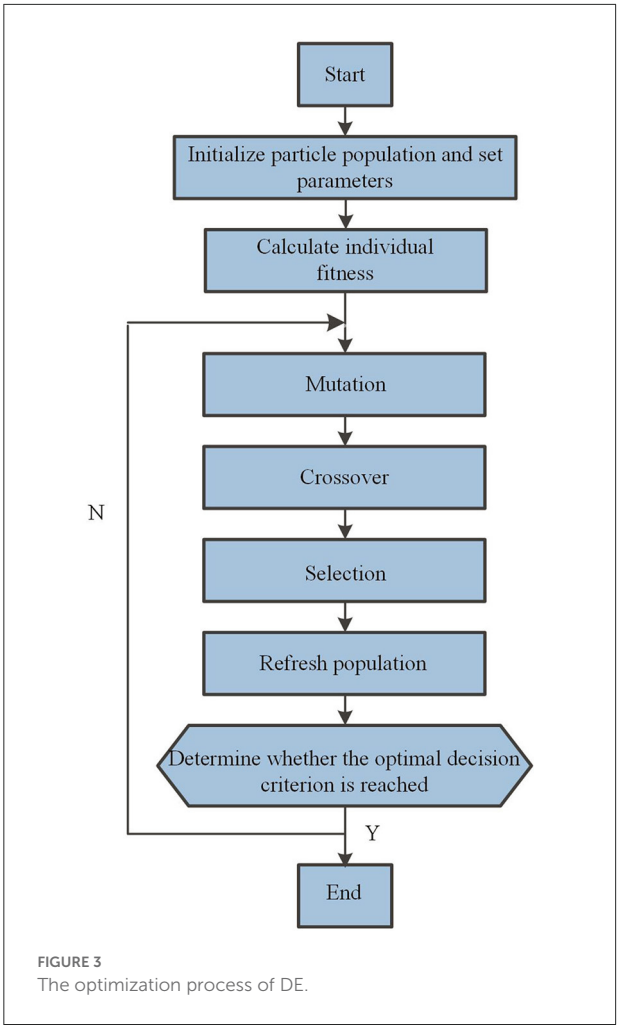

FIGURE 3
The optimization process of DE.

3.2. Differential evolution algorithm

Differential evolution algorithm is a bionic intelligent calculation method which simulates the natural evolutionary law to find the global optimal value. The optimization process of DE is shown in Figure 3. The main body of the algorithm generally includes three steps: mutation, crossover and selection (Yi et al., 2016; Bilal et al., 2020). The pseudo code of DE is presented in Table 2. Three different individual vectors are randomly selected to get a difference vector, denoted as X_{r1}, X_{r2}, and X_{r3}, and the mutation vector V_i was calculated according to Equation (5). The scaling factor F is set to control the influence of difference vector (X_{r2}-X_{r3}) on the evolution speed, and the value of F is generally between [0,1].

$$V_i = X_{r1} + F\,(X_{r2} - X_{r3}) \tag{5}$$

The function of crossover operation is to hybridize the current individuals of the population with the mutation vector

TABLE 2 Algorithm II: Conventional DE.

Conventional differential evolution algorithm

1. Set fitness function, population size *PN*, scaling factor *F*, and crossover probability *CR*
2. Generate the initial population
3. **For** *t* from 1 to T_{max} **do**
4. **For** *i* from 1 to *NP* **do**
5. Mutation: select X_{r1}, X_{r2} and X_{r3} randomly
6. Create the mutation vector V_i from Equation (5)
7. Crossover: the crossover operation is executed between a parent individual and mutation vector from Equation (6)
8. Selection from Equation (7)
9. Refresh fitness value
10. **end**
11. **If** $t = T_{max}$ or minimum error criteria is achieved **do**
12. Output the particle with best fitness value
13. **end if**
14. **end**

generated by the mutation, and generate new individuals randomly according to a certain probability, as shown in Equation (6). *CR* is the crossover probability, which determines the size of the hybridization probability, and its value is a real constant between [0,1].

$$u_{ij}(t+1) = \begin{cases} V_{ij}(t+1), \textit{if } r \textit{ and } (0,1) \le CR, \\ \text{or j} = \text{j and} \\ X_{ij}(t), \textit{ else} \end{cases} \tag{6}$$

The selection operation is based on the individual optimization objective to select one of the next generation individuals between the new vector generated by the crossover and the current vector. If the individual optimization objective of the new vector is better than the current vector, the new vector will be retained. Otherwise, the current vector individual will be inherited to the next generation. The selection formula is shown in Equation (7). In which, $X_i(t)$ is the particle position before the t th iteration, $U_i(t)$ is the new particle position after the differential evolution mutation operation of the t th iteration. $X_i(t+1)$ is the particle position after the t th iteration. $f(X_i(t))$ is the fitness value for particle position $X_i(t)$, $f(U_i(t))$ is the fitness value for particle position $U_i(t)$.

$$X_i(t+1) = \begin{cases} u_i(t), \textit{if } f(u_i(t)) < f(X_i(t)) \\ X_i(t), \textit{ if } f(u_i(t)) \ge f(X_i(t)) \end{cases} \tag{7}$$

3.3. Differential evolution particle swarm optimization

In the process of population updating, the DePSO iterates from the formula of the particle swarm optimization algorithm to get a new generation of individuals, and then hybridizes them from the formula of the differential evolution algorithm. The object of hybridization is the individual optimal value P_{best} and the global optimal value G_{best}, the selection formula after crossing is shown in Equations (8) and (9).

In order to find a better balance between optimal solution accuracy and convergence speed, the zoom factor F is adjusted adaptively with the number of iterations. The improved formula is demonstrated in Equation (10), in which F_{max} and F_{min} are the maximum and minimum values of the zoom factor, T_{max} is the maximum number of iterations, and t is the current number of iterations.

$$X_i(t+1) = \begin{cases} u_i(t), if\ f(u_i(t)) < f(P_{best}) \\ X_i(t),\ if\ f(u_i(t)) \geq f(P_{best}) \end{cases} \quad (8)$$

$$X_i(t+1) = \begin{cases} u_i(t), if\ f(u_i(t)) < f(G_{best}) \\ X_i(t),\ if\ f(u_i(t)) \geq f(G_{best}) \end{cases} \quad (9)$$

$$F = \frac{(F_{max} - F_{min})}{2}(1 + \frac{T_{max} - t}{T_{max}}) \quad (10)$$

4. USV path planning problem

4.1. PSO applying in TSP problems

In the TSP issue corresponding to automatic inspection and sampling of unmanned pollution detection vessel, a huge number of paths will be generated during flight path planning, and the number of paths will increase factorially with the number of sampling points. A path is equivalent to a particle in PSO, the position of the particle is represented by the path sequence composed of the distribution of sampling points, and the velocity of the particle is represented by the commutator. The calculation process of optimization is realized by exchanging the position of sampling points.

Exchange operator: for two path sequences $X_i = (X_{i1}, X_{i2}, ..., X_n)$ and $X_j = (X_{j1}, X_{j2}, ..., X_{jn})$, if the two sequences have different value at the same position, that is $X_{ia} \neq X_{ja}$, thus $V_{ij} = (X_{ia}, X_{ja})$ is called the exchange operator for the path sequence, as shown in Figure 4A.

Exchange sequence: a sequence composed by the exchange operators, such as $V = (V_1, V_2..., V_n)$, in which n is the number of locations corresponding to two cities with the same sequence but different values.

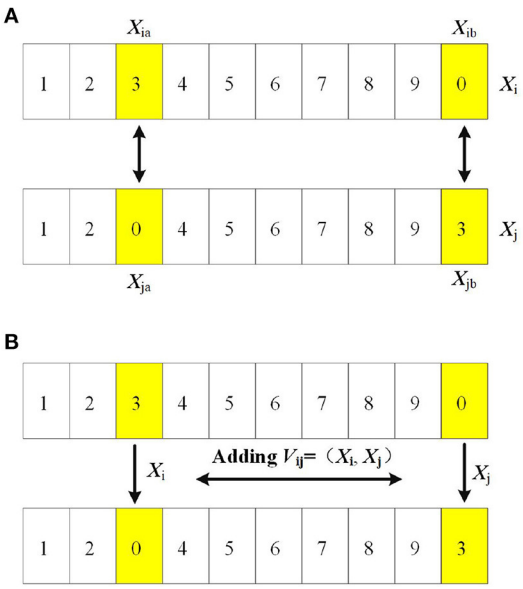

FIGURE 4
(A) Exchange operator. **(B)** Addition of position and velocity.

Position of the particle: the position of the particle is determined by the sequence of cities $X = (X_1, X_2..., X_m)$, in which m is the number of the cities.

Velocity of the particle: the velocity of the particles is given by the exchange sequence $V = (V_{1a}, V_{2b}..., V_{mn})$, in which V_{mn} is the exchange operator.

Addition of position and velocity: in TSP, we form a new path sequence by the addition of position and velocity to the parent individual, as shown in Equation (11). On the basis of the original path sequence, the new path sequence is obtained by exchanging the city position according to the exchange operator, as shown in Figure 4B.

$$\begin{aligned} X &= (X_1, X_2, \cdots, X_i, \cdots X_j, ..., X_n) + V_{ij}(X_i, X_j) \\ &= (X_1, X_2, \cdots, X_j, \cdots, X_i, \cdots, X_n) \end{aligned} \quad (11)$$

Subtraction between positions: the subtraction from a position to another position forms an exchange sequence that generates a new velocity. To form $V_{ij} = X_i - X_j$, where X_i and X_j are the number of the cities. First we find the city in the 2th sequence which is the same to the 1st element in the 1st sequence, and form an exchange operator $V(1,i)$. Then conduct this exchange operator to the first sequence and get a new 1st sequence. Later, find the first position in the 2st sequence with the same element in the new 1st sequence, and get an exchange operator $V(2,i)$, this course will continue sequentially until the exchange sequence of two city sequences is obtained, as is shown

in Equation (12).

$$X_1(4,3,6,2,1,5,7,8,9,0) - X_2(3,4,6,2,1,5,7,8,9,0) = V(v(1,5) + v(2,6) + v(3,6) + v(4,6) + v(5,6)) \quad (12)$$

Scalar multiplication of velocity: the scalar multiplication of velocity has a probabilistic meaning. For example, when V_{im}=cV_{ja}, the selection formula is shown in (13), in which c is a constant. During the calculation course, a random number rand is generated for each operator of V_{ja}, compare c and rand, the value of V_{im} is determined by the comparison result.

$$V_{\text{im}} = \begin{cases} V_{\text{ja}}, if \text{ r and } < \text{c} \\ 0, \text{else} \end{cases} \quad (13)$$

4.2. DePSO applied in USV path planning

The improved differential evolution particle swarm optimization algorithm is applied to the automatic inspection sampling path planning process of USV. The initial particles are chosen randomly from the possible paths generated at the beginning, and the optimal path is found by the approach of optimal value of DePSO. In the updating and iteration process of each generation of particle population, the idea PSO and DE are combined. The specific implementation steps of the algorithm are shown in Figure 5, and the pseudo code is presented in Table 3.

Step 1: Modeling the environment according to the map model and the distribution of sampling points.
Step 2: Randomly select a certain number of path sequences as the initial particle population, and set the parameter values used in the algorithm.
Step 3: Refresh the population and calculate the fitness value of the particles;
Step 4: Find the individual optimal path P_{best} and the population optimal path G_{best};
Step 5: Generate mutation vector according to mutation formula of differential evolution algorithm, and cross with P_{best} and G_{best}, respectively to get new path sequence. The fitness of the newly generated path sequence is calculated and compared with the fitness of P_{best} and G_{best}, respectively. If the path becomes shorter, the new particle position is retained, otherwise it is discarded. Update P_{best} and G_{best} while retaining the position of the new particle.
Step 6: Judge whether the optimal decision is reached; There are two judgment conditions: First, the minimum error criteria of fitness reaches the set accuracy; Second, reach the maximum number of iterations. If the one of the judgment conditions is met, the optimization ends. Otherwise, go back to the second step and the iteration continues.

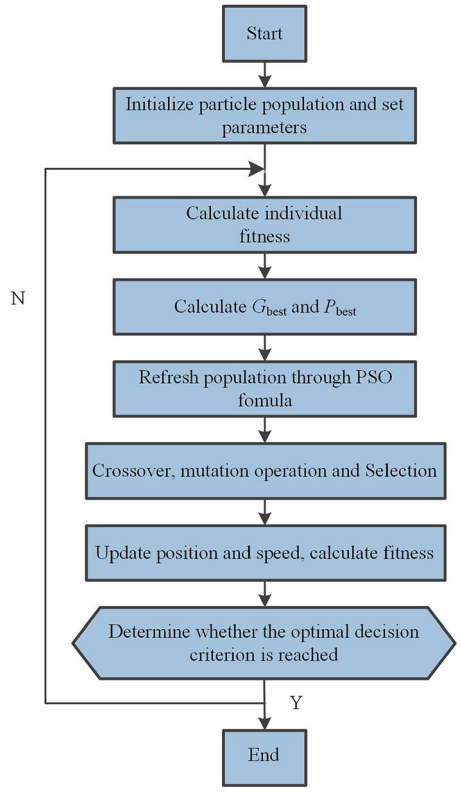

FIGURE 5
The optimization process of DePSO.

5. Numerical simulation results and analysis

In order to verify the flight path planning effect of the improved DePSO for USV, numerically simulation is done to imitation the path planning process. The hardware platform and software environment applied in this experiment is demonstrated in Table 4.

In this paper, a 500 × 500 m sea environment shown in Figure 6 is chosen as the inspection map model. The simulation environment is set to be ideal. Assuming there are no obstacles in the sea area. The USV is set as a particle without considering the factors such as its volume and position. The environmental factors such as wind and waves are also not considered. Firstly, the USV navigation environment is modeled according to the water environment map and the sampling points distribution. Secondly the improved particle swarm optimization algorithm is applied for global path planning. For comparison, the path planning process of PSO, DE and DePSO were simulated, respectively. The coordinate settings of the 30 sampling points is shown in Figure 7A. The particle population size was selected as 200. The parameter c_1 and c_2 are set to 1, and the inertia weight ω is set to 1. For conventional DE, CR and the zoom factor F are

TABLE 3 Algorithm III: DePSO.

Improved particle swarm optimization

1. Set fitness function, population size NP, acceleration coefficients c_1, c_2 and inertia weight ω. Set scaling factor F_{max}, F_{min}, and crossover probability CR
2. Generate the initial population
3. **For** t from 1 to T_{max} **do**
4. **For** i from 1 to NP **do**
5. Evaluate initial fitness value
6. **end**
7. Record initial P_{best} and G_{best}
8. **For** i from 1 to NP **do**
9. Refresh the velocity and position from Equations (1) and (2)
10. Refresh fitness value
11. Update P_{best} and G_{best}
12. **end**
13. **For** P_{best} and G_{best} **do**
14. Mutation: select X_{r1}, X_{r2} and X_{r3} randomly
15. Create the mutation vector V_i from Equation (5)
16. Crossover: the crossover operation is executed between P_{best} or G_{best} and mutation vector from Eq. 6
17. Selection from Equations (8) or (9)
18. Refresh fitness value
19. **end**
20. **If** $t = T_{max}$ or minimum error criteria is achieved **do**
21. Output the particle with best fitness value
22. **end if**
23. **end**

TABLE 4 Experimental hardware and software environment.

Project	Environmental set
CPU	Intelcore (TM) i7-10510U
Graphics Card	Intel UHD Graphics
Memory	16G
Operational System	Windows 10
Software Version	Matlab 2016R

set to 1. For DePSO, the maximum value of zoom factor F_{max} is set to 1 and the minimum value F_{min} is set to 0. Figures 7B–D give the comparison of the final results of route planning with conventional PSO, conventional DE, and DePSO, respectively, under the same map environment with 30 sampling points. Figure 7E gives the path length iterative process comparison for the three methods.

The coordinate settings of the 50 sampling points is shown in Figure 8A. Figures 8B–D demonstrate the comparison of the path planning results applying conventional PSO, conventional DE, and DePSO, respectively, under the same map environment with 50 sampling points, and Figure 8E gives the path length iterative process comparison.

FIGURE 6
The sea environment map.

As the number of sampling points increases, the number of iterations needed to find the optimal path will also increase gradually. In Figure 7, the maximum number of iterations is 1,000. In Figure 8, the maximum number of iterations is 3,000.

As can be seen from the figures, compared with the conventional PSO and DE, by applying the proposed DePSO, the number of overlaps of the final paths is significantly reduced and the total path length is significantly shortened. Because of the combination of the two algorithms, the computation amount for an iteration is increased, so that the computation time is slightly increased. But this increasing time is negligible when comparing with speed enhancement brought by the algorithm improvement.

Table 5 lists the path planning results applying conventional PSO, conventional DE, and DePSO proposed in this paper when the sampling points are 20, 30, 40, 50, and 60, respectively. For each algorithm, 5 times of simulation are done, among which the best, average and worst path length are listed. Applying the conventional PSO and DE algorithms, there are many crossings in the path and the total length of the path lengths are long. Applying the DePSO proposed in this paper, the path complexity is greatly reduced and the path lengths are significantly shorter.

Table 6 lists the path planning results applying the proposed DePSO with different population size when the sampling point number is 50. For each population size of 100, 200, 300, 400, and 500, 10 times of simulation are done, and the best, average and worst calculation time and path lengths are listed in Table 6. From this table we can get that, when population size rises, the optimization time becomes longer and the path lengths becomes shorter.

Table 7 lists the path planning results applying the proposed DePSO with different F ranges, when the population size is 200 and the sampling point number is 30. For each F range of [0,1], [0.05,0.95], [0.1,0.9], [0.15,0.85], and [0.2,0.8], 10 times

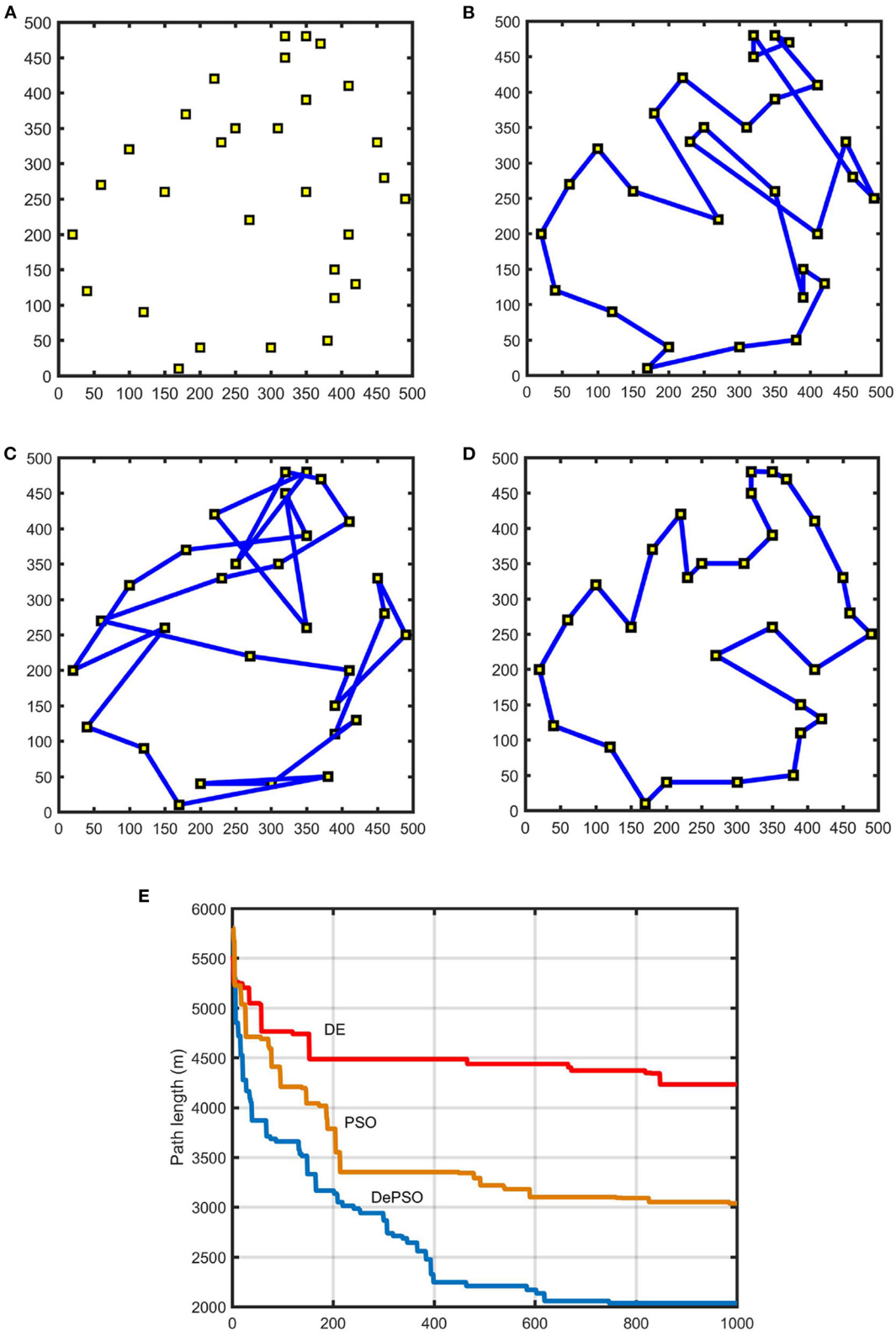

FIGURE 7
Path planning results for 30 sampling points: **(A)** Distribution of 30 sampling points. **(B)** Path planning result with conventional PSO. **(C)** Path planning result with conventional DE. **(D)** Path planning result with DePSO. **(E)** Path length iterative process comparison.

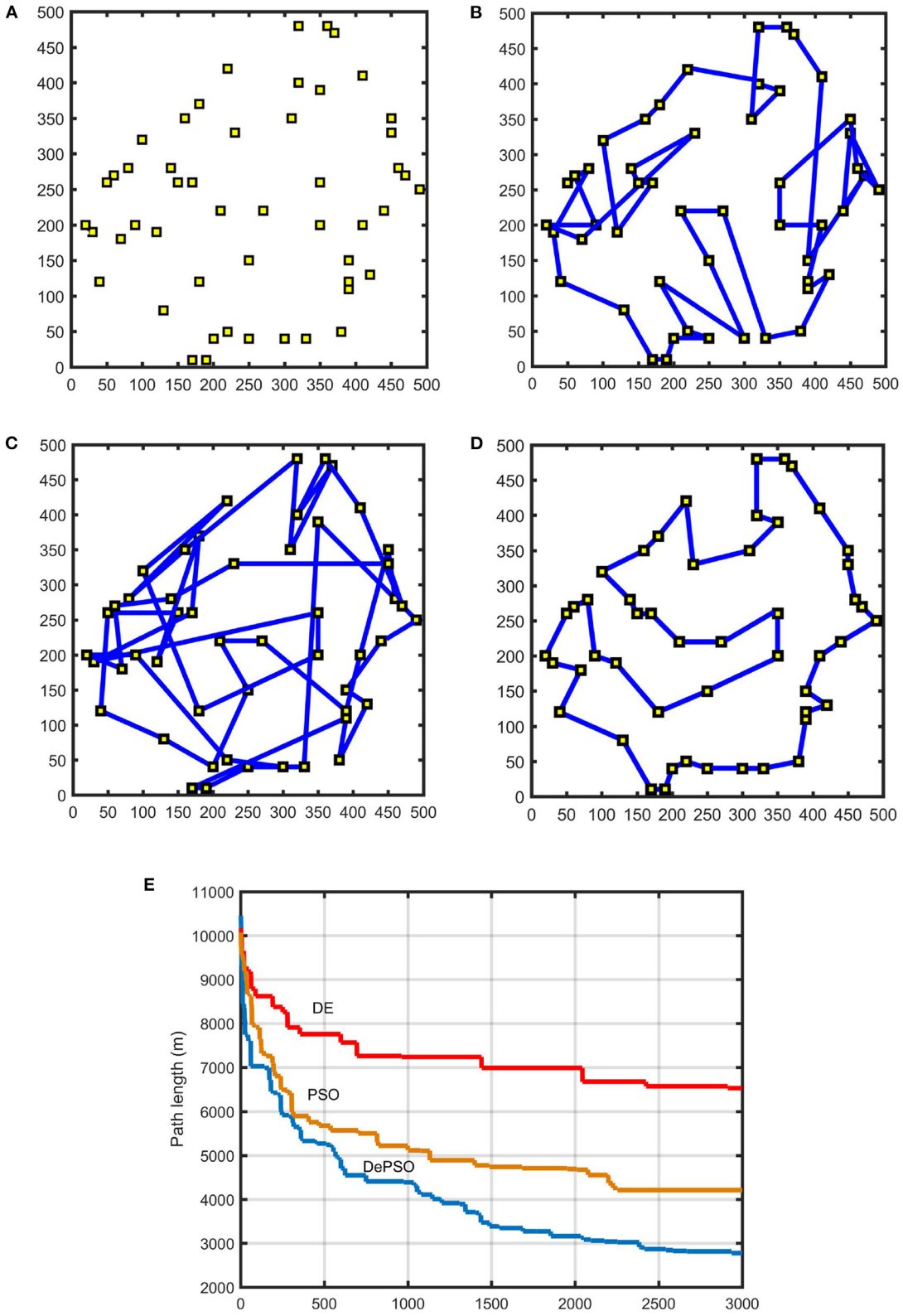

FIGURE 8
Path planning results for 50 sampling points: **(A)** Distribution of 50 sampling points. **(B)** Path planning result with conventional PSO. **(C)** Path planning result with conventional DE. **(D)** Path planning result with DePSO. **(E)** Path length iterative process comparison.

TABLE 5 Comparison of path lengths with PSO, DE, and DePSO.

Number of sample points	Conventional PSO			Conventional DE			DePSO		
	Best	Average	Worst	Best	Average	Worst	Best	Average	Worst
20	2,021	2126.4	2,215	2,413	2531.8	2,628	1,874	1911.6	1,939
30	2,769	2879.4	2,978	3,877	3,949	4,023	2,038	2096.8	2,207
40	3,670	3877.8	4,000	5,621	5736.8	5,837	2,391	2401.2	2,408
50	4,020	4102.6	4,195	6,705	6783.8	6,823	2,548	2655.2	2,760
60	4,761	4659.6	4,575	5,464	5,607	5,696	3,022	3086.2	3,186

TABLE 6 Path planning results with different population sizes.

Population size	Path length (m)			Computation time (s)		
	Best	Average	Worst	Best	Average	Worst
100	2,656	2866.6	3,167	56.88	58.95	59.95
200	2,525	2734.6	2,954	101.09	103.25	112.31
300	2534	2713.0	2903	146.10	150.86	155.32
400	2,529	2642.9	2,876	176.87	181.48	202.03
500	2,453	2629.4	2,742	221.68	243.81	259.94

TABLE 7 Path planning results with different *F* range.

F range	Path length (m)			Computation time (s)		
	Best	Average	Worst	Best	Average	Worst
[0,1]	2,038	2074.8	2,150	88.71	89.63	93.88
[0.05,0.95]	2,038	2235.1	2,283	86.75	87.89	90.69
[0.1,0.9]	2,550	2695.2	2,884	83.34	84.08	87.78
[0.15,0.85]	2,893	3,178.8	3,328	83.07	83.43	83.99
[0.2,0.8]	3,537	3772.4	3,905	79.85	81.92	83.94

of simulation are done, the best, average and worst calculation time and path lengths are listed. From Table 7 we can get that, the optimization result is also affected by the value range of zoom factor *F*. With small *F* range, the optimization course is fast, but the route length is longer. With the larger *F* value, the optimization time becomes longer and the path length is shorter.

6. Conclusion

Particle swarm optimization algorithm is an effective optimization method for USV global path planning. However, conventional PSO approaches cannot always find the global optima, particularly for complex scenes. DE algorithm has the strong global search ability and good robustness. The combination of PSO and DE can enhance their advantages thus enhance the global search ability of the algorithm. In this paper, an improved particle swarm optimization algorithm (DePSO) was used for global flight path planning of automatic inspection path of USVs. In the optimization process of DePSO, the current vector is differentially crossed with the local optimal value and the global optimal value, which can further enrich the population. In order to further balance the optimal solution accuracy and optimization speed, the mutation factor is adjusted adaptively with the number of iterations. The numerical simulation results show that the proposed DePSO can realize the global path planning of the USV and achieved shorter path length than conventional PSO and DE. Compared with the existing methods, the method proposed in this paper is more suitable for the flight path planning of USV applying in water environment automatic inspection.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YG proposed this contribution, verified, and concluded simulation results. ML gaves suggestions for simulation sets. SZ and SM gave suggestions for manuscript writing. All authors contributed to the article and approved the submitted version.

Funding

This work was partially supported by the National Natural Science Foundation of China (NSFC) Under Grant Nos. U2106202 and 12075142, the Shandong Provincial Natural Science Foundation Under Grant No. ZR2020MA102, and the Science and Technology Planning Project of Zhejiang Under Grant No. 2020C03101.

Conflict of interest

Author SM was employed by Zhejiang Jialan Ocean Electronics Co., Ltd.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., and Abraham, A. (2020). Differential evolution: a review of more than two decades of research. *Artif. Intell.* 90, 103479. doi: 10.1016/j.engappai.2020.103479

Chen, L., and Sun, H. (2021). Picking path optimization of mobile robotic arm based on differential evolution and improved a* algorithm. *IEEE Access* 9, 154413–154422. doi: 10.1109/ACCESS.2021.3060738

Khayati, G. R., Rajabi, Z., Ehteshamzadeh, M., and Beirami, H. (2019). A hybrid particle swarm optimization with dragonfly for adaptive anfis to model the corrosion rate in concrete structures. *Int. J. Concrete Struct. Mater.* 16, 28. doi: 10.1186/s40069-022-00517-9

Krell, E., King, S. A., and Carrillo, L. R. G. (2022). Autonomous surface vehicle energy-efficient and reward-based path planning using particle swarm optimization and visibility graphs. *Appl. Ocean Res.* 122, 103125. doi: 10.1016/j.apor.2022.103125

Laporte, G. (2010). A concise guide to the traveling salesman problem. *J. Operat. Res. Soc.* 61, 35–40. doi: 10.1057/jors.2009.76

Li, X., Dai, G., Wang, M., Liao, Z., and Ma, K. (2019). A two-stage ensemble of differential evolution variants for numerical optimization. *IEEE Access* 7, 56504–56519. doi: 10.1109/ACCESS.2019.2909743

Park, J., Kim, S., Noh, G., and Kim, H. (2021). Mission planning and performance verification of an unmanned surface vehicle using a genetic algorithm. *Int. J. Naval Arch. Ocean Eng.* 13, 575–584. doi: 10.1016/j.ijnaoe.2021.07.002

Pehlivanoglu, Y. V. (2012). A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. *Aerospace Sci. Technol.* 16, 47–55. doi: 10.1016/j.ast.2011.02.006

Rauf, H. T., Shoaib, U., Lali, M. I., Alhaisoni, M., Irfan, M. N., and Khan, M. A. (2020). Particle swarm optimization with probability sequence for global optimization. *IEEE Access* 8, 110535–110549. doi: 10.1109/ACCESS.2020.3002725

Wang, L., Wang, H., Yang, X., Gao, Y., Cui, X., and Wang, B. (2022). Research on smooth path planning method based on improved ant colony algorithm optimized by floyd algorithm. *Ships Offshore Struct.* 16, 955179. doi: 10.3389/fnbot.2022.955179

Wu, Q., Song, T., Liu, H., and Yan, X. (2017). Particle swarm optimization algorithm based on parameter improvements. *J. Comput. Methods Sci. Eng.* 17, 557–568. doi: 10.3233/JCM-170742

Wu, Z., Zhang, Y., Dong, Z., and Li, D. (2022). Fault monitoring and diagnosis of high-pressure heater system based on improved particle swarm optimization and probabilistic neural network. *Meas. Sci. Technol.* 33, 1–12. doi: 10.1088/1361-6501/ac8367

Xiao, J., Liu, S., Liu, H., Wang, M., Li, G., and Wang, Y. (2022). A jerk-limited heuristic feedrate scheduling method based on particle swarm optimization for a 5-dof hybrid robot. *Robot. Comput. Integr. Manuf.* 78, 1–12. doi: 10.1016/j.rcim.2022.102396

Xin, J., Li, S., Sheng, J., Zhang, Y., and Cui, Y. (2019). Application of improved particle swarm optimization for navigation of unmanned surface vehicles. *Sensors* 19, 47–55. doi: 10.3390/s19143096

Yang, Z. Y., Tang, K., and Yao, X. (2011). Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Comput.* 15, 2141–2155. doi: 10.1007/s00500-010-0643-6

Yi, W., Zhou, Y., Gao, L., Li, X., and Mou, J. (2016). An improved adaptive differential evolution algorithm for continuous optimization. *Expert. Syst. Appl.* 44, 1–12. doi: 10.1016/j.eswa.2015.09.031

Yildiz, A. R. (2013). A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations. *Appl. Soft Comput.* 13, 1561–1566. doi: 10.1016/j.asoc.2011.12.016

Zhang, G., and Xing, K. (2019). Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion. *Comput. Operat. Res.* 108, 33–43. doi: 10.1016/j.cor.2019.04.002

Zhao, L., Bai, Y., Wang, F., and Bai, J. (2022). Path planning for autonomous surface vessels based on improved artificial fish swarm algorithm: a further study. *Ships Offshore Struct.* doi: 10.1080/17445302.2022.2116765

Zheng, R., Zhang, Y., and Yang, K. (2022). A transfer learning-based particle swarm optimization algorithm for travelling salesman problem. *J. Comput. Design Eng.* 9, 933–948. doi: 10.1093/jcde/qwac039