IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2020, accepted December 24, 2020, date of publication December 28, 2020,
date of current version January 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3047816

An Improved Beetle Swarm Optimization Algorithm for the Intelligent Navigation Control of Autonomous Sailing Robots

LIN ZHOU, KAI CHEN, HANG DONG, SHUKAI CHI, AND ZHEN CHEN
College of Engineering, Ocean University of China, Qingdao 266100, China

Corresponding authors: Shukai Chi (chishukai@163.com) and Zhen Chen (chenzhen1989@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51779237, and in part by the Shandong Provincial Natural Science Foundation under Project 2019GHY112035.

ABSTRACT Autonomous sailing robots are a new type of green ship that use wind energy to maintain continuous cruising operations. Compared with traditional algorithms, swarm intelligence optimization algorithms have better intelligence and adaptation. An intelligent algorithm acts as one of the most important solutions to the path planning problem of autonomous sailing robots. The beetle swarm optimization, which is a novel intelligent method that combines the search mechanism of a single beetle with the particle swarm optimization algorithm, is utilized to obtain the optimal path. In this study, the track navigation control of an improved mathematical model of a sailing ship is introduced, and the navigation is tested using a downsized prototype of an autonomous sailing robot. The improved beetle swarm optimization is proposed here by dynamically changing the step size factor and the inertia weight formula. In the iteration of the improved beetle swarm optimization algorithm, the location update cooperates with the beetle monomer search mechanism to learn the update strategy of the particle swarm optimization algorithm. Combinatorial strategies can speed up the overall iterative convergence speed and reduce the possibility that the population will fall into a locally optimal solution. The simulation results demonstrate the robustness, efficiency, and feasibility of the improved beetle swarm optimization in different cases. The research results can provide some references and ideas for the autonomous intelligent navigation control design of autonomous sailing robots.

INDEX TERMS Autonomous sailing robot, beetle antennae search, improved beetle swarm optimization, path planning.

I. INTRODUCTION

Currently, the main methods of marine environmental monitoring and resource exploration include onsite detection with ocean buoys [1], underwater robots [2]–[5], unmanned surface vessels [6], and remote monitoring with aircraft and satellites. Manned motorboats, which are the main equipment for long-term marine monitoring, are limited by the power of batteries or fuels and the high manpower and material resource requirements. Fixed-point offshore buoys also have limitations. In particular, if there is the need to monitor a large area of the sea, many offshore fixed-point buoys are required, which is very costly. Furthermore, fixed detection devices lack maneuverability and need manual deployment. With the increasing demand of marine monitoring and exploration, traditional monitoring and exploration methods have revealed a large number of defects; therefore, it is necessary to develop new monitoring equipment.

In the last decade, autonomous sailing robots have shown good potential as alternative monitoring and exploration equipment. First, they use wind energy to maintain continuous cruising operations; thus, they do not have fuel or battery restrictions, which greatly extends their operation time. Second, they can collect data at fixed points during the voyage, and their collection range is much larger than that of static acquisition equipment, such as buoys. Third, they can greatly reduce fuel consumption and environmental pollution.

Autonomous sailing robots are mainly propelled by wind power and only require little energy to control their rudders

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

and sails. They rely on renewable energy sources, such as solar and wind energy, and are essential for long-range cruising and long-term continuous operations. Extensive research is being conducted on these marine vehicles. The navigation system of a simple autonomous sailing robot consists of two parts: navigation (high-level control) and execution (low-level control). The latter includes the independent control of the sails and the steering gear.

In the past decade, many methods have been proposed to solve the navigation and obstacle avoidance problems of autonomous sailing robots. The path planning for autonomous sailing robots can be divided into unobstructed and obstructed path planning, depending on whether obstacles are considered. Because autonomous sailing robots are completely powered by wind, their path planning is challenging even when there are no obstacles in the environment, since the target position, wind field, and side change strategies need to be considered. Stelzer R proposed a speed optimization algorithm that can give the fastest direction of the current sailing speed according to the speed curve of the autonomous sailing robot. In other words, this algorithm ensures that the speed component of the autonomous sailing robot in the direction connecting the current position with the target is the largest. The output path of this algorithm is generally not optimal, but it is more optimized than a straight path, and the planning speed is fast [7]. Furthermore, Stelzer added obstacles on the basis of velocity made good (VMG) and modified the speed pole diagram of the sailing boat according to the distance of the obstacle from the autonomous sailing robot, i.e., he added the direction in which the autonomous sailing robot cannot sail or he reduced the speed of the autonomous sailing robot in certain directions. However, this method inherits the disadvantages of VMG [8]. Considering the optimality of the planning, Erckens designed a local path planning algorithm for the sailboat based on the A$*$ algorithm and incorporated the sailing strategy of the sailboat in the algorithm. He then tested the proposed algorithm on an autonomous sailing robot named Avalon [9]. Romero M designed a local obstacle avoidance algorithm for sailboats based on fuzzy logic and added sailing rules based on empirical sailing knowledge [10]. Sauze C included the obstacles into the polar coordinates centered on the sailing boat. When approaching the obstacles, a new course was obtained by comprehensively considering the target heading angle, subsequent obstacles, the wind direction, and other factors [11]. Nevertheless, this method could not guarantee the optimal path. Other path planning algorithms include the PRM-Dijkstra algorithm [11], interval analysis [12], the cost function method [13], and the artificial potential field method [14]–[17].

In the tracking control of autonomous sailing robots, the waypoints are generally set, as is straight-line tracking control between the two waypoints. The controller has a PID heading controller [18], a speed direction controller [11], and a fuzzy logic controller [19], [20]. FOSSEN proposed to use the Line of Sight (LOS) projection LOS algorithm to track and correct the deviation of the track, which can complete the underdriven ship's linear path tracking control [21].

Generally, the existing path planning methods can be divided into two categories: classic algorithms and meta-heuristic optimization algorithms. Classic algorithms include cell decomposition, artificial potential fields, and sampling-based methods. However, classic methods are very time-consuming and require sufficient storage memory. Meta-heuristic algorithms have become very popular due to their stability and flexibility and their ability to better avoid local optimizations. Thus, meta-heuristic optimization algorithms are used frequently to optimize the path planning problem. These algorithms can also be grouped in four main categories (see Table 1): evolution-based, physics-based, swarm-based methods, and human-based algorithms.

Evolution-based methods are inspired by the laws of natural evolution. The most popular evolution-inspired technique is genetic algorithms (GA) [22]. GA is an optimization algorithm based on natural genetics, including natural selection, crossover, and mutation. Other popular algorithms are probability-Based incremental learning (PBIL) [23], evolution strategy (ES) [24], the biogeography-based optimizer (BBO) [25], and genetic programming (GP) [26].

Physics-based methods imitate the physical rules in the universe. The most popular algorithms include simulated annealing (SA) [27], the gravitational local search algorithm (GLSA) [28], curved space optimization (CSO) [29], charged system search (CSS) [30], the small-world optimization algorithm (SWOA) [31], central force optimization (CFO) [32], the artificial chemical reaction optimization algorithm (ACROA) [33], big-bang big-crunch (BBBC) [34], the gravitational search algorithm (GSA) [35], black hole algorithm (BH) [36], and ray optimization algorithm (RO) [37].

The third group of methods inspired by nature includes group techniques that mimic the social behavior of animal groups. The most popular algorithms are the particle swarm optimization algorithm (PSO) [38], ant colony optimization (ACO) [39], artificial bee colony (ABC) [40], the fruit fly optimization algorithm (FOA) [41], the artificial fish-swarm algorithm (AFSA) [42], and gray wolf optimization (GWO) [43].

It's worth mentioning here that there are other meta-heuristics inspired by human behavior. Some of the most popular algorithms are seeker optimization algorithm (SOA) [44], group search optimizer (GSO) [45], league championship (LCA) [46], social-based algorithm (SBA) [47], mine blast algorithm (MBA) [48], and harmony search (HS) [49].

The PSO algorithm is a random search algorithm based on group cooperation developed by simulating the foraging behavior of birds. Thabit and Mohades [50] proposed a new method for multirobot path planning in unknown environments. This method is inspired by particle swarm optimization and is called multirobot MOPSO. It considers shortness,

TABLE 1. Classification of meta-heuristic algorithms.

Evolutionary Algorithms	Genetic Algorithm [22]
	Probability-Based Incremental Learning [23]
	Evolution Strategies [24]
	Biogeography-Based Optimizer [25]
	Genetic Programming [26]
Physics-based Algorithms	Simulated Annealing [27]
	Gravitational Local Search [28]
	Curved Space Optimization [29]
	Charged System Search [30]
	Small-World Optimization Algorithm [31]
	Central Force Optimization [32]
	Artificial Chemical Reaction Optimization Algorithm [33]
	Big-Bang Big-Crunch Curved Space Optimization [34]
	Gravitational Search Algorithm [35]
	Black Hole Algorithm [36]
	Ray Optimization Algorithm [37]
Swarm-based Algorithms	Particle Swarm Optimization Algorithm [38]
	Ant Colony Optimization [39]
	Artificial Bee Colony [40]
	Fruit Fly Optimization Algorithm [41]
	Artificial Fish-Swarm Algorithm [42]
	Gray Wolf Optimization [43]
Human-based Algorithms	Seeker Optimization Algorithm [44]
	Group Search Optimizer [45]
	League Championship [46]
	Social-Based Algorithm [47]
	Mine Blast Algorithm [48]
	Harmony Search [49]

safety, and smoothness. ACO is an algorithm inspired by the foraging behavior of ants. Although the ability of a single ant is limited, after multiple ants mark the path, the entire ant colony will tend to the optimal path. In the application of path planning, Zhu *et al.* [51] combined the artificial potential field method with the ant colony algorithm, introduced inducing heuristic factors, and dynamically adjusted the state transition rules of the ant colony algorithm, which has higher global search capabilities and a faster convergence speed. The ABC is a global optimization algorithm based on swarm intelligence. Its intuitive background comes from the honey-collecting behavior of bee colonies. Bees perform different activities according to their respective division of labor and share and communicate bee colony information to find the optimal solution to the problem. Xu *et al.* [40] introduced the coevolution framework into ABC and designed a leading artificial bee colony algorithm, which has an improved strategy that can quicken its convergence and overcome size dependence. The GWO is a swarm intelligence optimization algorithm. The algorithm is inspired by the prey hunting activities of gray wolves and developed an optimized search method. It has a strong convergence performance, a few parameters, and easy implementation. Qu *et al.* [52] proposed a novel gray wolf optimizer algorithm based on reinforcement learning, called RLGWO. In the proposed algorithm, reinforcement learning is inserted, which is to control individuals to switch operations adaptively based on accumulated performance.

The beetle antennae search (BAS) algorithm is a novel meta-heuristic optimization algorithm proposed by Jiang and Li [53]. Its idea stems from the simulation of the beetle's foraging behavior. The BAS algorithm has the advantages of having few parameters and an easy implementation, and the calculation process of the algorithm is simple, versatile, robust, less subject to initial condition constraints, and can be used to solve complex nonlinear optimization problems. After the algorithm was proposed, it was applied in many fields. Wu applied the improved algorithm of basic BAS to the route planning of unmanned aerial vehicles and mobile robots [54]. Wang used BAS's improved algorithm, BSAS, for parameter estimation of the RC model [55]. Lin *et al.* [56] proposed a new algorithm – WSBAS based on the BAS algorithm. The change strategy of inertia weight enhances the global search and local search capabilities of the method and applies the improved BAS algorithm to the Economic load distribution problem. Based on the BAS algorithm, Wang *et al.* [57] designed the adaptive mutated beetle particle swarm algorithm, replacing artificial experience with a new optimization algorithm. The optimal control parameters can be quickly determined in the control algorithm, and the heading angle control under the optimal parameters can be realized. Jiang applied the BAS algorithm and its improved algorithm to 3D path planning research [58]. According to the size of the adaptive step size, BAS can effectively jump out of the local optimal value in the early stage of exploration and quickly converge at the end of the search. Therefore, it is very suitable for solving path planning problems. However, the BAS algorithm is also affected by the initial search step and a completely random search direction and requires complex initialization parameter adjustment. The performance of the BAS algorithm in processing high-dimensional functions is not very satisfactory, and the iteration result is very dependent on the initial position of the beetle.

Wang *et al.* [59] proposed the beetle swarm optimization algorithm (BSO), which combined BAS and PSO algorithms. Compared with other group intelligent optimization algorithms, the BSO algorithm has a higher convergence speed, can avoid falling into the local optimal solution, and has certain advantages in solving many optimization problems. However, no research has been conducted on applying the BSO algorithm to the field of autonomous sailing robots path planning.

In this paper, the improved beetle swarm optimization (IBSO) algorithm is proposed by dynamically changing the step size factor and the inertia weight formula. This study applies the beetle swarm optimization algorithm to autonomous sailing robots path planning for the first time and a path planning simulation is conducted. To verify the effectiveness of the improved beetle swarm optimization algorithm, it is compared with the Particle Swarm Optimization Algorithm and the beetle swarm optimization algorithm in different cases. Compared with other swarm intelligence optimization algorithms, the improved beetle swarm optimization algorithm has a better convergence speed and convergence accuracy. It was concluded that an improved beetle swarm optimization algorithm generates the optimal path with a higher quality.

The contributions of this work are as follows. First, the track navigation control of the sailboat based on the improved mathematical model of sailing was determined. Second, a navigation test on the sailboat in the sea area of Ningbo, China was conducted. Last, the improved beetle swarm optimization algorithm for the path planning of autonomous sailing robots was proposed.

The remainder of this paper is organized as follows. Section II describes the autonomous sailing robot system. Section III describes the improved beetle swarm optimization algorithm and its application to the path planning problem. In Section IV, the track simulation and path planning simulation experiments are performed, as are the actual measurement experiments of the sailboat. In the last section conclusions are drawn.

II. AUTONOMOUS SAILING ROBOT SYSTEM

In this section, sailing modeling, track navigation control, and a small autonomous sailing robot are introduced.

A. SAILBOAT DYNAMICS AND KINEMATICS (WITH A FOUR-DEGREE-OF-FREEDOM SIMULATOR)

To validate our proposed method, the sailing model developed by researchers at the Department of Mechanical Engineering, Kanazawa Institute of Technology [60] is improved upon.

The original sailing parameters to meet the specifications of our sailing boat are changed. Table 2 describes some of the improved parameters used in the Simulink, and Table 3 explains the symbols used in this paper.

Due to the lack of data on the aerodynamics of the sailboat, the sail of the simulated model ship adopts the aerodynamic characteristic curve of the typical sail mentioned in Wang Yifu's Theory of Ship Wing [61]. The controller writes the control rules according to the classic sail wing limit diagram given in [61].

Therefore, the simulated model ship is a sailboat with the hull of the Fujin ship [60] and a sail that conforms to the aerodynamic characteristic curve.

TABLE 2. Adopted parameters.

Class	Numerical value	Class	Numerical value
Loa (m)	1.5	*Lwl (m)*	1.311
Bmax (m)	0.476	*Bwl(m)*	0.364
I_{xx}	1.277	I_{yy}	1.628
I_{zz}	0.333	—	—

TABLE 3. Adopted symbols.

Symbol	Description	Symbol	Description
$\overline{GM}$	Metacentric height of boat	$I_{xx,yy,zz}$	Moments of inertia of boat about x_b-, y_b-, and z_b-axis in general body axis system
$J_{xx,yy,zz}$	Added moments of inertia about x_b-, y_b-, andz_b-axis in general body axis system	K,N	Moments about x- and z-axis in horizontal body axis system
L	Length on design waterline	m	Mass of boat
$m_{x,y,z}$	Added masses of boat along x_b-, y_b-, and z_b-axis in general body axis system	U,V	Velocity components of boat along x- and y-axis in horizontal body axis system
V_B	Boat velocity	X,Y	Force components along x- and y-axis in horizontal body axis system
α_R	Effective attack angle of rudder	δ	Rudder angle
β	Leeway angle	ρ	Density of water
φ	Heel angle or roll angle	γ_R	Decreasing ratio of inflow Angle for rudder
ψ	Heading angle	—	—

The simulator model describes the dynamics of a four-degree-of-freedom sailboat. The equations of the mathematical model are as follows:

Surge

$$(m+m_x)\,\dot{U}-\left(m+m_y\cos^2\varphi+m_z\sin^2\varphi\right)V\dot{\psi} = X_0+X_H+X_{V_{\dot{\psi}}}V\dot{\psi}+X_R+X_s \quad (1)$$

Sway

$$\left(m+m_y cos^2\varphi+m_z sin^2\varphi\right)\dot{V}+(m+m_x)\,U\dot{\psi} +2\left(m_z-m_y\right)sin\varphi cos\varphi\cdot V\varphi = Y_H+Y_{\dot{\varphi}}\dot{\varphi}+Y_{\dot{\psi}}\dot{\psi}+Y_R+Y_s \quad (2)$$

Roll

$$\left(m+m_y cos^2\varphi+m_z sin^2\varphi\right)\dot{V}+(m+m_x)\,U\dot{\psi} +2\left(m_z-m_y\right) sin\varphi cos\varphi\cdot V\dot{\varphi}=Y_H+Y_\phi\dot{\varphi}+Y_\psi\dot{\psi}+Y_R+Y_s \tag{3}$$

Yaw

$$\left\{\left(I_{yy}+J_{yy}\right)\sin^2\varphi+\left(I_{zz}+J_{zz}\right)\cos^2\varphi\right\}\ddot{\psi} +2\left\{\left(I_{yy}+J_{yy}\right)-\left(I_{zz}+J_{zz}\right)\right\}\sin\varphi\cos\varphi\cdot\dot{\psi}\dot{\varphi} =N_H+N_\psi\dot{\psi}+N_R+N_S \tag{4}$$

The steady forces on the canoe body and fin keel are described using the following hydrodynamic derivatives:

$$X_H=\left(X'_{VV}V'^2+X'_{\varphi\varphi}\varphi^2+X'_{VVV}V'^2\right)\left(\frac{1}{2}\rho V_B^2\mathrm{LD}\right) \tag{5}$$

$$Y_H=\left(Y'_V V'+Y'_\varphi\varphi+Y'_{V_{\varphi\varphi}}V'^2\varphi+Y'_{VV}V'^3\right)\left(\frac{1}{2}\rho V_B^2\mathrm{LD}\right) \tag{6}$$

$$K_H=\left(K'_V V'+K'_\varphi\varphi+K'_{V_\infty}V'^2\varphi+K'_{VV\varphi}V'^2\varphi+K'_{VV}V'^3\right)\times\left(\frac{1}{2}\rho V_B^2\,LD^2\right) \tag{7}$$

$$N_H=\left(N'_V V'+N'_\varphi\varphi+N'_{V_{\varphi\varphi}}V'^2\varphi+N'_{VV}V'^2\varphi+N'_{VV}V'^3\right)\times\left(\frac{1}{2}\rho V_B^2\,LD^2\right) \tag{8}$$

where

$$V'=-\frac{V_B sin\beta}{V_B}=-sin\beta \tag{9}$$

The hydrodynamic forces and moments on the rudder are expressed as

$$X_R=C_{X\delta}sin\alpha_R sin\delta\left(\frac{1}{2}\rho V_B^2\mathrm{LD}\right) \tag{10}$$

$$Y_R=C_{Y\delta}sin\alpha_R cos\delta cos\delta\left(\frac{1}{2}\rho V_B^2\mathrm{LD}\right) \tag{11}$$

$$K_R=C_{K\delta}sin\alpha_R cos\delta\left(\frac{1}{2}\rho V_B^2 LD^2\right) \tag{12}$$

$$\mathrm{N_R}=\mathrm{C_{N\delta}sin\alpha_R cos\delta cos\varphi}\left(\frac{1}{2}\rho\mathrm{V_B^2L^2D}\right) \tag{13}$$

where $C_{X\delta}$ to $C_{N\delta}$ are the coefficients determined from the rudder angle tests. The effective attack angle of the rudder, α_R, is given by

$$\alpha_R=\delta-\gamma_R\cdot\beta-\tan^{-1}\left(\frac{1_R\dot{\psi}}{U}\right) \tag{14}$$

where γ_R is the decreasing ratio of the inflow angle, which is mainly caused by the downwash from the fin keel.

As previously stated, it mainly introduces the mathematical model of the sailboat and improves the parameters to better suit our simulation needs.

B. TRACK NAVIGATION CONTROL

Based on the mathematical model of the sailboat, a MATLAB simulation program is established to realize the track navigation control. The track navigation control mainly involves tracking of the sailboat's rudder and sail controller.

In our simulation, the rudder and sail are controlled in two separate single-input single-output systems that are assumed to be independent of one another.

The development of a low-order controller to guide a sailboat to its desired location is a relatively easy task, usually solved by a simple, static proportional integral differential (PID) course controller. The PID controller can easily set the process. In general, it is sufficient for the sailboat to use a set of (static) control parameters (K_p, K_i, K_d), which are usually found through trial and error, throughout the mission to reach the target location.

Fig. 1 shows the relationship between the sailing direction and the wind direction. When there is no wind, the sails float freely, and the boat slows down until it stops. Few sailboats can sail at less than 45 degrees from the direction of the wind, so the area between the wind direction and the 45 degrees angle (on both sides) is called the "no-go-zone". In this study, the upwind 45 degrees are considered to be the boundary lines of the no-go-zone. In the no-go-zone area, autonomous sailing robots are driven by the wind and can be driven normally. When the target coordinates received by the autonomous sailing robots cause the heading angle to be in the "no-go zone", the autonomous sailing robots cannot obtain the necessary speed and therefore cannot sail in a straight line. It is necessary to plan the route into a "Z" shape to avoid the "no-go zone".

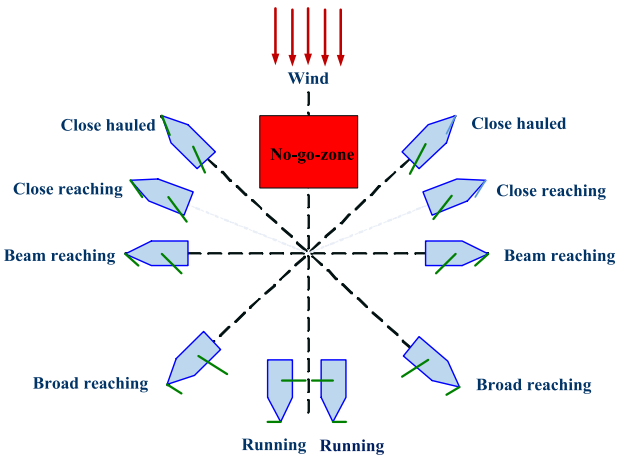

FIGURE 1. Relationship between the sailing direction and the wind direction.

For the sail, the controller's input and output variables are the relative wind direction angle and the sail's angle, respectively. The aerodynamic force of a sail is usually expressed in terms of the lift coefficient Cl and the drag coefficient Cd. The sail's lift coefficient and drag coefficient data in this study are obtained according to the typical sail wing polar diagram

(Cl-Cd diagram) reported in [61]. The best angle of attack is the difference between the relative wind direction angle and the optimal sail angle. In simulation control, linear interpolation is used when using values between the data points. In our simulation, it is assumed that the sail is adjusted at any time according to the specified data, i.e., the optimal sail angle can be achieved at any moment to obtain the best angle of attack. The sail's lift resistance coefficient data is shown in Table 4.

TABLE 4. Sail lift coefficient and drag coefficient data.

Relative wind direction angle (°)	Cl	Cd	attack (°)	Cx	Cy
15	0.8	0.2	12	0.01	0.8
30	1.1	0.3	18	0.28	1.1
50	1.2	0.4	20	0.6	1.08
70	1.3	0.6	28	1.1	1.12
90	1.4	0.7	30	1.4	0.72
110	1.4	0.8	34	1.6	0.3
130	0.9	1.1	50	1.4	0.25
150	0.5	1.4	65	1.46	0.23
180	0	1.3	90	1.38	0

For the rudder, a heading error minimizing the PID controller was designed in a control process using the MATLAB simulation. The PID controller consists of a proportional unit P, an integrating unit I, and a differential unit D.

C. A SMALL AUTONOMOUS SAILING ROBOT

In this section, the small autonomous sailing robot shown in Fig. 2 is introduced. The hardware and software of the sailboat are also introduced.

FIGURE 2. Photograph of the small-scale autonomous sailing robot.

The advantage of the small-scale sailboat is that it can have the behavior of a large ship and is easy to test.

The GPS of the sailing robot adopts the ATK-S1216F8-BD high performance positioning module of ALIENTEK and is used to locate the current position of the sailboat and provide data for the calculation of the path and control. The electronic compass uses the low-cost module GY953. In this compass, a gyroscope and acceleration sensor are used; the magnetic field sensor obtains the direct angle data through the data fusion algorithm and can directly output the heading angle. The wind direction sensor consists of a high precision single-turn absolute encoder and a wind vane.

All electronic devices are managed via Raspberry Pi. The autonomous sailing robot has two modes of operation, namely, automatic and manual modes, in which the sailboat can sail autonomously and can be operated by remote control, respectively.

Fig. 3 illustrates the framework of the autonomous sailing robot software, which can be divided into four main parts. The perceptual modules are used to collect the internal state and environmental data; the internal data include information on the sail and rudder angle, heading, and absolute position of the sailboat, and the environmental data include information on the wind direction. The angle of the sail and rudder are controlled by the control module.

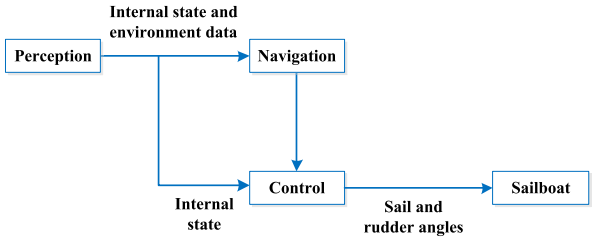

FIGURE 3. Software framework.

III. PATH PLANNING

In this section, the Beetle Antennae Search algorithm (BAS), the adaptive factor, the improved beetle swarm optimization algorithm (IBSO), and the path planning algorithm based on IBSO are introduced.

A. BEETLE ANTENNAE SEARCH ALGORITHM

The Beetle Antennae Search (BAS) algorithm is an intelligent optimization algorithm that simulates beetle foraging behavior. When a beetle forages, it will use its left and right antennae to sense the odor intensity of food. If the odor intensity received by the left antennae is large, it will fly to the left with the strong odor intensity; otherwise, it will fly to the right.

The modeling process of the BAS algorithm is as follows:

1) THE POSITION AND ORIENTATION OF THE BEETLE ARE RANDOMLY GENERATED AND NORMALIZED

$$\vec{b} = \frac{rands(Dim, 1)}{\|rands(Dim, 1)\|} \tag{15}$$

where Dim is the spatial dimension.

2) THE SPACE COORDINATES OF THE BEETLE'S LEFT AND RIGHT SIDES AND ITS ANTENNAE ARE CREATED

$$\begin{cases} x_{rt} = x^t + d_0 * \vec{b}/2 \\ x_{lt} = x^t - d_0 * \vec{b}/2 \end{cases} \tag{16}$$

Among them, x^t represents the position of the beetle's antennae at the t-th iteration, x_{rt} represents the position of the beetle's right antennae at the t-th iteration, x_{lt} represents the position of the beetle's left antennae at the t-th iteration, and d_0 represents the beetle's two positions.

3) THE DIRECTION IN WHICH THE BEETLE IS MOVING IS DETERMINED

According to the selected fitness function, the respective fitness values of the left and right antennae are calculated, and the beetle moves towards the antennae with a small fitness value.

4) THE LOCATION OF THE BEETLE IS ITERATIVELY UPDATED

$$x^{t+1} = x^t + \delta^t * \vec{b} * sign\left(f\left(x_{rt}\right) - f\left(x_{lt}\right)\right) \tag{17}$$

$$\delta^t = eta * \delta^{t-1} \tag{18}$$

Among them, δ^t is the step factor, sign is a sign function, and eta is the parameter step factor, which is usually 0.95.

B. ADAPTIVE FACTOR

In the BAS algorithm, the parameter step factor eta is the key to controlling the convergence speed of the algorithm, and the step size is closely related to the factor eta. The specific reasons are as follows: If the step size decays sufficiently slowly, the global search capability is stronger, but the convergence rate is too slow. Alternatively, if the step size decays too fast, the global optimal solution may not be obtained.

The step size factor controls the convergence speed of the algorithm. The larger the step size factor (towards 1), the slower the convergence speed, but the global search ability is strong; otherwise, the smaller the factor (towards 0), the faster the convergence speed, but it can easily fall into the local minimum value. However, the step factor in the basic BAS is fixed during the optimization process. To make the algorithm obtain a better optimization ability, this paper proposes an improved method to dynamically change the step size factor, which is an adaptive factor. Specifically, in the early stage of optimization, in order to expand the overall search range in the solution space and increase the speed of optimization, a larger step factor should be used; in the later stage of optimization, the search solution tends to be stable. To improve the accuracy of the solution, the step factor should be reduced. Additionally, the smaller the initial step size factor, the easier it is to fall into the local minimum value, so a higher initial value should be given, such as 0.95.

Based on the above considerations, the following adjustment mechanism is set:

$$\begin{cases} \beta = eta - 0.2 * ((i+1)/(5*n) + 0.5), & f_i > f_{\min} \\ \beta = eta, & f_i \le f_{\min} \end{cases} \tag{19}$$

where f_i is the current fitness value, $f_{\min}$ is the historical optimal fitness value, i is the current number of iterations, n is the total number of iterations, eta is the default step factor (generally 0.95), and β is the current step factor. When the current fitness value is less than the historical optimal fitness value, this indicates that the current optimization performance is good. At this time, the default value of the step factor is maintained to ensure a global search. In contrast, it is considered that the optimization performance is not good, and the step size factor is reduced to accelerate the convergence speed; as the number of iterations increases, the minimum fitness value tends to be stable, and the decrease in the step factor should be expanded.

C. IMPROVED BEETLE SWARM OPTIMIZATION ALGORITHM

The iteration results of the BAS algorithm have a great relationship with the initial position of the beetle. In other words, the choice of the initial position greatly affects the efficiency and effect of optimization. PSO simulates birds in a flock by designing massless particles, where each particle represents a potential solution to the problem, and each particle corresponds to a fitness value determined by a fitness function.

Inspired by the PSO algorithm, the BAS algorithm is further improved upon to expand individuals into groups. That is, the BSO algorithm to be introduced. The basic principle of the algorithm is to replace the particles in the particle swarm algorithm with the beetle, that is, to use BAS optimization to replace the comparison of individual optimal values in the particle swarm algorithm. The initial position and velocity parameters of the beetle are the same as in a particle swarm optimization. In the iterative process, the way to update the position of the beetle not only depends on the current global optimal solution of the individual beetle, but also on each iteration, where the beetle judges the odor concentration. $X = (X_1, X_2, \ldots, X_n)$ is used to represent the beetle swarm of size n in the S-dimensional search space, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{iS})^T$ is an S-dimensional vector representing the position attribute of beetle i in the S-dimensional search space and is a potential solution to the optimization problem. $V_i = (v_{i1}, v_{i2}, \ldots, v_{iS})^T$ is the speed attribute of the beetle i. The individual extreme is represented by $P_i = (p_{i1}, p_{i2}, \ldots, p_{iS})^T$, and the global extreme is represented by $P_g = (p_{g1}, p_{g2}, \ldots, p_{gS})^T$. The speed and position updates of the BSO algorithm can be expressed as follows [59]:

$$x_{is}^{k+1} = x_{is}^k + \lambda v_{is}^k + (1-\lambda)\xi_{is}^k \tag{20}$$

$$v_{is}^{k+1} = \omega v_{is}^k + c_1 r_1 \left(p_{is}^k - x_{is}^k\right) + c_2 r_2 \left(p_{gs}^k - x_{gs}^k\right) \tag{21}$$

$$\xi_{is}^{k+1} = \delta^k * v_{is}^k * sign\left(f\left(x_{rs}^k\right) - f\left(x_{ls}^k\right)\right) \tag{22}$$

$$x_{rs}^{k+1} = x_{rs}^k + v_{is}^k * \frac{\vec{d}}{2}; x_{ls}^{k+1} = x_{ls}^k - v_{is}^k * \frac{\vec{d}}{2} \tag{23}$$

where $s = 1, 2, \ldots, S; i = 1, 2, \ldots, n$; and k is the current number of iterations. V_{is} is expressed as the speed of the beetles, and ξ_{is} represents the increase in the beetle position

movement. The loosening factor (λ) and inertia weight (ω) are adjustable parameters, and r_1, r_2 are two random functions in the range of [0, 1]. The parameters c_1 and c_2 determine the impact degree of the individual and global extremes on the beetle.

In the BSO algorithm, the greater the value of the inertial weight ω, the wider the search range corresponding to the particle; that is, a stronger global search ability and a weaker local search ability. The smaller the value of the inertial weight ω, the greater the search range corresponding to the particle narrow; that is, a stronger local search ability and a weaker global search ability. This article modifies the BSO from the perspective of improving the inertia weight. In this paper, the inertia weight formula is adjusted as follows:

$$\omega(k) = rand * \omega_{\min} * (1 - \cosh) + \omega_{\max} * \cosh \quad (24)$$

where rand is a random function in the range [0, 1], $\omega_{min} = 0.4$, $\omega_{max} = 0.9$, $h = \pi t / 2k_{max}$, and k_{max} is the maximum number of iterations during the entire iteration. The improved strategy has a larger value and a slower change rate ω in the early stage of the search, which is conducive to the algorithm for a longer global search time. The possibility of finding a globally optimal solution is greatly improved. The fast ω enhances the ability to continuously approach the global optimal solution in the later stage of the search, find the global optimal solution, and improve the accuracy of the algorithm.

According to the improvement of the adaptive factor in the BAS algorithm, the improved BAS algorithm is combined with the PSO algorithm to obtain the IBSO.

In the iteration of the IBSO algorithm, the location update cooperates with the beetle monomer search mechanism to learn the update strategy of the PSO algorithm. Combinatorial strategies can quicken the overall iterative convergence speed and reduce the possibility that the population will fall into a locally optimal solution. The IBSO algorithm includes exploration and development capabilities and belongs to global optimization. Additionally, the linear combination of speed and beetle search improves the speed and accuracy of the population optimization, making the algorithm more stable.

The pseudo code of the IBSO algorithm is presented.

Algorithm 1 IBSO Algorithm

input: Initialize the swarm $X_i(i = 1, 2, \ldots, n)$, v, δ, K, velocity range v_{min} and v_{max}
output: x_{best}, f_{best}

```
Calculate the fitness of each search agent
While (k < K)
  Set inertia weight ω according to (24)
  for each search agent
      Calculate f(X_rs) and f(X_ls) according to (23)
      Update the incremental function ξ according
      to (22)
      Update the speed formula V according to (21)
      Update the position of the current search
      agent according to (20)
  end for
  Calculate the fitness of each search agent f(x)
  Record and store the location of each search agent
  for each search agent
    if f(x) < f_pbest then
      f_pbest ← f(x)
    end if
    if f(x) < f_gbest then
      f_gbest ← f(x)
    end if
  end for
  Update x* if there is a better solution
  Update step factor δ according to (18) and (19)
end while
Return x_best, f_best
```

D. PATH PLANNING ALGORITHM BASED ON THE IMPROVED BEETLE SWARM OPTIMIZATION ALGORITHM

The autonomous sailing robot path planning problem is an NP-hard optimization problem. The main purpose of solving this problem is that the robot should reach the target position from the start position with the shortest path without encountering any obstacles. This consists of the start and target positions, the size of the obstacle, the shape of the obstacle, the number of obstacles, and the boundary of the area. The objective function of the path planning problem is given below:

$$J = \min_{\mathbf{x},\mathbf{y}} Q(1 + \eta V) \quad (25)$$

where V is the violation cost; η is violation coefficient, which is a positive constant; and Q denotes the total distance between the start and target positions. The pseudo code is as follows:

Algorithm 2 Violation's Calculation

```
Violation ← 0
for each obstacle
  Calculate distance vector between the obstacle's center
and path
  a ← max (1 – distance/radius_obs, 0)
  Violation ← Violation + mean (a)
end for
```

The flowchart of the route planning based on the IBSO algorithm is shown in Fig. 4.

IV. SIMULATION STUDIES AND EXPERIMENTAL RESULTS

In this section, the results of tracking navigation control, the autonomous sailing robot experiment, and path planning are introduced.

A. TRACKING NAVIGATION CONTROL

Fig. 5 shows the results of the tracking navigation simulation of a rectangular path. It can be seen that the autonomous sailing robot works well under the conditions of headwind, crosswind, and downwind. In the following simulations,

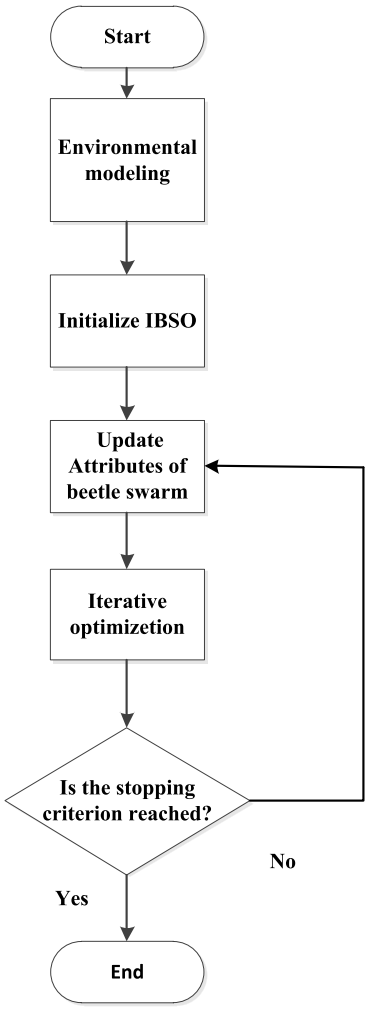

FIGURE 4. Flowchart of path planning based on the IBSO algorithm.

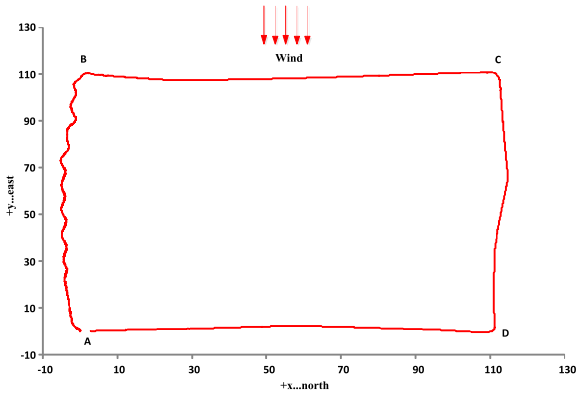

FIGURE 5. Sailboat rectangular trajectory with a given wind direction.

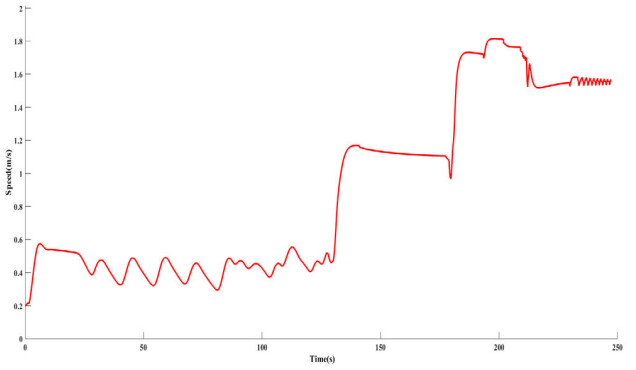

FIGURE 6. Plot of the boat speed as a function of U-time.

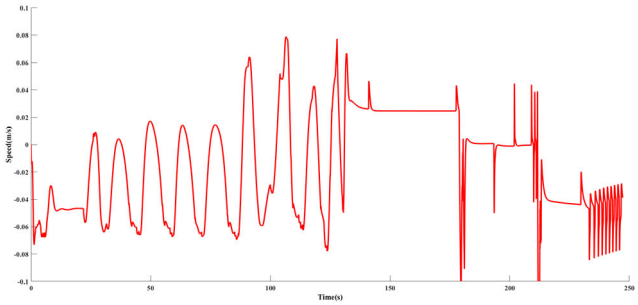

FIGURE 7. Plot of the boat speed as a function of V-time.

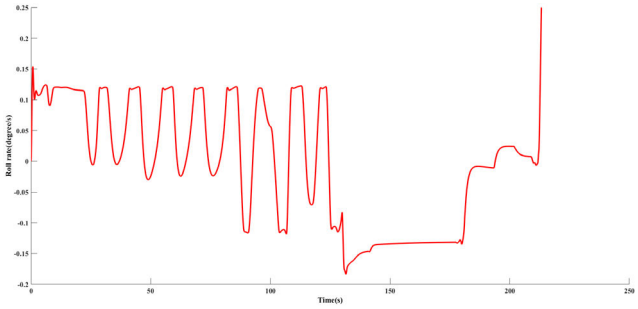

FIGURE 8. Plot of the roll rate as a function of time.

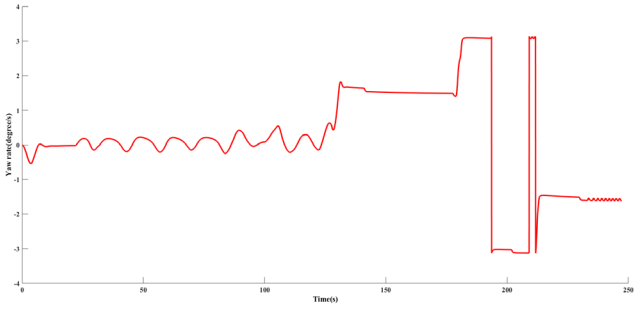

FIGURE 9. Plot of the yaw rate as a function of time.

the default parameters for the sailboat are set as follows: P = 1.2; I = 0.07; D = 0.4; true wind speed = 5 m/s; and true wind angle = 0°. The data of the surge, roll, sway, and yaw are displayed in Figs. 6–9.

In Fig. 5, point A to point B is the upwind sailing area, and the route is "Z" shaped; point C to point D is sailing downwind; and point B to point C and point D to point A

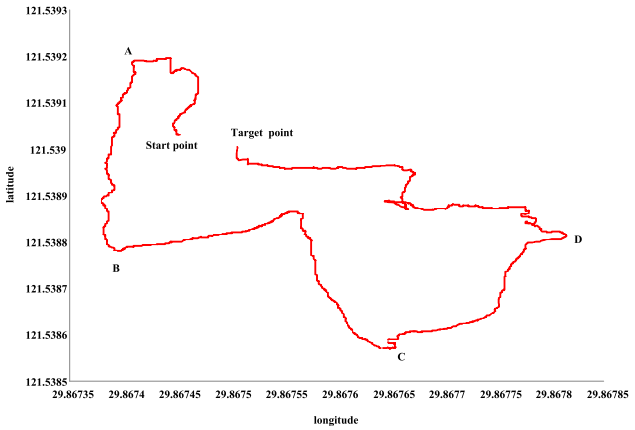

FIGURE 10. Sailing track in Ningbo, China.

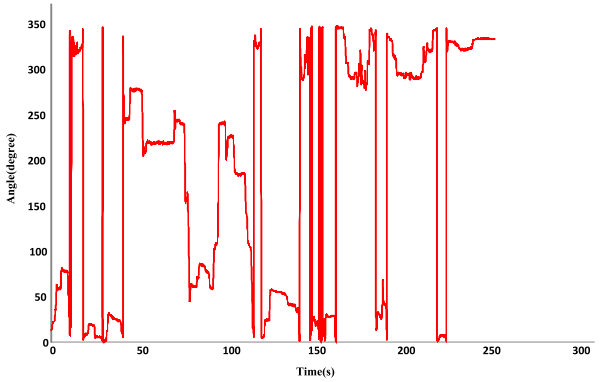

FIGURE 11. Plot of the relative wind angle as a function of time.

are cross-wind driving, where the sailboat is approximately straight.

The test results prove that the controller can control the sailboat to maintain a good straight-line navigation ability and turn in time to reach a new route and can also plan a new Z-shaped route to maintain a good speed when heading upwind.

As seen from the figures above, the controller makes the autonomous sailing robot follow the specified route, which proves the feasibility of the controller.

B. EXPERIMENT

The experiments were conducted on a lake in Ningbo, China. One purpose of these experiments was to test the navigation and control modules of the sailboat without the obstacle detection module. The purpose of the test was mainly to evaluate the performance of different sails in different winds and record the wind direction, bow orientation, GPS data, and navigation data. Fig. 10 shows a four-minute trajectory of the autonomous sailboat; Fig. 11 shows the true wind direction in the actual course of the sailboat; and Fig. 12 displays the heading angle of the sailboat in the experiment.

The autonomous sailing robot starts from the start point, travels a certain distance in the manual control mode, and

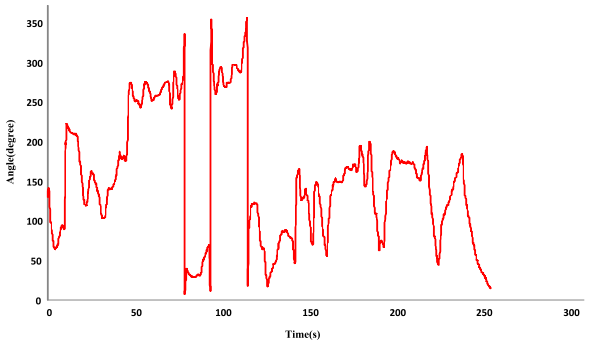

FIGURE 12. Plot of the heading angle as a function of time.

then switches to the autonomous navigation mode. It then passes the designated points A, B, C, and D, and successfully reaches the target point. The autonomous sailing robot successfully sailed to four predetermined points without any manual intervention.

C. PATH PLANNING

1) THE EXPERIMENT COMPARISON RESULTS

In the simulation, the influence of such factors as currents and waves is ignored, and it is assumed that the sailboat moves forward under the action of constant wind. The obstacle was set as static. In the simulations, a volumeless and massless particle is used to simulate an autonomous sailing robot. The start points of the path planning are S_1 (10, 0) , $S_2(-30, 0)$, and the target points of the path planning are T_1 (10, 70) , $T_2(40, 60)$. This section is divided into four cases according to the start points, target points, and the number of obstacles. The information of the four cases is shown in Table 5.

TABLE 5. Information for the four cases.

Case number	Start point	Target point	Number of obstacles
1	S_1	T_1	5
2	S_1	T_1	8
3	S_2	T_2	5
4	S_2	T_2	8

Some simulation experiments were given here to verify the good performances of IBSO in the path planning problem. To show the superiority of the proposed algorithm, the path planning problem using IBSO algorithms is solved and its performance has been compared with PSO and BSO.

For a fair comparison, the population sizes of all algorithms are set to 20, and the maximum number of iterations is set to 100. For each algorithm, a total of 30 runs were performed for each experiment. These algorithms are compared based on the solution's quality, stability, and convergence speed. The quality of the solution can be represented by the average optimal fitness value, which is the average of the 30 optimal

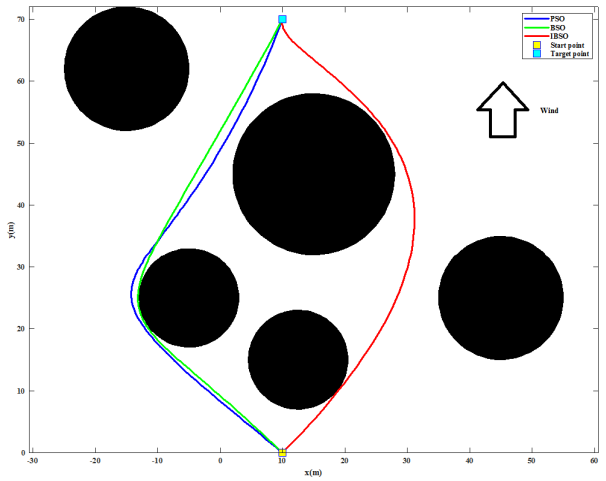

FIGURE 13. Sailboat trajectories in Case 1.

TABLE 6. The related parameter values.

Algorithm	Parameters
PSO	$c_1 = c_2 = 1.5; \omega = 1$
BSO	$c_1 = c_2 = 1.5; \omega = 1; \delta = 5; eta = 0.95$
IBSO	$c_1 = c_2 = 1.5; \omega_{\min} = 0.4, \omega_{\max} = 0.9; \delta = 5; eta = 0.95$

fitness values produced by 30 trials. The stability of the algorithm is determined by the standard deviation. The convergence speed of the algorithm depends on the number of iterations required for the algorithm to converge to the optimal solution. The initial parameters of the three algorithms are listed in Table 6. Four sets of simulation experiments were conducted at different start points, target points and obstacles, and the simulation results are shown in Figs. 13-24. The path planning comparison results are shown in Table 7.

For the first case, Fig. 13 shows some differences in the experimental results of the three algorithms. Although all three algorithms can successfully generate a collision-avoidance path, IBSO found a path close to the optimal path and performed better. Fig. 14 shows the convergence curves of the three algorithms in Case 1. The results show that the convergence speed and accuracy of this algorithm are better than those of other algorithms. PSO reached the best optimal value in the 79th iteration, BSO obtained the best optimal value in the 49th iteration, and IBSO found its best optimal value in the 38th iteration. The statistical results are shown in Fig. 15 and Table 6. Among them, the best PSO value is 85.8227, the worst is 87.4808, the average is 86.5876, the median is 86.5599, and the standard deviation is 0.4934; the best BSO value is 83.1233, the worst is 87.0604, the average is 84.5372, the median is 83.8286, and the standard deviation is 1.2618; the best IBSO value is 82.2377, the worst is 85.5946, the average is 82.9455, the median is 82.6548, and the standard deviation is 0.8604. Compared with the other algorithms, the simulation results of the IBSO

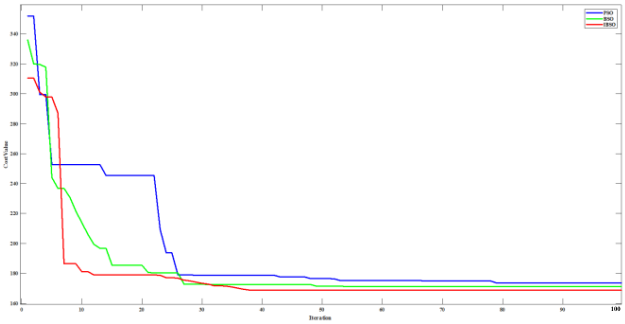

FIGURE 14. The convergence cures of the three algorithms in Case 1.

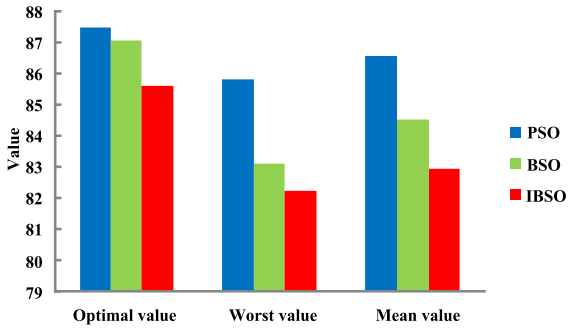

FIGURE 15. The statistical results of the three algorithms in Case 1.

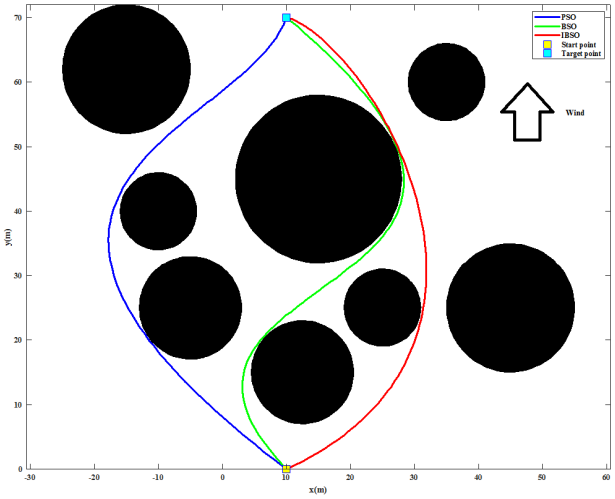

FIGURE 16. The sailboat trajectories in Case 2.

algorithm have a smaller best value, median value, worst value, average value, and the optimal number of iterations.

For the second case, the number of obstacles is slightly increased. Fig. 16 shows that IBSO can find a feasible path that meets the path planning requirements at the lowest cost. Fig. 17 shows the convergence curves of the three algorithms in Case 2. It can be seen from these curves that the IBSO algorithm has the best convergence speed and accuracy. The PSO algorithm reached the best optimal value in the 81st iteration, BSO obtained the best optimal value in the 57th iteration,

TABLE 7. Comparison results for the four cases.

	Algorithm	Best	Mean	Median	Worst	Standard deviation	Number of optimal iterations
Case 1	PSO	85.8227	86.5876	86.5599	87.4808	0.4934	79
	BSO	83.1233	84.5372	83.8286	87.0604	1.2618	49
	IBSO	**82.2377**	**82.9455**	**82.6548**	**85.5946**	0.8604	**38**
Case 2	PSO	89.3238	92.4022	92.5985	93.7377	0.9383	81
	BSO	85.0565	89.5392	89.4087	92.5399	2.1681	57
	IBSO	**82.2766**	**85.9451**	**86.5485**	**90.1005**	2.1536	**31**
Case 3	PSO	100.3225	105.1325	105.2048	107.7842	1.1257	56
	BSO	95.6328	100.8290	100.1502	105.0621	2.5626	46
	IBSO	**94.8834**	**97.2346**	**96.2335**	**101.2114**	2.2452	**35**
Case 4	PSO	106.0596	106.7795	106.7145	107.6076	0.4150	44
	BSO	95.3259	101.3478	100.2084	106.6331	3.2420	39
	IBSO	**94.8942**	**96.9031**	**95.3183**	**104.5172**	2.6783	**10**

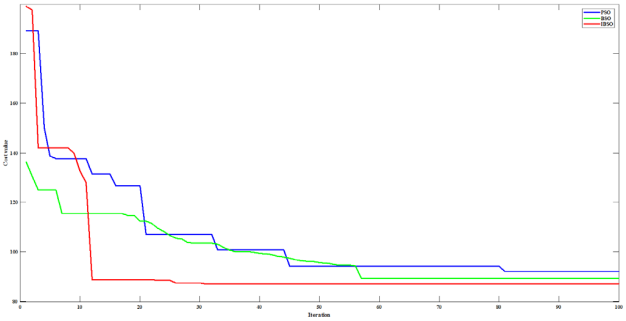

FIGURE 17. The convergence cures of the three algorithms in Case 2.

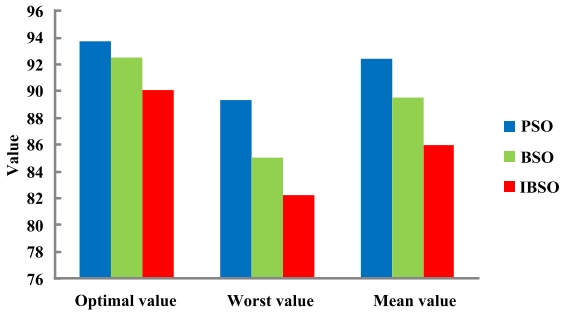

FIGURE 18. The statistical results of the three algorithms in Case 2.

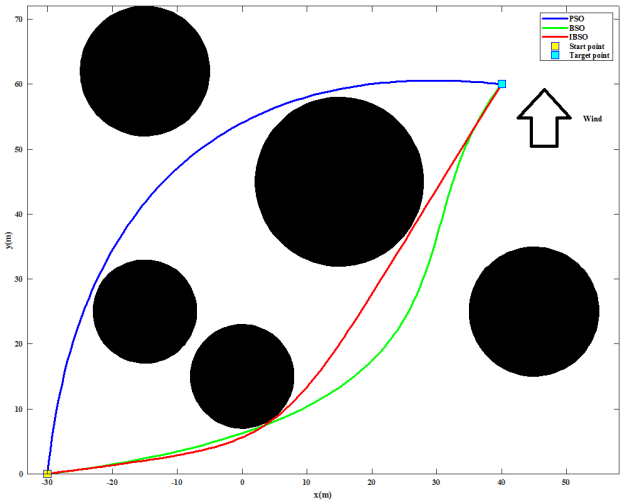

FIGURE 19. The sailboat trajectories in Case 3.

and IBSO found its best optimal value in the 31st iteration. Among them, the best IBSO value is 82.2766, the worst is 90.1005, the average is 85.9451, the median is 86.5485, and the standard deviation is 2.1536. For the statistical results shown in Fig. 18 and Table 6, the simulation results of the IBSO algorithm have a smaller best value, median value, worst value, average value, and the optimal number of iterations, which demonstrate its good performance.

For the third and fourth cases, Figs. 19 and 22 show that the experimental results of the three algorithms have some differences. Compared with PSO and BSO, IBSO can obtain the shortest path. It can be seen from Figs. 20 and 23 that IBSO has the best convergence. As the statistical results are shown in Figs. 21 and 24 and Table 6, the IBSO still provides the best results in terms of the best, worst, mean, and average values.

In summary, from the above experimental results, it can be seen that the IBSO algorithm can search for a satisfactory path. The excellent performance of IBSO is verified. The results show that the convergence speed and accuracy of the IBSO are better than the PSO and BSO. Compared with the PSO and BSO, the IBSO algorithm has a smaller

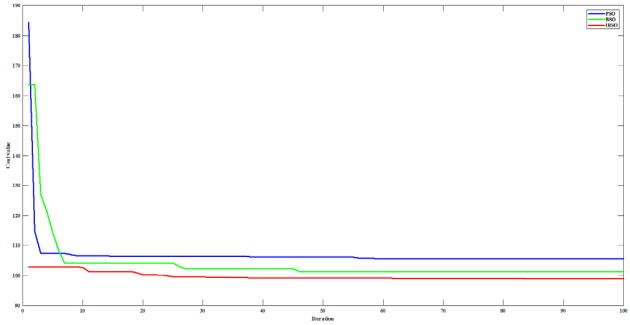

FIGURE 20. The convergence cures of the three algorithms in Case 3.

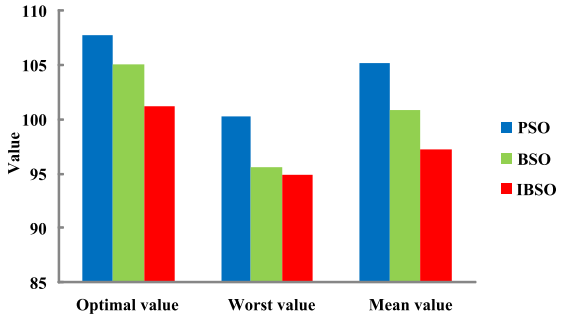

FIGURE 21. The statistical results of the three algorithms in Case 3.

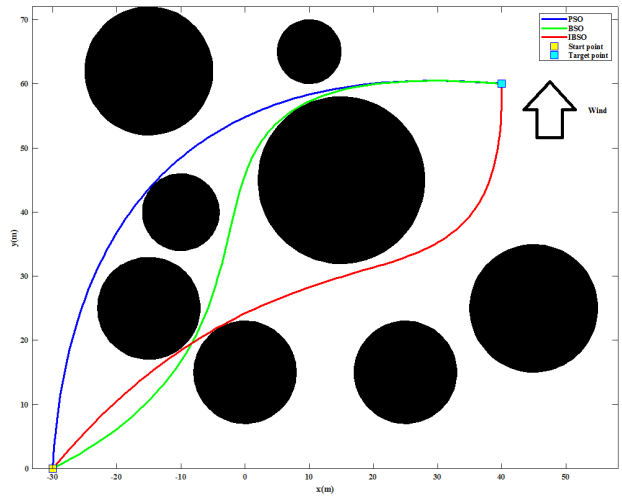

FIGURE 22. Sailboat trajectories in Case 4.

best value, median value, worst value, average value, and the optimal number of iterations, which demonstrate its superior performance and effectiveness. In other words, the IBSO algorithm is superior to the PSO and BSO algorithms in path planning.

2) COMPUTATIONAL COMPLEXITY COMPARISON

In this section, the computational complexities of the three algorithms are compared. To compare their performances, a more specific assessment needs to be done. Therefore,

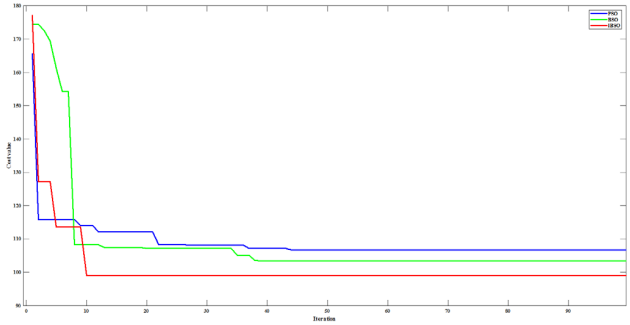

FIGURE 23. The convergence cures of the three algorithms in Case 4.

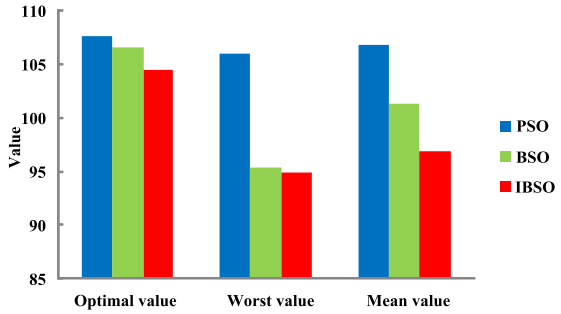

FIGURE 24. The statistical results of the three algorithms in Case 4.

TABLE 8. The time complexity of the three algorithms.

Algorithm	Complexity in one iteration	Complexity in M iteration
PSO	O(ND)	O(MND)
BSO	O(ND)	O(MND)
IBSO	O(ND)	O(MND)

TABLE 9. The space complexity of the three algorithms.

Algorithm	Complexity of algorithm
PSO	O(ND)
BSO	O(ND)
IBSO	O(ND)

the computational complexity is compared in two parts: time and space.

a: TIME COMPLEXITY

The time complexity of the algorithm measures the time it takes for the function to run the algorithm, excluding coefficients and low-order terms. Some parameters are indicators and the growth time of these parameters are analyzed and selected. In the case of time complexity, three parameters are set: population size (N), parameter dimension (D), and number of iterations (M). Table 8 shows the comparison results of time complexity.

b: SPACE COMPLEXITY

The space complexity of a program refers to the amount of memory required to run a program. Using the space

TABLE 10. Benchmark functions.

Benchmark functions	Equations	Search area
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	$x_n\in[-100 \quad 100]$
Rosenbrock	$f_2(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$x_n\in[-30 \quad 30]$
Quartic with noise	$f_3(x)=\sum_{i=1}^{n}ix_i^4+\text{random}[0,1)$	$x_n\in[-1.28 \quad 1.28]$
Rastrign	$f_4(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$x_n\in[-5.12 \quad 5.12]$
Ackley	$f_5(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	$x_n\in[-32 \quad 32]$
Griewank	$f_6(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$x_n\in[-600 \quad 600]$

TABLE 11. Mean values of the PSO, BSO, and IBSO.

	PSO	BSO	IBSO
f_1	0	0	0
f_2	1.51E+04	0.6578	0.4074
f_3	2.98E-04	5.17E-04	1.67E-04
f_4	5.1785	0.4311	0.3649
f_5	4.6379	0.1097	5.49E-02
f_6	0.1348	0.1267	0.1045

complexity of the program, a pre-estimation of how much memory the program needs to run can be known. In the case of space complexity, the number of iterations does not affect the results. The comparison results of space complexity are shown in TABLE 9.

From TABLE 8 and TABLE 9, it can be seen that these three algorithms have no significant differences in time complexity and space complexity. In other words, IBSO achieves a better path planning performance without increasing the computational complexity of the algorithm.

V. CONCLUSION

A simulation based on a mathematical model of an autonomous navigation robot was conducted. The controller was successfully used to drive the autonomous navigation robot through the designated route. A navigation test on the sailboat was conducted in a sea area. Furthermore, the improved beetle swarm optimization algorithm was proposed here by dynamically changing the step size factor and the inertia weight formula. The improved beetle swarm optimization algorithm for the path planning of autonomous sailing robots was improved. This algorithm has the advantages of a simple modeling, fewer adjustment parameters, a small calculation amount, and a fast convergence speed. IBSO allows for the obstacle avoidance path planning of autonomous sailing robots. Two aspects were mainly compared. One was the path planning performance, which is measured by the planning model; the other was the computational performance, which is measured by computational complexity. Through a comparison of the path planning performance, the proposed IBSO algorithm outperforms both the PSO and BSO algorithms. Through a comparison of the computational performance, the IBSO algorithm has the same computational complexity as PSO and BSO in terms of time complexity and space complexity.

Currently, the sails and rudder of autonomous sailing robots are controlled separately. Because the change in the sail angle inevitably leads to a change in the boat direction, a joint control of the sails and rudder is needed. Future plans include testing the proposed algorithm on an autonomous sailing robot with a 2.38 m-long and 1.1 m-wide hull and 2.7 m-high sails. Preliminary tests were conducted on a river near Qingdao, China, and the autonomous sailing robot was successfully controlled. In addition, the study of path

planning problems of heuristic optimization algorithms in autonomous sailing robots will be continued.

APPENDIXES

COMPARISONS OF THE PROPOSED IBSO, THE ORIGINAL BSO, AND THE PSO WITH BENCHMARK FUNCTIONS

To comprehensively verify the optimization performance of the improved beetle swarm optimization (IBSO) algorithm, several benchmark tests are conducted in this section. The original beetle swarm optimization (BSO) algorithm [59] and the original particle swarm optimization (PSO) algorithm are used for comparisons. 6 benchmark functions with 30 dimensions, 'Sphere', 'Rosenbrock', 'Quartic with noise', 'Rastrigin', 'Ackley', and 'Griewank' are selected. The three algorithms were tested 30 times and the mean values were obtained. It can be seen from Table 11 that the proposed IBSO can obtain smaller mean values for the optimal fitness results than PSO and BSO in all benchmark functions.

ACKNOWLEDGMENT

The authors of this article sincerely acknowledge the help of anonymous reviewers.

REFERENCES

[1] Y. Liu, X. Wang, J. You, and C. Chen, "Ocean wave buoy based on parallel six-dimensional accelerometer," *IEEE Access*, vol. 8, pp. 29627–29638, 2020.
[2] K. Isaka, K. Tsumura, T. Watanabe, W. Toyama, M. Sugesawa, Y. Yamada, H. Yoshida, and T. Nakamura, "Development of underwater drilling robot based on earthworm locomotion," *IEEE Access*, vol. 7, pp. 103127–103141, 2019.
[3] R. A. S. Fernandez, D. Grande, A. Martins, L. Bascetta, S. Dominguez, and C. Rossi, "Modeling and control of underwater mine explorer robot UX-1," *IEEE Access*, vol. 7, pp. 39432–39447, 2019.
[4] P. Ridao, M. Carreras, D. Ribas, P. J. Sanz, and G. Oliver, "Intervention AUVs: The next challenge," *Annu. Rev. Control*, vol. 40, pp. 227–241, Jan. 2015.
[5] D. Ribas, P. Ridao, A. Turetta, C. Melchiorri, G. Palli, J. J. Fernandez, and P. J. Sanz, "I-AUV mechatronics integration for the TRIDENT FP7 project," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 5, pp. 2583–2592, Oct. 2015.
[6] J. M. Larrazabal and M. S. Peñas, "Intelligent rudder control of an unmanned surface vessel," *Expert Syst. Appl.*, vol. 55, pp. 106–117, Aug. 2016.
[7] R. Stelzer and T. Pröll, "Autonomous sailboat navigation for short course racing," *Robot. Auto. Syst.*, vol. 56, no. 7, pp. 604–614, Jul. 2008.
[8] R. Stelzer, K. Jafarmadar, H. Hassler, and R. Charwot, "A reactive approach to obstacle avoidance in autonomous sailing," in *Proc. Int. Robotic Sailing Conf.*, 2010, pp. 13–16.
[9] H. Erckens, G.-A. Busser, C. Pradalier, and R. Y. Siegwart, "Avalon: Navigation strategy and trajectory following controller for an autonomous sailing vessel," *IEEE Robot. Autom. Mag.*, vol. 17, no. 1, pp. 45–54, Mar. 2010.
[10] M. Romero, Y. Guo, S.-H. Ieng, F. Plumet, R. Benosman, and B. Gas, "Omni-directional camera and fuzzy logic path planner for autonomous sailboat navigation," in *Proc. Iberoamer. Conf. Electron. Eng. Comput. Sci.*, 2011, pp. 335–346.
[11] H. Saoud, M.-D. Hua, F. Plumet, and F. Ben Amar, "Routing and course control of an autonomous sailboat," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2015, pp. 1–6.
[12] L. Jaulin and F. Le Bars, "An interval approach for stability analysis: Application to sailboat robotics," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 282–287, Feb. 2013.
[13] M. Tranzatto, A. Liniger, S. Grammatico, and A. Landi, "The debut of Aeolus, the autonomous model sailboat of ETH zurich," in *Proc. OCEANS Genova*, May 2015, pp. 1–62.
[14] F. Plumet, C. Petres, M.-A. Romero-Ramirez, B. Gas, and S.-H. Ieng, "Toward an autonomous sailing boat," *IEEE J. Ocean. Eng.*, vol. 40, no. 2, pp. 397–407, Apr. 2015.
[15] F. Plumet, H. Saoud, and M.-D. Hua, "Line following for an autonomous sailboat using potential fields method," in *Proc. MTS/IEEE OCEANS Bergen*, Jun. 2013, pp. 1–6.
[16] H. Saoud, F. Plumet, and F. B. Amar, "Adaptive sampling with a fleet of autonomous sailing boats using artificial potential fields," in *Marine Robotics and Applications*. Cham, Switzerland: Springer, 2018, pp. 15–27.
[17] C. Petres, M.-A. Romero-Ramirez, and F. Plumet, "Reactive path planning for autonomous sailboat," in *Proc. 15th Int. Conf. Adv. Robot. (ICAR)*, Jun. 2011, pp. 112–117.
[18] N. A. Cruz and J. C. Alves, "Auto-heading controller for an autonomous sailboat," in *Proc. OCEANS IEEE SYDNEY*, May 2010, pp. 1–6.
[19] R. Stelzer, T. Proll, and R. I. John, "Fuzzy logic control system for autonomous sailboats," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, Jun. 2007, pp. 1–6.
[20] Q. Wang, M. Kang, J. Xu, and J. Xu, "Autonomous sailboat track following control," in *Proc. Int. Robot. Sailing Conf.*, 2016, pp. 125–136.
[21] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proc. Volumes*, vol. 36, no. 21, pp. 211–216, Sep. 2003.
[22] J. H. Holland, *Genetic Algorithms*. New York, NY, USA: Scientific American, 1992, pp. 66–73.
[23] D. Dasgupta and Z. Michalewicz, "Evolutionary algorithms in engineering applications," *Int. J. Evol. Optim.*, vol. 1, no. 1, pp. 93–94, 1999.
[24] I. Rechenberg, "Evolutionsstrategien," in *Simulationsmethoden der Medizin und Biologie*. Berlin, Germany: Springer, 1978, pp. 83–114.
[25] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
[26] J. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. New York, NY, USA, 2005, pp. 127–164.
[27] E. Aarts, J. Korst, and W. Michiels, "Simulated annealing," in *Search Methodologies*. Dordrecht, The Netherlands: Springer, 2005, pp. 187–210.
[28] B. Webster and P. J. Bernhard, "A local search optimization algorithm based on natural principles of gravitation," in *Proc. Int. Conf. Inf. Knowl. Eng.*, 2003, pp. 255–261.
[29] F. Farrahi Moghaddam, R. Farrahi Moghaddam, and M. Cheriet, "Curved space optimization: A random search based on general relativity theory," 2012, *arXiv:1208.2214*. [Online]. Available: http://arxiv.org/abs/1208.2214
[30] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: Charged system search," *Acta Mechanica*, vol. 213, nos. 3–4, pp. 267–289, Sep. 2010.
[31] H. Du, X. Wu, and J. Zhuang, "Small-world optimization algorithm for function optimization," in *Proc. Int. Conf. Natural Comput.*, 2006, pp. 264–273.
[32] R. A. Formato, "Central force optimization," *Prog. Electromagn. Res*, vol. 77, pp. 425–491, Aug. 2007.
[33] B. Alatas, "ACROA: Artificial chemical reaction optimization algorithm for global optimization," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13170–13180, Sep. 2011.
[34] O. K. Erol and I. Eksin, "A new optimization method: Big bang–big crunch," *Adv. Eng. Softw.*, vol. 37, no. 2, pp. 106–111, 2006.
[35] H. Shah-Osseini, "Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation," *Int. J. Comput. Sci. Eng.*, vol. 6, nos. 1–2, pp. 132–140, 2011.
[36] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2013.
[37] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray optimization," *Comput. Struct.*, vols. 112–113, pp. 283–294, Dec. 2012.
[38] J. Kennedy and E. Russell, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[39] S. Gülcü, M. Mahi, Ö. K. Baykan, and H. Kodaz, "A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem," *Soft Comput.*, vol. 22, no. 5, pp. 1669–1685, Mar. 2018.
[40] F. Xu, H. Li, C.-M. Pun, H. Hu, Y. Li, Y. Song, and H. Gao, "A new global best guided artificial bee colony algorithm with application in robot path planning," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106037.
[41] W.-T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012.
[42] X. L. Li, "A new intelligent optimization-artificial fish swarm algorithm," M.S. thesis, Zhejiang Univ., Zhejiang, China, 2003, p. 27.

[43] Y. Wang, P. Yao, and Y. Dou, "Monitoring trajectory optimization for unmanned surface vessel in sailboat race," *Optik*, vol. 176, pp. 394–400, Jan. 2019.
[44] C. Dai, Y. Zhu, and W. Chen, "Seeker optimization algorithm," in *Computational Intelligence and Security*. Berlin, Germany: Springer, 2006, pp. 167–176.
[45] S. He, Q. H. Wu, and J. R. Saunders, "A novel group search optimizer inspired by animal behavioural ecology," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 1272–1278.
[46] A. H. Kashan, "League championship algorithm: A new algorithm for numerical function optimization," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.*, 2009, pp. 43–48.
[47] F. Ramezani and S. Lotfi, "Social-based algorithm (SBA)," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2837–2856, May 2013.
[48] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2592–2612, May 2013.
[49] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[50] S. Thabit and A. Mohades, "Multi-robot path planning based on multi-objective particle swarm optimization," *IEEE Access*, vol. 7, pp. 2138–2147, 2019.
[51] S. Zhu, W. Zhu, X. Zhang, and T. Cao, "Path planning of lunar robot based on dynamic adaptive ant colony algorithm and obstacle avoidance," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 3, May 2020, Art. no. 172988141989897.
[52] C. Qu, W. Gai, M. Zhong, and J. Zhang, "A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106099.
[53] X. Jiang and S. Li, "BAS: Beetle antennae search algorithm for optimization problems," *Int. J. Robot. Control*, vol. 1, no. 1, p. 1, Apr. 2018.
[54] Q. Wu, H. Lin, Y. Jin, Z. Chen, S. Li, and D. Chen, "A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability," *Soft Comput.*, vol. 24, no. 3, pp. 2369–2380, Feb. 2020.
[55] J. Wang, H. Chen, Y. Yuan, and Y. Huang, "A novel efficient optimization algorithm for parameter estimation of building thermal dynamic models," *Building Environ.*, vol. 153, pp. 233–240, Apr. 2019.
[56] M. Lin, Q. Li, F. Wang, and D. Chen, "An improved beetle antennae search algorithm and its application on economic load distribution of power system," *IEEE Access*, vol. 8, pp. 99624–99632, 2020.
[57] L. Wang, Q. Wu, J. Liu, S. Li, and R. R. Negenborn, "Ship motion control based on AMBPS-PID algorithm," *IEEE Access*, vol. 7, pp. 183656–183671, 2019.
[58] X. Jiang, Z. Lin, T. He, X. Ma, S. Ma, and S. Li, "Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies," *IEEE Access*, vol. 8, pp. 15459–15471, 2020.
[59] T. Wang, L. Yang, and Q. J. A. Liu, "Beetle swarm optimization Algorithm: Theory and application," vol. abs/1808.00206, pp. 1–19, Aug. 2018.
[60] Y. Masuyama and T. Fukasawa, "Tacking simulation of sailing yachts with new model of aerodynamic force variation during tacking maneuver," *J. Sailboat Technol.*, vol. 2, pp. 1–34, Oct. 2011.
[61] F. X. Wang, *Theory of Ship Wing*. Beijing, China: National Defense Industry Press, 1998.

LIN ZHOU received the Ph.D. degree from the Ocean University of China, in 2011. He is currently an Associate Professor with the College of Engineering, Ocean University of China. His current research interests include intelligent information processing and underwater acoustic positioning technology.

KAI CHEN was born in Jining, Shandong, China, in 1995. He received the B.S. degree in automation from the Shandong University of Science and Technology, Shandong, China, in 2017, where he is currently pursuing the M.S. degree in control engineering. His research interest includes autonomous sailing robots.

HANG DONG was born in Qingdao, Shandong, China, in 1997. He received the B.S. degree in automation from the Ocean University of China, Shandong, China, in 2019, where he is currently pursuing the M.S. degree in control engineering. His research interests include autonomous sailing robots, ocean measurement, and control technology.

SHUKAI CHI received the Ph.D. degree from the Ocean University of China. He is currently a Senior Engineer with the College of Engineering, Ocean University of China. His current research interests include intelligent instrumentation, underwater unmanned systems, and embedded technology.

ZHEN CHEN was born in 1990. He received the B.S. and M.S. degrees from the Ocean University of China, Qingdao, China, in 2012 and 2015, respectively. He is currently working with the College of Engineering, Ocean University of China. His research interests include intelligent control, ocean measurement, and control technology.