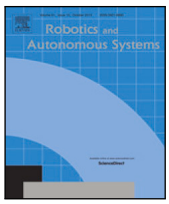
Robotics and Autonomous Systems 177 (2024) 104678

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



A novel hybrid swarm intelligence algorithm for solving TSP and desired-path-based online obstacle avoidance strategy for AUV

Yixiao Zhang, Yue Shen, Qi Wang, Chao Song, Ning Dai, Bo He *

School of Information Science and Engineering, Ocean University of China, Qingdao, Shandong, 266000, China

ARTICLE INFO

Keywords:
Ant colony optimization
Seagull optimization algorithm
Traveling salesman problem
Parameters optimization
Obstacle avoidance

ABSTRACT

Aiming at the problem that Ant Colony Optimization (ACO) is subject primarily to the parameters, we propose a hybrid algorithm SOA-ACO-2Opt to optimize the ACO parameter combination through Seagull Optimization Algorithm (SOA) to strengthen ACO's search capability. To obtain a uniform initial distribution of the ACO parameter combination, we incorporated the Kent Chaos Map (KCM) to randomly initialize the seagull's position, reducing the tendency of SOA to fall into the local optimum. To avoid slow calculation speed and premature convergence of ACO, we improved the adaptive multi-population mechanism to reduce repeated redundant calculations and used the $\epsilon - greedy$ and default strategy, respectively, to update the ants' position. 2Opt is applied to find shorter paths in each iteration. In addition, when AUV navigates on the planned path, it may encounter obstacles. Therefore, this paper proposes an autonomous obstacle avoidance algorithm based on forward-looking sonar to ensure safety during tasks. SOA-ACO-2Opt is verified against twelve different problems extracted from TSPLIB and compared with some state-of-the-art algorithms. Furthermore, sea trials were carried out for several representative marine engineering applications of TSP and obstacle avoidance. Experimental results show that this work can significantly improve AUV's work efficiency and intelligence and protect the AUV's safety.

1. Introduction

As technology advances, Autonomous Underwater Vehicle (AUV), as shown in Fig. 1, is becoming increasingly important in exploring and developing marine resources [1]. As an effective and safe unmanned underwater detection equipment, AUV can perform various tasks such as marine pollutant monitoring [2], marine bioprospecting [3], landform survey [4], and pipeline tracking [5]. One of the primary tasks of AUV is to search for an optimal or sub-optimal path from the initial position in a given set of navigation target points according to different mission environments (Fig. 2) and accomplish the given work. Without considering other constraints, the path planning of AUV can be equivalent to a Traveling Salesman Problem (TSP) in many above applications, which is a paradigmatic NP-hard combinatorial optimization problem where a salesman starting from a home city travels to all the other cities and returns to the home city in the shortest possible path [6–8]. Fig. 2(b) depicts a kind of scenario of Complete Coverage Path Planning (CCPP), a variant of TSP. The goal of CCPP is to find a series of path points so that the robot or sensor can cover the whole area [9]. The combinatorial optimization problem is a category of the optimization problem, and its description is commonly simple and has a strong engineering representation, but the optimization solution is complicated [10].

Collective intelligence refers to the phenomenon that the group could make better decisions than individuals by sharing information [11] by studying the characteristics of individuals and their relationship with the group; the algorithm that formalizes the corresponding mechanism is called "Swarm intelligence (SI)" [12]. In the last decades, various optimization algorithms have emerged one after another, including Ant Colony Algorithm (ACO) [13], Particle Swarm Algorithm (PSO) [14], Artificial Fish Swarm algorithm (AFS) [15] and Bacterial foraging optimization [16], et al.. Essentially, SI algorithms are iteratively based random search algorithms in which heuristic information is shared to perform a search for a certain number of iterations [17]. Such a self-organizing mechanism has been shown to solve many high-complexity problems. Since these problems require solutions that may be sub-optimal but achievable within a reasonable time [18], SI is still gaining prominence as a practical solution.

Many scholars used TSP to test the algorithm performance by default when studying SI-based meta-heuristic algorithms [19,20]. For instance, [7] proposed a variant of SMO (Spider Monkey Optimization)

* Corresponding author.
E-mail addresses: zyx_unique@stu.ouc.edu.cn (Y. Zhang), shenyue@ouc.edu.cn (Y. Shen), wangqi7757@stu.ouc.edu.cn (Q. Wang), songchao@stu.ouc.edu.cn (C. Song), dn5700@stu.ouc.edu.cn (N. Dai), bhe@ouc.edu.cn (B. He).

https://doi.org/10.1016/j.robot.2024.104678
Received 27 December 2022; Received in revised form 26 August 2023; Accepted 27 February 2024
Available online 1 March 2024

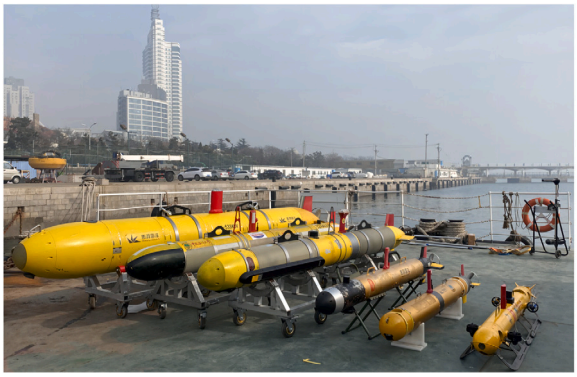


Fig. 1. AUVs independently developed by the Underwater Vehicle Laboratory (UVL) of Ocean University of China.

called discrete SMO (DSMO). In DSMO, each spider monkey denotes a solution, and the optimal one can be obtained by the Swap Sequence (SS) and Swap Operator (SO) interaction. DSMO is relatively novel and has good performance. [21] proposed an improved discrete bio-inspired chicken swarm (DCSO) algorithm. Based on the combination of the exchange operator and exchange sequence strategy, the DCSO for solving continuous space problems is mapped to discrete path optimization. A local search method 2-Opt is applied to shorten the path length of the solutions. The algorithm has an excellent performance in solving 34 kinds of TSP instances. [22] combined a discrete Grey Wolf Optimizer (GWO) and 2-Opt to solve TSP. In addition, there are many classic SI algorithms such as Genetic Algorithm (GA) [23,24], PSO [25–27] and ACO [6,8].

ACO algorithm was first proposed by M. Dorigo in his Ph.D. thesis in 1992. ACO belongs to a growing collection of nature-inspired metaheuristics that can be applied to solve various optimization problems [12]. Due to the slow convergence rate of the basic ant colony algorithm, it is likely to fall into the local suboptimal. So, in order to solve the problems of the basic ant colony algorithm, many researchers conducted studies with the improved algorithm search strategy and pheromone regulation mechanism; for instance, some variants of ACO such as rank-based ant algorithm [28], elite ACO [29], and Best–Worst Ant System [30]. By assigning corresponding levels to ants, these algorithms perform path searches according to priority, thereby reducing the negative impact of bad search results. In addition, there are some studies to improve by adjusting the way ants search, such as greedy-levy ACO algorithm [31], adaptive ACO based on shared information [32], adaptive polymorphic ant colony algorithm [33]. [31] uses the $\varepsilon - greedy$ strategy to construct a pseudo-random scale rule to replace the ant's default state transition formula and integrates levy flight into the ant search, thereby speeding up the convergence speed and promoting the search in a better current direction. [32] proposed that numerous ants performing repeated searches in the iterations will cause much computational redundancy. Therefore, the author divides the ants into two groups based on the optimal solution in each iteration, and the public information about the suboptimal solution reduces the search pressure of the ants, thereby speeding up the search process and achieving good results. The multi-swarm mechanism [33,34] is also a practical variant of the optimized ant colony algorithm.

The parameter value significantly impacts the performance [35]. Excellent parameters can help the ants to fall into local optimum as little as possible and converge quickly. However, adjusting parameters often depends on manual settings and expert experience, so many simulation experiments are needed to correct the values of critical parameters. [36] applied the Taguchi method to determine the parameter values of ACO. Besides, some researchers have combined the ant colony algorithm with other swarm intelligence algorithms [30,37–39] to solve TSP. The main idea is to use other algorithms to search for suitable ant colony algorithm parameters or TSP solutions in advance and inspire the initial ant colony search. [19] proposed a TSP hybrid algorithm that combines ACO and PSO, where PSO utilizes pheromone deposition in ACO as particle weight to improve performance. [25] integrated PSO and ACO and used the PSO algorithm to search the two-dimensional vital parameters in the ACO algorithm.

The experimental results proved its searchability. [38] proposed a novel hybrid algorithm. The author combines symbiotic organisms search (SOS) with ACO. Based on assigning specific ACO parameters, the remaining parameters are adaptively optimized by SOS. The experimental results show that SOS-ACO has an excellent adaptive ability to various values of these parameters and can often find a competitive solution close to the optimum. However, sometimes the optimal solution cannot be found. The above algorithm usually searches for the two-dimensional parameters in the ant colony algorithm. However, ACO also has crucial parameters, such as pheromone intensity and evaporation factor, which significantly affect the algorithm's performance. In addition, ACO's performance needs to be improved.

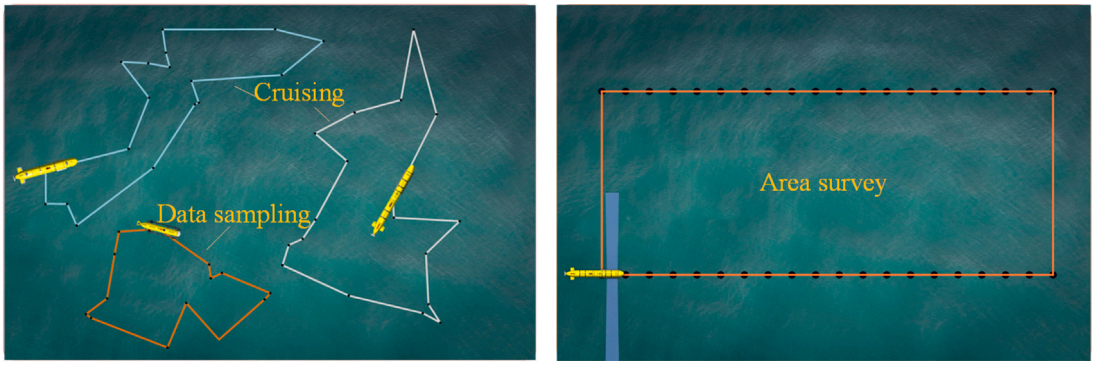
In order to find the optimal high-dimensional parameter combination to maximize the ability of the ACO algorithm, this paper proposes a hybrid SOA-ACO-2Opt algorithm, which can reduce the human resources input in parameter tuning and speed up the algorithm convergence. The Seagull Optimization Algorithm (SOA) is a novel biologically inspired algorithm. It has a simple principle and few parameters and can quickly and accurately find the optimal value in a continuous space [40,41], suitable for parameter tuning. In SOA-ACO-2Opt, the position of SOA is used to represent the critical parameter combination of ACO. With each iteration of ACO, the seagull moves in the best direction to optimize the ACO parameters. ACO uses optimized parameters to search for optimal or suboptimal solutions.

Robot obstacle avoidance is one of the most essential and critical functions. It aims to ensure that robots do not collide during their actions. The core of robot obstacle avoidance technology includes the selection of sensors and the selection of planning algorithms. The first step of automatic obstacle avoidance is to enable the robot to perceive the parameters of the surrounding environment, such as the size, shape, and position of obstacles. Various sensors are currently used for obstacle avoidance, and their characteristics and scope of application are also different. According to different principles, they can be divided into laser sensors [42], visual sensors [43], and acoustic sensors. Sensors commonly used for AUV obstacle avoidance include forward-looking sonar, camera, etc.

AUV needs to conduct online processing and analysis of sensor data and use this as a guide to autonomously mark the mutual position of obstacles and itself [43] to plan a collision-free path from the starting point to the target point precise navigation [1]. [44] uses the obstacle avoidance algorithm based on deep reinforcement learning (DRL) based on the obstacle detection results obtained from sonar images to plan a reasonable obstacle avoidance path for the AUV. However, algorithms based on reinforcement learning require extensive training and are likely only suitable for specific tasks.

The main innovations and contributions of this paper are as follows:

(1) It is the first time to combine the seagull optimization algorithm (SOA) with the ant colony optimization (ACO) algorithm solve TSP, especially in the AUV field. The SOA is simple in principle, easy to implement, with few adjustable parameters and strong searchability, and it has been widely utilized in many engineering applications. Using SOA to find the best parameter combination of ACO can significantly enhance the performance of ACO.
(2) The Kent Chaos Map (KCM) is used to initialize the seagulls' position, which can avoid SOA search to a certain extent stuck in a local optimum. It can generate a more uniform and reasonable initialization distribution based on ACO parameter value combinations in the search space.



(a) Applications that can be seen as TSP (b) Complete coverage path planning (CCPP)

Fig. 2. Typical applications of AUV in engineering.

(3) This paper adopted and improved the multi-population mechanism and divided the ants into three groups to avoid the ACO falling into a local optimum. As the number of iterations changes, the number of ants in each group is adaptively adjusted. Each group of ants has a different task. Depending on the division of labor, ants update their positions based on different state transition strategies ($\epsilon - greedy$ strategy and default strategy), which can speed up the search process, significantly reduce the probability of falling into local optimum, and reduce a large number of redundant calculations.
(4) In the SOA-ACO hybrid algorithm, after one iteration is completed, the 2Opt algorithm is used to shorten the path length further.
(5) When the AUV sails to perform tasks according to the path planned by the SOA-ACO-2Opt hybrid algorithm, it may encounter obstacles. This paper proposes an obstacle avoidance algorithm based on the initial desired path and forward-looking sonar (FLS), which can ensure the AUV effectively avoids obstacles and return to the initially expected path as soon as possible to ensure the smooth execution of the task.

The experimental results show that SOA-ACO-2Opt can search for the best solution or an approximation quite close to the best solution. Compared with the primary ACO and ACO with a local optimization strategy (ACO-2Opt), SOA-ACO-2Opt finds a better solution and improves the convergence speed. Based on the combination of parameters searched by the proposed algorithm, the performance of ACO is also improved. Finally, the performance of the hybrid algorithm is compared by comparing some algorithms in the literature. The experimental results show that SOA-ACO-2Opt can obtain satisfactory solutions in different instances and find suitable parameters. The report stated that SOA-ACO-2Opt has an excellent adaptive ability to solve the TSP problem.

The remainder of this paper is organized as follows. Section 2 briefly introduces ACO and $\epsilon - greedy$ strategy, SOA, and 2-Opt. Section 3 describes the details of SOA-ACO-2Opt and the obstacle avoidance strategy. In Section 4, experiments are executed to validate the hybrid SOA-ACO-2Opt and the obstacle avoidance strategy. In order to verify that the algorithm proposed in this paper can be applied to engineering applications, the AUV is used to conduct sea test verification, and the corresponding results and analysis are described in detail in Section 5. In Section 6, conclusions are drawn.

2. Background

2.1. ACO and $\epsilon - greedy$ strategy

2.1.1. ACO for TSP

The process of basic ACO for solving the TSP is as follows:

(1) **Initialization**. The m ants are randomly placed on n cities. Each edge of the city has an initial pheromone $\tau_{ij}(0)$, and the first element of each ant's tabu list is set to its initial city.
(2) **Each ant decides which city to go to next.** In the path selection stage, the probability of ants determining which city to go to depends on the distance between cities and pheromone intensity according to the probability function. The probability function is as follows,

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{t \in allowed_t}[\tau_{it}(t)]^{\alpha}[\eta_{it}]^{\beta}} & j \in allowed_k \\ 0 & otherwise \end{cases} \tag{1}$$

where $\tau_{ij}(0)$ represents pheromone intensity on edge (i, j), and i is the starting city, j is the arriving city. η_{ij} represents the distance factor between city i and j, which usually takes the value of $1/d_{ij}$, d_{ij} represents the distance between city i and j, α refers to the role of pheromone in the selection probability, and β represents the role of path length in the selection probability. $allowed_k$ represents the list of all candidate cities that ant k can visit at the current moment.
(3) After n iterations, the tabu list of all ants has been filled. At this time, **calculate the length of the path passed by each ant and save the shortest path**. The path is recorded, and the pheromone on the path is changed according to Eqs. (2) and (3). Determine whether it reaches the maximum number of iterations. If so, go to (4); otherwise, (2).

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \tag{2}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

where $\Delta\tau_{ij}$ denotes the sum of accumulated new pheromones on a certain edge, ρ denotes the level of pheromone dissipation, and $0 \le \rho < 1$. $\Delta\tau_{ij}^{k}$ represents the number of pheromones left by the ant k between t and $t+n$ on this edge. If the ant k between t and $t+n$ uses this edge, then,

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k} \tag{4}$$

where Q represents the total amount of information released by the ant moves in one iteration, and L_k is the length of the path of the kth ant in this iteration. Otherwise, the value is 0.
(4) **Output the best path and its path length**.

2.1.2. $\epsilon - greedy$ strategy

ACO is a reinforcement learning algorithm [45], and $\epsilon - greedy$ strategy is a vital exploration strategy in reinforcement learning [46].

The definition of the $\varepsilon - greedy$ strategy is shown in Eq. (5) [31].

$$p = \begin{cases} \arg\max \left\{ (\tau_{ij})^{\alpha} (\eta_{ij})^{\beta} \right\} & \text{if } p \le \varepsilon \\ p_{ij}^{k} & \text{otherwise} \end{cases} \tag{5}$$

In this algorithm, in each step of constructing a feasible loop, the ant k on city i selects the city j to move by applying the rules given in Eqs. (1) and (5). The $\varepsilon - greedy$ strategy is also utilized as a pseudo-random mechanism [32] in an improved ACO named ant colony system (ACS). In the case of $1-\varepsilon$, the original $\varepsilon - greedy$ strategy uses uniform distribution to select a candidate. This strategy tends to choose short paths and a large pheromone concentration side as motion direction, which can accelerate the search process of ants.

2.2. Basic SOA

The SOA is a novel bionic algorithm [40], it runs as following steps:

(1) In order to avoid collision between seagulls, parameter A is introduced to limit seagulls in a safe search space when calculating the new position of seagulls.

$$C_s(t) = A * P_s(t) \tag{6}$$

$$A = f_c - \left(t * \left(f_c / Max_{iteration}\right)\right) \tag{7}$$

where $C_s(t)$ represents the seagull's new position, which will not collide with other seagulls, $P_s(t)$ represents the location of the seagull, and t represents the number of iterations. A indicates that the range of motion of seagulls is limited to a specific space; f_c is the parameter of the control variable A, its value is linear decreasing, and the decreasing interval is $[0, 2]$.

(2) **Calculate the best direction**: under the premise of ensuring that there is no collision between each other, the seagulls will change the direction of motion to keep close to the best position.

$$M_s(t) = B * \left(P_{bs}(t) - P_s(t)\right) \tag{8}$$

$$B = 2 * A^2 * r_d \tag{9}$$

where $M_s(t)$ represents the best direction; B is a random number, ensuring the balance between global and local search; r_d is a control parameter with a range between $[0, 1]$.

(3) **Calculation of the latest position**: the current position of seagulls and other seagulls will not collide, then begins to move to the best position near and arrives at new positions, which are the new positions of gulls.

$$D_s(t) = |C_s(t) + M_s(t)| \tag{10}$$

(4) **Close to the best position**: When the new position of the gull does not overlap with the other gulls, it begins to move in the direction of the best position to reach the new position.

(5) **Local attack**: Seagulls wait for the best time to attack prey by changing the angle and speed of attack during migration. They use their wings and their weight to maintain height during flight. When attacking prey, they attack prey in the air in a spiral flight state. The equations of motion behavior in the x, y, and z planes are described as follows:

$$x = r * \cos\theta \tag{11}$$

$$y = r * \sin\theta \tag{12}$$

$$z = r * \theta \tag{13}$$

$$r = u * e^{\theta v} \tag{14}$$

where r is the radius of each spiral; θ is a random angle value in the $[0, 2\pi]$ range; u and v are related constants of spiral shape; e is the bottom of the natural logarithm.

2.3. 2-Opt algorithm

The main ideas are as follows :

A traveler must start from A city through B, C, D, E, F, G, H these cities and finally return to A city. The objective function is the shortest distance.

First, we can choose a feasible solution s_1: $A \to B \to C \to D \to E \to F \to G \to H \to A$, and assume the best solution $S_{\min} = s_1$. Then, the 2-Opt algorithm is used to solve the problem. Two points i and k, are randomly selected, and the path before i is added to the new path without change. The path between i and k is flipped, numbered, and then added to the new path. The path after k is added to the new path without change.

If $i = 4$, $k = 7$, then the new path s_2: $A \to B \to C \to (G \to F \to E \to D) \to H \to A$, s_2 is a new feasible solution. The objective function value can be obtained by substituting the feasible solution with the objective function. Compared with the objective function value of $S_{\min}$, the feasible solution with a smaller objective function value of both is taken as $S_{\min}$ until the function value smaller than $S_{\min}$ is not found. So far, the TSP problem has been solved by the 2-Opt algorithm.

3. Methodology

3.1. Hybrid SOA-ACO-2Opt algorithm

SOA has efficient optimization ability, fewer parameters, and simple principles and can solve challenging large-scale constrained problems [47,48]. In this paper, we employed the improved SOA algorithm to search the critical parameters of the enhanced ACO during the ant colony search. Finding the most suitable parameter combination in ant search is significant for finding the best path.

Due to the coupling relationship between ACO parameters, finding accurate parameter values for a given problem often depends on expert experience. In ACO, the parameter values are usually fixed. Hence, if the parameter setting is reasonable, ACO is prone to fall into the local minimum, and the final search effect could be better. In this paper, the optimization process is defined as solving TSP by ACO. After each iteration, SOA calculates the fitness according to the path length and transfers to a new position until it finds an best position so ACO shows the best performance. The optimization process is expressed as $F(*)$:

$$\left(\alpha_{new}, \beta_{new}, \rho_{new}, Q_{new}\right) = F\left(\alpha_{pre}, \beta_{pre}, \rho_{pre}, Q_{pre}\right) \tag{15}$$

Since solving TSP based on ACO is an NP problem, finding the best parameter combination of ACO in the iterative process of ACO based on SOA is also an NP problem.

The flowchart of the hybrid SOA-ACO-2Opt algorithm is shown in Fig. 3, and the algorithm details are shown in Algorithm 1. In SOA-ACO-2Opt, the position of a seagull is expressed as P_s, $P_s \leftarrow (\alpha, \beta, Q, \rho)$. Each dimension of the position of the seagull represents the corresponding parameters. and the fitness function of P_s is

$$Fitness\left(P_s\right) = \frac{L_{P_s}}{L_{\min}} \tag{16}$$

where L_{P_s} represents the parameter represented by P_s in the current iteration, the length of the shortest path that the ants can find, and $L_{\min}$ represents the globally best path length.

The initialization of the fundamental SOA is to randomly generate the position of the gull individual in the upper and lower bounds. Such initialization may make the position of the gull in the space unevenly distributed, leading to premature convergence of the algorithm and falling into local best solutions. In order to solve this problem, this paper uses the initialization form based on Kent Chaotic Mapping (KCM) [49] in Section 3.1.1.

In the search process of ACO, the ants easily fall into local optimum in the iterations, which will cause plenty of redundant calculations.

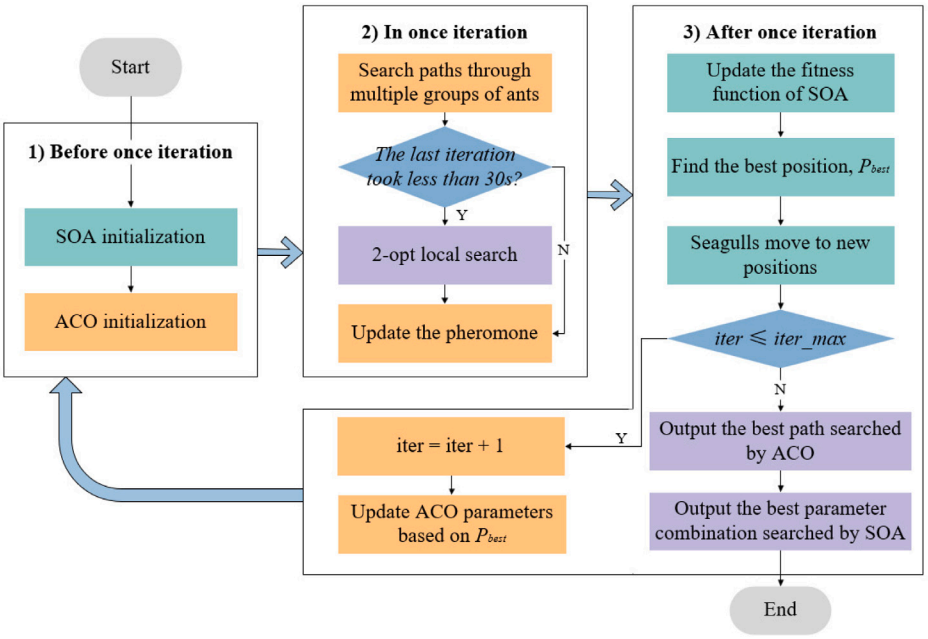


Fig. 3. The flowchart of the proposed algorithm.

Algorithm 1 The proposed hybrid SOA-ACO-2Opt algorithm.

Input: 1) Ants' number m, 2) seagulls' number sg_num, 3) maximum iterations $iter\_max$, 4) α's value range $Range\_\alpha$, 5) $Range\_\beta$, 6) $Range\_\rho$, 6) $Range\_Q$, 7) the collection of n nodes to be visited $Cities$, 8) SOA's parameters $\{f_c, u, v\}$.

Output: The searched best path R_op, the best parameter combination P_{best};

SOA initialization: $sg\_num = m$, and then initialize the seagulls' position according to Kent Chaotic Mapping in a given range, $Ps \leftarrow (\alpha, \beta, \rho, Q)_{m\times n}$;
ACO initialization: Initialize the ACO and divide m ants into three groups (m_1, m_2 and m_3) according to the initial criteria;
while $iter \leq iter\_max$ **do**
    Generate the start city of each ant randomly.
    The first group of ants searches the path according to the $\varepsilon - greedy$ strategy (Eq. (5)).
    Find the current best solution r_op and second best solution r_subop, and obtain public information CI.
    The second group of ants searches the paths according to the $\varepsilon - greedy$ strategy (Eq. (5)) and CI.
    Update r_op, r_subop and CI.
    The third group of ants searches the paths according to CI and the default strategy (Eq. (1)).
    Update r_op, r_subop and CI.
    if Last iteration took less than 30s **then**
        Utilize 2-opt to optimize the paths.
    end if
    Update the pheromone of ACO according to Eq. (2).
    Update the fitness function of SOA according to Eq. (16)
    Find the best position of SOA, P_{best}.
    The seagulls move to new positions according to Eq. (6) to (10).
    $iter = iter + 1$
    Set the critical parameters for each ant according to relative seagull's position.
end while
Return R_op and P_{best};

This paper adopted an improved multi-populations mechanism in Section 3.1.2 to enhance the performance of ACO. In addition, when an SOA-ACO cycle takes more than the 30 s, 2Opt is no longer used, which can save a lot of time load.

According to our algorithm design, in one iteration, the population size of ACO is m, and correspondingly, the number of parameter combinations used by each ant is also m; that is, the population size of SOA is also m. We regard the ith ant and the corresponding ith seagull as a mixed population creature (ant-gull); that is, in each iteration, the scale of these ant-gulls is constant, and the fitness is re-adapted and updated according to the search path length corresponding location. The number of evaluations is the number m of ant-gulls multiplied by the number of iterations, which is the same as the number of evaluations of ACO and ACO-2Opt. And because the best parameters found by SOA-ACO-2Opt can make it easier to find a shorter path, the shortened path is very important for the time and energy savings brought by the actual navigation of the AUV, so it is very worthwhile to integrating SOA into ACO.

3.1.1. Improved SOA based on Kent chaotic mapping initialization

The improved SOA is designed to explore the best combination of ACO parameters in the process of ant colony search.

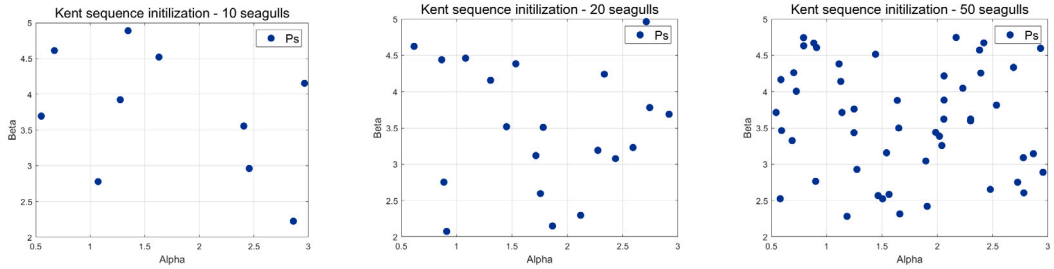
A chaotic mapping is a nonlinear mapping that can generate a random sequence. It is sensitive to initial values, ensuring that the encoder can generate an unrelated encoding sequence. Due to the randomness and ergodicity, the KCM can be used to initialize the swarm intelligence algorithm population. The mathematical model of the KCM is as follows:

$$x_{n+1} = \begin{cases} \frac{x_n}{a} & 0 < x_n \leq a \\ \frac{1-x_n}{1-a} & a < x_n < 1 \end{cases} \tag{17}$$

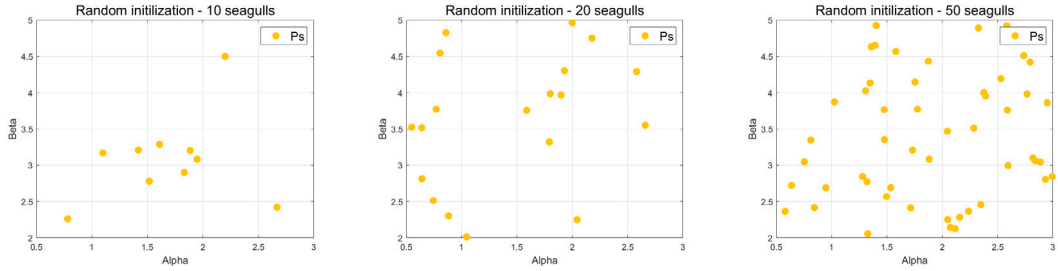
where, a is a variable value, x is the initial value of the $x(0)$. In this paper, $a = 0.5$. The model initialized by the chaotic mapping is given by Eq. (18).

$$x = x_{\min} + Chaos * \left(x_{\max} - x_{\min}\right) \tag{18}$$

where $x_{\min}$ and $x_{\max}$ represent the lower limit and upper limit of independent variables, respectively, $Chaos$ is a chaotic factor generated by the Kent mapping function.



(a) Initialization of 10 seagulls based on KCM. (b) Initialization of 20 seagulls based on KCM. (c) Initialization of 50 seagulls based on KCM.



(d) Initialization of 10 seagulls based on the random method. (e) Initialization of 20 seagulls based on the random method. (f) Initialization of 50 seagulls based on the random method.

Fig. 4. Position initialization comparison between KCM and random method.

The principle is to use the sequence between $[0, 1]$ generated by the chaotic mapping and then initialize the gull population according to the chaotic factor, which can generate a more reasonable initial distribution of gulls to avoid convergence prematurely.

Fig. 4 shows the 2-dimension position (α, β) distribution maps based on random and KCM initialization with different numbers of seagulls. The value range of α is [0.5, 3], and the value range of β is [2, 5]. It can be seen that the KCM initialization method generated a more uniform distribution than the random initialization.

3.1.2. Improved ACO with multi-population mechanism

When ants use pheromones to find best paths, some paths must be traversed repeatedly. The repeatedly found path is called the Common path, and its definition is shown in Fig. 5. When finding the best path, the second best and best paths tend to be the same in most path segments. Sometimes the search for an optimization algorithm requires only minor tweaks to improve results. Therefore, allocating all ants for new explorations in an iterative search is unnecessary. Some ants can be appropriately mobilized to analyze the current best and second best paths to obtain the common information CI. After the analysis, the search is performed according to different transfer strategies, which can simplify the ants' work and improve the search's efficiency.

This paper divides the ants into three groups. In each iteration, the first group of ants searches for the best path according to the $\epsilon - greedy$ strategy (as shown in Eq. (5)). The second group finds new paths regarding the shared information of the current best and the second best solution based on the $\epsilon - greedy$ strategy. The third group will search for new paths based on shared information and default probability function (see as in Eq. (1)).

As shown in Fig. 6, there are 20 ordinary ants divided into three groups.

(1) The first group of ants is named *workers*. The task is to search for paths as fast as possible with less concern about the length of paths. The *workers* use the strategy of Eq. (5) to find solutions and then calculate and record the Common paths between the best and best solutions according to their search results.
(2) The second group is named *managers*. The task is to search for paths fast and consider the length of paths. Based on the public information provided by *workers*, further exploration can be carried out by the *managers*. In addition, the *managers* use the formula Eq. (5) for state transition to improve the calculation speed.
(3) The third group is named *bosses*. The most significant task is to search for paths that break the limit of the experiences of other ants based on the common paths provided by the *managers*. The *bosses* will complete the path search further according to the default transfer strategy in Eq. (1) - jump out of the restriction of the local greedy idea.

The number of three groups of ants varies with the iterations. The grouping method is shown in the Eq. (19)–(21).

$$m_2 = \begin{cases} [m/5] & iter \le 6n \\ [m/3] & 6n < iter \le 10n \\ [m/2] & iter > 10n \end{cases} \tag{19}$$

$$m_3 = \begin{cases} [m/10] & iter \le 6n \\ [m/8] & 6n < iter \le 10n \\ [m/6] & iter > 10n \end{cases} \tag{20}$$

$$m_1 = m - m_2 - m_3 \tag{21}$$

where m represents the total number of ants, $[*]$ denotes represents the rounding of elements, $m_i\,(i = 1, 2, 3)$ denotes the number of ants in group i, n denotes the number of cities, and $iter$ denotes the number of current iterations.

The calculation speed and search ability of the *managers* and *bosses* is very considerable, especially in the late iterations. Using the common path algorithm can get a better search direction, and the solution will gradually approach the actual best solution.

3.2. Obstacle avoidance based on forward-looking sonar

When the AUV navigates on the desired path, based on Multibeam forward-looking sonar (Multibeamforward-looking sonar, MFLS) continuously detects obstacles within a specific range of AUV heading. MFLS sends multi-channel sound wave signals at a time, propagates

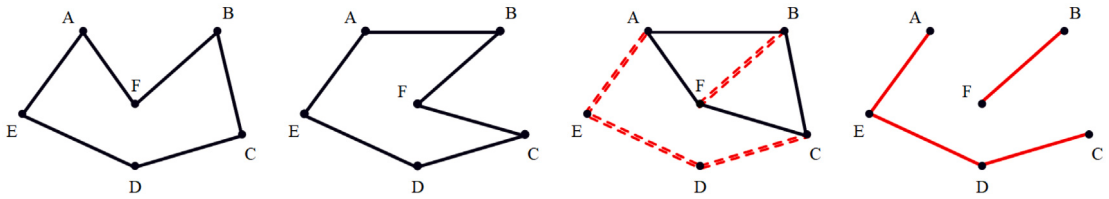


Fig. 5. Find common paths between best and second best solutions.

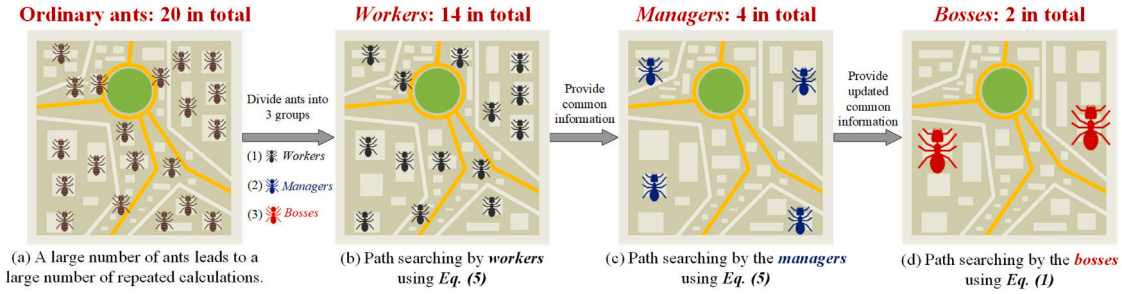


Fig. 6. The multi-population mechanism search process.

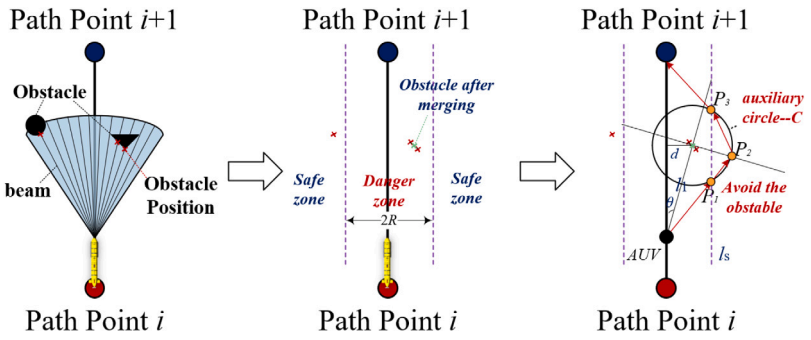


Fig. 7. The AUV avoids obstacles while navigating on the desired path.

through the water environment, and generates echo signals when encountering obstacles. Therefore, MFLS can provide AUV with the position, distance, and shape information of objects in the environment. By filtering and integrating the environmental information collected by sonar, AUV can obtain information such as the relative orientation, distance, and speed of obstacles in the working environment and itself.

This study proposes the following obstacle avoidance strategy (see Fig. 7) based on comprehensive obstacle position, AUV position, heading, and target point position, considering energy consumption and safety factors.

When the AUV navigates between the path points i and $i+1$, the FLS finds circular and triangular obstacles and determines the position information of the obstacles (red cross) according to the detection results of the beam. If multiple consecutive beams detect obstacles, the merge operation outputs the merged coordinates of continuous obstacles (as shown by the green cross). The $2R$ range at both ends of the AUV path is set as a dangerous zone. The range beyond $2R$ is the safe area. When the obstacle is in the danger zone, that is, when the distance between the obstacle and the expected path of the AUV is d, AUV needs to avoid obstacles:

(1) Let the line connecting the AUV and the nearest obstacle be l_1, l_2 is perpendicular to l_1, and the angle between l_1 and l_2 is θ. Take the obstacle's position as the center, R_{ac} as the radius to calculate the obstacle avoidance auxiliary circle, C. R_{ac} is adjusted according to the possible size of obstacles.

(2) Calculate the intersection points of C and the boundary line l_s of the danger zone, and record them as P_1 and P_3. Calculate the intersection point of C and l_2, denoted as P_2. Then $P = \{P_1, P_2, P_3\}$ is the set of obstacle avoidance path points.

(3) Merge the waypoints: Due to the limitation of the accuracy of the position sensor configured by the AUV and the influence of environmental loads (waves, currents), sometimes the AUV can only reach each waypoint closely but not accurately. In the actual work of AUV, we designed a circle for judging whether the AUV has reached the waypoint. As shown in Fig. 8, the decision circle takes the path point as the center and a specific constant as the radius. The radius of the judgment point is r, and the specific value needs to be determined according to the size and motion characteristics of the AUV. When the AUV enters the judgment circle of a specific path point, it is considered that the AUV has reached the target point. Based on the judgment circle, the positional relationship between the obstacle avoidance path point P_j $(j = 1, 2, 3)$ and the path point i is determined, and the points falling in the same judgment circle are merged into one.

4. Simulation experiments and analysis

4.1. Experimental environment

In this section, we conducted a detailed experimental evaluation of the proposed hybrid SOA-ACO-2Opt algorithm. We selected 12 TSP

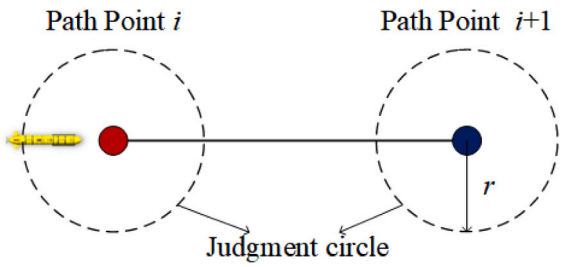


Fig. 8. The judgment circle for AUV. If the AUV sails to the decision circle of the current waypoint, it means the AUV has arrived at this point.

Table 1
Basic information of the TSP instances for resolving.

TSP	n	Number of edges($(n(n-1)/2)$)	BKS
Eil51	51	1275	426
Berlin52	52	1326	7542
Eil76	76	2850	538
Rat99	99	4851	1224
Kroa100	100	4950	21,282
Lin105	105	5460	14,379
Ch150	150	11,175	6528
Pr226	226	25,425	80,369
Gil262	262	34,191	2378
Pr264	264	34,716	49,315
u1432	1432	2,049,192	152,970
u2319	2319	5,375,442	234,256

instances of different sizes in TSPLIB [50] for validation. The experiments were performed on the same computer platform and simulation environment:

- Simulation Platform: Personal Computer
- Processor: Intel(R) Core(TM) i5-10210U CPU @ 1.60 GHz 2.11 GHz
- RAM: 16.0GB
- System type: 64-bit operating system, x64-based processor
- Software: MATLAB R2019b

The basic information of these TSP instances is shown in Table 1. The experimental results of SOA-ACO-2Opt are presented in Section 4.2, in which we compare the search results for 2D parameter combinations (α, β) and 4D parameter combinations (α, β, ρ, Q). In Section 4.3, the experimental results of SOA-ACO-2Opt(α, β, ρ, Q) are compared with those of some other algorithms, such as ACO-2Opt, ACO, and others in additional literature. Furthermore, we conducted ACO re-experiment with the best parameter combination searched by the hybrid algorithm to verify the parameters' effectiveness. In Section 4.4, we programmed and simulated the obstacle avoidance strategy based on the MOOS-IvP environment carried by our self-developed AUV.

4.2. Experimental results of SOA-ACO-2Opt

Table 1 summarizes the results for twelve different instances. To make the test more adequate, SA_2 and SA_4 were run 50 times for each instance to record the experimental results. BKS represents the length of the known best path for the current instance. The improved SOA searches for two-dimensional parameter combinations (α, β) and four-dimensional combinations (α, β, ρ, Q), respectively. In order to convenience, SOA-ACO-2Opt(α, β) is referred to as SA_2, and SOA-ACO-2Opt(α, β, ρ, Q) is referred to as SA_4. The parameters setting are shown in Table 2.

Among them, α, β, m, ρ, and Q are related parameters of ACO, and we set these parameters in this paper referring to other literature [25, 38]. sg_num, f_c, u, v are related parameters of SOA. The number of seagulls sg_num is set to be the same as m so that the position of each seagull corresponds to an ant; as shown in Eq. (7), f_c controls the movement behavior of seagulls in a given search space by controlling the frequency of A. Generally, set $f_c = 2$. As it iterates, the value decays linearly from 2 to 0, which means that the seagulls are constantly shrinking their search range. u and v are related parameters of the seagull's spiral shape during the attack process and generally take a constant value of 1. However, we found in simulation experiments that setting u and v to 0.5 can speed up the local development of SOA.

The experimental results are shown in Table 3. In the comparison, all fractional parts of BKS are ignored. The $Best$, $Worst$, and $Average$ in the solutions for each TSP instance in the table represent the best, worst, and average path lengths, respectively, and SD represents the standard deviation calculated based on twelve solutions of each instance. $Error(\%)$ is the relative error of $Average$ with respect to BKS. See Eq. (22) for the calculation method of $Error(\%)$, which can evaluate the algorithm's performance. $PE(\%)$ is the relative error between $Average$ and $Best$. $PE(\%)$ evaluates to Eq. (23).

$$Error(\%) = \frac{Average - BKS}{BKS} \times 100\% \tag{22}$$

$$PE(\%) = \frac{Avergae - Best}{Best} \times 100\% \tag{23}$$

SA_2 found BKS for Eil51, Berlin52, Eil76, KroA100, Lin105, Ch150, and Pr264. For Rat99, Pr226, Gil262, $Best$ based on SA_2 is also very close to BKS. For u1432 and u2319, SOA-ACO-2Opt did not find the optimal path, and the average error between the average result of SA_2 and SA_4 and BKS was 22.83%. For the instances whose scale is less than 1000, the $Error(\%)$ of SA_2 does not exceed 2%. It shows that SOA-ACO-2Opt can find high-quality solutions in most experiments. Compared with SA_2, SA_4 also found BKS for Rat99, Pr226, Gil262. In instances without BKS is found, the resulting $Best$ based on SA_4 is also closer to BKS. Also, all instances have $PE(\%)$ within 1%, which means that the solutions for each instance are slightly different, and SOA-ACO-2Opt has steady performance. Through comparing SA_2 with SA_4, it can be seen that,

(1) The SA_4 has generated the closest results to the optimal solution;
(2) From the perspective of $Best$, $Worst$ and $Average$, the path length of solutions obtained by SA_4 is superior, which shows the critical role of suitable parameters for ant colony performance;
(3) Even in individual instances, such as u1432, SA_4 produced a larger $Worst$, but overall, it confirms that a higher appropriate parameter combination of dimensions can bring more significant and effective help to path search.

The results illustrated that according to the more suitable parameters, our algorithm performs better, and SOA-ACO-2Opt has strong robustness for TSP. The best solutions of several TSP instances computed by SOA-ACO-2Opt are shown in Fig. 9.

4.3. Comparison with some other algorithms

Under the same environments, the experiments were carried out for the same twelve TSP instances to compare the performance of SA_4, ACO, and ACO with the 2Opt local optimization strategy (ACO-2Opt). In addition, SA_4 was compared with some hybrid algorithms in the previous literature, such as PSO-ACO-3Opt [25], PACO-3Opt [39], SOS-ACO [38] and DSMO [7]. Each instance was run ten times with the above different algorithms.

4.3.1. Comparison with ACO-2Opt and ACO

Refer to [51], in basic ACO and ACO-2Opt, alpha is set to 1, beta is 5, and the rest of the involved parameter settings are shown in Table 4. The experimental results of the comparison between SA_4 and these two algorithms are shown in Table 5. It can be seen that SA_4 is significantly better than ACO and ACO-2Opt, especially the large-scale instances.

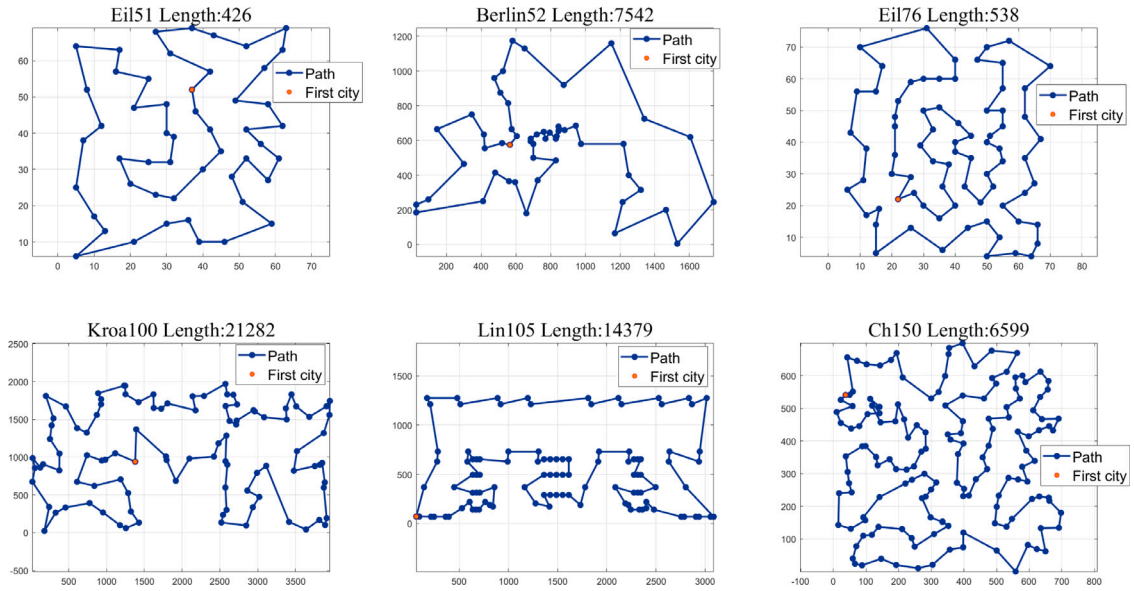


Fig. 9. The best solutions computed by SOA-ACO-2Opt(α, β).

Table 2
The assignments of parameters(SOA-ACO-2Opt).

Parameter	$Range_a$	$Range_\beta$	$iter\_max$	m	ρ	Q	ε	sg_num	f_c	u	v	a
SA_2	[0.5, 3]	[2, 5]	1000	20	0.2	300	0.9	20	2	0.5	0.5	0.5
SA_4	[0.5, 3]	[2, 5]	1000	20	[0.1, 0.3]	[200, 400]	0.9	20	2	0.5	0.5	0.5

Table 3
Comparison of SA_2 and SA_4 after fifty runs.

TSP	BKS	SA_2						SA_4					
		Best	Average	Worst	SD	Error(%)	PE(%)	Best	Average	Worst	SD	Error (%)	PE (%)
Eil51	426	426	426.9	429	0.82	0.22	0.22	426	426.7	429	0.90	0.16	0.16
Berlin52	7542	7542	7543.9	7547	2.07	0.02	0.02	7542	7543.6	7547	2.01	0.02	0.02
Eil76	538	538	540.6	544	2.55	0.48	0.48	538	540.3	544	2.38	0.43	0.43
Rat99	1211	1212	1224.1	1230	4.83	1.08	1.00	1211	1222.3	1230	6.02	0.85	0.85
KroA100	21,282	21,282	21,316.5	21,408	36.81	0.16	0.16	21,282	21,311.7	21,381	45.37	0.14	0.14
Lin105	14,379	14,379	14,480.9	14,706	132.24	0.71	0.71	14,379	14,380.5	14,383	1.94	0.01	0.01
Ch150	6528	6528	6652.5	6763	82.54	1.91	1.91	6528	6588.2	6763	89.32	0.92	0.92
Pr226	80,369	80,382	81,022.2	81,245	327.39	0.81	0.80	80,369	80,775.8	81,245	403.88	0.51	0.51
Gil262	2378	2406	2418.7	2432	7.45	1.71	0.53	2378	2390.5	2406	8.49	0.53	0.53
Pr264	49,135	49,135	49 219.7	49,310	49.51	0.17	0.17	49,135	49,212.8	49,310	48.89	0.16	0.16
u1432	152,970	187,689	187,889.3	188,045	94.22	22.83	0.10	183,707	186,144.2	188,776	1528.52	21.69	1.28
u2319	234,256	285,857	290,288.3	295,170	2578.38	23.92	1.48	283,607	286,544.0	289,602	1572.35	22.32	1.02

Table 4
The assignments of parameters ACO-2Opt and ACO.

Parameter	α	β	$iter\_max$	m	ρ	Q
ACO-2Opt	1	5	1000	20	0.2	300
ACO	1	5	1000	20	0.2	300

Table 6 shows the best parameter values that the SA_4 algorithm searches for each instance. Based on these parameters, we re-tested ACO using the best parameters for the corresponding instance, running ten times per instance. Other parameter settings are unchanged except for the parameters in Table 4. The experimental results are shown in Table 7. It demonstrates that the parameters found by our algorithm have an excellent effect on ACO. Based on the appropriate parameter combination, the performance of ACO has also been improved to some extent, especially for the large-scale instances u1432 and u2319.

The convergence curves of ACO, ACO-2Opt and SOA-ACO-2Opt for the six TSP instances in Fig. 9 are shown in Fig. 10. According to Fig. 10, we can draw the following conclusions:

(1) Neither ACO nor ACO-2Opt can find the optimal solution for the six TSP instances;
(2) SOA-ACO-2Opt found a significantly better solution than ACO and ACO-2opt in each instance;
(3) SOA-ACO-2Opt can effectively jump out of the local optimal and find a better solution by adjusting the ACO parameters.

4.3.2. Comparison with other algorithms in the literature

PSO-ACO-3Opt and PACO-3Opt contain local optimization strategies, and PSO-ACO-3Opt and SOS-ACO contain ACO parameter optimization. In PACO-3Opt, the solution is first found by ACO, which is then improved using 3-Opt. In PSO-ACO-3OPT, PSO is used to optimize the parameters of ACO, and 3-Opt is used to improve the solution. In SOS-ACO, SOS is used to optimize ACO's key parameters $alpha$ and β. In DSMO, each spider monkey represents a TSP solution. The monkeys interact to obtain the best TSP solution through the operation based on the exchange sequence (SS) and exchange operator (SO). The SOA-ACO-2OPT proposed in this paper uses improved SOA to find the best

Table 5
Comparison between SOA-ACO-2Opt(α, β, ρ, Q), ACO-2opt and ACO after ten runs.

TSP	BKS	SA_4			ACO-2Opt			ACO		
		Best	Average	Error (%)	Best	Average	Error(%)	Best	Average	Error(%)
Eil51	426	426	426.2	0.05	433	435.3	2.18	453	460.6	8.12
Berlin52	7542	7542	7543	0.01	7547	7594.7	0.7	7786	7958.9	5.53
Eil76	538	538	538.4	0.07	541	555.3	3.22	571	587.6	9.22
Rat99	1211	1212	1222.4	0.94	1224	1249.6	3.19	1317	1354	11.82
KroA100	21,282	21,282	21,298.2	0.08	21,404	21,460.5	0.84	23,968	24,503.1	15.14
Lin105	14,379	14,379	14,381.5	0.02	14,383	14,581	1.41	15,422	15,690.3	9.12
Ch150	6528	6528	6612.5	1.29	6763	6817.5	4.43	7015	8376.4	28.31
Pr226	80,369	80,382	80,617.4	0.31	80,382	83,569.1	3.98	93,096	93,331.8	16.13
Gil262	2378	2378	2383.1	0.21	2521	2543.2	6.95	2777	2821.9	18.67
Pr264	49,135	49,135	49,182.7	0.1	49,432	51,032.7	3.86	53,560	54,521.1	10.96
u1432	152,970	183,707	186,828.6	22.13	245,483	253,726	65.87	855,995	868,841.7	467.98
u2319	234,256	283,607	287,891.8	22.90	377,760	384,482	64.13	415,702	419,659	79.15

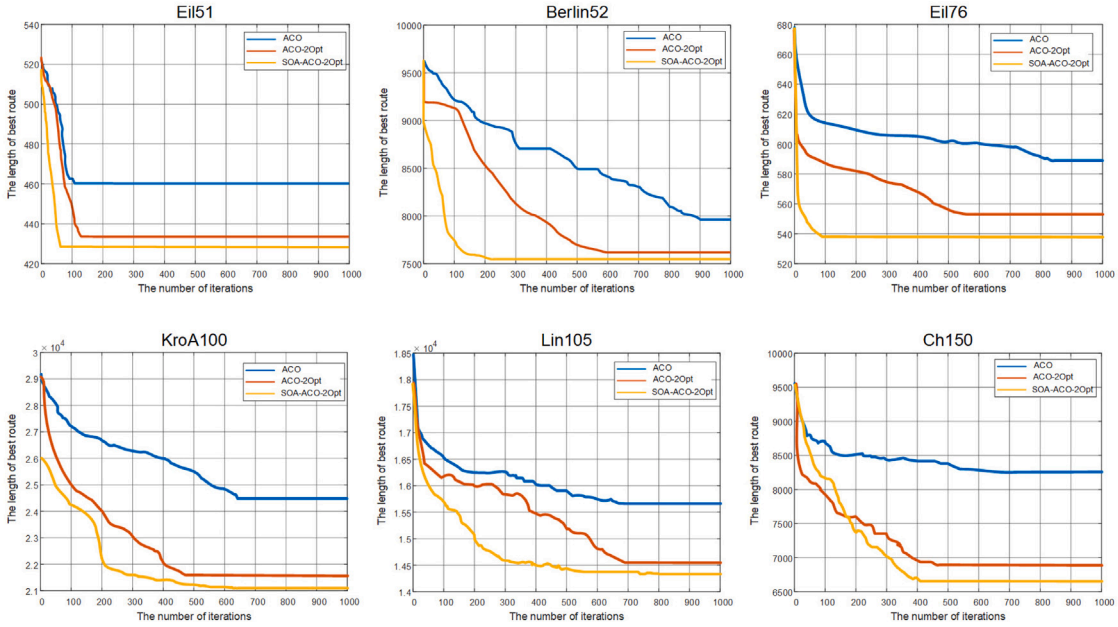


Fig. 10. Convergence curve graphs of the average value of the ten times running of ACO, ACO-2Opt, and SOA-ACO-2Opt.

Table 6
The best parameters values for ACO.

	α	β	ρ	Q
Eil51	1.3466	2.6897	0.1829	351.9053
Berlin52	2.8145	2.0705	0.2832	209.1953
Eil76	1.4756	3.6090	0.2488	379.7358
Rat99	1.3722	3.2375	0.2738	306.8875
KroA100	1.6752	2.9573	0.1502	311.2833
Lin105	0.9388	2.5620	0.1355	399.1526
Ch150	1.2646	3.6488	0.1398	281.1063
Pr226	2.3709	2.3023	0.2736	361.4942
Gil262	1.3644	4.0119	0.1123	283.6094
Pr264	1.0757	3.6488	0.1398	281.1063
u1432	1.4519	2.0845	0.2373	352.0830
u2319	1.8576	3.8514	0.2271	309.1617

parameter of the ACO and uses the $\epsilon - greedy$ strategy and the multi-population mechanism to improve the ACO, and the algorithm performs better.

The performance of SOA-ACO-2Opt is compared with four other algorithms from the literature that have been used to solve TSP. The statistics of results obtained for SOA-ACO-2Opt after ten runs, along with the reported statistics of other algorithms, are detailed in Table 8. The best results are given in bold.

As will be seen from Table 8, these results are better than the studies in the literature. As the examined results in the literature illustrate, the proposed method has produced closer results to the optimum.

4.4. Obstacle avoidance simulation experiments

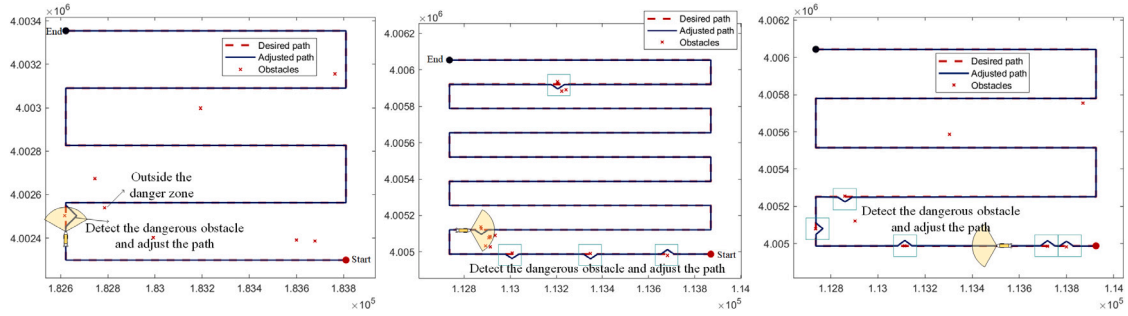
We performed simulations based on the obstacle avoidance strategy proposed in Section 3.2. The results are shown in Fig. 11. In the simulation and practical application, the maximum detection distance of the forward-looking sonar is set to 60 m, the horizontal opening angle is 120 degrees, and the number of beams is 10. The width of the dangerous zone is set to 10 m. The AUV starts from the "Start" position and advances along the desired path of the red dotted line. In simulating the position movement of an AUV, several obstacles (red crosses) will be found one after another. The path planner checks whether the obstacle hinders the progress of the preset task, judges whether the obstacle is in the dangerous zone, and, if so, avoids the obstacle. The simulation results show that the obstacle avoidance algorithm applied in this paper can help the AUV avoid obstacles effectively and return to the desired path. It can help AUVs equipped with forward-looking sonar to avoid collisions in actual work to ensure the smooth execution of tasks.

Table 7
Comparison of the effects of the default parameters and the best parameters by ACO after ten runs.

TSP	Default parameters			Best parameters		
	Best	Average	Error(%)	Best	Average	Error(%)
Eil51	453	460.6	8.12	433	447.4	5.02
Berlin52	7786	7958.9	5.53	7548	7870.5	4.36
Eil76	571	587.6	9.22	547	579.4	7.7
Rat99	1317	1354.2	11.82	1317	1356.2	11.99
KroA100	23,968	24,503.1	15.14	22,697	23,412.6	10.01
Lin105	15,422	15,690.3	9.12	15,373	15,612.3	8.58
Ch150	7015	8376.4	28.31	6815	7746.7	18.67
Pr226	93,096	93,331.8	16.13	86,430	90,439.8	12.53
Gil262	2777.2	2821.9	18.67	2698	2713.7	14.12
Pr264	53,630	54,521.1	10.96	52,576	53,681.2	9.25
u1432	855,995	868,841.7	467.98	225,239	227,621.1	48.80
u2319	415,702	419,659	79.15	335,388	338,007	44.29

Table 8
The computational results of the proposed method and other methods in the literature after ten runs.

Algorithm	TSP	Eil51	Berlin52	Eil76	Rat99	KroA100	Lin105	Ch150	Pr226	Gil262	Pr264
	BKS	426	7542	538	1211	21,282	14,379	6528	80,369	2378	49,135
SOA-ACO-2Opt	Best	**426**	**7542**	**538**	**1212**	**21,282**	**14,379**	**6528**	**80,832**	**2378**	**49,135**
	Average	**426.2**	7543	538.4	1222.4	21,298.2	14,381.5	6612.5	**80,617.4**	2383.1	49,182.7
	Error(%)	**0.05**	0.01	0.07	0.94	0.08	0.02	1.29	**0.31**	**0.21**	**0.10**
PSO-ACO-3Opt	Best	**426**	**7542**	**538**	1224	21,301	**14,379**	6538	–	–	–
	Average	426.5	7543.20	538.3	1227.4	21,445.10	14,379.15	6563.95	–	–	–
	Error(%)	0.12	0.02	**0.06**	1.35	0.77	**0.00**	**0.55**	–	–	–
PACO-3Opt	Best	**426**	**7542**	**538**	1213	**21,282**	**14,379**	6570	–	–	–
	Average	426.4	**7542**	539.85	1217.10	21,326.80	14,393.00	6601.40	–	–	–
	Error(%)	0.09	**0**	0.34	**0.5**	0.21	0.10	1.12	–	–	–
SOS-ACO	Best	**426**	–	**538**	–	**21,282**		6558	–	–	49,135
	Average	428.1	–	541.7	–	21,290.1	–	6571.2	–	–	49,250.7
	Error(%)	0.49	–	0.69	–	**0.04**	–	**0.20**	–	–	0.24
DSMO	Best	428.86	7544.37	558.68	–	21,298.21	14,383	–	83,587.98	2543.15	49,135
	Average	436.96	7633.6	572.7	–	22,024.27	15,114	–	85,935.69	2627.87	49 250.7
	Error(%)	2.57	1.21	6.45	–	0.08	5.11	–	6.93	10.50	0.24



(a) Group 1 (Obstacle avoidance effect display. $R_{ac} = 50m$)
(b) Group 2 (Multiple obstacles when AUV moves on the desired path. $R_{ac} = 30m$)
(c) Group 3 (Multiple obstacles when AUV moves on another desired path. $R_{ac} = 30m$)

Fig. 11. Obstacle avoidance simulation experiments.

5. Sea trials and analysis

To prove that the SOA-ACO-2Opt algorithm can be used in practical engineering applications and further prove the algorithm's robustness in different situations, we carried out a series of sea experiments based on QIYU-260 AUV. The experimental platform is introduced in this section, and the experimental results are presented and analyzed.

5.1. QIYU-260 AUV

QIYU-260 AUV (Fig. 12) is a small AUV system independently developed by the Underwater Vehicle Laboratory (UVL) of Ocean University of China. Since its development, it has been used as a test platform for various engineering applications, such as scientific research, marine parameter monitoring, marine geological/geomorphic exploration, and collaborative observation.

5.2. Sea trials design

In order to verify the applicability and performance of the algorithm in engineering applications (Fig. 2), we designed three different typical examples (as shown in Fig. 13) based on the three application backgrounds of "marine cruise", "data sampling", and "area survey".

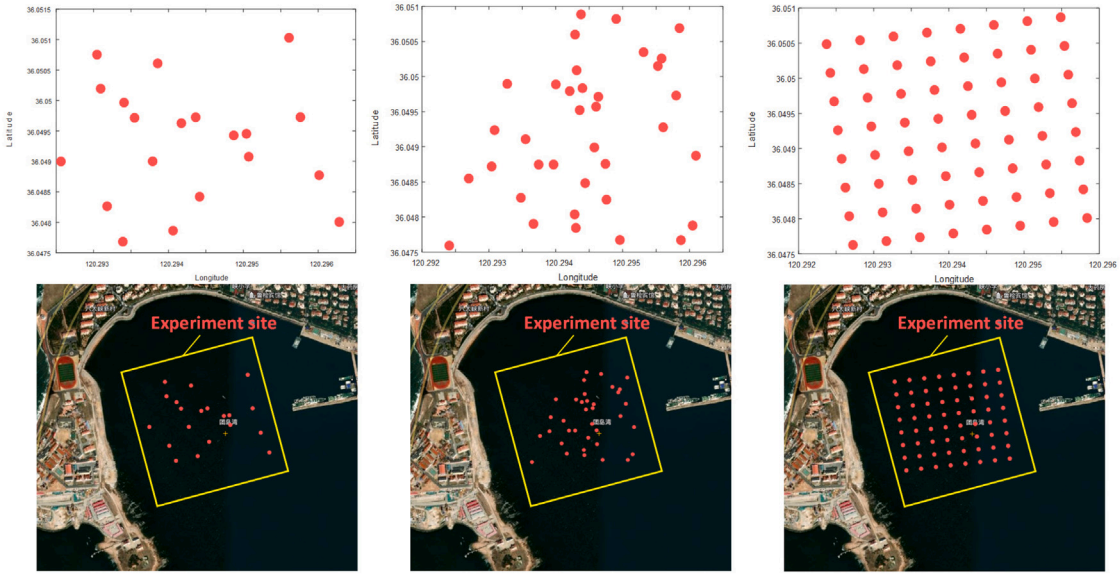
Case 1 AUV needs to traverse 20 cities, and its corresponding application scenario is the cruise of AUV.

Case 2 The style is similar to the example in TSPLIB. There are 36 cities, and the distribution is denser than in the other two cases. It represents the data sampling at given locations.

Case 3 There are 64 cities to visit, and the cities are usually concatenated [52,53] in a path based on a lawnmower algorithm, often set up for topographic/geological surveys. However, its

(a) QIYU-260 AUV (b) Sea trial photograph

Fig. 12. QIYU-260 AUV.



(a) Case 1 : 20 cities (a) Case 2 : 36 cities (a) Case 3 : 64 cities

Fig. 13. Three cases of the sea trials in TuanDao.

start and end points often do not overlap, and the distance is relatively long; from the perspective of solving TSP, intelligent path planning can ensure that the starting point is the last path point, significantly reducing the path length.

In order to verify the obstacle avoidance ability of AUV, as shown in Fig. 14, we selected oil barrels and bridge pillars as two kinds of obstacles to carry out sea tests. The oil barrel is about 1 meter in diameter and 1.5 meters high. The obstacle bridge column selected for the test has a diameter of about 1.5 m, and the bridge hole formed between the two pillars is about 8 m.

5.3. Sea trial results and analysis

Table 9 summarizes the simulation results for three instances.

For each case, ten searches were performed based on ACO and SA_4, respectively. The parameter settings of the algorithm were the same as those in Tables 2 and 4. The results are shown in Table 10. η represents the efficiency improvement of SA_4 relative to ACO, which is calculated as Eq. (24). The results show that the performance of the algorithm proposed in this paper is significantly better than ACO in terms of the best solution, stability, and robustness.

$$\eta = \frac{Average\,(\mathrm{ACO})}{Average\,(\mathrm{SA}_4)} \tag{24}$$

Table 9
Basic information of the TSP instances for resolving.

TSP	n	Number of edges ($(n(n-1)/2)$)
Case 1	20	190
Case 2	36	630
Case 3	64	2016

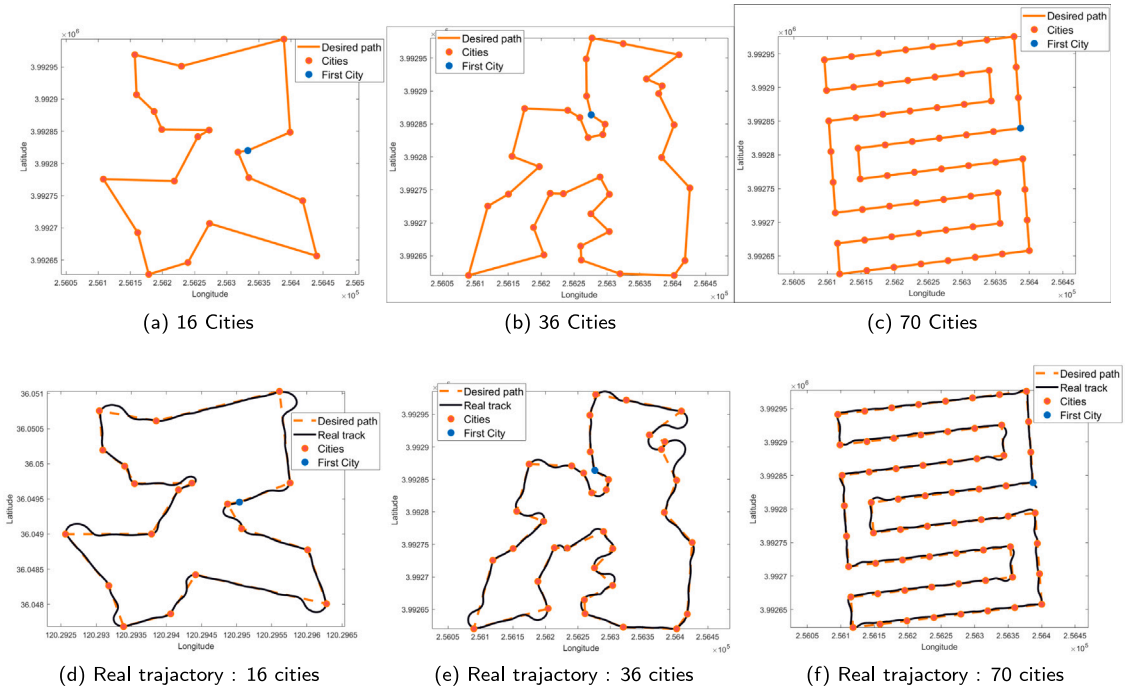
Based on the QIYU-260 AUV, three trials were conducted at Tuandao Bay in Qingdao, China. Before each experiment, the AUV reads the coordinates of cities corresponding to the cases and uses SA_4 to search ten times. Finally, the AUV executes the task according to the best path among these ten searches. In the actual tests, due to the small distance (several meters) between some path points, It may be challenging for AUV to achieve ideal path tracking. In order to reduce the impact of GPS positioning error and the restriction of AUV control performance, AUV sailed with appropriate adaptive deceleration to achieve a better control effect: in the path between two waypoints, the desired speed when navigating between points is 3kn, but if the AUV is less than 15 m from the next waypoint, the desired speed is adjusted to 2kn. The sea

(a) Obstacle: Oil barrel (b) Obstacle: Bridge pillars

Fig. 14. The chosen obstacles in the experiments.

Table 10
Comparison of the effects of the default parameters and the best parameters by ACO.

TSP	ACO				SA_4				η
	Best	Average	Worst	SD	Best	Average	Worst	SD	(%)
Case 1	1580	1582.6	1588	3.56	1580	1580	1580	0	**0.16**
Case 2	1853	1915.5	1976	32.92	1798	1803.8	1817	7.45	**5.83**
Case 3	3031	3122.5	3207	50.03	2694	2697	2724	9	**13.63**



(a) 16 Cities (b) 36 Cities (c) 70 Cities

(d) Real trajectory : 16 cities (e) Real trajectory : 36 cities (f) Real trajectory : 70 cities

Fig. 15. Best paths based on the proposed algorithm.

trial results, such as each route search's average and total mission time, are shown in Table 11. It can be seen that the time-consuming caused by the calculation is simply a drop in the bucket compared to the task time. The actual trajectory of the AUV in the experiments is shown in Fig. 15.

The sea trial results demonstrate that regarding and solving the AUV path planning as a TSP is reasonable and beneficial in engineering applications.

Compared with other spatial domains, the marine environment is more complex and changeable, and the application of AUV also faces various constraints, such as its energy consumption, deployment, and recycling. When sailing on the water surface, the safety of AUVs may be threatened by a collision with passing ships or other floating things; in actual marine applications, the deployment and recovery of AUVs will take up much time in marine operations, which leads to a less but more valuable time for AUVs to perform tasks.

Therefore, regarding safety and efficiency, AUV is usually deployed as close to the preset starting point as possible. If considering the AUV path planning as solving a TSP, as it is solved, the termination and starting points of the task are the same, and it is conducive to a safe and

Table 11
Sea trial results.

	Actual work time(s)	Path length (m)	Desired speed (kn)	Average time per operation based on SA_4 (s)
Case 1	1093	1580	3	7.25
Case 2	1270	1798	3	13.18
Case 3	1924	2694	3	19.5

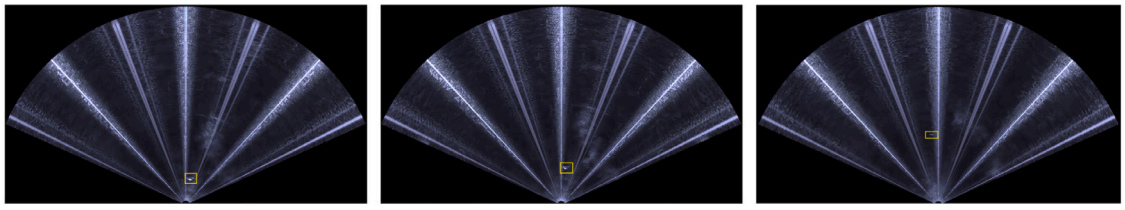
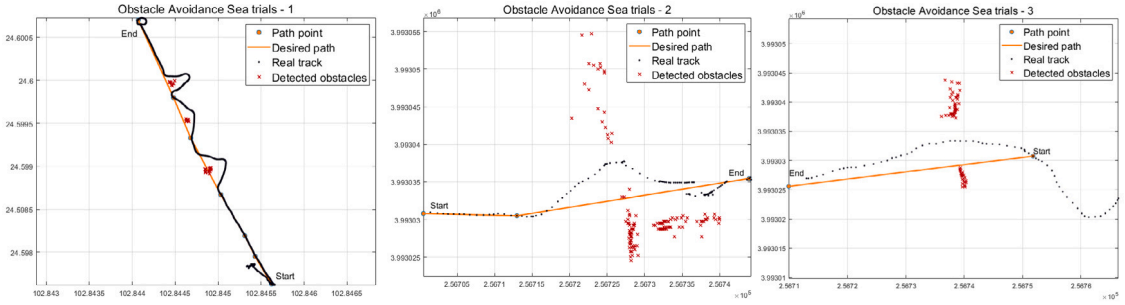


Fig. 16. Forward-looking sonar images. Obstacles are framed by yellow rectangles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Obstacle: Oil barrel (b) Obstacle: Bridge pillars (c) Obstacle: Bridge pillars

Fig. 17. Result analysis of AUV offshore obstacle avoidance tests.

quick recovery because the AUV will return to the starting point once the mission is accomplished. Besides, shorter paths mean less mission time and energy consumption and enhance AUV's working efficiency. A total of three sets of obstacle avoidance experiments were carried out.

The first set of experiments is shown in Fig. 17(a). Typical scanning images of oil barrels by forward-looking sonar are shown in Fig. 16. In this group of experiments, we deployed three oil barrels, and the oil barrels were suspended at a fixed height through floating balls and counterweights. When the AUV travels along the desired path (plotted in orange line), it obtains real-time information concerning obstacles (marked as the red cross) based on the forward-looking sonar. It plans the obstacle avoidance path according to the proposed strategy. The actual track of the AUV is marked in the black dot, and the results showed that the AUV successfully avoided three oil barrels.

The other two sets of experiments used bridge pillars as obstacles. The bridge pillars belong to a discrete and multi-obstacle environment, and the AUV needs to pass between the bridge pillars to complete the obstacle avoidance task. The red cross in Figs. 17(b) and 17(c) is the position of the bridge pillar detected by the forward-looking sonar. The two sets of experiments are in the same obstacle area. In the two sets of experiments, different expected paths are set on both sides of the bridge pillar to ensure that the initially expected path of the AUV will intersect with the position of the bridge pillar. In Fig. 17(b), the AUV deflects to the left, passes through the bridge hole, and finally reaches the desired termination; in Fig. 17(c), the AUV deflects to the right and bypasses the bridge pillar, indicating that the obstacle avoidance algorithm also completes the autonomous obstacle avoidance in a multi-obstacle environment.

The sea trials prove that the obstacle avoidance strategy proposed in this paper is safe and effective, has strong real-time performance and specific generalization ability, and can realize obstacle avoidance in different environments.

6. Conclusion

This paper proposes a new hybrid algorithm, combining improved SOA, enhanced ACO, and 2-Opt to solve TSP in AUV path planning to improve AUV's work efficiency in marine survey and patrol protection. Considering that the AUV may encounter obstacles during its travel, this paper also proposes an obstacle avoidance strategy, which can effectively avoid obstacles based on the obstacle information and the expected path to ensure the equipment's safety.

In this paper, the fundamental parameters α, β, ρ, Q of ACO are found through the improved SOA so that ACO has a more powerful search ability and more critical adaptation and robustness. The main contributions of our proposed algorithm are as follows: (1) Combining novel SOA and classic ACO for AUV to solve TSPs; (2) In order to reduce SOA's search for ACO parameters into a locally best, we use the Kent chaos graph to initialize the position of the seagull to make its initialization distribution more uniform; (3) In order to effectively prevent the ACO search from falling into local optimum and improve the convergence speed of the algorithm, we improved the adaptive multi-population mechanism of ants' division of labor and cooperation. Ants with different divisions of labor selectively adopt the $\epsilon - greedy$ strategy and the default strategy for path search. (4) To further shorten the planned path's length, we apply 2-Opt to perform additional optimization on the search results of the SOA-ACO algorithm. (5) The obstacle avoidance strategy is proposed based on the expected path to ensure the safety of AUV.

This paper investigates the method's performance by considering the best and average path length, standard deviation, and relative error percentage values based on the average of twelve test questions taken from TSPLIB. The SOA-ACO-2Opt algorithm has been applied to the marine engineering application of AUV and has passed the actual sea test verification. Experimental results show that the proposed

method has excellent search ability and performance, which is better than or similar to other comparative methods in the literature. More importantly, the algorithm is practical and can improve the efficiency of AUVs in performing tasks in engineering applications.

Based on the multibeam FLS information, combined with the details of AUV's actual work, this paper proposes an autonomous obstacle avoidance algorithm based on the expected path, which effectively solves the problem of real-time obstacle avoidance for AUVs using forward-looking sonar information in unknown environments. The obstacle avoidance strategy is simple, and the amount of calculation is small, which can realize AUV online fast path replanning. The experimental results show that the algorithm can calculate the appropriate obstacle avoidance path points when dealing with various types of obstacles, realize online path replanning, successfully bypass or pass through obstacles, and ensure the safety of AUV.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work is supported by The National Key Research and Development Program of China (Project number: 2016YFC0301400).

References

[1] C. Cheng, Q. Sha, B. He, G. Li, Path planning and obstacle avoidance for AUV: A review, Ocean Eng. 235 (2021) 109355.
[2] H. Flores, N.H. Motlagh, A. Zuniga, M. Liyanage, M. Passananti, S. Tarkoma, M. Youssef, P. Nurmi, Toward large-scale autonomous marine pollution monitoring, IEEE Internet Things Mag. 4 (1) (2021) 40–45.
[3] M. Aranganathan, Analytical techniques in marine biotechnology, 2022, ScienceOpen Posters.
[4] J.A. Howe, K. Husum, M.E. Inall, J. Coogan, A. Luckman, R. Arosio, C. Abernethy, D. Verchili, Autonomous underwater vehicle (AUV) observations of recent tidewater glacier retreat, western Svalbard, Mar. Geol. 417 (2019) 106009.
[5] M. Jacobi, D. Karimanzira, Multi sensor underwater pipeline tracking with AUVs, in: 2014 Oceans-St. John's, IEEE, 2014, pp. 1–6.
[6] P. González, R.R. Osorio, X.C. Pardo, J.R. Banga, R. Doallo, An efficient ant colony optimization framework for HPC environments, Appl. Soft Comput. 114 (2022) 108058.
[7] M. Akhand, S.I. Ayon, S. Shahriyar, N. Siddique, H. Adeli, Discrete spider monkey optimization for travelling salesman problem, Appl. Soft Comput. 86 (2020) 105887.
[8] R. Skinderowicz, Improving ant colony optimization efficiency for solving large TSP instances, Appl. Soft Comput. 120 (2022) 108653.
[9] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robot. Auton. Syst. 61 (12) (2013) 1258–1276.
[10] D. Du, P.M. Pardalos, Handbook of Combinatorial Optimization, vol. 4, Springer Science & Business Media, 1998.
[11] T. Sasaki, D. Biro, Cumulative culture can emerge from collective intelligence in animal groups, Natu. Commun. 8 (1) (2017) 1–6.
[12] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms, Comput. Ind. Eng. 137 (2019) 106040.
[13] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoret. Comput. Sci. 344 (2–3) (2005) 243–278.
[14] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.
[15] M. Neshat, G. Sepidnam, M. Sargolzaei, A.N. Toosi, Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications, Artif. Intell. Rev. 42 (4) (2014) 965–997.
[16] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. Mag. 22 (3) (2002) 52–67.
[17] J. Tang, G. Liu, Q. Pan, A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends, IEEE/CAA J. Autom. Sin. 8 (10) (2021) 1627–1643.
[18] A. Chakraborty, A.K. Kar, Swarm intelligence: A review of algorithms, in: Nature-Inspired Computing and Optimization, Springer, 2017, pp. 475–494.
[19] W. Elloumi, H. El Abed, A. Abraham, A.M. Alimi, A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP, Appl. Soft Comput. 25 (2014) 234–241.
[20] S. Ebadinezhad, DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem, Eng. Appl. Artif. Intell. 92 (2020) 103649.
[21] F. Chebihi, M.E. Riffi, A. Agharghor, S.C.B. Semlali, A. Haily, Improved chicken swarm optimization algorithm to solve the travelling salesman problem, Indonesian J. Electr. Eng. Comput. Sci. 12 (3) (2018) 1054–1062.
[22] K. Panwar, K. Deep, Discrete Grey Wolf Optimizer for symmetric travelling salesman problem, Appl. Soft Comput. 105 (2021) 107298.
[23] E. Ahmadi, B. Goldengorin, G.A. Süer, H. Mosadegh, A hybrid method of 2-TSP and novel learning-based GA for job sequencing and tool switching problem, Appl. Soft Comput. 65 (2018) 214–229.
[24] M. Gunduz, M. Aslan, DJAYA: A discrete jaya algorithm for solving traveling salesman problem, Appl. Soft Comput. 105 (2021) 107275.
[25] M. Mahi, Ö.K. Baykan, H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem, Appl. Soft Comput. 30 (2015) 484–490.
[26] B.A.S. Emambocus, M.B. Jasser, M. Hamzah, A. Mustapha, A. Amphawan, An enhanced swap sequence-based particle swarm optimization algorithm to solve TSP, IEEE Access 9 (2021) 164820–164836.
[27] B. Wei, Y. Xing, X. Xia, L. Gui, A novel particle swarm optimization with genetic operator and its application to tsp, Int. J. Cogn. Inf. Nat. Intell. (IJCINI) 15 (4) (2021) 1–17.
[28] F.J. Vasko, J. Bobeck, M. Governale, D. Rieksts, J. Keffer, A statistical analysis of parameter values for the rank-based ant colony optimization algorithm for the traveling salesperson problem, J. Oper. Res. Soc. 62 (6) (2011) 1169–1176.
[29] M. Yousefikhoshbakht, M. Sedighpour, A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem, Proc. Roman. Acad. A 13 (4) (2012) 295–302.
[30] M.S. Qamar, S. Tu, F. Ali, A. Armghan, M.F. Munir, F. Alenezi, F. Muhammad, A. Ali, N. Alnaim, Improvement of traveling salesman problem solution using hybrid algorithm based on best-worst ant system and particle swarm optimization, Appl. Sci. 11 (11) (2021) 4780.
[31] Y. Liu, B. Cao, H. Li, Improving ant colony optimization algorithm with epsilon greedy and levy flight, Complex Intell. Syst. 7 (4) (2021) 1711–1722.
[32] Y. Liu, X. Shen, H. Chen, An adaptive ant colony algorithm based on common information for solving the traveling salesman problem, in: 2012 International Conference on Systems and Informatics, ICSAI2012, IEEE, 2012, pp. 763–766.
[33] Z. Jiao, K. Ma, Y. Rong, P. Wang, H. Zhang, S. Wang, A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs, J. Comput. Sci. 25 (2018) 50–57.
[34] H. Zhang, X. You, Multi-population ant colony optimization algorithm based on congestion factor and co-evolution mechanism, IEEE Access 7 (2019) 158160–158169.
[35] K.Y. Wong, Komarudin, Parameter tuning for ant colony optimization: A review, in: 2008 International Conference on Computer and Communication Engineering, 2008, pp. 542–545.
[36] M. Peker, B. Şen, P.Y. Kumru, An efficient solving of the traveling salesman problem: The ant colony system having parameters optimized by the Taguchi method, Turk. J. Electr. Eng. Comput. Sci. 21 (7) (2013) 2015–2036.
[37] N. Rokbani, P. Kromer, I. Twir, A.M. Alimi, A new hybrid gravitational particle swarm optimisation-ACO with local search mechanism, PSOGSA-ACO-Ls for TSP, Int. J. Intell. Eng. Inform. 7 (4) (2019) 384–398.
[38] Y. Wang, Z. Han, Ant colony optimization for traveling salesman problem based on parameters optimization, Appl. Soft Comput. 107 (2021) 107439.
[39] Ş. Gülcü, M. Mahi, Ö.K. Baykan, H. Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem, Soft Comput. 22 (5) (2018) 1669–1685.
[40] G. Dhiman, V. Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, Knowledge-Based Syst. 165 (2019) 169–196.
[41] W. Long, J. Jiao, X. Liang, M. Xu, M. Tang, S. Cai, Parameters estimation of photovoltaic models using a novel hybrid seagull optimization algorithm, Energy 249 (2022) 123760.
[42] L. Lindner, O. Sergiyenko, M. Rivas-López, M. Ivanov, J.C. Rodríguez-Quiñonez, D. Hernández-Balbuena, W. Flores-Fuentes, V. Tyrsa, F.N. Muerrieta-Rico, P. Mercorelli, Machine vision system errors for unmanned aerial vehicle navigation, in: 2017 IEEE 26th International Symposium on Industrial Electronics, ISIE, 2017, pp. 1615–1620.
[43] X.d.J. Garcıa-Gutierrez, R. Alaniz-Plata, G. Trujillo-Hernández, I.Y. Alba-Corpus, W. Flores-Fuentes, J.C. Rodríguez-Quiñonez, D. Hernández-Balbuena, O. Sergiyenko, F.F. González-Navarro, P. Mercorelli, J.E. Miranda-Vega, F.N. Murrieta-Rico, Obstacle coordinates transformation from TVS body-frame to AGV navigation-frame, in: 2022 IEEE 31st International Symposium on Industrial Electronics, ISIE, 2022, pp. 589–593.

[44] X. Cao, L. Ren, C. Sun, Research on obstacle detection and avoidance of autonomous underwater vehicle based on forward-looking sonar, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[45] L.M. Gambardella, M. Dorigo, Ant-q: A reinforcement learning approach to the traveling salesman problem, in: Machine Learning Proceedings 1995, Elsevier, 1995, pp. 252–260.
[46] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[47] H. Jia, Z. Xing, W. Song, A new hybrid seagull optimization algorithm for feature selection, IEEE Access 7 (2019) 49614–49631.
[48] G. Dhiman, K.K. Singh, A. Slowik, V. Chang, A.R. Yildiz, A. Kaur, M. Garg, EMoSOA: A new evolutionary multi-objective seagull optimization algorithm for global optimization, Int. J. Mach. Learn. Cybern. 12 (2) (2021) 571–596.
[49] W. Qiao, Z. Yang, Modified dolphin swarm algorithm based on chaotic maps for solving high-dimensional function optimization problems, IEEE Access 7 (2019) 110472–110486.
[50] G. Reinelt, TSPLIB–A traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384.
[51] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.
[52] E.M. Arkin, S.P. Fekete, J.S. Mitchell, Approximation algorithms for lawn mowing and milling a preliminary version of this paper was entitled "the lawnmower problem" and appears in the Proc. 5th Canad. Conf. Comput. Geom., Waterloo, Canada, 1993, pp. 461–466, Comput. Geometry 17 (1) (2000) 25–50, URL https://www.sciencedirect.com/science/article/pii/S0925772100000158.
[53] R. Chang, Y. Wang, J. Hou, S. Qiu, R. Nian, B. He, A. Lendasse, Underwater object detection with efficient shadow-removal for side scan sonar images, in: OCEANS 2016 - Shanghai, 2016, pp. 1–5.

Yixiao Zhang received a B.S. degree from Qufu Normal University in 2018, and he is a doctoral student at the Ocean University of China. He mainly engaged in the intelligent control system for AUV, including decision-making, path planning, and AUV control technology.

Yue Shen received his M.S. degree from Henan University of Science and Technology in 1998 and his Ph.D. from X'an Jiaotong University, China, in 2003, respectively. In 2004, A.Prof. SHEN joined Ocean University of China (OUC), and now he is a full associate professor of OUC at the College of Information Science and Engineering. His research interests include AUV design and applications, decision making, and control.

Qi Wang received an M.S. degree in electronic and information engineering from the Ocean University of China in 2020. She is currently pursuing a Ph.D. degree in intelligent communication and information systems at the Ocean University of China. Her research interests include intelligent detection systems for autonomous underwater vehicles (AUV), neural networks, and machine learning.

Chao Song, a graduate student at Qingdao University of Science and Technology (QUST), received his B.Eng degree from QUST in 2020. He is committed to the research on collaborative path planning of AUV.

Ning Dai, a graduate student at the North University of China (NUC), received his B.Eng degree from NUC in 2021. He is committed to the research on path planning of AUV.

Bo He received his M.S. and Ph.D. degrees from the Harbin Institute of Technology, China, in 1996 and 1999, respectively. From 2000 to 2003, he was a PostDoctoral Fellow at Nanyang Technological University, Singapore, where he worked on mobile robots and unmanned vehicles. His research included precise navigation, control, and communications. In 2004, he joined the Ocean University of China (OUC), where he is currently a Full Professor and Deputy Head of the Department of Electronics Engineering, College of Information Science and Engineering. His research interests include AUV design and applications, AUV SLAM, AUV control, and machine learning.