Research Article
Cubic Spline Interpolation-Based Robot Path Planning Using a Chaotic Adaptive Particle Swarm Optimization Algorithm

Jianfang Lian,1 Wentao Yu,1 Kui Xiao,1 and Weirong Liu2

1College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha 410004, China
2Country School of Computer Science and Engineering, Central South University, Changsha 410083, China

Correspondence should be addressed to Wentao Yu; wtyu_csuf@126.com

Received 15 October 2019; Revised 15 December 2019; Accepted 14 January 2020; Published 20 February 2020

Abstract
This paper proposed a cubic spline interpolation-based path planning method to maintain the smoothness of moving the robot’s path. Several path nodes were selected as control points for cubic spline interpolation. A full path was formed by interpolating on the path of the starting point, control points, and target point. In this paper, a novel chaotic adaptive particle swarm optimization (CAPSO) algorithm has been proposed to optimize the control points in cubic spline interpolation. In order to improve the global search ability of the algorithm, the position updating equation of the particleswarm optimization (PSO) is modified by the beetle foraging strategy. Then, the trigonometric function is adopted for the adaptive adjustment of the control parameters for CAPSO to weigh global and local search capabilities. At the beginning of the algorithm, particles can explore better regions in the global scope with a larger speed step to improve the searchability of the algorithm. At the later stage of the search, particles do fine search around the extremum points to accelerate the convergence speed of the algorithm. The chaotic map is also used to replace the random parameter of the PSO to improve the diversity of particle swarm and maintain the original random characteristics. Since all chaotic maps are different, the performance of six benchmark functions was tested to choose the most suitable one. The CAPSO algorithm was tested for different number of control points and various obstacles. The simulation results verified the effectiveness of the proposed algorithm compared with other algorithms. And experiments proved the feasibility of the proposed model in different dynamic environments.

1. Introduction

Human beings always have an urge to complete all the works and jobs automatically through automated machines, which inspires the researchers to focus on the design of mobile robots. Path planning is one of the most critical skills for mobile robots. It has been used in different applications such as robot rescue [1], robot service [2], and robot patrol [3]. The main goal of path planning is to find the shortest and smooth path between the starting and target points. However, it is hard to find the shortest and smooth collision-free path for robots because of the complex robot working environment.

In general, the existing robot path planning methods can be classified into two categories: classical algorithms and heuristic optimization algorithm. The main classical algorithms include cell decomposition, artificial potential field, and sampling-based methods [4]. However, classic methods are very time consuming and require ample storage memory [5]. Thus, heuristic optimization algorithms [6, 7] are used frequently to optimize the path planning problem, such as differential evolutionary (DE) algorithm [8], genetic algorithm (GA) [9], A* algorithm [10], artificial bee colony (ABC) algorithm [11], annealing (SA) [12], particle swarm optimization (PSO) [13], and ant colony optimization (ACO) [14].

GA is an optimization algorithm based on natural genetics, which includes natural selection, crossover, and variation [9]. Premature convergence may occur because genetic algorithms operate in grid maps and do not control
population diversity. Therefore, in the robot path planning application, GA is integrated with other algorithms such as artificial potential field method, artificial bee population algorithm, etc., to achieve better results [15]. However, the main disadvantage of improved genetic algorithms in the robot path planning field is that it takes a long time.

The A* algorithm is an effective and direct method to search path in a static environment. However, when the A* algorithm was used to search in the neighborhood of the current node, the path beyond the threshold length is abandoned due to the local path length limitation, resulting in the final path being not optimal. Also, if there are not enough sample points to build the graph, the search success rate may be low. Nevertheless, increasing the number of sampling points increases costs. Hence, there are many improved A* algorithms to improve the optimal range of robot paths and search success rates [16]. These improved A* algorithms are very inefficient in dynamic environments. Because when the environment changes, such as new barriers, the A* algorithm and the improved A* algorithms must research from the current node to the end of the path. Hence, if the environment changes and the robot’s current position is still quite a long distance from the final target, then the A* algorithm and the improved A* algorithms will take a lot of time to redraw its path.

The ABC algorithm is based on a simulation of the foraging behavior of the bee colony [11]. ABC is easy to implement. ABC also has some drawbacks like early convergence, stagnation, and slow convergence. The ABC performs very well while exploring the feasible search space, but it shows poor performance for exploitation. Besides, ABC is first applied to solve the problems of function optimization, while the path planning of a mobile robot is a combinatorial optimization problem. Therefore, there are some difficulties in algorithm construction.

SA is a stochastic global optimization algorithm simulated by Kirkpatrick. The convergence probability of SA algorithm is 1 [12]. This characteristic can guarantee the achievement of global optimization when it is used for robot global path planning while it has the disadvantages of slow convergence speed. The ACO is a random search optimization algorithm with characteristics of positive feedback and parallel computing, which applies to various problems such as traveling salesman problem [17], quadratic programming problem [18], and production scheduling problem [19]. However, the optimal path of ant colony algorithm planning is time-consuming.

Traditional robot path planning attempts to find an optimal path, which may contain some sharp turns and polygonal lines. However, this algorithm is not flexible because it adds extra workload by moving the robot along the sharp turn by stopping, rotating, or restarting, even switching between different modes. Therefore, traditional robot path planning is time-consuming and labor-intensive for specific tasks requiring smooth motion.

Spline interpolation curve is a smooth curve passing through a series of given points. Quadratic spline interpolation is a quadratic polynomial interpolation. When the coefficient of the highest degree term is 0, the quadratic spline interpolation curve is a straight line [20]. Therefore, the quadratic interpolation is not applicable. However, cubic spline interpolation has the convergence property of the first and second derivatives. When the highest term coefficient of cubic function is 0, the cubic spline interpolation curve is still a curve, and the interpolation effect should be better. In addition, compared with high-order interpolation, cubic spline interpolation has the advantages of simple calculation and good stability.

Cubic spline interpolation has been widely applied. For example, cubic spline interpolation was used to control the quality of air temperature [21]. By interpolating the temperature observations for each altitude segment, suspicious values in the temperature observation data can be more effectively marked. Also, cubic spline interpolation was used to simulate the trajectory of the UAV [22] and study the path smoothness of the manipulator. In [23], the cubic spline interpolation was applied in the temperature compensation aspect of alcohol concentration measurement results.

In this paper, the moving robot’s path can be drawn as a series of line segments that join the path nodes. The cubic spline function was used to interpolate on the path of the starting point, control points, and target point. Thus, a full path that was formed by connecting all interpolation points was obtained. The control points’ position determines the length of the path, the number of which determines the number of spline curves and the maximum turnings of the path. Therefore, these control points are used to optimize the path of mobile robots. Heuristic optimization algorithms are an alternative to optimize control points.

Particle swarm optimization (PSO) is a computational technology. Like other evolutionary algorithms, PSO achieves the search for optimal spatial solutions through individual collaboration and competition. However, compared with evolutionary algorithms, PSO is simple and easy to implement. Moreover, PSO needs to adjust relatively few parameters, which is convenient and practical. PSO has been widely used in many applications. For example, the PSO algorithm is applied to deal with mathematical problems [24], which shows excellent performance in solving some classical function optimization problems and even some nonlinear functions. Also, the PSO algorithm is introduced into the fault diagnosis to propose a novel intelligent diagnosis method, which is applied to diagnose the faults of the motor bearing [25]. Then, PSO is also used to solve the problem of optimal deployment of wireless sensor network (WSN) nodes and to solve many problems caused by the optimized deployment of static nodes [26].

However, in some cases, particle swarm optimization cannot find a global optimal solution due to particle depletion. Moreover, the search strategy of the particle swarm algorithm is mainly based on a random walk, so it cannot always successfully find the optimal solution. To date, some variants of PSO have been presented to improve the original version PSO. Most of the current existing PSOs can be roughly divided into three categories: parameter selection, hybrid versions, and topology structure, respectively [27].

The proper selection of control parameters, such as inertia weight and acceleration coefficient, can significantly
influence the convergence of PSO. In [28], Jiao et al. proposed a new inertia weight particle swarm optimization (IWPSO) algorithm, which uses Boltzmann search (BPSO) to adaptively adjust inertia weight $C_1$ and $C_2$ in the velocity update equation. The IWPSO algorithm can guide particles to find the most promising regions in the search space. In [29], Melin et al. proposed a parameter self-tuning based on fuzzy logic. They proposed a method of dynamically adjusting the iterative coefficients of the inertia weights $C_1$ and $C_2$ using fuzzy control, where the iterative coefficients are the learning factors. However, the design of such algorithms is very complicated, which increases the workload. Moreover, the optimization accuracy of these algorithms is not improved, and the global search ability of the algorithm is weak.

Many studies improve the performance of particle swarm optimization algorithms by combining particle swarm optimization with other search techniques, such as differential evolution (DE) [30], artificial potential field (APF) [31], genetic algorithm (GA) [32], and neighborhood search [33]. In [34], Ding et al. combined a quantum-behavior PSO with the simplex algorithm to solve the load flow problem. However, these algorithms are inherently complex, so the combination of these algorithms with particle swarm optimization algorithms will be more complicated and time-consuming.

Different types of topology structures have been studied in the literature to enhance the performance of PSO. In [35], Kong et al. presents a simplified particle swarm optimization (SPSO). SPSO is an improved particle swarm optimization algorithm that combines the particle updating formula into one and introduces the group of optimal terms. The algorithm performance has been improved to a certain extent. However, too many different terms in the algorithm affect the algorithm performance, and the searchability of the algorithm cannot keep up with the convergence speed after the updating formula is merged.

In [36], Nagra et al. presented a dynamic multiswarm particle swarm optimizer (DMPSO). Different from PSO, the swarms in DMPSO are dynamic, and DMPSO often regroups with different regrouping plans and exchanges information between subgroups. Multipopulation particle swarm optimization (MPSO) is introduced into a niche generation and evolution strategy [37]. Firstly, PSO containing $N$ particles is divided into two niche subpopulations to generate independent search space. These niche subpopulations are not isolated in space, and all the particles evolve within their subpopulations, according to PSO. When evolution ends, the two subpopulations will chase different extremes. However, this kind of algorithm is complex and time-consuming.

Chaos concept is an alternative strategy to solve the problem of particle depletion, which has many excellent extrinsic properties, such as periodicity, ergodicity, and pseudorandomness. These excellent properties are important to ensure that the generated solutions by the algorithm can be diverse enough to reach every mode in the multimodal objective search space potentially. Hence, chaos search can escape more easily from a local optimal solution than the standard stochastic search. Also, chaotic systems have been widely used in various applications, and it was already combined with some optimization algorithms [38]. This combination may generate solutions that are more flexible and diverse than standard optimization algorithms [39]. To date, there are some widely used chaotic maps, such as Singer map [40], Kent map [41], Logistic map [42], and Sine map [43].

In this paper, a cubic spline interpolation-based path planning method has been proposed to maintain the smoothness of moving the robot’s path. Several path nodes were selected as control points for cubic spline interpolation. A full path was formed by interpolating on the path of the starting point, control points, and target point. The main contribution is to use chaotic adaptive particle swarm optimization (CAPSO) algorithm to present a novel algorithm that is used to optimize control points in cubic spline interpolation. The fitness function of CAPSO synthesizes two evaluation functions that consider path length after cubic spline interpolation and obstacle risk degree separately. The main improvement of the CAPSO algorithm is illustrated below.

To strengthen the global searchability of the algorithm, we introduced the beetle foraging strategy to modify the particle position update formula of PSO. In [44], Jiang and Li proposed the beetle antennae search algorithm according to beetle foraging strategy. The beetle antennae search algorithm is extremely simple and efficient. Therefore, CAPSO improves the searching ability without affecting the convergence speed.

Besides, the parameter $\omega$ in PSO is used to balance the search capabilities of local and global search of particles in the swarm [45]. The parameter $C_1$ symbolizes self-cognition, and $C_2$ symbolizes social influence. It is advisable to take a higher value of $C_1$ in the beginning than $C_2$ and gradually reversing during the search. The three control parameters in the PSO algorithm are adaptively adjusted by using the trigonometric function. $C_1$ and $\omega$ are both adjusted between linear decrement and nonlinear decrement while $C_2$ is adjusted between linear increase and nonlinear increase; thus, at the beginning of the algorithm, particles can explore better regions in the global scope with a larger speed step to improve the searchability of the algorithm. At the later stage of the search, particles do excellent search around the extremum points to accelerate the convergence speed of the algorithm.

Since the traversal of chaotic variables is not repeated within a specific range, chaotic maps are used instead of random parameters in the PSO algorithm. After chaotic processing, the diversity of particle group traversal is optimized, and the original random characteristics of the standard PSO algorithm are retained, which is able to effectively prevent the PSO from plunging into local optimal and make the particles proceed with searching in other regions of the solution space. As different chaotic maps may lead to different behaviors of the proposed algorithm, we tested the performance of six benchmark functions to choose the most suitable one. We also tested the influence of the number of control points on path planning. Experiments
have been conducted to compare the proposed algorithms with PSO, SA, GA, DE, and some other improved PSO algorithms in different environments. Experiments also tested the feasibility of the proposed model in different dynamic environments.

The rest of this paper is organized as follows. Section 2 introduces the background. Section 3 introduces the problem formulation for robot path planning. Section 4 introduces the chaotic adaptive particle swarm optimization algorithm, and Section 5 introduces the application of CAPSO algorithm in path planning. Section 6 introduces experiments and result analysis. Finally, Section 7 summarizes the whole paper.

2. Background

2.1. Particle Swarm Optimization. PSO is inspired by the social behavior of some biological organisms, especially the group’s ability of some animal species to locate a desirable position in the given area. In the PSO algorithm, each particle has a memory that tracks the best position of the previous iteration: the particle’s optimal position pbest and the particle’s global optimal position gbest, each with a velocity \( v_i \) and position \( X_i \). The speed update method for the \( i \)th particle at \( t+1 \) iteration is

\[
v_i(t + 1) = \omega v_i(t) + c_1 r_1(t) [p_{\text{best}}(t) - x_i(t)] + c_2 r_2(t) [g_{\text{best}}(t) - x_i(t)],
\]

where \( v_i(t + 1) \) is the new velocity of the particle at time \( t + 1 \), \( x_i(t) \) is the position of the current particle at time \( t \), \( \omega \) represents the weight, \( C_1 \) and \( C_2 \) are the learning factors, and \( r_1 \) and \( r_2 \) are random numbers of \([0, 1] \), which increases the randomness of particle flight. The position update method for the \( i \)th particle at \( t+1 \) iteration is

\[
x_i(t + 1) = x_i(t) + v_i(t + 1).
\]

In this paper, the particle coding is coordinates of several path nodes in the environment. The update of particle positions in two successive iterations is shown in Figure 1.

2.2. Cubic Spline Interpolation. The path of the mobile robot should be smooth to reduce the shaft wear of real robots’ wheels and energy consumption. We used cubic spline interpolation to achieve this goal. And cubic spline interpolation is to form a smooth curve through a series of shape points. Take \((n+1)\) nodes on the interval \([a, b]\):

\[
a = x_0 < x_1 < \cdots < x_n = b.
\]

A function \( f(x) \) on \([a, b]\) becomes an interpolated cubic spline function if the following two conditions are met:

In each interval \([x_{i-1}, x_i]\), \( f(x) \) is a cubic polynomial function.

\[
f_i(x) = a_i + b_i (x - x_i) + c_i (x - x_i)^2 + d_i (x - x_i)^3,
\]

where \( f(x) \) is continuous in the interval \([a, b]\).

In this paper, cubic spline interpolation is used to interpolate at the starting point, three path nodes, and the target point. Thus, a completely smooth path is formed by connecting all interpolation points.

3. Problem Formulation for Robot Path Planning

Given a robot and a two-dimensional workplace including obstacles and danger sources, path planning problem is typically stated as follows: to find an optimal collision-free path from starting and ending points according to some performance merits, such as the length, the time, the smoothness, and the energy. In this paper, we pay attention to the length and the risk degree (safety). To model the path planning problem, we model the workplace of robots first. The length can be expressed as a mathematical formula:

\[
\min \|P(t)\| = \sum_{i=1}^{d} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2},
\]

where \((x_i, y_i)\) is a path node after interpolation, and there is a total of \( d \) path nodes after interpolation. In this paper, the interpolation points of each spline curve are 1000. The number of curves is \((n + 1)\), where \( n \) is the number of control points. Thus, the number of path nodes \( d \) is \((1000 \times (n + 1))\). \( P(t) \) is the sum of the lengths of adjacent path nodes after
interpolation at time \( t \), which represents the length of the path at time \( t \).

For the risk degree, we suppose the obstacle is a rigid body, denoted by \( O_k \). For the sake of simplicity, these obstacles are represented by circles, and the center is \( O_k \), where \( k \) is the number of obstacles in the problem. Even with an irregular obstacle, we can seek its circumscribed circle. To obtain a collision-free path, the safety distance \( d_{\text{safe}} \) between the path and the obstacle should be greater than the threshold \( d_{\text{min}} \), which represents the minimum distance between the path and the obstacle.

\[
d_{\text{safe}} = \sqrt{(x_i - O_k^x)^2 + (y_i - O_k^y)^2 - y_{OP}^2},
\]

where \( O_k^x \) and \( O_k^y \) represent the horizontal and vertical coordinates of the \( k \)th obstacle, respectively, and \( y_{OP} \) represents the radius of the \( k \)th obstacle. The path is feasible only when \( d_{\text{safe}} < d_{\text{min}} \); otherwise, it is not feasible. The risk degree can be expressed in a mathematical formula:

\[
P(t) = \begin{cases} 
\text{infeasible path}, & d_{\text{safe}} < d_{\text{min}}, \\
\text{feasible path}, & d_{\text{safe}} < d_{\text{min}}. 
\end{cases}
\]

The objective function of the algorithm is to achieve the above two objectives. The fitness function of CAPSO is in the following mathematical formula:

\[
\begin{align*}
\text{Min} \quad & \| P(t) \| \\
\text{s.t.} \quad & P(t) \in C^2 \\
& P(t) \in P_{\text{free}},
\end{align*}
\]

where \( C^2 \) represents a set of second-order differentiable functions and \( P_{\text{free}} \) represents a set of collision-free paths satisfying the constraint of formula (11).

4. Chaotic Adaptive Particle Swarm Optimization Algorithm

Although the PSO algorithm proved efficient for solving different optimization problems, it still has drawbacks. In some cases, particle swarm optimization cannot find a global optimal solution due to particle depletion. Besides, at the beginning of the optimization process, particles can almost walk randomly in the entire search space, while particles can walk faster randomly in the search space, which may lead to the algorithm selecting the suboptimal solution. In other words, the search strategy of the particle swarm algorithm is mainly based on a random walk, so it cannot always successfully find the optimal solution. CAPSO makes the following improvements.

4.1. Modification Based on Chaotic Map. A chaotic system is similar to a random system, but it is different from the random phenomena that arise from the random term or coefficient of the system itself. For a real random system, the value from a given moment does not know the determining value of any subsequent moment, i.e., the system is unpredictable in the short term. For a chaotic system, due to its sensitivity to the dependence of the original value, its short-term behavior is completely certain, but its inherent randomness makes it impossible to predict the exact operation in the long run. Therefore, chaotic systems have better dynamics and statistical properties, which is crucial to ensure that chaotic variables can traverse all states within a specific range without repeating. In other words, the chaotic orbital passes every state point in the chaotic region in a finite time. These dynamic characteristics are essential to ensure that the solutions generated by the particle swarm algorithm are sufficiently diverse. In this paper, the one-dimensional chaotic map is used to replace the random parameter \( r_1 \) and \( r_2 \) in (1), allowing for the required mix between exploitation and exploration. Because different chaotic maps may lead to different behaviors of the proposed algorithm, this paper uses four well-known chaotic maps, namely, Kent, Sine, Singer, and Logistic, which constitute four different algorithms: CAPSO-Kent, CAPSO-Sine, CAPSO-Singer, and CAPSO-Logistic. By comparing the advantages and disadvantages of the four algorithms for path planning, a mapping is selected as the optimal alternative.

4.1.1. Singer Map. The definition of this map is as follows:

\[
x_{k+1} = \mu(7.86x_k - 23.31x_k^3 + 28.75x_k^3 - 13.3x_k^3),
\]

where \( \mu \) is a parameter of 0.9 to 1.08; in this paper, \( \mu \) is set to 1.01.

4.1.2. Sine Map. The definition of this map is as follows:

\[
x_{k+1} = \frac{a}{4}\sin(\pi x_k),
\]

where \( 0 < a < 4 \); in this paper, \( a \) is set to 4.

4.1.3. Kent Map. The definition of this map is as follows:

\[
x_{k+1} = \begin{cases} 
\frac{x_k}{a}, & 0 < x < a, \\
1 - \frac{x_k}{1 - a}, & a < x < 1,
\end{cases}
\]

where \( 0 < a < 4.5 \); in this paper, \( a \) is set to 0.3.

4.1.4. Logistic Map. The definition of this map is as follows:

\[
x_{k+1} = ax_k(1 - x_k),
\]

where \( 3.1 < a < 3.99 \); in this paper, \( a \) is set to 3.99.

The speed update method for the \( i \)th particle at \( t + 1 \) iteration is

\[
v_i(t + 1) = \omega v_i(t) + c_1x_{k+1} [p_{\text{best}}(t) - x_i(t)] + c_2x_{k+1} [g_{\text{best}}(t) - x_i(t)].
\]
over the 4096 cycles, while in the Sine map and Logistic map, value 0 and value 1 appear more frequently over the 4096 cycles. In Singer map, values from 0.75 to 0.98 appear more frequently over the 4096 cycles. We use chaotic maps to update $r_1$ and $r_2$ random numbers to improve the diversity of the particle swarm. In other words, we want chaos values to be traversed rather than one value constantly appearing. Hence, we choose Kent map as the optimal alternative. Also, experiments verified to select Kent map is best suitable for the algorithm.

4.2. Beetle Search Strategy-Based Location Update. In order to improve the searchability of the algorithm, we introduced the beetle search strategy into the position update formula of the particle swarm algorithm. The beetle’s antennae have two essential functions: one is to detect the surrounding environment. For example, when an obstacle is encountered, the antennae can sense its size, shape, and stiffness. The second is to capture the taste of food. When a higher concentration of odor is detected on one side of the antennae, the beetles will rotate in the same direction. Otherwise, they will turn to the other side. According to this simple principle, beetles can effectively find food.

The main advantage of the beetle search strategy is that its design is simple and can solve optimization problems in a short time. The beetle search strategy is as follows:

Step 1 generates a random vector and normalizes it:

$$\text{dir} = \text{rands} (n, 1),$$

$$\text{dir} = \frac{\text{dir}}{\text{norm} (\text{dir})}. \quad (18)$$

Step 2 calculates the position of the left and right whiskers separately:

$$x_l(t) = x(t) + d_0 \times \frac{\text{dir}}{2},$$

$$x_r(t) = x(t) - d_0 \times \frac{\text{dir}}{2}. \quad (20)$$

where $d_0$ is the search step size of the antennae and the ratio of the step size step to the search step size $d_0$ is a fixed constant, that is, $c$ is a constant. Step can be a fixed value and a variable value. This article sets it to a variable step size as a mathematical formula:

$$\text{step} = \eta \times \text{step}, \quad (21)$$

eeta can be a constant or a variable. The eta variable in this article is a mathematical formula (19).

$$\text{eta} = \text{step} \times \left( \text{step} \times (10 \times \frac{k}{k_{\max}}) \right). \quad (22)$$

$k_{\max}$ is the maximum number of iterations and $k$ is current iteration times. Both step 0 and step 1 are constants. eta is the coefficient of step size change and less than 1. In formula (22), eta decreases with the increase of $k$. A large search step size means a large search area. Thus, at the beginning of the algorithm, the larger search step size can explore better areas in the global scope, and in the late stage of the search, the smaller step size can do a good search near the extremum point.
Step 3 calculates the odor intensity of the left and right whiskers separately:

\[
\begin{align*}
& f_{\text{left}} = f(x_l), \\
& f_{\text{right}} = f(x_r).
\end{align*}
\]  

(23)

\[
\begin{align*}
x_b(t + 1) &= \begin{cases} 
  x_b(t) + \text{step} \cdot \text{sign}(x_l(t) - x_r(t)), & f_l(t) \leq f_r(t), \\
  x_b(t) - \text{step} \cdot \text{sign}(x_l(t) - x_r(t)), & f_l(t) > f_r(t),
\end{cases}
\end{align*}
\]

which is

\[
x_b(t + 1) = x_b(t) - \text{step} \cdot \text{direction} \cdot \text{sign}(f_l(t) - f_r(t)).
\]  

(25)

In order to improve the global search ability of the algorithm, the position of the particle in CAPSO is updated as follows:

\[
x_i(t + 1) = x_i(t) + Mv_i(t + 1) + (1 - M)x_p(t),
\]  

(26)

where \(M\) is a constant; in this paper, \(M\) is set to 0.34.

4.3. Trigonometric Function Adaptation. PSO includes three parts: current motion influence, individual particle influence, and particle swarm influence. The first part is controlled by the weighting factor \(\omega\). The second and third parts are controlled by the acceleration factors \(C_1\) and \(C_2\). A larger \(\omega\) is useful for jumping out of the local optimal, while a smaller \(\omega\) is suitable for the algorithm to converge. The optimal value of the particle swarm is crucial in the early stage of the algorithm optimization, while the optimal value of particle individual is significant in the later stage. Therefore, in this paper, the inertia weight factor and the acceleration factor are adaptively adjusted by using the trigonometric function at each stage of the algorithm operation.

\[
\omega(k) = \frac{\omega_{\text{max}} - \omega_{\text{min}}}{2} \cos \left( \frac{\pi \cdot k}{k_{\text{max}}} \right) + \frac{\omega_{\text{max}} + \omega_{\text{min}}}{2},
\]  

(27)

where \(k_{\text{max}}\) is the number of final iterations; \(k\) is the number of iterations of the algorithm; and \(\omega(k)\) is the inertia weighting factor corresponding to the \(k\)th iteration. In this paper, \(k_{\text{max}}\) is set to 100, \(\omega_{\text{max}}\) is set to 0.9, and \(\omega_{\text{min}}\) is set to 0.4. The changes in the weighting factor in the algorithm operation are shown in Figure 3, where the maximum number of iterations is 500.

\[
\begin{align*}
C_1(k) &= \cos \left( \frac{\pi \cdot k}{k_{\text{max}}} \right) + a, \\
C_2(k) &= -\cos \left( \frac{\pi \cdot k}{k_{\text{max}}} \right) + a,
\end{align*}
\]  

(28)

(29)

where \(a\) is set to 1.5. The changes in the three parameters in the algorithm operation are shown in Figure 1, where the maximum number of iterations is 500.

The parameter \(\omega\) in PSO is used to balance the search capabilities of local and global search of particles in the swarm. The parameter \(C_1\) symbolizes self-cognition, and \(C_2\) symbolizes social influence. It is advisable to take a higher value of \(C_1\) in the beginning than \(C_2\) and gradually reversing during the search. In other words, a higher value of \(C_1\), a higher value of \(\omega\), and a lesser value of \(C_2\) are advisable at the beginning of the algorithm, while a lesser value of \(C_1\), a lesser value of \(\omega\), and a higher value of \(C_2\) are advisable at the later stage of the search. The three control parameters in the PSO algorithm are adaptively adjusted by using the trigonometric function. \(C_1\) and \(\omega\) are both adjusted between linear decrement and nonlinear decrement while \(C_2\) is adjusted between linear increase and nonlinear increase.

As shown in Figure 3, at the beginning of the algorithm, \(A > B, E > F > G > K > L\). Also, \(C > D, M > N > O > P > Q\) at the later stage of the search. Thus, changing to the trigonometric functions make it better compared to the linear adaptation and constant. At the beginning of the algorithm, particles can explore better regions in the global scope with a more massive speed step to improve the searchability of the algorithm. At the later stage of the search, particles do excellent search around the extremum points to accelerate the convergence speed of the algorithm.

5. The Application of CAPSO Algorithm in Path Planning

Through the introduction of the above sections, this section proposes the CAPSO algorithm and wants to compare the traditional PSO algorithm, and the CAPSO algorithm has a higher global searchability and search speed. Figure 4 shows the flowchart of CAPSO in robot path planning.

The path planning process based on the CAPSO algorithm is as follows:

Step 1: initialize the particle group, including the population size \(N\) and the velocity \(V_i\) of each particle \(p_i\).

Step 2: calculate the fitness value of each particle \(\text{fit}[i]\).

Step 3: for each particle, compare its fitness value \(\text{fit}[i]\) with the individual extremum \(p_{\text{best}}[i]\). If \(\text{fit}[i] < p_{\text{best}}[i]\), replace \(p_{\text{best}}[i]\) with \(\text{fit}[i]\).

Step 4: for each particle, compare its fitness value \(\text{fit}[i]\) with the global extremum \(g_{\text{best}}\). If \(\text{fit}[i] < g_{\text{best}}[i]\), replace \(g_{\text{best}}[i]\) with \(\text{fit}[i]\).
Start

Initialize the parameters of CAPSO, particle velocity, and position

Calculate the individual optimum

Calculate the population optimum

Update r1 and r2

Update the velocity of the particle

Update the particle’s position

Choose a map

Logistic map

Sine map

Sine r map

Kent map

Update the path

Boundary processing

Update the path

Updated parameter

Satisfy the end condition

Y

Output result

End

No

Figure 4: Flowchart of robot path planning based on CAPSO algorithm.

Figure 3: (a) The variation diagram of ω in algorithm operation. (b) The variation diagram of C1 and C2 in algorithm operation.
Step 5: iteratively update the position of the particle according to formula (26).

Step 6: use the chaotic map to update \( r_1 \).

Step 7: iteratively update the velocity of the particle according to updated formula (17).

Step 8: boundary condition processing.

Step 9: update parameters according to formulae (27)–(29).

Step 10: determine whether the algorithm termination condition is satisfied: if yes, end and output the optimization result; otherwise, return to Step 2.

6. Experiments

In this section, the performance of the proposed CAPSO algorithm is evaluated by experiments, and the optimal path planning method based on CAPSO is obtained. In the first section, the experimental environment and parameter settings are given. In the second section, the performances of CAPSO algorithms with different chaotic maps were tested. In the second section, the performance of the CAPSO algorithm for selecting chaotic optimal substitution and traditional heuristic algorithms was tested. In the third section, the performance of the CAPSO algorithm for selecting chaotic optimal replacement and improved PSO algorithms was tested.

6.1. Experimental Environment and Parameter Settings

The environment or workplace was two-dimensional. In order to get an unbiased comparison of CPU time, all experiments were performed on the same PC. In the environment, the domains of \( x \) and \( y \) were between 0 and 15. Meanwhile, the positions of the start point and the endpoint were (1.5, 6.5) and (7.2, 1.2), respectively, so the straight-line distance between the start point and the endpoint is 7.7833. There are five circular obstacles scattered in the environment. They are located on \([x_\text{obs}_i, y_\text{obs}_i]\), where \( x_\text{obs}_1 = [21.2, 36.5, 50.0, 60.0] \) and \( y_\text{obs}_1 = [5.2, 2.5, 4.5, 2.0] \). The radius of the obstacles is set to \([0.7, 0.7, 1.0, 0.7]\).

In environment 2, the domains of \( x \) and \( y \) were between \(-5\) and 30. Meanwhile, the positions of the start point and the endpoint were (0, 0) and (20, 20), respectively, so the straight-line distance between the start point and the endpoint is 28.2843. There are nine rectangular obstacles scattered in the environment. Their outer circle is located on \([x_\text{obs}_i, y_\text{obs}_i]\), where \( x_\text{obs}_1 = [344.4571011.515.517] \) and \( y_\text{obs}_1 = [31795125916] \). The radius of the obstacles is set to \([1.1423 \ 0.5356 \ 1.5811 \ 1.4141 \ 4.4721 \ 1.1180 \ 1.1180 \ 2.2360]\).

In environment 3, we set up 100 random collision-free obstacles. The domains of \( x \) and \( y \) were between \(-200\) and 1300. Meanwhile, the positions of the start point and the endpoint were (0, 0) and (1000, 1000), respectively, so the straight-line distance between the start point and the endpoint is 1414.21365.

In environment 4, the domains of \( x \) and \( y \) were between \(-4\) and 24. Meanwhile, the positions of the start point and the endpoint were (0, 0) and (12, 13), respectively, so the straight-line distance between the start point and the endpoint is 17.6918. There are nine rectangular obstacles scattered in the environment. Their outer circle is located on \([x_\text{obs}_i, y_\text{obs}_i]\), where \( x_\text{obs}_1 = [2257799.510.5] \) and \( y_\text{obs}_1 = [2862.7510.58110.5] \). The radius of the obstacles is set to \([1.4142 \ 3.3536 \ 1.5811 \ 1.4141 \ 4.4721 \ 1.1180 \ 1.1180 \ 2.2360]\).

In environment 5, first, we set up 50 random collision-free obstacles and then randomly selected random obstacles to make them disappear. The positions of \( x \) and \( y \) were between 50 and 1100. Meanwhile, the positions of the start point and the endpoint were (0, 0) and (1000, 1000), respectively, so the straight-line distance between the start point and the endpoint is 1414.21365.

In environment 6, first, we set up four static obstacles and four dynamic obstacles which move in a straight line, respectively. The positions of \( x \) and \( y \) were between 0 and 12. Meanwhile, the positions of the start point and the endpoint were (0, 0) and (12, 10), respectively, so the straight-line distance between the start point and the endpoint is 15.6205.

In environment 7, we set up 30 random collision-free obstacles and then randomly selected random obstacles to make them disappear. Besides, the destination vertex is randomly transformed three times during the algorithm run.

The parameters of the CAPSO algorithm are as follows: the number of particles is \( \text{Pop} \), which is set to 30. The maximum velocity of the particles is \( \text{Vel}_{\text{Max}} \), which is set to 5.12. The minimum velocity \( \text{Vel}_{\text{Min}} \) is set to \(-5.12\). \( c \) is set to 20, and the step is set to 200; Step 0 and Step 1 are set to 0.29 and 0.24, respectively.

6.2. First Experiment: Comparison between Different CAPSO Algorithms

In this section, three groups of test functions with different characteristics are used to benchmark the performance of the proposed different CAPSO algorithms, which are unimodal functions [46, 47], multimodal functions [48, 49], and fixed-dimension multimodal functions [50, 51]. The specific form of the function is given in Table 1, where \( \text{Dim} \) represents the dimension of the function. Range represents the range of independent variables, that is, the range of population, and \( f_{\text{min}} \) represents the minimum value of the function.

Figure 5 shows the two-dimensional versions of a unimodal function, multimodal function, and fixed-dimension multimodal function, respectively. The unimodal test function has only one global optimal solution, which is helpful to find the global optimal solution in the search space, and it can test the convergence speed and efficiency of the algorithm thoroughly, while the multimodal function and the fixed-dimensional multimodal test function have multiple local optimal solutions, which can be used to test the algorithm to avoid the performance of the optimal local solution, and the fixed-dimensional multimodal function compared with unimodal test function is more challenging. Convergence curves of CAPSO-Kent, CAPSO-Singer, and CAPSO-Logistic are compared in Figure 6 for all of the test functions. The figure shows that CAPSO-Kent has good processing ability for unimodal functions, multimodal functions, and fixed-dimension functions, and the processing process is very stable. Especially when solving more complex fixed-dimension...
functions, CAPSO-Kent shows a more obvious advantage than other algorithms. Therefore, we choose the Kent map to represent \( r_1 \) and \( r_2 \).

### 6.3. Second Experiment: Different Numbers of Control Points.

This experiment aims to test the influence of the number of control points on the proposed algorithm CAPSO with Kent map. In this experiment, the number of control points ranged from one to ten control points. The population size was 30. Figure 7 shows the results of experiment 2. Also, Figure 8 shows the convergence curve of the CAPSO algorithm when the number of control points ranged from one to nine and when the population size was 30.

As shown, the best results were achieved when the number of control points was small, i.e., \( n = 3 \) or \( 4 \), and the results dramatically decreased when the value of \( n \) was more than five, and the worst results were achieved when \( n = 8 \) and \( n = 9 \). The larger the number of control points, the larger the search space. Hence, the larger population is required to obtain good results. Also, when the number of control points is small, too few turns of the curve will limit the choice of paths such as \( n = 1 \) or \( n = 2 \).

Therefore, three to six control points are relatively mature. When the number of obstacles increases sharply, the number of control points can be increased appropriately.

### 6.4. Third Experiment: CAPSO versus Traditional Heuristic Algorithms in Path Planning.

This section tested the path planning of robots in environment 2 based on GA [9], SA [12], PSO [13], and CAPSO algorithms. For a fair comparison, the population size in all algorithms, the number of control points, and the maximum number of iterations are set to 30, 3, and 500, respectively. Each experiment ran 15000 objective function evaluations, equivalent to 500 iterations using 30 population sizes. For each algorithm, a total of 25 runs were performed for each experiment. Figure 9(a) shows the best paths in environment 2 generated by the CAPSO, PSO, GA, DE, and SA during 25 trials. The yellow square and green pentacle in Figure 9(a) are the start point and endpoint of the path, respectively. Also, Figure 9(b) shows the convergence curves of the best fitness values during 25 trials.

Based on the three factors of solution quality, stability, and convergence speed, the performances of these algorithms
Figure 6: (a–f) Comparison of convergence curves of different CAPSO algorithms obtained in the benchmark function ($F_1$–$F_6$).
The solution quality can be expressed by the average optimal fitness value, which is defined as the average of 25 optimal fitness values produced by 25 trials. The smaller the average optimal fitness value is, the higher the solution quality is. The stability of the algorithm is determined by the standard deviation. The smaller the standard deviation is, the more stable the algorithm is. The convergence speed of the algorithm is determined by the number of iterations required for the algorithm to converge to the optimal or suboptimal solution. Given the maximum number of iterations, the fewer iterations the algorithm converge to the optimal solution, the faster the convergence speed.

As we can see, although path planners based on five algorithms can successfully generate a collision-avoidance path, CAPSO found the near-optimal path and performs better. The path planners based on DE and SA have slow convergence speed. In contrast, the path planners based on PSO, CAPSO, and GA have fast convergence speed. Moreover, the solution quality of path planners based on DE and SA is inferior to that based on PSO, CAPSO, and GA.

In 25 trials, the complete performance analysis for the best performance of different algorithms in path planning is presented in Table 2. From the statistical data in Table 2, it can be seen that the CAPSO-based path planner has the smallest average fitness value, execution time, and standard deviation, which further proves the high searchability, simplicity, and robustness of CAPSO. In addition, the number of optimal iterations of CAPSO is the smallest, which further proves the rapid convergence of CAPSO. Compared with other algorithms, the CAPSO algorithm has the shortest execution time. From the six statistics listed in Tables 3 and 4, we proved the high searchability, simplicity, and robustness of CAPSO. Also,
CAPSO is not time-consuming. All in all, the performance of CAPSO is better than the other five tested algorithms.

6.5. Fourth Experiment: CAPSO versus Improved PSO Algorithms in Path Planning. The experiment was divided into two parts according to two types of environments (environment 3 and environment 4). This section tested the path planning of robots based on FAPSO [29], IWPSO [28], DEPSO [30], SPSO [35], MPSO [37], and CAPSO algorithms.
Example 1. In this example, the population size in all algorithms, the number of control points, and the maximum number of iterations are set to 30, 7, and 100, respectively. Each experiment ran 3000 objective function evaluations, equivalent to 100 iterations using 30 population sizes. For each algorithm, a total of 25 runs were performed for each experiment. Figure 10(a) shows the best paths in environment 3 generated by the CAPSO (green line), DEPSO (black line), MPSO (red line), SPSO (rose red line), FAPSO (blue line), and IWPSO (yellow line) during 25 trials. The yellow square and green pentacle in Figure 10(a) are the start point and endpoint of the path, respectively. Also, Figure 10(b) shows the convergence curves of the best fitness values during 25 trials.

As shown in Figures 10 and 11, although path planners based on six algorithms can successfully generate a collision-avoidance path, CAPSO found the near-optimal path and performs better. Also, all paths are smooth and there are no large corners.

In 25 trials, the mean fitness value and standard deviation in environment 3 and environment 4 are shown in Tables 3 and 4. From the statistical data in Tables 3 and 4, we can see that the CAPSO-based path planner has the smallest average fitness value, execution time, and standard deviation, which further proves the high searchability, simplicity, and robustness of CAPSO. In addition, the number of optimal iterations of CAPSO is the smallest, which further proves the rapid convergence of CAPSO. Also, CAPSO is the algorithm with the shortest time compared with other algorithms. From the six statistics listed in Tables 3 and 4, we proved the high searchability, simplicity, and robustness of CAPSO. Also, CAPSO is not time-consuming. All in all, the performance of CAPSO is better than the other five tested algorithms.

Example 2. In this example, the population size in all algorithms, the number of control points, and the maximum number of iterations are set to 30, 4, and 200, respectively. Each experiment ran 6000 objective function evaluations, equivalent to 200 iterations using 30 population sizes. For each algorithm, a total of 25 runs were performed for each experiment. Figure 11(a) shows the best paths in environment 4 generated by the CAPSO (green line), DEPSO (black line), MPSO (red line), SPSO (rose red line), FAPSO (blue line), and IWPSO (yellow line) during 25 trials. The yellow square and green pentacle in Figure 11(a) are the start point and endpoint of the path, respectively. Also, Figure 11(b) shows the convergence curves of the best fitness values during 25 trials.

As we can see, the overall path is globally optimal; at the same time, the local path is smoother at each stage, the corner is small and gentle, and it conforms to the mobile control of the robot, and there is no obstacle unobstructed near the target point.

### Table 3: Path planning results in environment 3.

| Algorithms | Best fitness value | Mean fitness value | Worst fitness value | Average execution time | Standard deviation | Number of optimal iterations |
|------------|--------------------|--------------------|---------------------|------------------------|-------------------|----------------------------|
| CAPSO      | 1414.43            | 1429.83            | 1416.24             | 15.1812                | 0.9064            | 46                         |
| DEPSO      | 1695.77            | 1731.87            | 1697.76             | 41.0869                | 0.9968            | 73                         |
| FAPSO      | 1726.20            | 1823.28            | 1728.99             | 19.0295                | 1.3953            | 35                         |
| SPSO       | 1722.87            | 1802.55            | 1725.90             | 16.9308                | 1.5131            | 45                         |
| IWPSO      | 1814.76            | 1888.91            | 1816.82             | 26.6719                | 1.0275            | 74                         |
| MPSO       | 1610.44            | 1710.44            | 1613.83             | 19.4110                | 1.7239            | 80                         |

### Table 4: Path planning results in environment 4.

| Algorithms | Best fitness value | Mean fitness value | Worst fitness value | Average execution time | Standard deviation | Number of optimal iterations |
|------------|--------------------|--------------------|---------------------|------------------------|-------------------|----------------------------|
| CAPSO      | 18.4155            | 19.7854            | 20.0875             | 7.1129                 | 0.8360            | 46                         |
| DEPSO      | 19.0657            | 20.8703            | 20.8497             | 15.9505                | 0.8920            | 39                         |
| FAPSO      | 20.5181            | 22.4203            | 22.5167             | 13.9275                | 0.9993            | 35                         |
| SPSO       | 20.5665            | 22.8267            | 22.7347             | 9.4258                 | 1.0841            | 41                         |
| IWPSO      | 20.3603            | 21.5655            | 22.3353             | 9.1382                 | 0.9875            | 42                         |
| MPSO       | 18.8765            | 20.5745            | 21.2343             | 7.7203                 | 1.1789            | 21                         |

6.6. Fifth Experiment: CAPSO Algorithms in Dynamic Path Planning under Environment 5. This section tested the feasibility of the proposed model in the dynamic environment 5. Figures 12(a)–12(d) show the best paths in environment 5 generated by the CAPSO. Figure 13 shows the convergence curves of the best fitness values.

As we can see, the overall path is globally optimal; at the same time, the local path is smoother at each stage, the corner is small and gentle, and it conforms to the mobile control of the robot, and there is no obstacle unobstructed near the target point.

6.7. Sixth Experiment: CAPSO Algorithms in Dynamic Path Planning under Environment 6. This section tested the feasibility of the proposed model in dynamic environment 6, which contains four static obstacles and four dynamic obstacles. Static obstacles are represented by filling graphs, while dynamic obstacles are represented by hollow blue circles. Figures 14(a)–14(f) show the best paths in environment 6 generated by CAPSO.

As we can see, the overall path is globally optimal; at the same time, the local path is smoother at each stage, the corner is small and gentle, and it conforms to the mobile control of the robot, and there is no obstacle unobstructed near the target point.
Figure 10: (a) Comparison of path planning results in environment 3. (b) Convergence curves of different algorithms in environment 3.

Figure 11: (a) Comparison of path planning results in environment 4. (b) Convergence curves of different algorithms in environment 4.
6.8. Sixth Experiment: CAPSO Algorithms in Dynamic Path Planning under Environment 7. This section tested the feasibility of the proposed model in dynamic environment 7, which contains four dynamic obstacles and dynamic object vertex. Obstacles are represented by hollow blue and red circles. Figures 15(a)–15(d) show the best paths in environment 7 generated by CAPSO.

Figure 12: (a–d) Dynamic Path planning results in environment 5.

Figure 13: Convergence curves of CAPSO algorithms in environment 5.
Figure 14: (a–f) Dynamic path planning results in environment 6.

Figure 15: Continued.
As we can see, the overall path is globally optimal; at the same time, the local path is smoother at each stage, the corner is small and gentle, and it conforms to the mobile control of the robot, and there is no obstacle unobstructed near the target point.

7. Conclusions

This paper used the chaotic adaptive particle swarm optimization (CAPSO) algorithm to present a novel algorithm which is used to optimize Cubic spline interpolation to get the shortest, smooth, and collision-free path. The main improvement of this algorithm is illustrated below. The CAPSO is an improved PSO algorithm. It introduces the beetle foraging strategy to modify the particle position updating equation to strengthen the global search ability of the algorithm. Then, the three control parameters in the PSO algorithm are adaptively adjusted by using the trigonometric function to make CAPSO adaptively adapt control parameters between linearly decreasing and nonlinearly decreasing strategies so that control parameters of CAPSO reach the optimal level in the iterative process. Also, the chaotic map is used to replace the random parameter of the PSO to improve the diversity of particle swarm and maintain the original random characteristics. Experiments show that the Kent map is the most suitable. With CAPSO, the global search ability and search speed are improved, and optimal robot path planning under static environment is realized. Moreover, experiments verified the feasibility of the proposed model in different dynamic environments. The simulation results proved the effectiveness and robustness of the proposed algorithm.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This research was funded by the National Natural Science Foundation (grant nos. 61602529 and 61672539), Planned Science and Technology Project of Hunan Province (grant no. 603286580059), and Scientific Innovation Fund for Graduates of Central South University of Forestry and Technology (grant no. CX20192077).

References

[1] F. Bayat, S. Najafinia, and M. Aliyari, “Mobile robots path planning: electrostatic potential field approach,” Expert Systems with Applications, vol. 100, pp. 68–78, 2018.
[2] X. Liu, L. Wang, D. Wang et al., “Research on substation PD monitoring method based on patrol robot,” in Proceedings of the 2019 International Conference on High Performance Big Data and Intelligent Systems (HPBDe-IS), Shenzhen, China, May 2019.
[3] C. Feng, X. Sun, and D. Wei, “Inertia weight particle swarm optimization with Boltzmann exploration,” in Proceedings of the 2011 Seventh International Conference on Computational Intelligence and Security, Sanya, Hainan, China, December 2011.
[4] T. M. Cabreira, C. D. Franco, P. R. Ferreira, and G. C. Buttazzo, “Energy-aware spiral coverage path planning for uav photogrammetric applications,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3662–3668, 2018.
[5] J. Lian, W. Yu, and W. Liu, “A chaotic adaptive particle swarm optimization for robot path planning,” in Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, July 2019.
[6] L. K. Panwar, S. Reddy K, A. Verma, B. K. Panigrahi, and R. Kumar, “Binary grey wolf optimizer for large scale unit commitment problem,” Swarm and Evolutionary Computation, vol. 38, pp. 251–266, 2018.
[7] PanwarL. Kumar et al., “Optimal schedule of plug in electric vehicles in smart grid with constrained parking lots,” in Proceedings of the 6th International Conference on Power Systems (ICPS), New Delhi, India, March 2016.
[8] M. Wang, J. Luo, J. Fang, and J. Yuan, “Optimal trajectory planning of free-floating space manipulator using differential evolution algorithm,” Advances in Space Research, vol. 61, no. 6, pp. 1525–1536, 2018.
[9] R. Roy, M. Mahadevappa, and C. S. Kumar, "Trajectory path planning of eeg controlled robotic arm using GA," *Procedia Computer Science*, vol. 84, pp. 147–151, 2016.

[10] F. Duchoň, A. Babinec, M. Kajan et al., "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[11] H. Liu, B. Xu, D. Lu, and G. Zhang, "A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm," *Applied Soft Computing*, vol. 68, pp. 360–376, 2018.

[12] R. S. Tavares, T. C. Martins, and M. S. G. Tsuzuki, "Simulated annealing with adaptive neighborhood: a case study in off-line robot path planning," *Expert Systems with Applications*, vol. 38, no. 4, pp. 2951–2965, 2011.

[13] H.-T. Hsieh and C.-H. Chu, "Improving optimization of tool path planning in 5-axis flank milling using advanced PSO algorithms," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 3, pp. 3–11, 2013.

[14] S. ª Guëlç, M. Mahi, ªO. 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 Computing*, vol. 22, no. 5, pp. 1669–1685, 2018.

[15] M. Elhoseny, A. Harwat, A. E. Hassanien, and Aboul Ella Hassanien, "A new hybrid method based on modified genetic algorithm," *Journal of Computational Science*, vol. 25, pp. 339–350, 2018.

[16] J. Yao, "Path planning for virtual human motion using improved A* star algorithm," in *Proceedings of the Seventh International Conference on Information Technology: New Generations*, Washington, DC, USA, April 2010.

[17] M. Mahi, O. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, 2015.

[18] M. Zangari, A. Pozo, R. Santana, and A. Mendiburu, "A decomposition-based binary ACO algorithm for the multi-objective UBQP," *Neurocomputing*, vol. 246, pp. 58–68, 2017.

[19] S. Yu, J. Zhang, S. Zheng, and H. Sun, "Provincial carbon intensity abatement potential estimation in China: a PSO-GA-optimized multi-factor environmental learning curve method," *Energy Policy*, vol. 77, pp. 46–55, 2015.

[20] U.-Y. Huh and S.-R. Chang, "AG 2 continuous path-smoothing algorithm using modified quadratic polynomial interpolation," *International Journal of Advanced Robotic Systems*, vol. 11, no. 25, p. 2, 2014.

[21] B. Parmentier, B. J. McGill, A. M. Wilson et al., "Using multiscale methods and satellite-derived land surface temperature for the interpolation of daily maximum air temperature in Oregon," *International Journal of Climatology*, vol. 35, no. 13, pp. 3862–3878, 2015.

[22] W. Kang, S. Yu, S. Ko, and J. Paik, "Multisensor super resolution using directionally-adaptive regularization for UAV images," *Sensors*, vol. 15, no. 5, pp. 12053–12079, 2015.

[23] M. Gülmü, M. Kadir Yesilyurt, and A. Bilgin, "The performance assessment of cubic spline interpolation and response surface methodology in the mathematical modeling to optimize biodiesel production from waste cooking oil," *Fuel*, vol. 255, p. 115778.

[24] W. Deng, R. Yao, H. Zhao, X. Yang, and G. Li, "A novel intelligent diagnosis method using optimal LS-SVM with improved PSO algorithm," *Soft Computing*, vol. 23, no. 7, pp. 2445–2462, 2019.

[25] Y. Gao, "A hybrid method for mobile agent moving trajectory scheduling using ACO and PSO in WSNs," *Sensors*, vol. 19, no. 3, 2019.

[26] J. Wang, "A PSO based energy efficient coverage control algorithm for wireless sensor networks," *Computers, Materials and Continua*, vol. 56, no. 3, pp. 433–446, 2018.

[27] C. Sahu, P. B. Kumar, and D. R. Parhi, "An intelligent path planning approach for humanoid robots using adaptive particle swarm optimization," *International Journal on Artificial Intelligence Tools*, vol. 27, no. 5, 2018.

[28] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm," *Chaos, Solitons & Fractals*, vol. 37, no. 3, pp. 698–705, 2008.

[29] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic," *Expert Systems with Applications*, vol. 40, no. 8, pp. 3196–3206, 2013.

[30] T. d. F. Araújo and W. Uturkey, "Performance assessment of PSO, DE and hybrid PSO-DE algorithms when applied to the dispatch of generation and demand," *International Journal of Electrical Power & Energy Systems*, vol. 47, pp. 205–217, 2013.

[31] T. Y. Abdalla, A. A. Abed, and A. A. Ahmed, "Mobile robot navigation using PSO-optimized fuzzy artificial potential field with fuzzy control," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 6, pp. 3893–3908, 2017.

[32] A. Benvidi, S. Abbasi, S. Ghahramani, M. Dehghan Tezerjani, and S. Masoum, "Spectrophotometric determination of synthetic colorants using PSO-GA-ANN," *Food Chemistry*, vol. 220, pp. 377–384, 2017.

[33] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[34] W. Ding, C. T. Lin, M. Prasad, Z. Cao, and J. Wang, "A layered-coevolution-based attribute-boosted reduction using adaptive quantum-behavior PSO and its consistent segmentation for neonates brain tissue," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1177–1191, 2017.

[35] X. Kong, L. Gao, H. Ouyang, and S. Li, "Solving the redundancy allocation problem with multiple strategy choices using a new simplified particle swarm optimization," *Reliability Engineering & System Safety*, vol. 144, pp. 147–158, 2015.

[36] A. A. Nagra, F. Han, Q.-H. Ling, and S. Mehta, "An improved hybrid method combining gravitational search algorithm with dynamic multi-swarm particle swarm optimization," *IEEE Access*, vol. 7, pp. 50384–50399, 2019.

[37] F. Q. Lu, M. Huang, W. K. Ching, X. W. Wang, and X. L. Sun, "Multi-swarm particle swarm optimization based risk management model for virtual enterprise," in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 387–392, Shanghai, China, July 2019.

[38] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Applied Intelligence*, vol. 48, no. 10, pp. 3462–3481, 2018.

[39] S. Wang, Y. Zhang, G. Ji, J. Yang, J. Wu, and L. Wei, "Fruit classification by wavelet-entropy and feedforward neural network trained by fitness-scaled chaotic abc and biogeography-based optimization," *Entropy*, vol. 17, no. 8, pp. 5711–5728, 2015.
optimization techniques for environmental opf problems in the power system,” *Chaos, Solitons & Fractals*, vol. 69, no. 69, pp. 271–284, 2014.

[41] H. L. Bian, “Study on chaotic time series prediction algorithm for kent mapping based on particle swarm optimization,” *Applied Mechanics and Materials*, vol. 511-512, pp. 941–944, 2014.

[42] J. Cao, “A novel fixed point feedback approach studying the dynamical behaviors of standard logistic map,” *International Journal of Bifurcation and Chaos*, vol. 29, no. 1, 2019.

[43] E. Farri and P. Ayubi, “A blind and robust video watermarking based on iwt and new 3d generalized chaotic sine map,” *Nonlinear Dynamics*, vol. 93, no. 4, pp. 1875–1897, 2018.

[44] X. Jiang and S. Li, “BAS: beetle antennae search algorithm for optimization problems,” 2017, http://arxiv.org/abs/1710.10724.

[45] R. Jensii and G. W. Jiji, ”An enhanced particle swarm optimization with levy flight for global optimization,” *Applied Soft Computing*, vol. 43, pp. 248–261, 2016.

[46] M. R. Tahir, Q. X. Tran, and M. S. Nikulin, “Comparison of hypertabastic survival model with other unimodal hazard rate functions using a goodness-of-fit test,” *Statistics in Medicine*, vol. 36, no. 12, pp. 1936–1945, 2017.

[47] M. Parashar, “Optimization of benchmark functions using A nature inspired bird Swarm algorithm,” in *Proceedings of the International Conference on Computational Intelligence & Communication Technology*, pp. 9-10, Ghaziabad, India, February 2017.

[48] B. K. Hassani, ”Distortion risk measures or the transformation of unimodal distributions into multimodal functions,” *Documents De Travail Du Centre Deconomie De La Sorbonne*, vol. 211, no. 5, pp. 71–88, 2015.

[49] B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, and P. N. Suganthan, “Novel benchmark functions for continuous multimodal optimization with comparative results,” *Swarm and Evolutionary Computation*, vol. 26, pp. 23–34, 2016.

[50] R. Moghdani and K. Salimifard, “Volleyball premier league algorithm,” *Applied Soft Computing*, vol. 64, pp. 161–185, 2018.

[51] S. Reddy, ”Solution to unit commitment in power system operation planning using binary coded modified moth flame optimization algorithm (BMMFOA): a flame selection based computational technique,” *Journal of Computational Science*, vol. 25, pp. 298–317, 2015.