10.24425/acs.2024.149659

Archives of Control Sciences
Volume 34(LXX), 2024
No. 1, *pages 225–242*

Novel path planning method using marine predator algorithm for mobile robot

Qiang WANG and Yinghui HUANG

The main goal of robot path planning is to design an optimal path for a robot to navigate from its starting point to its goal while avoiding obstacles and optimizing certain criteria. A novel method using marine predator algorithm which is used in the field of robot path planning is presented. The proposed method has two steps. First step is to build a mathematical model of path planning while second step is optimization process using marine predator algorithm. Simulation results show that the proposed method works well and has good performance in different situations. Therefore, this method is an effective method for robot path planning and related applications.

Key words: robot path planning, marine predator algorithm, metaheuristic, optimization, autonomous driving

1. Introduction

Robot path planning is a field of study in robotics that deals with finding the optimal path for a robot to navigate from its starting point to its goal while avoiding obstacles and optimizing certain criteria such as time, energy consumption, or safety [1]. It plays a critical role in the development of autonomous robots that can operate in complex and dynamic environments. Robot path planning enables robots to avoid obstacles, navigate around complex structures, and optimize their movements to conserve energy and time. In addition, robot path planning also enhances safety by minimizing the risk of collisions and accidents [2]. In this

Q. Wang (corresponding author, e-mail: wq@bbc.edu.cn) is with College of Electronic and Electrical Engineering, Bengbu University, Bengbu 233030, China.

Y. Huang is with College of Computer and Information Engineering, Bengbu University, Bengbu 233030, China.

The generous support of Anhui Science and Technology Research Project (2022AH040255), Bengbu Science and Technology Research Plan Project (2022gx23), and scientific research project of Bengbu University (2021ZR04zd) are gratefully acknowledged.

Received 28.07.2023. Revised 4.02.2024.

paper, a novel robot path planning method using marine predator algorithm [3] is proposed, which can solve the shortest path planning in any scenes with the low computation complexity. This method can be applied to the automatic routing design of robot application.

In recent years, the various algorithms are proposed in robot path planning and their applications in different domains. In general, most proposed algorithms are classified into several classes of methods, such as heuristic search-based methods, potential field-based methods, sampling-based methods, hybrid methods, and evolution methods. Heuristic search-based methods [4] are widely used in robot path planning due to their efficiency and effectiveness in finding optimal paths. These methods use a heuristic function to estimate the distance from a given state to the goal and guide the search towards the optimal path. Examples of heuristic search algorithms used in robot path planning include A* [5–7], Dijkstra's algorithm [8, 9], and breadth-first search [10]. These methods have been applied in various applications such as autonomous driving, mobile robots, and unmanned aerial vehicles (UAVs) [11, 12].

Potential field-based methods [13] use attractive and repulsive forces to guide the robot towards its goal while avoiding obstacles. These methods have been widely used in robot path planning due to their simplicity and computational efficiency. However, potential field-based methods can be prone to getting stuck in local minima, and the performance can be sensitive to the choice of potential field parameters. Examples of potential field-based methods include the artificial potential field method [13], the virtual potential field method [14], and the elastic bands method [15]. These methods have been applied in various applications such as mobile robots, UAVs, and underwater robots.

Sampling-based methods [16] generate a roadmap of the environment by randomly sampling points and connecting them with feasible paths. These methods have become increasingly popular in robot path planning due to their ability to handle high-dimensional spaces and environments with complex geometries. Examples of sampling-based methods include the probabilistic roadmap method (PRM) [17, 18], the rapidly exploring random tree (RRT) method [19], and its variants such as RRT* [20] and RRT*-Smart [21]. These methods have been applied in various applications such as mobile robots, UAVs, and robotic manipulation.

Hybrid methods [22] combine multiple approaches to overcome the limitations of individual methods and improve their performance. These methods have become increasingly popular in robot path planning due to their ability to handle complex environments and dynamic obstacles. Examples of hybrid methods include the APRM [23], the anytime dynamic A algorithm [24], and the sampling-based roadmap [25]. These methods have been applied in various applications such as autonomous driving, mobile robots, and UAVs.

Evolutionary methods [26] are a class of optimization algorithms that simulate the process of natural selection to find optimal solutions to complex problems, including robot path planning. In evolutionary methods, a population of potential solutions is generated and iteratively evolved through a process of selection, mutation, and reproduction. Each solution is evaluated based on a fitness function that measures its performance in achieving the desired objectives, such as minimizing the path length or avoiding obstacles. One popular evolutionary method for robot path planning is the genetic algorithm (GA) [27], which is based on the principles of genetics and evolution. In a GA, a population of potential paths is generated randomly, and each path is represented as a chromosome. The fitness of each chromosome is evaluated using the fitness function, and the fittest chromosomes are selected for reproduction. The reproduction process involves crossover and mutation, where parts of the fittest chromosomes are combined to generate new offspring, which are then evaluated and added to the population. This process is repeated for several generations until an optimal solution is found. Another evolutionary method for robot path planning is the particle swarm optimization (PSO) [28–30], which is based on the behavior of bird flocks or fish schools. In a PSO, a swarm of particles is used to represent the potential paths, and each particle moves in the search space according to its velocity and the position of the fittest particle in the swarm. The fitness of each particle is evaluated using the fitness function, and the particle swarm is iteratively updated until an optimal solution is found. In addition, Grey Wolf Optimization (GWO) is applied for global path planning which is used in autonomous underwater vehicle (AUV) [31]. Evolutionary methods for robot path planning have several advantages. They can handle complex environments with non-convex shapes and dynamic obstacles, and they can generate multiple solutions that satisfy different objectives simultaneously. They can also handle high-dimensional search spaces and are less sensitive to the choice of parameters. However, evolutionary methods can be computationally expensive and may require significant computing resources, especially for large-scale problems. As an evolutionary algorithm, Marine Predator Algorithm (MPA) [3] is a nature-inspired optimization algorithm that is based on the predatory behavior of marine predators such as sharks and dolphins. The algorithm has been successfully applied to a variety of optimization problems, including feature selection, data clustering, and function optimization. MPA is selected as the optimization method to realize robot path planning in this paper.

2. Marine predator algorithm

The MPA algorithm simulates the behavior of marine predators in a virtual environment [32], where each predator represents a candidate solution to an

optimization problem. The hunting behavior of the predators is modeled based on the following three strategies. At first, the predator moves randomly in the search space to explore new areas. Then the predator moves towards the position of the prey to catch it. Finally, the predator waits in a specific location for the prey to come close and then attacks it. In the survival of the fittest theory, the top predators are more talented in foraging in nature. So, the fittest solution is defined as a top predator to construct an elite matrix. Arrays of this matrix manage searching and finding the prey based on the information on prey's positions. The elite matrix is defined as follows:

$$\boldsymbol{Elite} = \begin{bmatrix} X^I_{1,1} & X^I_{1,2} & \cdots & X^I_{1,d} \\ X^I_{2,1} & X^I_{2,2} & \cdots & X^I_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X^I_{n,1} & X^I_{n,2} & \cdots & X^I_{n,d} \end{bmatrix}_{n\times d}, \tag{1}$$

where n is the number of search agents, d is the number of dimensions, $\boldsymbol{X^I}$ is the top predator, which is replicated n times to construct the elite matrix, respectively. Furthermore, another $n \times d$ matrix is defined as prey matrix which the predators update their positions based on it. The prey matrix is shown as follows:

$$\boldsymbol{Prey} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}_{n\times d}, \tag{2}$$

where $X_{i,j}$ presents the j-th dimension of i-th prey. In MPA, the whole process of the optimization is related to elite matrix and prey matrix.

Like most metaheuristics, MPA is a population-based approach, in which initial solutions are uniformly distributed in the search space as first trials as follows:

$$X_0 = X_{\min} + rand\,(X_{\max} - X_{\min}), \tag{3}$$

where $X_{\min}$ and $X_{\max}$ are the lower and upper bound for variables and *rand* is a uniform random vector in the range of 0 to 1. The same as the hunting behavior of the predators in nature, the process of MPA is divided into three stages of optimization considering different velocity ratio and at the same time mimicking the entire life of a predator and prey.

In the first stage, when predator is moving faster than prey, the exploration is important. The mathematical model of this rule is applied as:

$$\begin{aligned} &\text{While } \ Iter < \frac{1}{3}\max\_Iter \\ &\boldsymbol{stepsize}_i = \boldsymbol{R}_B \otimes \left(\boldsymbol{Elite}_i - \boldsymbol{R}_B \otimes \boldsymbol{Prey}_i\right), \quad i = 1, 2, \ldots, n \\ &\boldsymbol{Prey}_i = \boldsymbol{Prey}_i + P \cdot \boldsymbol{R} \otimes \boldsymbol{stepsize}_i, \end{aligned} \tag{4}$$

where $\boldsymbol{R}_B$ is a vector containing random numbers based on normal distribution representing the Brownian motion, P is 0.5, $\boldsymbol{R}$ is a vector of uniform random numbers in $[0,\ 1]$, respectively. The notation $\otimes$ shows entry-wise multiplications. *Iter* is the current iteration while *Max_iter* is the maximum one.

In the second stage, when both predator and prey are moving at the same pace, it simulates that both are looking for their preys. Prey is responsible for exploitation while predator is responsible for exploration. So, both exploration and exploitation are important in this stage. Based on the rule, if prey moves in Lévy which is a special class of random walks [33], the best strategy for predator is Brownian. Thus, this study considers prey moves in Lévy while predator moves in Brownian. The mathematical model of this rule is applied as:

$$\begin{aligned} &\text{While } \ \frac{1}{3}\max\_Iter < Iter < \frac{2}{3}\max\_Iter \\ &\text{For the first half of population} \\ &\boldsymbol{stepsize}_i = \boldsymbol{R}_L \otimes \left(\boldsymbol{Elite}_i - \boldsymbol{R}_L \otimes \boldsymbol{Prey}_i\right), \quad i = 1, 2, \ldots, n/2 \\ &\boldsymbol{Prey}_i = \boldsymbol{Prey}_i + P \cdot \boldsymbol{R} \otimes \boldsymbol{stepsize}_i, \\ &\text{For the second half of population} \\ &\boldsymbol{stepsize}_i = \boldsymbol{R}_B \otimes \left(\boldsymbol{R}_B \otimes \boldsymbol{Elite}_i - \boldsymbol{Prey}_i\right), \quad i = n/2, \ldots, n \\ &\boldsymbol{Prey}_i = \boldsymbol{Elite}_i + P \cdot CF \otimes \boldsymbol{stepsize}_i, \end{aligned} \tag{5}$$

where $\boldsymbol{R}_L$ is a vector of random numbers based on Lévy distribution representing Lévy movement, P is 0.5, $\boldsymbol{R}$ is a vector of uniform random numbers in [0, 1], $\boldsymbol{R}_B$ is a vector containing random numbers based on Normal distribution representing the Brownian motion, $CF = \left(1 - \dfrac{Iter}{\max\_Iter}\right)^{\left(2\frac{Iter}{\max\_Iter}\right)}$ is considered as an adaptive parameter to control the step size for predator movement, respectively.

In the last stage, when predator is moving slower than prey, the best strategy for predator is Lévy. The mathematical model of this rule is applied as:

$$\begin{aligned} &\text{While } \ Iter > \frac{2}{3}\max\_Iter \\ &\boldsymbol{stepsize}_i = \boldsymbol{R}_L \otimes \left(\boldsymbol{R}_L \otimes \boldsymbol{Elite}_i - \boldsymbol{Prey}_i\right), \quad i = 1, 2, \ldots, n \\ &\boldsymbol{Prey}_i = \boldsymbol{Elite}_i + P \cdot CF \otimes \boldsymbol{stepsize}_i, \end{aligned} \tag{6}$$

where $\boldsymbol{R}_L$ is a vector of random numbers based on Lévy distribution representing Lévy movement, P is 0.5, $CF = \left(1 - \dfrac{Iter}{\max\_Iter}\right)^{\left(2\frac{Iter}{\max\_Iter}\right)}$, respectively.

The process of MPA is described as follows:

Step 1: Set the algorithm parameters and initialize the population.

Step 2: Calculate the fitness value and record the optimal position.

Step 3: The predator selects the corresponding update method from Eqs. 4–6 according to the iteration stage and updates the position of the predator.

Step 4: Calculate the fitness value and update the optimal position.

Step 5: Determine whether the stop condition is satisfied, if not, repeat steps Eqs. 4–6, otherwise output the optimal result of MPA.

The MPA algorithm has several advantages over other optimization algorithms. It is easy to implement, computationally efficient, and can handle both continuous and discrete search spaces. The algorithm is also less sensitive to the choice of parameters and has good convergence properties. However, the performance of the MPA algorithm depends on the specific characteristics of the optimization problem and the available computational resources.

3. Robot path planning method

In our design, a robot moves in a 2D Euclidean workspace $\boldsymbol{W}$ ($\boldsymbol{W} = \boldsymbol{R}^2$) which is populated with rigid obstacles $\boldsymbol{O}$ where $\boldsymbol{O}$ denotes the set of all points in $\boldsymbol{W}$ that are in the obstacles i.e. $\boldsymbol{O} \subseteq \boldsymbol{W}$. The robot is allowed to move in the workspace $\boldsymbol{W}$ but cannot go through the obstacles $\boldsymbol{O}$. Moreover, a start point and a destination point are designated which are located in the workspace $\boldsymbol{W}$. The objective of path planning is to find out a shortest collision-free path between the start point and destination point for the robot while avoiding any contact with the obstacles $\boldsymbol{O}$.

In order to realize the robot path planning,N middle points are defined as $p_1\left(p_{1x}, p_{1y}\right) p_2\left(p_{2x}, p_{2y}\right), \ldots, p_N\left(p_{Nx}, p_{Ny}\right)$, which are randomly designated in the workspace $\boldsymbol{W}$. If the start point is $p_s\left(p_{sx}, p_{sy}\right)$ while the destination point

is $p_d\left(p_{dx}, p_{dy}\right)$, a simple result without the obstacles is shown as follows:

$$L = \sqrt{\left(p_{1x} - p_{sx}\right)^2 + \left(p_{1y} - p_{sy}\right)^2} + \sum_{i=1}^{N-1} \sqrt{\left(p_{(i+1)x} - p_{ix}\right)^2 + \left(p_{(i+1)y} - p_{iy}\right)^2} + \sqrt{\left(p_{dx} - p_{Nx}\right)^2 + \left(p_{dy} - p_{Ny}\right)^2}. \tag{7}$$

Obviously, the result in Eq. (7) is not a good result. To search for the optimal result, there are two problems to be solved firstly. The first one is the path from start point to destination point should be a smooth curve while the second one is all obstacles that must be avoided by the robot. For the first problem, we use the cubic spline function interpolation to smooth the path between the start point and destination point. If the number of interpolation points for the whole planning path are M, the result is shown as follows:

$$L = \sum_{i=1}^{M-1} \sqrt{\left(pp_{(i+1)x} - pp_{ix}\right)^2 + \left(pp_{(i+1)y} - pp_{iy}\right)^2}, \tag{8}$$

where $pp_i\left(pp_{ix}, pp_{iy}\right)$ is the i-th interpolation point.

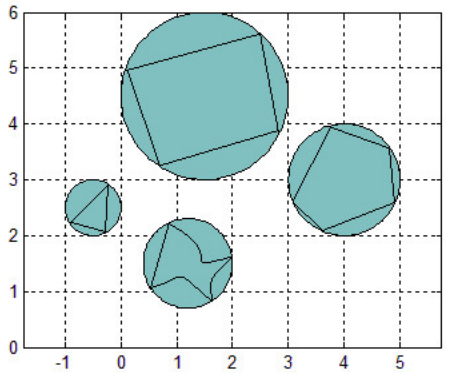


Figure 1: Circumscribed circles of obstacles

For the second problem, if a part of middle points are located in the obstacles, the result is not accepted. If all the middle points are not located in the obstacles, the result is accepted. However, if all the middle points are not located in the obstacles while some interpolation points are located in the obstacles, the result has to be modified. In our method, every independent obstacle area has a circumscribed circle. Fig. 1 shows the circumscribed circles of obstacles. We define center of the i-th circumscribed circle as $ob_i\left(ob_{ix}, ob_{iy}\right)$, radius of the i-th circumscribed circle as ob_{ir}. If interpolation point i is in the obstacle j, we use

the following equations to calculate the violation:

$$d_{ij} = \sqrt{\left(pp_{ix} - ob_{jx}\right)^2 + \left(pp_{iy} - ob_{jy}\right)^2}, \tag{9}$$

$$v_{ij} = \max\left(1 - \frac{d_{ij}}{ob_{jr}}, 0\right), \tag{10}$$

$$V = \sum_{i,j} v_{ij}\,, \tag{11}$$

where d_{ij} is the distance between interpolation point i and center of the circumscribed circle j, v_{ij} is the violation if interpolation point i is in the obstacle j, and V is total violation, respectively. Therefore, a modified result is shown as follows:

$$L_m = L\,(1 + \rho V) \tag{12}$$

where L_m is the modified result, ρ is an adjustment factor, respectively. Obviously, if some interpolation points are in the obstacles, the modified result path is prolonged and is longer than the original result path. Therefore, an objective function L_f is introduced as follows for MPA optimization.

$$L_f = \begin{cases} L, & \text{all interpolation points are not in obstacles,} \\ L_m, & \text{some interpolation points are in obstacles,} \\ \infty, & \text{some middle points are in obstacles.} \end{cases} \tag{13}$$

We define the middle points $P = \left[p_1(p_{1x}, p_{1y}), p_2(p_{2x}, p_{2y}), \ldots, p_N(p_{Nx}, p_{Ny}\right]$ as optimization variable, L_f as objective function, the edge of target area as variable constraints. Thus, we can use MPA optimization to realize the robot path planning in a given target area.

4. Simulation

To test the efficiency of proposed method, 4 simulations are shown in this section. Our simulation circumstance includes 2.5 GHz Intel(R) Core(TM) i5-7200U CPU, 8GB RAM, and MATLAB 7.0 R2009b. The first simulation is aimed to test the basic performance of the method. The second simulation checks the performance with the different parameters. The third simulation compares the performance between proposed method and other typical methods. In the last simulation, we consider the realization of proposed method under the safe distance between the robot and the obstacles as one of practical situation. In our method, we use the circumscribed circle to model the independent obstacle. Therefore, the circle obstacles are used for these simulations.

4.1. Simulation 1

In simulation 1 the basic performance is tested. The following parameter settings are selected in simulation 1: start point is $(0, 0)$, number of middle points is 4, adjustment factor ρ is 100, number of search agents in MPA is 25, and maximum number of iterations is 500, $p_x \in [-10, 10]$, $p_y \in [-10, 10]$. Three scenes are used for test as shown in Fig. 2. In Fig. 2, the yellow square is start point and the green pentagram is destination point.

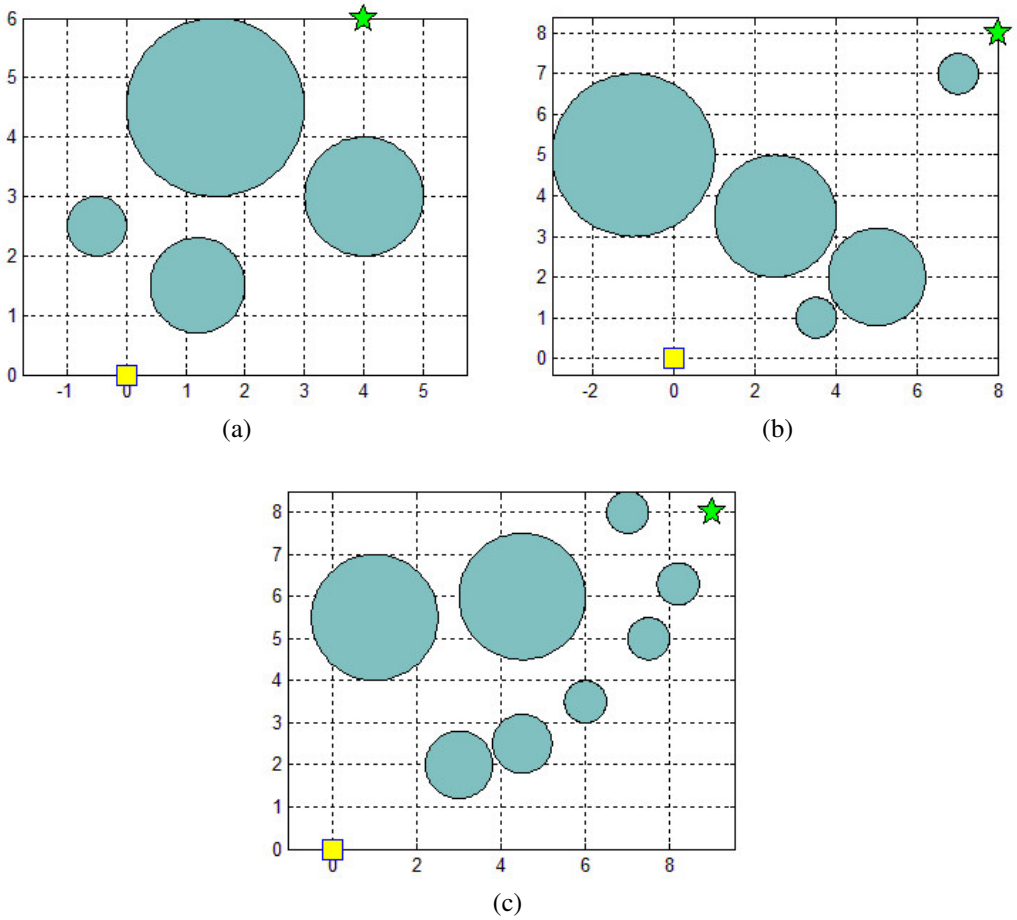


Figure 2: Schemes for simulation 1

The result of simulation 1 is shown in Fig. 3 and illustrates that the path planning is well realized. Fig. 4 shows the iteration process in simulation 1. Table 1 shows the details of simulation 1, which includes the coordinate of destination point as well as the result of middle points and path length.

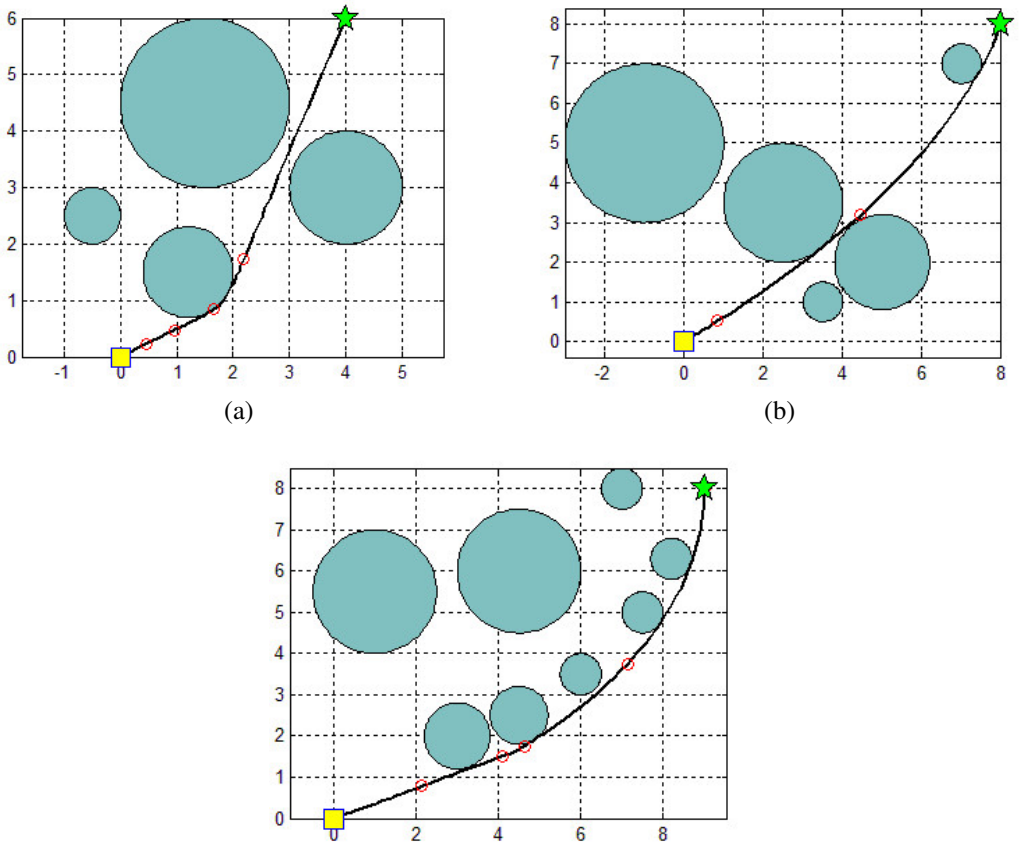
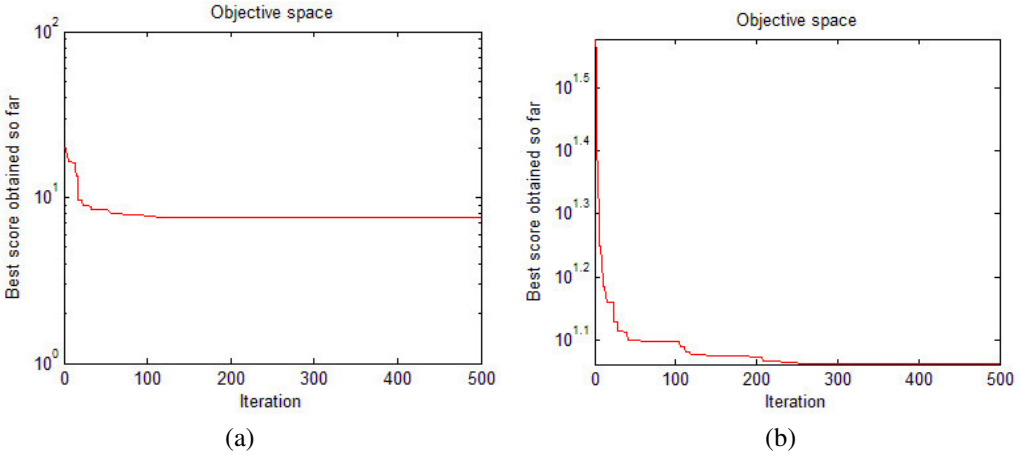
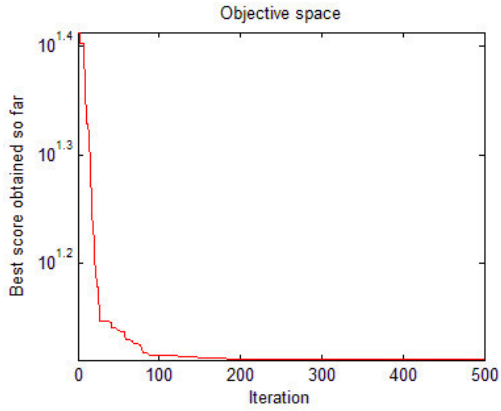


Figure 3: Result of simulation 1





(c)

Figure 4: Iteration process of simulation 1

Table 1: Details of simulation 1

	Destination Point	Obstacles	Middle Points	Result
Simulation 1a	(4,6)	ob_1(–0.5, 2.5) $r_1 = 0.5$ ob_2(1.2, 1.5) $r_2 = 0.8$ ob_3(1.5, 4.5) $r_3 = 1.5$ ob_4(4, 3) $r_4 = 1$	p_1(0.227340, 0.112805), p_2(0.510527, 0.251189), p_3(1.524739, 0.763690), p_4(2.151584, 1.645890)	7.546620
Simulation 1b	(8,8)	ob_1(–1, 5) $r_1 = 2$ ob_2(2.5, 3.5) $r_2 = 1.5$ ob_3(3.5, 1) $r_3 = 0.5$ ob_4(5, 2) $r_4 = 1.2$ ob_5(7, 7) $r_5 = 0.5$	p_1(0.321401, 0.190319), p_2(0.441454, 0.264442), p_3(1.197911, 0.745012), p_4(4.572792, 3.300611)	11.516975
Simulation 1c	(9,8)	ob_1(1, 5.5) $r_1 = 1.5$ ob_2(3, 2) $r_2 = 0.8$ ob_3(4.5, 2.5) $r_3 = 0.7$ ob_4(4.5, 6) $r_4 = 1.5$ ob_5(6, 3.5) $r_5 = 0.5$ ob_6(7, 8) $r_6 = 0.5$ ob_7(7.5, 5) $r_7 = 0.5$ ob_8(8.2, 6.3) $r_8 = 0.5$	p_1(0.723644, 0.764604), p_2(1.547514, 1.646202), p_3(2.590836, 2.699956), p_4(4.403291, 4.012246)	12.069715

4.2. Simulation 2

The performance for different parameters is studied in simulation 2 and scene 1a from the simulation 1 is used. Firstly, the effect of number of search agents in MPA on the result is studied. Other selected values of parameters are the same as in simulation 1. We use 10, 15, 20, 25, 30, 40, and 50 as the number

of search agents for test. The result of this test is shown as Table 2. From Table 2, when the number of search agents is bigger, the result is better while the elapsed time is longer.

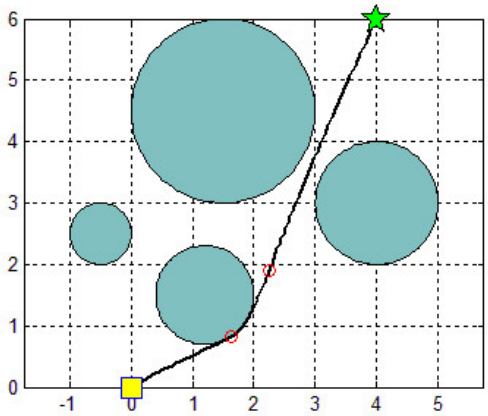
Table 2: Relations between number of search agents and result

Number of search agents	Result	Elapsed time (s)
10	7.547240	11.053363
15	7.546858	15.822105
20	7.546753	21.370520
25	7.546620	26.456755
30	7.546609	31.438441
40	7.546380	42.970323
50	7.545805	51.397946

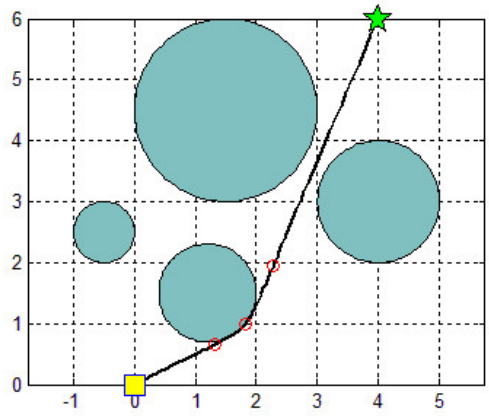
Secondly, the relations between number of middle points and result are studied. Number of search agents is 25. Other selected values of parameters are same as simulation 1. We use 2, 3, 4, 6, 9, and 12 middle points in the test. The result is shown in Table 3. Figure 5 shows the results of path planning with different number of middle points. It can be seen from Table 3 that when the number of middle points is less than 6, the result is acceptable. This turns out to be a good choice that the number of middle points is 4. Moreover, if the number of middle points is bigger, the elapsed time is longer. When the number of middle points is 9 or 12, some middle points overlap approximately. Therefore, too many points are not necessary.

Table 3: Relations between number of middle points and result

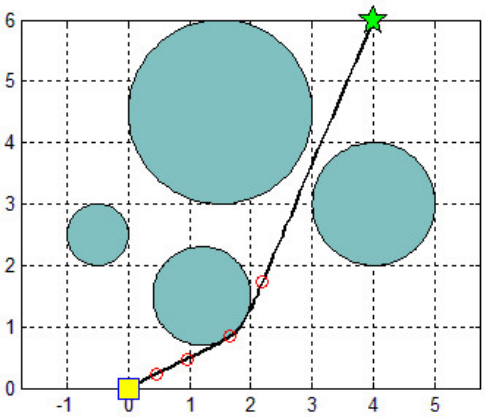
Number of middle points	Result	Elapsed time (s)
2	7.547740	25.573069
3	7.546996	26.190904
4	7.546620	26.456755
6	7.546692	26.644421
9	7.639523	26.782447
12	7.645375	26.973696



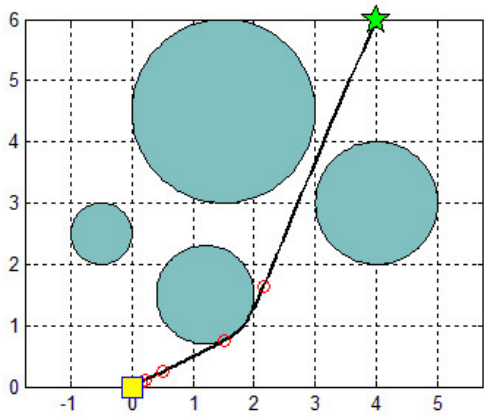
(a) number = 2



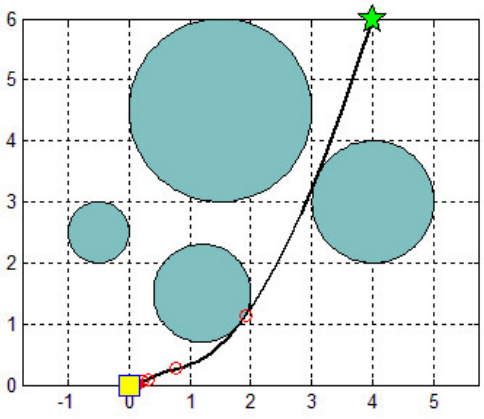
(b) number = 3



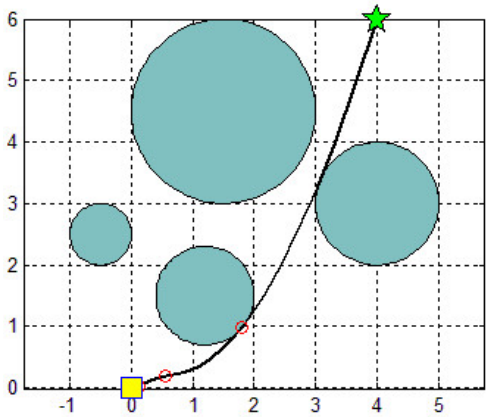
(a) number = 4



(b) number = 6



(a) number = 9



(b) number = 12

Figure 5: Results of path planning with different number of middle points

4.3. Simulation 3

In simulation 3 we compare the performance of our method and robot path planning using PSO. PSO is a famous bionic intelligent algorithm which is widely used in optimization. The following parameter settings of PSO are selected in simulation 3: maximum number of Iterations is 500, inertia weight is 1, inertia weight damping ratio is 0.98, personal learning coefficient is 1.5, global learning coefficient is 1.5, and α is 0.1. Other selected values of parameters are the same as in simulation 1. We use scene 1a as test object. We use 10, 15, 20, 25, 30, 40, and 50 as the population size in PSO for test and compare with Table 2. It follows for Table 4, that the result of our method is better than robot path planning using PSO. When the number of middle points in MPA or population size in PSO is small (< 30), the elapsed time of our method is shorter. Therefore, our method shows efficiency and proves a good performance for the robot path planning.

Table 4: Relations between population size and result in PSO optimization

Population size	Result	Elapsed time (s)
10	7.708715	21.657349
15	7.693902	23.730421
20	7.571579	24.388451
25	7.561598	31.448218
30	7.561551	31.646596
40	7.548479	39.384715
50	7.548475	51.397946

4.4. Simulation 4

In the last simulation we consider the realization of proposed method under the safe distance between the robot and the obstacles. In the simulation the radius of circumscribed circles is increased what causes overlap. We also use three scenes to test the proposed method. Every scene has some overlap area. Fig. 6 shows the result of simulation 4 and Table 5 shows the detail of this simulation. It is clear that the proposed method also works well in this simulation.

From the results of simulation 1 to 4, we can learn that proposed robot path planning method realizes the desired design and works well in different situations. Therefore, this method is an efficient method which is well applied to the automatic routing design of robot application.

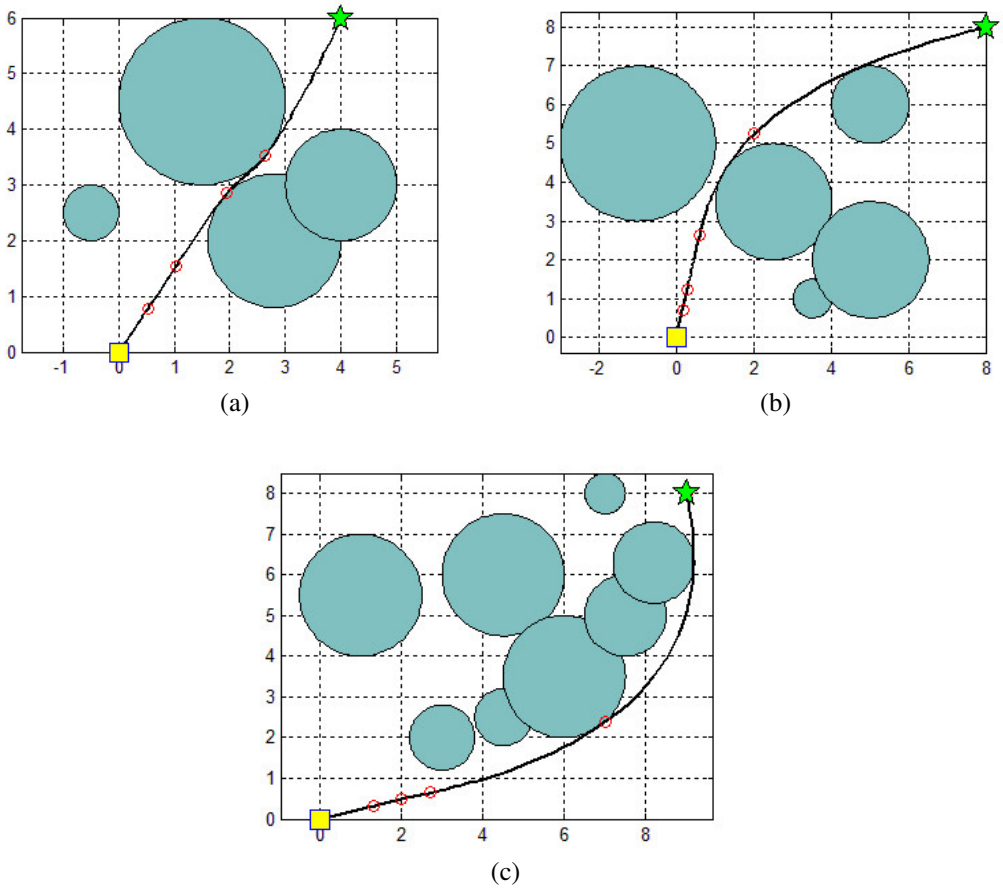


Figure 6: Result of simulation 4

5. Conclusion

Robot path planning is a critical research area in robotics that involves finding a feasible path for a robot to move from its current position to its goal position while avoiding obstacles. Graph search algorithms, potential field methods, and sampling-based methods are popular techniques used for robot path planning. Each technique has its advantages and limitations, and the choice of technique depends on the specific requirements of application. A novel method for robot path planning is proposed which uses marine predator algorithm as the optimization tool. The simulation results demonstrate that our method works well in different situations. Therefore, this method is an efficient tool which can be applied in the robot path planning and relative applications.

Table 5: Details of simulation 4

	Destination Point	Obstacles	Middle Points	Result
Simulation 4a	(4, 6)	ob_1(–0.5, 2.5) r_1 =0.5 ob_2(1.2, 1.5) r_2 =0.8 ob_3(2.8, 2) r_3 =1.2 ob_4(4, 3) r_4 =1	p_1(0.306584, 0.462852), p_2(0.697968, 1.060690), p_3(1.908223, 2.804900), p_4(2.621655, 3.500493)	7.250949
Simulation 4b	(8, 8)	ob_1(–1, 5) r_1 =2 ob_2(2.5, 3.5) r_2 =1.5 ob_3(3.5, 1) r_3 =0.5 ob_4(5, 2) r_4 =1.5 ob_5(5, 6) r_5 =1	p_1(0.015088, 0.047861), p_2(0.071338, 0.260807), p_3(0.298513, 1.485034), p_4(1.515279, 4.681381)	12.400865
Simulation 4c	(9, 8)	ob_1(1, 5.5) r_1 = 1.5 ob_2(3, 2) r_2 = 0.8 ob_3(4.5, 2.5) r_3 = 0.7 ob_4(4.5, 6) r_4 = 1.5 ob_5(6, 3.5) r_5 = 1.5 ob_6(7, 8) r_6 = 0.5 ob_7(7.5, 5) r_7 =1 ob_8(8.2, 6.3) r_8 = 1	p_1(1.363796, 0.386311), p_2(2.672261, 0.752883), p_3(3.290054, 0.900366), p_4(7.175943, 2.495619)	13.826446

References

[1] S.S. Ge and Y.J. Cui: New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, **16**(5), (2000), 615–620. DOI: 10.1109/70.880813

[2] N. Sariff and N. Buniyamin: An overview of autonomous mobile robot path planning algorithms. *IEEE 4th Student Conference on Research and Development*, (2006). DOI: 10.1109/scored.2006.4339335

[3] A. Faramarzi, M. Heidarinejad, S. Mirjalili and A.H. Gandomi: Marine predators algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications*, **152** (2020), 113377. DOI: 10.1016/j.eswa.2020.113377

[4] T.T. Mac, C. Copot, D.T. Tran and R. De Keyser: Heuristic approaches in robot path planning: A survey. *Applied Mathematics and Computation*, **222** (2013), 420–437. DOI: 10.1016/j.robot.2016.08.001

[5] A.K. Guruji, H. Agarwal and D.K. Parsediya: Time-efficient A* algorithm for robot path planning. *Procedia Technology*, **23** (2016), 144–149. DOI: 10.1016/j.protcy.2016.03.010

[6] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai and H. Pan: An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Robotics and Autonomous Systems*, **106** (2018), 26–37. DOI: 10.1016/j.robot.2018.04.007

[7] L. Zuo, Q. Guo, X. Xu and H. Fu: A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots. *Neurocomputing* **170** (2015), 257–266. DOI: 10.1016/j.neucom.2014.09.092

[8] H. Wang, Y. Yu and Q. Yuan: Application of Dijkstra algorithm in robot path-planning. *IEEE Second International Conference on Mechanic Automation and Control Engineering*, (2011). DOI: 10.1109/mace.2011.5987118

[9] J.L. Solka, J.C. Perry, B.R. Poellinger and G.W. Rogers: Fast computation of optimal paths using a parallel Dijkstra algorithm with embedded constraints. *Neurocomputing*, **8**(2), (1995), 195–212. DOI: 10.1016/j.compeleceng.2021.107327

[10] H.K. Tripathy, S. Mishra, H.K. Thakkar and D. Rai: CARE: A collision-aware mobile robot navigation in grid environment using improved Breadth First Search. *Computers and Electrical Engineering*, **94** (2021), 107327. DOI: 10.1016/j.compeleceng.2021.107327

[11] A. Basiri, V. Mariani, G. Silano, M. Aatif, L. Iannelli and L. Glielmo: A survey on the application of path-planning algorithms for multi-rotor UAVs in precision agriculture. *The Journal of Navigation*, **75**(2), (2022), 364–383. DOI: 10.1017/s0373463321000825

[12] S. Bortoff: Path planning for UAVs. In *Proceedings of the 2000 American Control Conference*, **1** (2000), 364–368. DOI: 10.1109/acc.2000.878915

[13] G. Li, Y. Tamura, A. Yamashita and H. Asama: Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning. *International Journal of Mechatronics and Automation*, **3**(3), (2013), 141–170. DOI: 10.1504/ijma.2013.055612

[14] M.H. Jaryani: An effective manipulator trajectory planning with obstacles using virtual potential field method. *2007 IEEE International Conference on Systems, Man and Cybernetics*, (2007). DOI: 10.1109/icsmc.2007.4413685

[15] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang and X. Feng: A novel potential field method for obstacle avoidance and path planning of mobile robot. *3rd International Conference on Computer Science and Information Technology*, (2010). DOI: 10.1109/iccsit.2010.5565069

[16] M. Elbanhawi and M. Simic: Sampling-based robot motion planning: A review. *IEEE Access*, **2** (2014), 56–77. DOI: 10.1109/access.2014.2302442

[17] L.E. Kavraki, M.N. Kolountzakis and J.-C. Latombe: Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, **14**(1), (1998), 166–171. DOI: 10.1109/70.660866

[18] G. Chen, N. Luo, D. Liu, Z. Zhao and C. Liang: Path planning for manipulators based on an improved probabilistic roadmap method. *Robotics and Computer-Integrated Manufacturing*, **72** (2021), 102196. DOI: 10.1016/j.rcim.2021.102196

[19] S. Rodriguez, X. Tang, J.M. Lien and N.M. Amato: An obstacle-based rapidly-exploring random tree. *Proceedings 2006 IEEE International Conference on Robotics and Automation*, (2006). DOI: 10.1109/robot.2006.1641823

[20] I. Noreen, A. Khan and Z. Habib: Optimal path planning using RRT* based approaches: A survey and future directions. *International Journal of Advanced Computer Science and Applications*, **7**(11), (2016), 97–107. DOI: 10.14569/ijacsa.2016.071114

[21] I. Noreen, A. Khan and Z. Habib: A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security*, **16**(10), (2016), 20–27.

[22] L. Zhang, Y.J. Kim and D. Manocha: A hybrid approach for complete motion planning. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2007). DOI: 10.1109/IROS.2007.4399064

[23] J. van den Berg and M. Overmars: Path planning in repetitive environments. *12th IEEE International Conference on Methods and Models in Automation and Robotics*, (2006) 657–662.

[24] K. Belghith, F. Kabanza, L. Hartman and R. Nkambou: Anytime dynamic path-planning with flexible probabilistic roadmaps. *Proceedings 2006 IEEE International Conference on Robotics and Automation*, (2006). DOI: 10.1109/ROBOT.2006.1642057

[25] B. Burns and O. Brock: Sampling-based motion planning using predictive models. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, (2005). DOI: 10.1109/ROBOT.2005.1570590

[26] M.N. Ab Wahab, S. Nefti-Meziani and A. Atyabi: A comparative review on mobile robot path planning: Classical or meta-heuristic methods? *Annual Reviews in Control*, **50** (2020), 233–252. DOI: 10.1016/j.arcontrol.2020.10.001

[27] Y. Hu and S.X. Yang: A knowledge based genetic algorithm for path planning of a mobile robot. *IEEE International Conference on Robotics and Automation*, (2004). DOI: 10.1109/ROBOT.2004.1302402

[28] G. Li and H. Shi: Path planning for mobile robot based on particle swarm optimization. *2008 Chinese Control and Decision Conference*, (2008). DOI: 10.1109/CCDC.2008.4597938

[29] B. Song, Z. Wang and L. Zou: An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. *Applied Soft Computing*, **100** (2021), 106960. DOI: 10.1016/j.asoc.2020.106960

[30] H.S. Dewang, P.K. Mohanty and S. Kundu: A robust path planning for mobile robot using smart particle swarm optimization. *Procedia Computer Science*, **133** (2018), 290–297. DOI: 10.1016/j.procs.2018.07.036

[31] M. Panda, B. Das and B. Bhusan Pati: Global path planning for multiple AUVs using GWO. *Archives of Control Sciences*, **30**(1), (2020), 77–100. DOI: 10.24425/acs.2020.132586

[32] S. Vaidyanathan, K. Benkouider, A. Sambas and P. Darwin: Bifurcation analysis, circuit design and sliding mode control of a new multistable chaotic population model with one prey and two predators. *Archives of Control Sciences*, **33**(1), (2023), 127–153. DOI: 10.24425/acs.2023.145117

[33] N.E. Humphries, N. Queiroz, J.R.M. Dyer, N.G. Pade, M.K. Musyl and K.M. Schaefer: Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature*, **465**(7301), (2010), 1066–1069. DOI: 10.1038/nature09116