Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 21(2): 2568–2586.
DOI: 10.3934/mbe.2024113
Received: 22 October 2023
Revised: 21 December 2023
Accepted: 29 December 2023
Published: 18 January 2024

Research article

A novel parallel ant colony optimization algorithm for mobile robot path planning

Jian Si and Xiaoguang Bao[*]

College of Information Technology, Shanghai Ocean University, Shanghai 201306, China

[*] **Correspondence:** Email: xgbao@shou.edu.cn.

Abstract: With the continuous development of mobile robot technology, its application fields are becoming increasingly widespread, and path planning is one of the most important topics in the field of mobile robot research. This paper focused on the study of the path planning problem for mobile robots in a complex environment based on the ant colony optimization (ACO) algorithm. In order to solve the problems of local optimum, susceptibility to deadlocks, and low search efficiency in the traditional ACO algorithm, a novel parallel ACO (PACO) algorithm was proposed. The algorithm constructed a rank-based pheromone updating method to balance exploration space and convergence speed and introduced a hybrid strategy of continuing to work and killing directly to address the problem of deadlocks. Furthermore, in order to efficiently realize the path planning in complex environments, the algorithm first found a better location for decomposing the original problem into two subproblems and then solved them using a parallel programming method-single program multiple data (SPMD)-in MATLAB. In different grid map environments, simulation experiments were carried out. The experimental results showed that on grid maps with scales of 20×20, 30×30, and 40×40 compared to nonparallel ACO algorithms, the proposed PACO algorithm had less loss of solution accuracy but reduced the average total time by 50.71, 46.83 and 46.03%, respectively, demonstrating good solution performance.

Keywords: ant colony optimization; decomposition; parallelism; single program multiple data; mobile robot; path planning

1. Introduction

Path planning is an important problem in the field of mobile robot research. It can be described as follows: Given a two-dimensional area filled with obstacles, the objective of the problem is to find a minimum-length collision-free path from a starting point to a target point. For the problem, there are many solution methods, such as artificial potential field (APF) method [1, 2], A* algorithm [3, 4], fuzzy logic (FL) [5,6], simulated annealing (SA) [7,8], genetic algorithm (GA) [9,10], particle swarm optimization (PSO) [11,12], ant colony optimization (ACO) [13,14], whale optimization algorithm (WOA) [15,16], slime mould algorithm (SMA) [17,18], moth-flame optimization (MFO) [19,20], Harris Hawks optimization (HHO) [21,22], grey wolf optimization (GWO) [23,24], and so on. Researchers are committed to effectively solving the mobile robot path planning by improving the components of individual algorithms or hybridizing different solution methods. For example, Li et al. [4] introduced the bidirectional alternating search strategy, improved the heuristic function, and employed the filtering function and Bézier curves in the A* algorithm to solve the problems of long calculation time, large turning angles, and unsmoothed path in large task spaces. Shi et al. [8] introduced the initial path selection method and deletion operation in the SA algorithm to reduce the computational effort. Zhang et al. [10] combined GA and the firefly algorithm (FA) to address the problem of FA easily trapping into the local optimal solution. Yuan et al. [12] proposed an improved PSO algorithm based on differential evolution to overcome the disadvantage of low convergence accuracy and easy maturity. Dai et al. [16] adopted adaptive technology, set virtual obstacles, and introduced improved potential field factor in WOA to improve the convergence speed, avoid local optimal traps, and enhance the dynamic obstacle avoidance ability of mobile robots, respectively. Zheng et al. [18] utilized a variable neighborhood Lévy flight and an individual rotation perturbation and variation mechanism in SMA to enhance the local optimization ability and prevent falling into local optimization. Dai and Wei [20] in MFO introduced the concept of historical best flame average to jump out of the local optimum and employed the quasi-opposition-based learning to enrich the population diversity and improve the convergence rate. Cai et al. [22] adopted linear path, local search update, and nonlinear control strategies to improve the performance of the HHO algorithm. Hou et al. [24] used improved chaotic tent mapping, the nonlinear convergence factor based on the Gaussian distribution change curve, and the dynamic proportional weighting strategy to enhance the performance of the GWO algorithm.

The ACO algorithm [25–27], inspired by the cooperative foraging behavior of ants in nature, is widely used to solve combinatorial optimization problems (such as disassembly sequence planning [28], vehicle routing problem [29], traveling salesman problem [30], airport taxiway planning [31], job shop scheduling [32], feature selection [33], and image segmentation [34]) due to its advantages of positive feedback, strong robustness, and parallelism. For the path planning of mobile robots, there are also numerous research results in the literature. Akka and Khabers [35] introduced a stimulating probability in the state transition rule to choose a safe and accessible grid. Meanwhile, the authors employed new heuristic information, pheromone updating rule, and a dynamic evaporation strategy to make the algorithm more competitive. You et al. [36] presented an improved ant colony system (ACS) algorithm, in which a new heuristic operator in the state transition rule was adopted to achieve a balance between the population diversity and convergence rate. Here, ACS is an ACO algorithm and is introduced by Dorigo and Gambardella [37]. Gao [38] proposed an enhanced heuristic ACO algorithm and utilized four strategies to improve the performance and efficiency of the algorithm, including local visibility enhancement, new pheromone diffusion, backtracking mechanism, and path merging. Luo et al. [13] proposed an improved ACO algorithm to overcome shortcomings of the basic ACO algorithm in terms of initial pheromone, heuristic information, pheromone update, state transition, and deadlock problem.

In addition to the above studies in which ants deposit pheromones on arcs, there is also the work of placing pheromones on nodes (see Deng [39]). Furthermore, there are also many scholars who combine the ACO algorithm with other methods to solve the path planning problem. For example, in Liu et al. [40], Dai et al. [41], Zhang et al. [42], and Li et al. [43], the ACO algorithm was combined with the APF method and geometric local optimization, A* heuristic method, PSO algorithm, and artificial bee colony algorithm, respectively.

In order to improve the global search ability and solution efficiency for the mobile robot path planning in complex environments, a novel parallel ant colony optimization (PACO) algorithm is proposed in this paper. The improvement and innovation of the algorithm includes the following three aspects. First, a new rank-based pheromone updating method in which the first ten short paths are rewarded and the last ten poor paths are punished is constructed to balance the search space exploration and faster convergence speed. Second, a hybrid strategy of continuing working and killing directly is introduced to address the deadlock problem. In the early iterations, ants in a deadlock state are allowed to continue searching to enrich the diversity of solutions, while in the later iterations the ants are directly killed to improve convergence speed of the algorithm. Third, the strategies of decomposition and parallelism are designed to achieve rapid problem solving. The algorithm first finds a better location based on the pheromone concentration to split the original problem into two subproblems, and then solves them using a parallel programming method-single program multiple data (SPMD)-in MATLAB. Simulation experiments were conducted in different grid map environments. The results verify the validity and superiority of the proposed algorithm.

The remainder of this paper is organized as follows. Section 2 describes the environmental model for the path planning of mobile robots. Traditional and parallel ACO algorithms are presented in Sections 3 and 4, respectively. Simulation experiments of different scales are carried out in Section 5. Finally, conclusions and further work are given in Section 6.

2. Environmental model

A two-dimensional grid map model is used to represent the working environment of the robot. In a grid map, all grids are numbered sequentially from left to right and from top to bottom. Meanwhile, the grids are divided into two colors: white and black. The white grids, in which the robot can move, are called feasible, while the black grids, which represent obstacles, are called infeasible. Figure 1 shows an example of a grid map.

In order to facilitate the implementation of the algorithm, 0 and 1 are used to represent feasible and infeasible grids, respectively. Furthermore, the conversion from the grid numbers to two-dimensional coordinates is given by Formula (2.1).

$$\begin{cases} x = \begin{cases} mod(n,N) - 0.5, & \text{if } mod(n,N) \neq 0 \\ N - 0.5, & \text{otherwise} \end{cases} \\ y = N + 0.5 - ceil\left(\frac{n}{N}\right) \end{cases} \tag{2.1}$$

Here, n is the grid number, N is the number of columns in the grid map, *mod* denotes the remainder function, and *ceil* denotes the rounding function toward positive infinity.

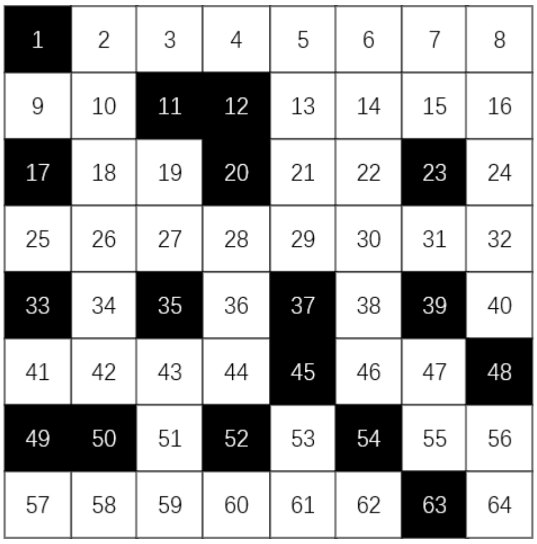


Figure 1. Grid map.

3. Traditional ACO algorithm

The ACO algorithm is a metaheuristic that simulates the foraging behavior of natural ants. During the process of searching for food source, the ants release pheromones as a communication medium to guide other members of the colony. In the ACO algorithm, the ants probabilistically construct solutions to a problem based on pheromones and heuristic information. After finding solutions, the ants will leave a certain amount of pheromones on the solutions.

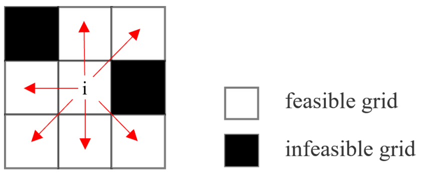


Figure 2. Feasible movements for next grid of robot.

In a grid map environment, each ant can move in no more than 8 directions. As shown in Figure 2, the ant k located at grid i can only choose the next moving position from 6 adjacent grids due to two obstacle grids. The state transition probability from grid i to the next grid j is calculated by Formula (3.1).

$$P_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{s\in N_k(i)} \tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)}, & j \in N_k(i) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Here, $\tau_{ij}(t)$ denotes the pheromone value of the edge (i, j) at time t, $\eta_{ij}(t)$ represents the heuristic information value given by Formula (3.2) where d_{ij} is the distance between grids i and j, α is the pheromone intensity factor, β is the expected heuristic factor, and $N_k(i)$ is the set of grids that are feasible and have not been visited by ant k.

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \tag{3.2}$$

After all the ants complete a search, the pheromone values of the edges in feasible paths will be updated according to Formula (3.3).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{3.3}$$

Here, ρ represents the pheromone evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}(t)$ is the amount of pheromone left by ant k on the edge (i, j), which is typically given by Formula (3.4).

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k}, & \text{if ant } k \text{ passes through edge } (i, j) \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Here, Q is a constant and L_k is the total length of the path of ant k.

4. Parallel ACO algorithm

In large-scale maps, as the number of grids increases, the solving efficiency of the traditional ACO algorithm decreases sharply. In order to solve this problem, decomposition and parallelism strategies are proposed. Before presenting these two strategies, based on the results of existing literature, we provide two small modifications in pheromone updating and deadlock handling to optimize the corresponding components of the algorithm.

4.1. Pheromone updating rule

For the pheromone updating, Bullnheimer et al. [44] proposed a rank-based method. The paths are first sorted according to their length, then some of the paths that rank higher are used. In addition, compared to Formula (3.4), the authors introduced a weight that is proportional to the rank of the path. On the other hand, Luo et al. [13] introduced a penalty mechanism for the iteration-worst solution.

Inspired by the above two approaches, we introduce a rank-based pheromone updating method, in which the first ten good and last ten poor paths are used. At the same time, in order to balance the exploration space and convergence speed, we gradually utilize the information of these paths. More specifically, the new pheromone updating strategy is designed as follows: When $1 \leq t \leq 10\%T$, each of the first ten good paths is rewarded whereas the worst path is punished; when $10\%T < t \leq 20\%T$, each of the first nine short paths is rewarded whereas each of the last two poor paths is punished; and so on. Here, t is the iteration counter and T is the maximum number of iterations. The values of the reward and punishment are calculated by Formula (4.1). Here, the meanings of *ceil* and L_k can be found in Formulas (2.1) and (3.4), respectively.

From the proposed pheromone updating strategy, it can be seen that: In the early iterations, more good paths are appropriately rewarded and fewer poor paths are punished, which helps the algorithm to explore extensively; in the later iterations, fewer short paths are greatly rewarded and more long paths are punished, which helps to improve the convergence speed of the algorithm.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} ceil(\frac{t}{10\%T}) * \frac{Q}{L_k}, & \text{if ant } k \text{ is rewarded and passes through edge } (i, j) \\ -\frac{Q}{L_k}, & \text{if ant } k \text{ is punished and passes through edge } (i, j) \\ & \text{and } 1 \le t \le 50\%T \\ -0.2 * \frac{Q}{L_k}, & \text{if ant } k \text{ is punished and passes through edge } (i, j) \\ & \text{and } 50\%T < t \le T \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$\tau_{ij}(t+1) = \begin{cases} (1-\lambda)^{ceil\left(\frac{M_{Lost}(i,j)}{3}\right)} * \tau_{ij}(t), & \text{if edge } (i, j) \text{ is in the last two steps} \\ \tau_{ij}(t), & \text{otherwise} \end{cases} \tag{4.2}$$

4.2. *Deadlock handling strategy*

Due to the presence of obstacles and the use of a taboo list, ants may fall into a state of no road to go, known as a deadlock. As shown in Figure 3, when an ant in grid S passes through T and enters D, it is trapped in a deadlock state because the eight adjacent grids are either obstacles or belong to the taboo list.

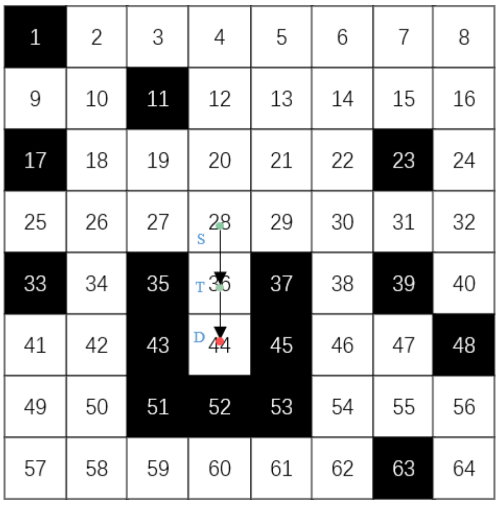


Figure 3. Status of deadlock.

For the deadlock problem, Wang and Yu [45] proposed a strategy called early death, in which ants trapped in a deadlock state are directly killed without any other actions. Qu et al. [46] presented a one-step backtracking method where ants are allowed to return one step and continue searching for a path, and the edge corresponding to this step is punished. Luo et al. [13] gave a compromise strategy, in which ants in a deadlock state are killed and the edges corresponding to the last two steps are punished according to Formula (4.2). Here, $0 < \lambda < 1$ is a penalty factor, $M_{Lost}(i, j)$ is the number of ants entering the same deadlock position, and the meaning of *ceil* can be found in Formula (2.1).

From Luo et al.'s method, it can be seen that the punishment not only alerts subsequent ants, but also adjusts its intensity based on the number of lost ants. However, killing these ants without continuing to work reduces the possibility of discovering feasible paths, leading to a decrease in the diversity of solutions. On the other hand, the subsequent search will also increase a certain amount of computation. In this paper, we follow Luo et al.'s method with minor change to balance extensive exploration and fast convergence. More specifically, the "continuing to work" strategy is adopted in the early iterations, while the "killing directly" strategy is used in the remaining iterations. The boundary between the two strategies varies with the scale of grid map, which will be specifically presented in the following experiments.

4.3. Decomposition and parallelism strategies

In order to efficiently complete the path planning in complex environments, a strategy of decomposing the original problem into two subproblems is proposed. The key to the strategy is the selection of a boundary line, such that the two subproblems obtained are as balanced as possible. Because any feasible path must pass through the middle row (column) between the rows (columns) corresponding to the starting and target points, which divides the grid map into approximately equal two parts, we choose it as the boundary line. Meanwhile, without loss of generality, suppose that the difference between the starting and target rows is not less than that between the columns. The middle row is selected as the boundary line in the following discussion.

Consider which grid in the middle row to use as an auxiliary point for partitioning the problem. Pheromone concentration is the key information for determining the importance of edges. Therefore, a method for selecting the auxiliary point can be carried out as follows: First, for each grid in the middle row, calculate the sum of the pheromone concentrations of the edges associated with it and choose the grid with the highest value. The second strategy is to directly count the number of feasible paths passing through each grid and select the grid with the maximum number as the auxiliary point. Due to the significant correlation between the two methods, the latter is adopted in this paper for ease of calculation. For determining the auxiliary point, a preprocessing stage, i.e. the first T_0 iterations of the algorithm, is executed. Here, T_0 is a parameter whose value will be given in the following text.

Now present the parallelism strategy of the algorithm. The proposed algorithm in this paper adopts the parallelism strategy in both stages. The first is in the preprocessing stage: In order to reduce the execution time of the algorithm and address the limitations of unidirectional search, two problems, one is from the starting point to the target point while the other is in opposite, are addressed in parallel, with each being executed T_0 iterations. The second is in the remaining stage: Two subproblems, one is from the starting point to the auxiliary point while the other is from the auxiliary point to the target point, are dealt with simultaneously. The MATLAB Parallel Computing Toolbox is an effective tool for developing a parallel program in a multiprocessor environment in which SPMD is one of the most used methods (see Cao et al. [47]). In our algorithm, the SPMD is used to implement parallel computing.

4.4. Parallel ACO procedure

The specific steps of the parallel ACO algorithm proposed in this paper are presented as follows:

Step 1. Import a two-dimensional grid environment. Set the starting and target points as N_{start} and N_{target}, respectively.

Step 2. Initialize the parameters. The ant number m, maximum iteration number T, preprocessing iteration number T_0, pheromone intensity Q, pheromone heuristic factor α, expected heuristic factor β, and deadlock penalty factor λ are initialized.

Step 3. Execute the preprocessing stage. Activate two CPU cores and address the problems from N_{start} to N_{target} and from N_{target} to N_{start}, respectively, using the proposed new relevant strategies. Set the auxiliary grid obtained and the time spent as $N_{auxiliary}$ and t_1, respectively.

Step 4. Execute the formal stage. Activate two CPU cores again and deal with the subproblems from N_{start} to $N_{auxiliary}$ and from $N_{auxiliary}$ to N_{target}, respectively, using the proposed new relevant strategies. Set the obtained subpaths as $Path_1$ and $Path_2$, respectively. Meanwhile, set the time spent as t_2.

Step 5. Output the final path $Path$ by connecting $Path_1$ and $Path_2$ and obtain the algorithm's total running time $t_{total} = t_1 + t_2$.

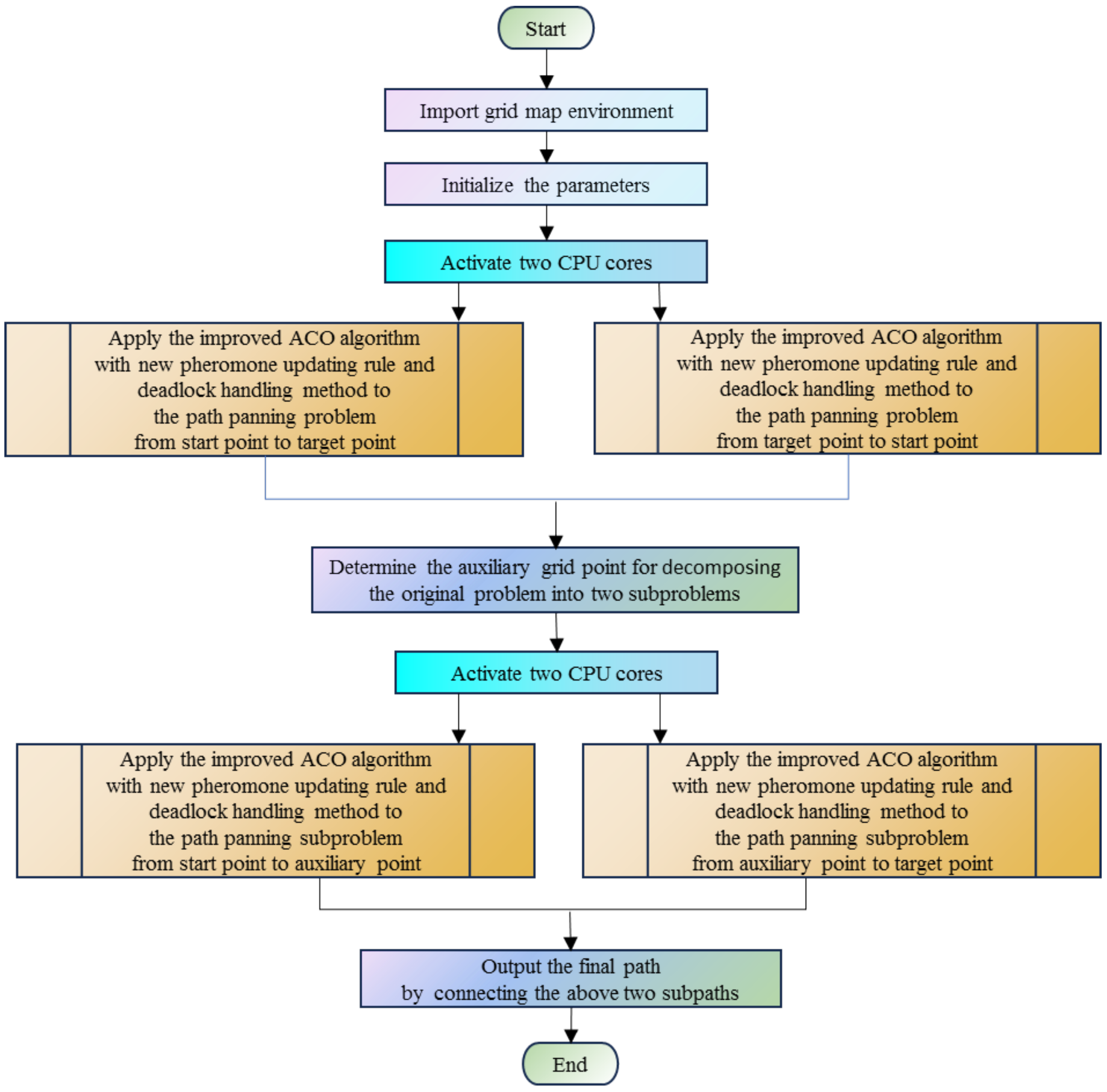

Start
Import grid map environment
Initialize the parameters
Activate two CPU cores
Apply the improved ACO algorithm with new pheromone updating rule and deadlock handling method to the path panning problem from start point to target point
Apply the improved ACO algorithm with new pheromone updating rule and deadlock handling method to the path panning problem from target point to start point
Determine the auxiliary grid point for decomposing the original problem into two subproblems
Activate two CPU cores
Apply the improved ACO algorithm with new pheromone updating rule and deadlock handling method to the path panning subproblem from start point to auxiliary point
Apply the improved ACO algorithm with new pheromone updating rule and deadlock handling method to the path panning subproblem from auxiliary point to target point
Output the final path by connecting the above two subpaths
End

Figure 4. Flowchart of the proposed parallel ACO algorithm.

The flowchart of the proposed parallel ACO algorithm is shown in Figure 4. Now, we analyze the time complexity of the algorithm. Because the search time of each ant in each iteration is at most

$O(N^2)$, the overall complexity of nonparallel ACO algorithms is $O(N^2 * m * T)$. However, due to the use of the decomposition and parallelism strategies, the total time consumption of the proposed parallel ACO algorithm is the sum of $O(N^2 * m * T_0)$ in the preprocessing stage and $O((\frac{N}{2})^2 * m * (T - T_0))$ in the formal stage, which is $O(N^2 * m * (\frac{T}{4} + \frac{3T_0}{4}))$.

5. Experimental results

In order to verify the effectiveness of the proposed parallel ACO algorithm for solving the path planning problem, simulation experiments of different scales were executed and compared with other ACO algorithms. The simulation environment is as follows: Windows10 64 bit; processor Intel (R) Core (TM) i5-8500; main frequency 3.00 GHz; memory 8 GB; simulation software MATLAB R2020a. By using the empirical and experimental method, a better parameter combination was determined. The values of these parameters are shown in Table 1.

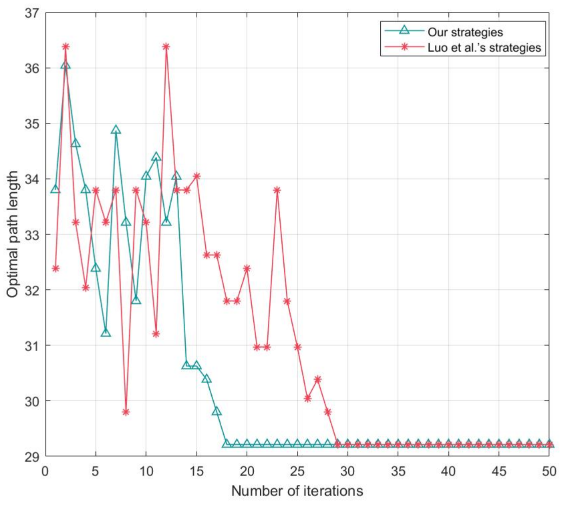
Table 1. Values of the main parameters in the proposed algorithm.

T	T_0	M	Q	α	β	λ
50	3	50	1	1.0	7.0	0.2

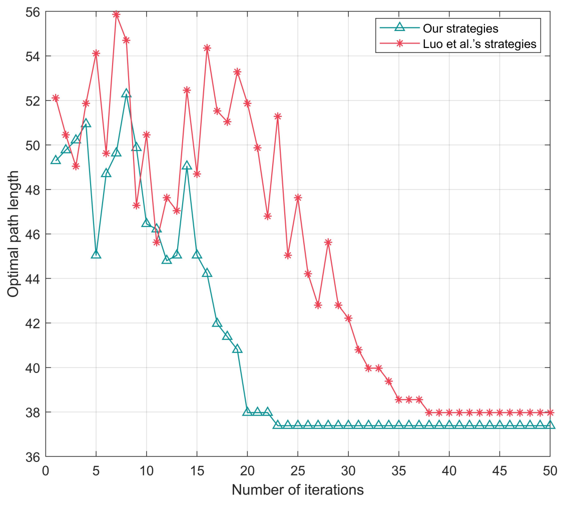
Next, in Section 5.1 we present two comparative experiments to verify the effectiveness of the proposed pheromone updating rule and deadlock handling strategy, respectively. The effectiveness of decomposition and parallelism strategies, as well as the comparison of PACO's performance with other algorithms, are presented in Section 5.2. In each comparative experiment, we used grid maps of three scales: 20×20, 30×30, and 40×40, all from Dai et al. [41]. Here, the boundaries between the "continuing to work" and "killing directly" strategies are 60, 70, and 80% of the total iteration number, respectively. Furthermore, in order to evaluate the stability of the algorithm, for each grid map with fixed start and target points as input, the algorithm was independently executed 10 times and the average and best results were used for comparison.

5.1. Effectiveness of pheromone updating rule and deadlock handling strategy

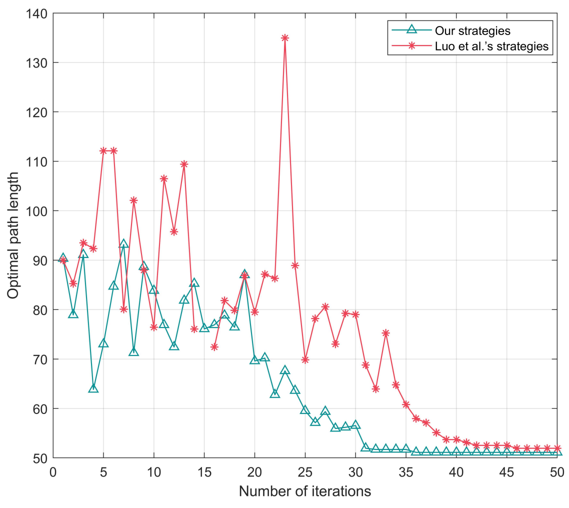
We first verify the effectiveness of the proposed pheromone updating rule. Due to the impact of deadlock handling strategy on the pheromone update method, we consider them as a whole and then validate their effectiveness. Since our method is based on that in Luo et al. [13], the combination of the two strategies proposed in this paper is compared with Luo et al.'s for the optimal path length. The results on three scale maps are shown in Figure 5, in which the solid lines marked by a blue triangle and red star represent the results obtained by ours and Luo et al.'s strategies, respectively. From Figure 6, it can be observed that: Our results are better than Luo et al.'s both in terms of convergence speed and solution accuracy, indicating the effectiveness of the combination of the two strategies proposed in this paper.



(a) 20 × 20



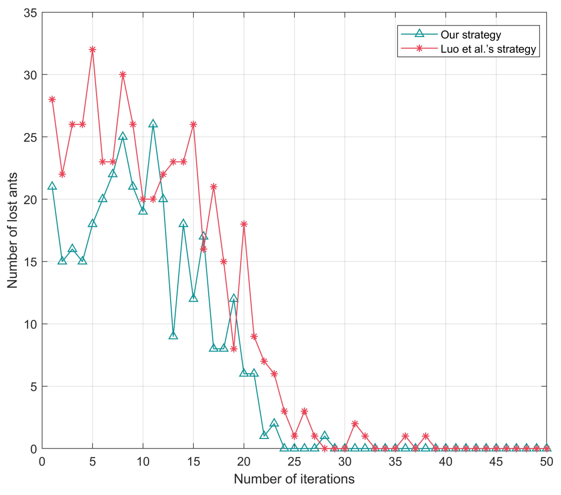
(b) 30 × 30



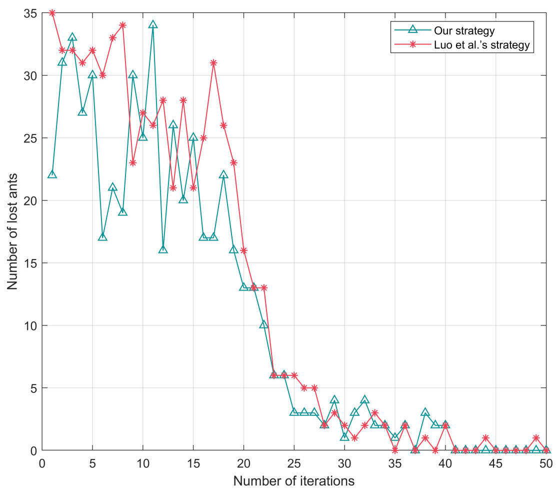
(c) 40 × 40

Figure 5. Comparisons of the optimal path length between ours and Luo et al.'s strategies.

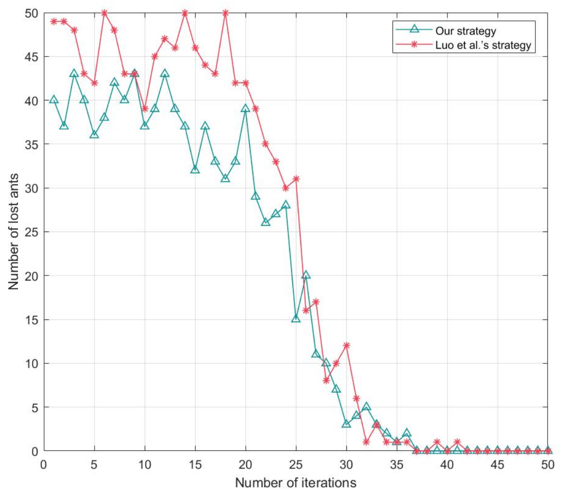
We now present a validation of the effectiveness of the proposed deadlock handling strategy. Because our method for dealing with the deadlock problem is based on that in Luo et al. [13], two strategies are compared for the number of lost ants. The results on three scale maps are shown in Figure 6, in which the solid lines marked by a blue triangle and red star represent the results obtained by ours and Luo et al.'s strategy, respectively. From Figure 6, it can be observed that: On the whole, the curve of the lost ant number using the method proposed in this paper is located below Luo et al.'s, which means that our strategy yields fewer lost ants, especially in the early iterations. Since the smaller the number of lost ants, the more the diversity of solution and the greater the probability of finding a better feasible path, indicating that the proposed method is effective.



(a) 20 × 20



(b) 30 × 30



(c) 40 × 40

Figure 6. Comparisons of the number of lost ants between ours and Luo et al.'s strategy.

5.2. *Effectiveness of decomposition and parallelism strategies as well as comparison of performance between PACO and other algorithms*

In order to verify the effectiveness of decomposition and parallelism strategies as well as the performance of the PACO algorithm, we compared the PACO algorithm with nonparallel ACO (NPACO) algorithm, Dai et al.'s ACO algorithm in [41], and the traditional ACO algorithm, respectively. Here, NPACO compared to PACO has the same components, except for not using decomposition and parallelism strategies. In the following three subsections, we conduct comparative analysis on the three maps of different scales.

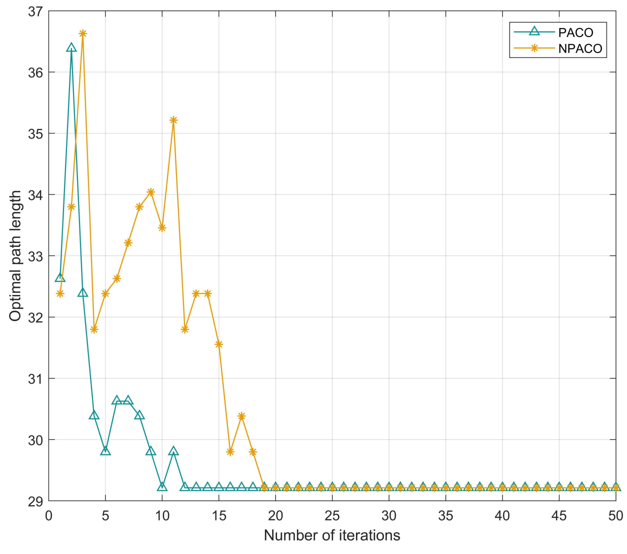
5.2.1. In 20 × 20 map environment

The first comparative experiment was conducted on a 20 × 20 grid map, where the starting and target points of the path planning problem are (0.5, 19.5) and (19.5, 0.5), respectively. The auxiliary point obtained by executing the preprocessing stage of the proposed PACO algorithm in this paper is (8.5, 10.5). The experimental results are as follows:

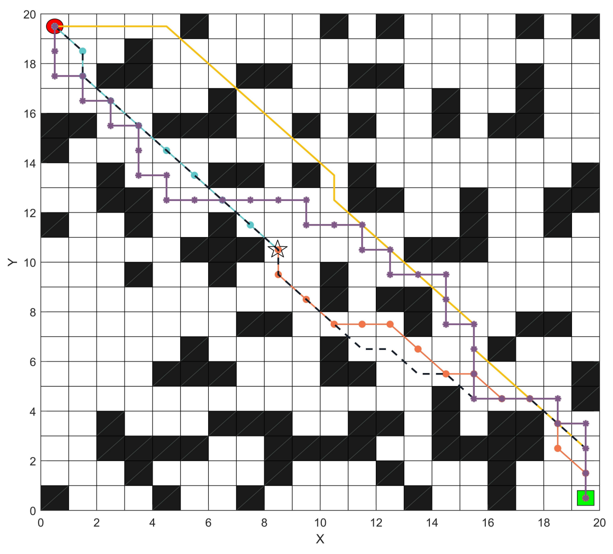
From Table 2, it can be observed that the average total time for PACO and NPACO is 1.6300 and 3.3070, respectively. The former is reduced by 50.71% compared to the latter, which means that the decomposition and parallelism strategies of PACO are effective. Meanwhile, for the shortest and average path lengths, the results of PACO are not inferior to those of NPACO, Dai et al.'s ACO [41], and traditional ACO. This indicates that PACO not only has a shorter running time, but also ensures better solution accuracy, further verifying its effectiveness.

Table 2. Experimental results on 20 × 20 map.

Evaluation Criteria	PACO	ACO in [41]	Traditional ACO	NPACO
Average total time /s	1.6300	4.89	9.22	3.3070
Shortest path length /m	29.2132	29.2133	37.4143	29.2132
Average path length /m	29.2132	29.3807	38.6335	29.2132



(a) Convergence curve



(b) Optimal path trajectory

Figure 7. Convergence curves and optimal path trajectories of the four algorithms on 20 × 20 map.

From Figure 7(a), it can be seen that in the early iterations, the result of PACO is also better than that of NPACO, indicating the effectiveness of the proposed strategies in this paper. Furthermore, optimal path trajectories are shown in Figure 7(b), where the solid lines with blue and orange circles, yellow solid line, black dashed line, and solid lines with purple circles represent the results obtained by PACO, Dai et al.'s ACO [41], NPACO, and traditional ACO, respectively. Meanwhile, the grid marked with a pentagram represents the auxiliary point in PACO.

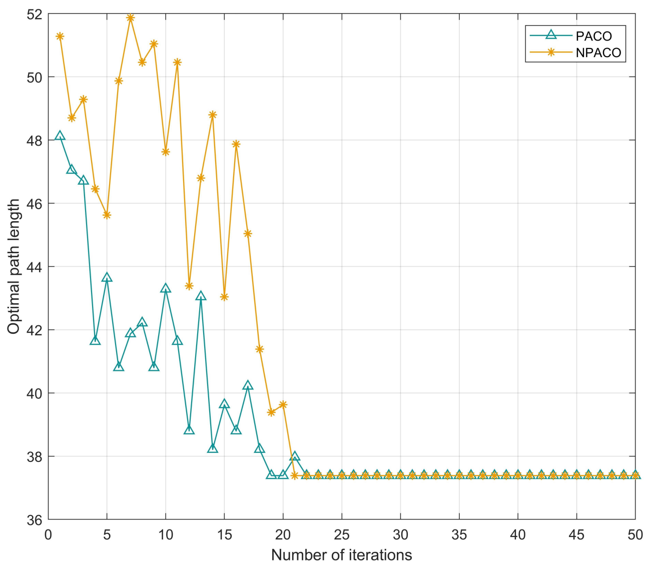
5.2.2. In 30 × 30 map environment

The second comparative experiment was on a 30 × 30 grid map with the starting and target points (0.5, 8.5) and (25.5, 28.5). The auxiliary point obtained by PACO is (20.5, 18.5). The experimental results are as follows:

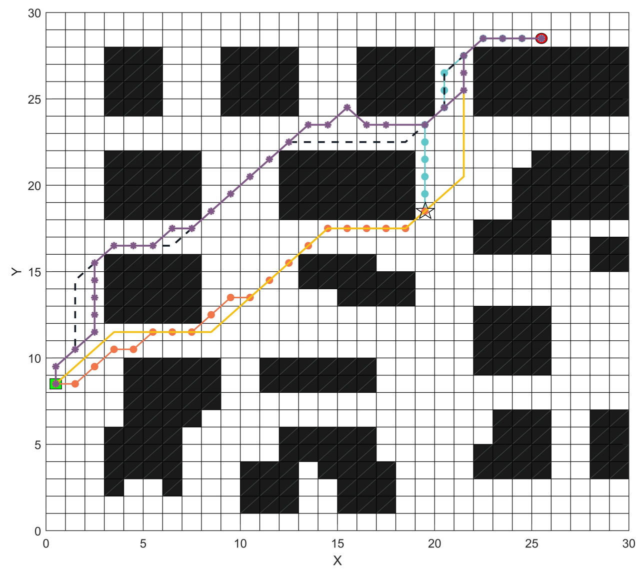
From Table 3, it can be observed that the average total time for PACO and NPACO is 7.2252 and 13.5880, respectively. The former is reduced by 46.83% compared to the latter, which again shows that the decomposition and parallelism strategies of PACO are effective. For the shortest path length, the result of traditional ACO is the worst, while the results of the other three algorithms are almost equal. For the average path length, the best and worst results are 37.3848 of NPACO and 38.6325 of traditional ACO, respectively; compared to the best result, the path length of PACO is increased by 1.42%. Combining the three indicators, it can be concluded that although PACO loses a small amount of solution accuracy, it significantly reduces the running time, proving its effectiveness.

Table 3. Experimental results on 30 × 30 map.

Evaluation Criteria	PACO	ACO in [41]	Traditional ACO	NPACO
Average total time /s	7.2252	17.97	26.92	13.5880
Shortest path length /m	37.3848	37.3849	38.2133	37.3848
Average path length /m	37.9173	38.1262	38.6325	37.3848



(a) Convergence curve



(b) Optimal path trajectory

Figure 8. Convergence curves and optimal path trajectories of the four algorithms on 30 × 30 map.

Figure 8(a) indicates the effectiveness of the PACO algorithm on the 30 × 30 map. Moreover, optimal path trajectories are shown in Figure 8(b), with the same line type representation as that on the 20 × 20 map environment.

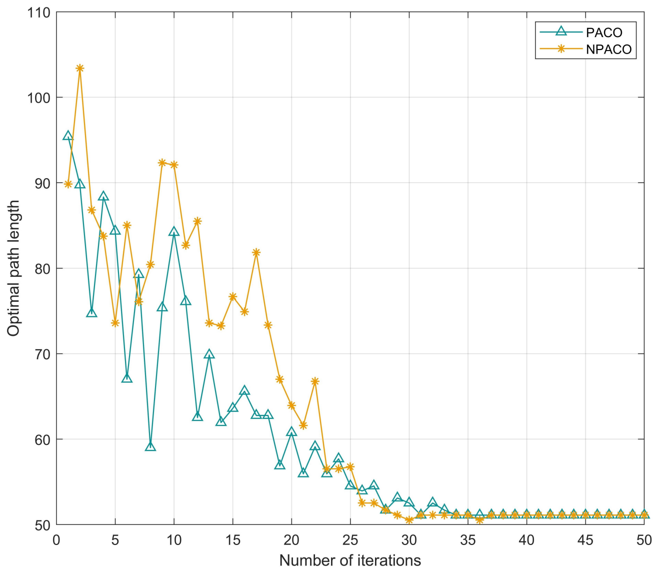
5.2.3. In 40 × 40 map environment

The third comparative experiment was on a 40 × 40 grid map with the starting and target points (5.5, 34.5) and (28.5, 5.5). The auxiliary point obtained by PACO is (11.5, 20.5). The experimental results are as follows:

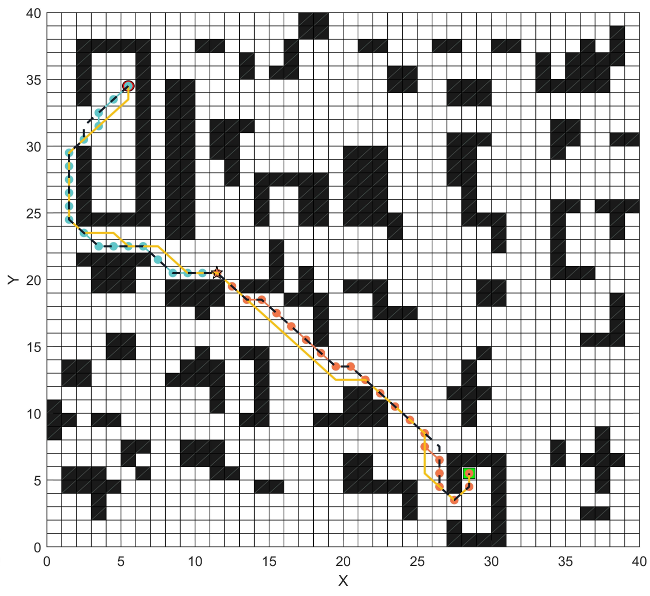
From Table 4, it can be observed that the average total time for PACO and NPACO is 22.8895 and 42.4130, respectively. The former is reduced by 46.03% compared to the latter. For the shortest path length, the results of the other three algorithms are almost equal, except for traditional ACO which has not obtained a feasible solution. For the average path length, apart from traditional ACO, the best and worst results of the other three algorithms are 51.7023 of NPACO and 52.5414 of PACO, respectively. Based on these three indicators, the effectiveness of PACO is once again demonstrated.

Table 4. Experimental results on 40 × 40 map.

Evaluation criteria	**PACO**	**ACO in [41]**	**Traditional ACO**	**NPACO**
Average total time /s	22.8895	88.20	-	42.4130
Shortest path length /m	51.1127	51.1128	-	51.1127
Average path length /m	52.5414	51.8471	-	51.7023



(a) Convergence curve



(b) Optimal path trajectory

Figure 9. Convergence curves and optimal path trajectories of the four algorithms on 40 × 40 map.

Figure 9(a) indicates the effectiveness of the PACO algorithm on the 40 × 40 map. Again, optimal path trajectories are shown in Figure 9(b), with the same line type representation as that on the 20 × 20 map environment.

In summary, as the scale of grid maps enlarges, the running time of nonparallel ACO algorithms significantly increases. In particular, on the 40 × 40 grid map, the traditional ACO algorithm is no longer able to obtain effective paths. However, the parallel ACO algorithm proposed in this paper not only has a significantly smaller running time, but also has little loss of solution accuracy, demonstrating good solution performance.

6. Conclusions

This paper proposed a parallel ACO algorithm for the path planning problem of mobile robots. A rank-based pheromone updating method was introduced to balance exploration space and convergence speed, while a hybrid approach of continuing searching and killing directly was constructed to deal with the deadlock problem. In order to quickly implement the path planning in complex environments, the decomposition and parallelism strategies were designed. Simulation experiments showed that the effectiveness of the proposed algorithm was verified. Because the PACO algorithm can greatly reduce computation time with smaller loss of solution accuracy, applying it to relevant practices can save computational resources and provide fast and better decision-making solutions for managers. For future research, one is to design a better decomposition method to split the original problem into two or more subproblems, so that they can be solved in parallel as evenly as possible. The other is to study the path planning problem in dynamic environments.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant number 11701363.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robot. Res.*, **5** (1986), 90–98. https://doi.org/10.1177/027836498600500106

2. X. Li, L. Wang, Y. An, Q. Huang, Y. Cui, H. Hu, Dynamic path planning of mobile robots using adaptive dynamic programming, *Expert Syst. Appl.*, **235** (2023), 121112. https://doi.org/10.1016/j.eswa.2023.121112

3. F. Duchon, A. Babinec, M. Kajan, P. Beno, M. Florek, T. Fico, et al., Path planning with modified A star algorithm for a mobile robot, *Proc. Eng.*, **96** (2014), 59–69. https://doi.org/10.1016/j.proeng.2014.12.098

4. C. Li, X. Huang, J. Ding, K. Song, S. Lu, Global path planning based on a bidirectional alternating search A* algorithm for mobile robots, *Comput. Ind. Eng.*, **168** (2022), 108123. https://doi.org/10.1016/j.cie.2022.108123

5. M. Shayestegan, M. H. Marhaban, Mobile robot safe navigation in unknown environment, in *Proceedings of the 2012 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2012)*, Penang, Malaysia, (2012), 44–49. https://doi.org/10.1109/ICCSCE.2012.6487113

6. J. Wang, Z. Xu, X. Zheng, Z. Liu, A fuzzy logic path planning algorithm based on geometric landmarks and kinetic constraints, *Inf. Technol. Control*, **51** (2022), 499–514. https://doi.org/10.5755/j01.itc.51.3.30016

7. H. Miao, Y. Tian, Dynamic robot path planning using an enhanced simulated annealing approach, *Appl. Math. Comput.*, **222** (2013), 420–437. https://doi.org/10.1016/j.amc.2013.07.022

8. K. Shi, Z. Wu, B. Jiang, H. R. Karimi, Dynamic path planning of mobile robot based on improved simulated annealing algorithm, *J. Frankl. Inst. Eng. Appl. Math.*, **360** (2023), 4378–4398. https://doi.org/10.1016/j.jfranklin.2023.01.033

9. A. Tuncer, M. Yildirim, Dynamic path planning of mobile robots with improved genetic algorithm, *Comput. Electr. Eng.*, **38** (2012), 1564–1572. https://doi.org/10.1016/j.compeleceng.2012.06.016

10. T. Zhang, G. Xu, X. Zhan, T. Han, A new hybrid algorithm for path planning of mobile robot, *J. Supercomput.*, **78** (2022), 4158–4181. https://doi.org/10.1007/s11227-021-04031-9

11. B. Song, Z. Wang, L. Zou, On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm, *Cogn. Comput.*, **9** (2017), 5–17. https://doi.org/10.1007/s12559-016-9442-4

12. Q. Yuan, R. Sun, X. Du, Path planning of mobile robots based on an improved particle swarm optimization algorithm, *Processes*, **11** (2023), 26. https://doi.org/10.3390/pr11010026

13. Q. Luo, H. Wang, Y. Zheng, J. He, Research on path planning of mobile robot based on improved ant colony algorithm, *Neural Comput. Appl.*, **32** (2020), 1555–1566. https://doi.org/10.1007/s00521-019-04172-2

14. Y. Shi, H. Zhang, Z. Li, K. Hao, Y. Liu, L. Zhao, Path planning for mobile robots in complex environments based on improved ant colony algorithm, *Math. Biosci. Eng.*, **20** (2023), 15568–15602. https://doi.org/10.3934/mbe.2023695

15. Z. Jin, G. Luo, R. Wen, J. Huang, WOA-AGA algorithm design for robot path planning, *Int. J. Comput. Commun. Control*, **18** (2023), 5518. https://doi.org/10.15837/ijccc.2023.5.5518

16. Y. Dai, J. Yu, C. Zhang, B. Zhan, X. Zheng, A novel whale optimization algorithm of path planning strategy for mobile robots, *Appl. Intell.*, **53** (2023), 10843–10857. https://doi.org/10.1007/s10489-022-04030-0

17. G. Hu, B. Du, G. Wei, HG-SMA: hierarchical guided slime mould algorithm for smooth path planning, *Artif. Intell. Rev.*, **56** (2023), 9267–9327. https://doi.org/10.1007/s10462-023-10398-3

18. L. Zheng, Y. Tian, H. Wang, C. Hong, B. Li, Path planning of autonomous mobile robots based on an improved slime mould algorithm, *Drones Basel*, **7** (2023), 257. https://doi.org/10.3390/drones7040257

19. M. Abdel-Basset, K. A. Eldrandaly, L. A. Shawky, M. Elhoseny, N. M. AbdelAziz, Hybrid computational intelligence algorithm for autonomous handling of COVID-19 pandemic emergency in smart cities, *Sust. Cities Soc.*, **76** (2022), 103430. https://doi.org/10.1016/j.scs.2021.103430

20. X. Dai, Y. Wei, Application of improved moth-flame optimization algorithm for robot path planning, *IEEE Access*, **9** (2021), 105914–105925. https://doi.org/10.1109/ACCESS.2021.3100628

21. C. Li, Q. Si, J. Zhao, P. Qin, A robot path planning method using improved harris hawks optimization algorithm, *Meas. Control*, (2023). https://doi.org/10.1177/00202940231204424

22. C. Cai, C. Jia, Y. Nie, J. Zhang, L. Li, A path planning method using modified harris hawks optimization algorithm for mobile robots, *PeerJ Comput. Sci.*, **9** (2023). https://doi.org/10.7717/peerj-cs.1473

23. R. Kumar, L. Singh, R. Tiwari, Path planning for the autonomous robots using modified grey wolf optimization approach, *J. Intell. Fuzzy Syst.*, **40** (2021), 9453–9470. https://doi.org/10.3233/JIFS-201926

24. Y. Hou, H. Gao, Z. Wang, C. Du, Improved grey wolf optimization algorithm and application, *Sensors*, **22** (2022), 3810. https://doi.org/10.3390/s22103810

25. M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in *New Ideas in Optimization*, McGraw Hill, London, (1999), 11–32.

26. M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, *Artif. Life*, **5** (1999), 137–172. https://doi.org/10.1162/106454699568728

27. M. Dorigo, T. Stützle, *Ant Colony Optimization*, MIT Press, Cambridge, 2004. https://doi.org/10.1109/MCI.2006.329691

28. H. Tseng, C. Chang, S. Lee, Y. Huang, Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning, *Eng. Appl. Artif. Intell.*, **83** (2019), 45–56. https://doi.org/10.1016/j.engappai.2019.04.015

29. Y. Wang, L. Wang, G. Chen, Z. Cai, Y. Zhou, L. Xing, An improved ant colony optimization algorithm to the periodic vehicle routing problem with time window and service choice, *Swarm Evol. Comput.*, **5**5 (2020), 100675. https://doi.org/10.1016/j.swevo.2020.100675

30. W. Gao, Modified ant colony optimization with improved tour construction and pheromone updating strategies for traveling salesman problem, *Soft Comput.*, **25** (2021), 3263–3289. https://doi.org/10.1007/s00500-020-05376-8

31. W. Deng, L. Zhang, X. Zhou, Y. Zhou, Y. Sun, W. Zhu, et al., Multi-strategy particle swarm and ant colony hybrid optimization for airport taxiway planning problem, *Inf. Sci.*, **612** (2022), 576–593. https://doi.org/10.1016/j.ins.2022.08.115

32. D. B. M. M. Fontes, S. M. Homayouni, J. F. Gonçalves, A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources, *Eur. J. Oper. Res.*, **306** (2023), 1140–1157. https://doi.org/10.1016/j.ejor.2022.09.006
33. T. T. Erguzel, C. Tas, M. Cebi, A wrapper-based approach for feature selection and classification of major depressive disorder-bipolar disorders, *Comput. Biol. Med.*, **64** (2015), 127–137. https://doi.org/10.1016/j.compbiomed.2015.06.021
34. A. Qi, D. Zhao, F. Yu, A. A. Heidari, Z. Wu, Z. Cai, et al., Directional mutation and crossover boosted ant colony optimization with application to COVID-19 X-ray image segmentation, *Comput. Biol. Med.*, **148** (2022), 105810. https://doi.org/10.1016/j.compbiomed.2022.105810
35. K. Akka, F. Khaber, Mobile robot path planning using an improved ant colony optimization, *Int. J. Adv. Robot. Syst.*, **15** (2018), 1729881418774673. https://doi.org/10.1177/1729881418774673
36. X. You, S. Liu, C. Zhang, An improved ant colony system algorithm for robot path planning and performance analysis, *Int. J. Robot. Autom.*, **33** (2018), 527–533. https://doi.org/10.2316/Journal.206.2018.5.206-0071
37. M. Dorigo, L. M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.*, **1** (1997), 53–66. https://doi.org/10.1109/4235.585892
38. W. Gao, Q. Tang, B. Ye, Y. Yang, J. Yao, An enhanced heuristic ant colony optimization for mobile robot path planning, *Soft Comput.*, **24** (2020), 6139–6150. https://doi.org/10.1007/s00500-020-04749-3
39. X. Deng, L. Zhang, L. Luo, An improved ant colony optimization applied in robot path planning problem, *J. Comput.*, **8** (2013), 585–593. https://doi.org/10.4304/jcp.8.3.585-593
40. J. Liu, J. Yang, H. Liu, X. Tian, M. Gao, An improved ant colony algorithm for robot path planning, *Soft Comput.*, **21** (2017), 5829–5839. https://doi.org/10.1007/s00500-016-2161-7
41. X. Dai, S. Long, Z. Zhang, D. Gong, Mobile robot path planning based on ant colony algorithm with A* heuristic method, *Front. Neurorobotics*, **13** (2019), 15. https://doi.org/10.3389/fnbot.2019.00015
42. Z. Zhang, J. Lu, Z. Xu, T. Xu, Mobile robot path planning based on hybrid ant colony optimization, *J. Intell. Fuzzy Syst.*, **45** (2023), 2611–2623. https://doi.org/10.3233/JIFS-231280
43. G. Li, C. Liu, L. Wu, W. Xiao, A mixing algorithm of ACO and ABC for solving path planning of mobile robot, *Appl. Soft. Comput.*, **148** (2023), 110868. https://doi.org/10.1016/j.asoc.2023.110868
44. B. Bullnheimer, R. F. Hartl, C. Strauss, A new rank based version of the ant system – A computational study, *Central Eur. J. Oper. Res. Econ.*, **7** (1999), 25–38.
45. D. Wang, H. Yu, Path planning of mobile robot in dynamic environments, in *Proceedings of the 2011 2nd International Conference on Intelligent Control and Information Processing (ICICIP 2011)*, (2011), 691–696. https://doi.org/10.1109/ICICIP.2011.6008338
46. H. Qu, L. Huang, X. Ke, Research of improved ant colony based robot path planning under dynamic environment, *J. Univ. Electron. Sci. Technol. China*, **44** (2015), 260–265. https://doi.org/10.3969/j.issn.1001-0548.2015.02.017

47. J. Cao, S. Fan, X. Yang, Spmd performance analysis with parallel computing of Matlab, in *Proceedings of the 2012 Fifth International Conference on Intelligent Networks and Intelligent Systems (ICINIS 2012)*, (2012), 80–83. https://doi.org/10.1109/ICINIS.2012.31