Research article

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

Jian Si and Xiaoguang Bao

1 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 important topics in the field of mobile robot research. This paper focuses on the study of path planning problem for mobile robot in complex environment based on ant colony optimization (ACO) algorithm. In order to solve the problems of local optimum, susceptibility to deadlocks, low search efficiency in traditional ACO algorithm, a novel parallel ACO (PACO) algorithm is proposed. The algorithm constructs a rank-based pheromone updating method to balance exploration space and convergence speed, and introduces 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 environment, the algorithm first finds a better location for decomposing the original problem into two subproblems, and then solves them using a parallel programming method-single program multiple data (SPMD)-in MATLAB. In different grid map environments, simulation experiments are carried out. The experimental results show that on grid maps with scales of 20 × 20, 30 × 30, and 40 × 40, compared to non parallel ACO algorithms, the proposed PACO algorithm has less loss of solution accuracy but reduces 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 robotic 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 bidirectional alternating search strategy and improved heuristic function and em-
ployed filtering function and Bézier curves, in 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 ini-
tial path selection method and deletion operation, in SA algorithm, to reduce the computational effort.
Zhang et al. [10] combined GA and firefly algorithm (FA) to address the problem of FA easily trapping
into 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 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 perform-
ance of HHO algorithm. Hou et al. [24] used improved chaotic tent mapping, nonlinear convergence
factor based on the Gaussian distribution change curve, and improved dynamic proportional weighting
strategy to enhance the performance of GWO algorithm.
The ACO algorithm [25, 26, 27], inspired by the cooperative foraging behavior of ants in na-
ture, is widely used to solve combinatorial optimization problems (such as disassembly sequence
planning [28], vehicle routing problem [29], traveling salesman problem [30], airport taxiway plan-
ning [31], job shop scheduling [32], feature selection [33], image segmentation [34] ) due to its
advantages of positive feedback, strong robustness, and parallelism. For the path planning of mobile
robot, 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, 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 were 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 algo-
ritm 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,
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 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 environment, a novel parallel ant colony optimization (PACO) algorithm is proposed in this paper. The improvement and innovation of the algorithm include the following three aspects. Firstly, 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. Secondly, 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. Thirdly, 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 robot. 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

![Grid map](image)

Figure 1. Grid map.

A two-dimensional grid map model is used to represent the working environment of the robot. In 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).

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

Here, \(n\) is the grid number, \(N\) is the number of columns in the grid map, \(\text{mod}\) denotes the remainder function, and \(\text{ceil}\) denotes the rounding function towards positive infinity.

### 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.

In 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 next grid \(j\) is calculated by Formula (3.1).

\[
P_{kj}^k(t) = \begin{cases} 
\frac{\tau_{ij}^t \eta_{ij}^t}{\sum_{\substack{s \in N_k(i) \\
s \neq j}} \tau_{is}^t \eta_{is}^t}, & j \in N_k(i) \\
0, & \text{otherwise}
\end{cases} \quad (3.1)
\]

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 \(j\), \(\alpha\) is the pheromone intensity factor, \(\beta\) is the expected heuristic factor, and \(N_k(i)\) is the set of grids that are feasible and have not been visited by ant \(k\).

\[
\eta_{ij}(t) = \frac{1}{d_{ij}} \quad (3.2)
\]

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

\[
\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \quad (3.3)
\]
Here, $\rho$ represents the pheromone evaporation rate, $m$ is the number of ants, and $\Delta r^k_{ij}(t)$ is the amount of pheromone left by ant $k$ on the edge $(i, j)$, which is typically given by Formula (3.4).

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

(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, and 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 $\text{ceil}$ and $L_k$ can be found in Formulas (2.1) and (3.4), respectively.

$$\Delta r^k_{ij}(t) = \begin{cases} \text{ceil} \left( \frac{t}{10\% T} \right) \ast \frac{Q}{L_k}, & \text{if ant } k \text{ is rewarded and passes through edge } (i, j) \\ -\frac{Q}{L_k}, & \text{if ant } k \text{ is punished and passes through edge } (i, j) \\ -0.2 \ast \frac{Q}{L_k}, & \text{if ant } k \text{ is punished and passes through edge } (i, j) \\ 0, & \text{otherwise} \end{cases}$$

(4.1)

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.

Mathematical Biosciences and Engineering
\[ \tau_{ij}(t + 1) = \begin{cases} (1 - \lambda)^{\text{ceil}(\frac{M_{Lost}(i,j)}{M_{Max}})}) \times \tau_{ij}(t), & \text{if edge } (i, j) \text{ in the last two steps} \\ \tau_{ij}(t), & \text{otherwise} \end{cases} \] (4.2)

4.2. Deadlock Handling Strategy

Due to the presence of obstacles and the use of 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 traps in a deadlock state, because the eight adjacent grids are either obstacles or belong to the taboo list.

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 allows to return one step and continue searching for 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 \( \text{ceil} \) 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.

Figure 3. Status of deadlock.
4.3. Decomposition and Parallelism Strategies

In order to efficiently complete the path planning in complex environment, a strategy of decomposing the original problem into two subproblems is proposed. The key to the strategy is the selection of 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.

Then 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 then 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 parallel program in a multiprocessor environment, in which single program multiple data (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 $N_{\text{start}}$ and $N_{\text{target}}$, 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 $\alpha$, expected heuristic factor $\beta$, and deadlock penalty factor $\lambda$ are initialized.

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

Step 4. Execute the formal stage. Activate two CPU cores again and deal with the subproblems from $N_{\text{start}}$ to $N_{\text{auxiliary}}$ and from $N_{\text{auxiliary}}$ to $N_{\text{target}}$ respectively, using the proposed new relevant strate-
gies. Set the obtained subpaths as $\text{Path}_1$ and $\text{Path}_2$, respectively. Meanwhile, set the time spent as $t_2$.

Step 5. Output the final path $\text{Path}$ by connecting $\text{Path}_1$ and $\text{Path}_2$ and obtain the algorithm’s total running time $t_{\text{total}} = t_1 + t_2$.

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 non parallel ACO algorithms is $O(N^2 \times m \times T)$. 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 $O(N^2 \times m \times T_0)$ in the preprocessing stage and $O((\frac{N}{2})^2 \times m \times (T - T_0))$ in the formal stage, which is $O(N^2 \times m \times (\frac{T}{4} - \frac{3T_0}{4}))$. 
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 algorithm. 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 were determined. The values of these parameters are shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>T₀</th>
<th>M</th>
<th>Q</th>
<th>α</th>
<th>β</th>
<th>λ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>3</td>
<td>50</td>
<td>1</td>
<td>1.0</td>
<td>7.0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

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

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, with 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 strategies on pheromone update methods, 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 was 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 blue triangle and red star represent the results obtained by our 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.
We now present a validation of the effectiveness of the proposed deadlock handling strategy. Because our method for dealing with deadlock problem is based on that in Luo et al. [13], two strategies were 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 blue triangle and red star represent the results obtained by our and Luo et al.’s strategies, respectively. From Figure 6, it can be observed that: on the whole, the curve of 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, the greater the probability of finding a better feasible path, indicating that the proposed method is effective.
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 non parallel ACO (NPACO) algorithm, Dai et al.’s ACO algorithm in [41], and 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.

Figure 6. Comparisons of the number of lost ants between our and Luo et al.’s strategies.
5.2.1. In $20 \times 20$ map environment

The first comparative experiment was conducted on a $20 \times 20$ grid map, where the starting and target points of the path planning problem were $(0.5, 19.5)$ and $(19.5, 0.5)$ respectively. The auxiliary point obtained by executing the preprocessing stage of the proposed PACO algorithm in this paper is $(10.5, 10.5)$. 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 shorter running time, but also ensures better solution accuracy and convergence speed, further verifying its effectiveness.

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>PACO</th>
<th>ACO in [41]</th>
<th>Traditional ACO</th>
<th>NPACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average total time /s</td>
<td>1.6300</td>
<td>4.89</td>
<td>9.22</td>
<td>3.3070</td>
</tr>
<tr>
<td>Shortest path length /m</td>
<td>29.2132</td>
<td>29.2133</td>
<td>37.4143</td>
<td>29.2132</td>
</tr>
<tr>
<td>Average path length /m</td>
<td>29.2132</td>
<td>29.3807</td>
<td>38.6335</td>
<td>29.2132</td>
</tr>
</tbody>
</table>

From Figure 7a, 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 7b, 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 $30 \times 30$ map environment

The second comparative experiment was on a $30 \times 30$ 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 $30 \times 30$ map.

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>PACO</th>
<th>ACO in [41]</th>
<th>Traditional ACO</th>
<th>NPACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average total time /s</td>
<td>7.2252</td>
<td>17.97</td>
<td>26.92</td>
<td>13.5880</td>
</tr>
<tr>
<td>Shortest path length /m</td>
<td>37.3848</td>
<td>37.3849</td>
<td>38.2133</td>
<td>37.3848</td>
</tr>
<tr>
<td>Average path length /m</td>
<td>37.9173</td>
<td>38.1262</td>
<td>38.6325</td>
<td>37.3848</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>PACO</th>
<th>ACO in [41]</th>
<th>Traditional ACO</th>
<th>NPACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average total time /s</td>
<td>22.8895</td>
<td>88.20</td>
<td>-</td>
<td>42.4130</td>
</tr>
<tr>
<td>Shortest path length /m</td>
<td>51.1127</td>
<td>51.1128</td>
<td>-</td>
<td>51.1127</td>
</tr>
<tr>
<td>Average path length /m</td>
<td>52.5414</td>
<td>51.8471</td>
<td>-</td>
<td>51.7023</td>
</tr>
</tbody>
</table>

Figure 9a indicates the effectiveness of PACO algorithm on the 40 × 40 map. Again, optimal path trajectories are shown in Figure 9b, 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 non parallel ACO algorithms significantly increases. In particular, on the $40 \times 40$ 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 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 robot. 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 environment, the decomposition and parallelism strategies are designed. Simulation experiments show that the effectiveness of the proposed algorithm is 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 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 environment.

Acknowledgments

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

Figure 9. Convergence curves and optimal path trajectories of the three algorithms on $40 \times 40$ map.
Conflict of interest

The authors declare there is no conflict of interest.

References


© the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)