Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots

TAN Guan-Zheng[1,3] HE Huan[2] SLOMAN Aaron[3]

Abstract A novel method for the real-time globally optimal path planning of mobile robots is proposed based on the ant colony system (ACS) algorithm. This method includes three steps: the first step is utilizing the MAKLINK graph theory to establish the free space model of the mobile robot, the second step is utilizing the Dijkstra algorithm to find a sub-optimal collision-free path, and the third step is utilizing the ACS algorithm to optimize the location of the sub-optimal path so as to generate the globally optimal path. The result of computer simulation experiment shows that the proposed method is effective and can be used in the real-time path planning of mobile robots. It has been verified that the proposed method has better performance in convergence speed, solution variation, dynamic convergence behavior, and computational efficiency than the path planning method based on the genetic algorithm with elitist model.

Key words Mobile robot, globally optimal path planning, ACS algorithm, MAKLINK graph, Dijkstra algorithm

1 Introduction

The globally optimal path planning is an important problem in navigation of autonomous mobile robots, which is to find an optimal collision-free path from a starting point to a goal in a given environment according to some criterion. For this problem, there are many solving methods now, such as the potential field methods[1], visibility graph methods[2], and grid methods[3]. The potential field methods have simpler structures and are used in real-time obstacle avoidance. But, this kind of methods has some inherent limitations, including trap situations due to local minima, no passage between closely spaced obstacles, oscillations in the presence of obstacles, and oscillations in narrow passages. The main problem of visibility graph methods is that they have more complicated search paths and lower search efficiency. In the grid methods where grids are used to form the map of the environment, the main problem is how to determine the size of grids. The smaller the size of grids, the more precise the representation of the environment. However, using smaller grids will result in exponential increase in memory space and search range. In recent years, the research on artificial intelligence has advanced rapidly, and many intelligent algorithms have been applied to the path planning of mobile robots, such as the fuzzy logic and reinforcement learning[4], neural networks[5], genetic algorithms[6], particle swarm optimization[7], and so on.

Ant system (AS) algorithm is a novel simulated evolutionary algorithm[8]. Its main characteristics include positive feedback search mechanism, distributed computation, and the use of a constructive greedy heuristic. So far, AS algorithm has been used successfully to solve many practical optimization problems, such as the traveling salesman problem[8], the quadratic assignment problem[9], the discrete optimization problem[10], the optimal controller design[11], and so on. The ant colony system (ACS) algorithm is an improvement of AS algorithm[12], and is more robust, faster, and has a better probability in achieving the globally optimal solution.

The global path planning of a mobile robot includes two sub-problems: the free space modeling and the path finding. In this paper, the MAKLINK graph theory is employed to establish the free space model, then the Dijkstra algorithm is used to find a sub-optimal collision-free path, finally the established ACS algorithm is utilized to optimize the location of the sub-optimal path so as to generate the globally optimal path.

2 Free space modeling of mobile robot

To use the MAKLINK graph theory[13] to establish the free space model, the following assumptions need to be made: 1) the heights of the environment and obstacles can be ignored; 2) there exist some known obstacles distributed in the environment, both the environment and the obstacles have a polygonal shape; 3) in order to avoid a moving path too close to the obstacles, the boundaries of every obstacle can be expanded by an amount that is equal to half of the greater size in the length and width of the robot′s body plus the minimum measuring distance of the relevant sensors. In this case, the size of the robot can be ignored.

Fig. 1 illustrates a moving environment of a mobile robot, which is a 350 by 350 meters square and includes six obstacles. Points S and T denote the starting point and the goal respectively.

For each of the obstacles, the black polygon denotes its original size and the white margin denotes its expanded part. A black part plus its white margin constitutes a so-called "grown obstacle". In Fig. 1, symbols B_1, B_2, ..., and B_{23} denote respectively the vertices of these grown obstacles. The (x, y) coordinates of vertices B_1, B_2, ..., and B_{23} are (40, 288), (66, 288), (40, 151), (66, 151), (115, 275), (95, 214), (123, 163), (170, 245), (90, 106), (90, 45), (183, 45), (183, 106), (238, 311), (212, 248), (274, 268), (258, 205), (234, 190), (234, 111), (296, 111), (296, 137), (170,170), (190,150), and (210,170), respectively. The (x, y) coordinates of the starting point S and the goal T are(15, 335) and (315, 35), respectively.

The free space of a mobile robot in an environment means the space in which the robot can move freely. A free space consists of some polygonal areas, each of which is enclosed by several free MAKLINK lines. A free MAKLINK line is defined as: 1) either its two end points are two vertices on two different grown obstacles or one point is a

Received August 31, 2005; in revised form August 28, 2006
Supported by National Natural Science Foundation of P. R. China (50275150) and National Research Foundation for the Doctoral Program of Higher Education of P. R. China (20040533035)
1. School of Information Science and Engineering, Central South University, Changsha 410083 P. R. China 2. Center for Space Science and Applied Research, Chinese Academy of Sciences, Beijing 100080 P. R. China 3. School of Computer Science, The University of Birmingham, Birmingham B15 2TT, The United Kingdom
DOI: 10.1360/aas-007-0279

vertex of a grown obstacle and the other is located on a boundary of the environment; 2) every free MAKLINK line cannot intersect any of the grown obstacles.

Using the MAKLINK graph theory to establish the free space model of a mobile robot means finding the free MAKLINK line/lines of every vertex on every grown obstacle. This process can be described as follows: 1) find out the peripheral boundaries of all the grown obstacles, which are free MAKLINK lines, for example, lines B_1B_{13}, $B_{15}B_{20}$, $B_{19}B_{11}$, and $B_{10}B_3$ in Fig. 1; 2) the perpendicular line from each vertex of the peripheral boundaries to its adjacent boundary of the environment is a free MAKLINK line; find out all of this kind of free MAKLINK lines; 3) find out all other free MAKLINK lines according to the definition of free MAKLINK line. Fig. 1 also shows the MAKLINK graph of this example.

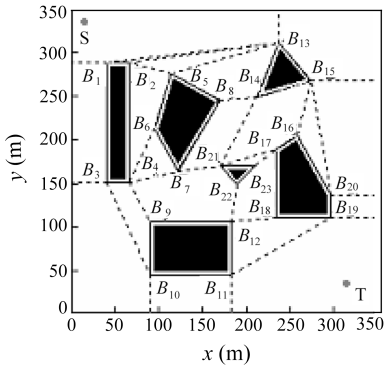


Fig. 1 An environment and its MAKLINK graph

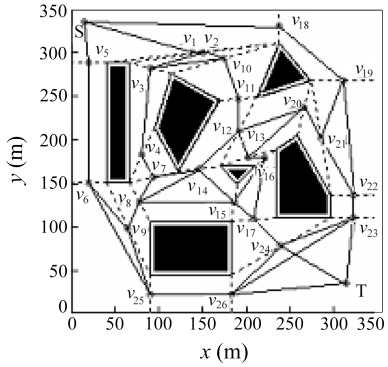


Fig. 2 Network graph for free motion of robot

Assume that symbol l denotes the total number of the free MAKLINK lines on a MAKLINK graph, the middle points of these free MAKLINK lines are denoted respectively by v_1, v_2, ..., v_l. If each pair of the middle points on two adjacent free MAKLINK lines are connected together, a network graph can be formed, which gives the possible paths for the free motion of the mobile robot. For the example shown in Fig. 1, the free motion network graph of the robot is shown in Fig. 2, where $l = 26$.

Points S and T are also denoted by v_0 and v_{l+1}, respectively. Fig. 2 is an undirected graph, and denoted by $G(V, E)$, where $V=\{v_0, v_1, \ldots, v_{l+1}\}$ is the set including points S and T and the middle points of all the free MAKLINK lines; E is a set of the lines which includes: the lines, each of which connects a pair of the middle points on two adjacent free MAKLINK lines; the lines connecting point S and the middle points on the free MAKLINK lines adjacent to S; and the lines connecting the goal T and the middle points on the free MAKLINK lines adjacent to T. Using the undirected graph $G(V, E)$ as the free space model, the globally optimal path planning can be solved by finding the shortest path between the given starting point S and goal Ton $G(V, E)$.

3 Searching for sub-optimal path using Dijkstra algorithm

The Dijkstra algorithm is widely used to search for the shortest path on a network graph [14]. In this section, this algorithm is used to find a sub-optimal path between the starting point S and goal T.

When using the Dijkstra algorithm, it is needed to calculate the cost function of a path. In general, each edge on a path will be given a weight, so the cost function can be defined as the sum of the weights of all the edges of the path. Here, the length of an edge is used as its weight.

Before using the Dijkstra algorithm, it is necessary to define the adjacency matrix with weights for the network graph $G(V, E)$, which is the basis for computing the shortest path. Each element of the matrix represents the length of the straight line between two adjacent path nodes on $G(V, E)$, where a path node means the intersection of the robot path and a free MAKLINK line. For the problem discussed here, each element of the adjacency matrix is defined as

$$adjlist[i][j] = \begin{cases} length(v_i, v_j), & \text{if } edge(v_i, v_j) \in E \\ \infty, & \text{others} \end{cases} \quad (1)$$

where *adjlist* $[i][j]$ is the element corresponding to the ith row and the jth column of the matrix, *length* (v_i, v_j) is the straight-line distance between the path nodes v_i and v_j, i and $j = 0, 1, 2, \ldots, l, l+1$. For the example given in Fig. 1, using the Dijkstra algorithm the sub-optimal path is obtained as S$\rightarrow v_1 \rightarrow v_2 \rightarrow v_{10} \rightarrow v_{11} \rightarrow v_{12} \rightarrow v_{13} \rightarrow v_{16} \rightarrow v_{17} \rightarrow v_{24} \rightarrow T$, which is shown in Fig. 3. The length of this path is 507.692 meters.

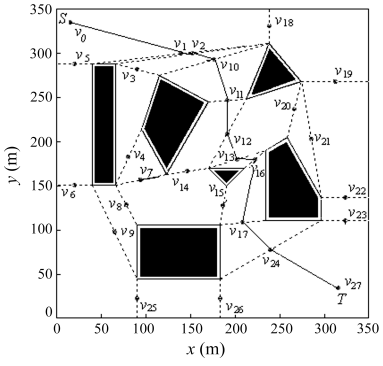


Fig. 3 Sub-optimal path generated by Dijkstra algorithm

4 Globally optimal path searching method based on ACS algorithm

Utilizing the Dijkstra algorithm, we obtain a feasible robot path between the starting point S and the goal T

and all the free MAKLINK lines which this path passes through. But, this path is just a sub-optimal path because it passes only through the middle points of those free MAKLINK lines. In this section, we will adopt the ant colony system(ACS) algorithm to adjust and optimize the locations of the path nodes on this sub-optimal path so as to generate the globally optimal path.

4.1 Description of optimization problem

Assume that the sub-optimal path generated by the Dijkstra algorithm is denoted in order by path nodes P_0, P_1, P_2, ..., P_d, and P_{d+1}, where P_0 and P_{d+1} denote the starting point S and the goal T respectively. At the beginning, these path nodes lie on the middle points of the relevant free MAKLINK lines. Now, we need to adjust and optimize their locations on their corresponding free MAKLINK lines. The adjustment method is described as follows. Introducing a parameter h_i, the location of P_i on its free MAKLINK line $P_{i1}P_{i2}$ can be expressed as

$$P_i = P_{i1}+(P_{i2}-P_{i1})\times h_i, \quad h_i \in [0,1], \quad i = 1,2,\ldots,d \quad (2)$$

Obviously, given a set of values to $\{h_1, h_2, \ldots, h_d\}$, together with the starting point S and the goal T, a new robot path can be generated. The different combinations of parameters h_1, h_2, ..., and h_d will generate the length-different paths. Rewriting P_i as $P_i(h_i)$ to reflect the influence of h_i on the location of P_i, the objective function of the optimization problem can be defined as

$$L = \sum_{i=0}^{d} length\{P_i(h_i), P_{i+1}(h_{i+1})\} \quad (3)$$

where *length* $\{P_i(h_i), P_{i+1}(h_{i+1})\}$ represents the straight-line distance between two adjacent path nodes P_i and P_{i+1}. In (3), when i=0, $P_0(h_0)$ means the starting point S; when $i = d$, $P_{d+1}(h_{d+1})$ means the goal T. Our purpose is to use the ACS algorithm to find the optimal parameter set $\{h_1^*, h_2^*, \ldots, h_d^*\}$ such that L has the minimum value.

4.2 Generating method of ant moving paths

When using the ACS algorithm, a discrete solving space is needed because the path selection number of an ant in each step is limited. In order to use the ACS algorithm conveniently, we express the values of parameters h_1, h_2, ..., and h_d on plane O-XY. As shown in Fig. 4, we draw d lines on O-XY which have equal length and equal interval and are perpendicular to axis X. The x coordinates of these lines are represented by 1, 2, ..., and d in order. Then, we divide equally each of these lines into ten portions, and thus eleven nodes are generated on each line. The y coordinates of the eleven nodes on each line are 0, 0.1, 0.2, ..., and 1.0, which represent the eleven possible values of parameter h_i (i=1, 2, ..., d). Corresponding to the MAKLINK graph, this means each of the relevant continuous free MAKLINK lines is transformed into the eleven discrete path nodes with equal interval. In Fig. 4, the starting point S and the goal T are also marked, which are located on axis X with point S at the origin.

Fig. 4 is a grid graph on which there are $d\times 11$ nodes in total. We use n_{ij} to denote node j on line h_i. Let an ant depart from the starting point S. In its each step forward, it chooses a node from the next line h_i (i=1, 2, ..., d) and then moves to this node along the straight line. When it arrives at the goal T, it completes one tour. Its moving path can be expressed as $Path$=$\{S, n_{1j}, n_{2j}, \ldots, n_{dj}, T\}$. Fig. 4 shows a moving path of an ant.

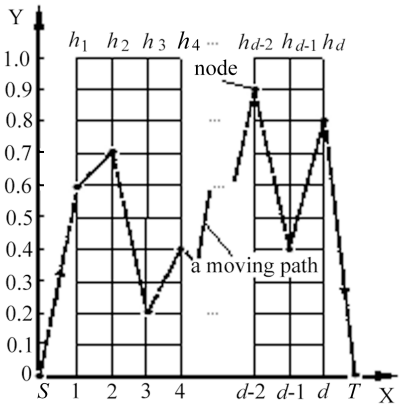


Fig. 4 Generating of nodes and moving paths

4.3 Selection of nodes on a moving path

For an ant colony, assume that from any node on line h_i to any node on the next line h_{i+1}, each ant has the same moving time. So, if all ants depart from the starting point S at the same time, they will arrive on each line h_i (i=1, 2, ..., d) at the same time too, and finally arrive at the goal T at the same time. To this moment, they all complete a tour, and the ACS algorithm completes an iteration. Let $\tau_{ij}(t)$ represent the concentration of pheromone at node n_{ij}, where t is the iteration counter of the ACS algorithm. Assume that at initial time t =0 all the nodes have the same pheromone concentration τ_0, that is, $\tau_{ij}(0)$= τ_0 (i=1, 2, ..., d; j=0, 1, 2, ..., 10).

Assume the number of ants is m. In moving process, for an ant k, when it locates on line h_{i-1}, it will choose a node j from the eleven nodes of the next line $h_i$to move to according to the following random transition rule [12]:

$$j = \begin{cases} \arg\ \max_{u\in A}\left\{[\tau_{iu}(t)]\cdot[\eta_{iu}]^{\beta}\right\}, & if\ q \le q_0 \\ J \qquad\qquad\qquad\qquad\quad , & if\ q > q_0 \end{cases} \quad (4)$$

where A represents the set: {0, 1, 2, ..., 10}; $\tau_{iu}(t)$ is the pheromone concentration of node n_{iu}; η_{iu} represents the visibility of node n_{iu} and is computed by the following (6); β is an adjustable parameter which controls the relative importance of visibility η_{iu} versus pheromone concentration $\tau_{iu}(t)$; q is a random variable uniformly distributed over [0, 1]; q_0 is a tunable parameter ($0\le q_0 \le 1$); and J is a node which is selected according to the following probability formula[12] and "Roulette Wheel Selection Method":

$$p_{iJ}^k(t) = \frac{[\tau_{iJ}(t)]\cdot[\eta_{iJ}]^{\beta}}{\sum_{w=0}^{10}[\tau_{iw}(t)]\cdot[\eta_{iw}]^{\beta}} \quad (5)$$

For the visibility η_{ij} of noden_{ij}, we define it as

$$\eta_{ij} = \frac{1.1 - |y_{ij} - y_{ij}^*|}{1.1} \quad (6)$$

where y_{ij} is the y coordinate of node n_{ij}; the values of y_{ij}^* are set in the following way. In the first iteration of the ACS algorithm, the values of y_{ij}^* are set respectively to the values of parameters h_1, h_2, ..., and h_d which are corresponding to the path nodes on the free MAKLINK lines found by the Dijkstra algorithm, from Section 3 we know all of them are equal to 0.5. In each of the following iterations, the values of y_{ij}^* are set to the values of parameters

h_1, h_2, ..., and h_d which are corresponding to the path nodes on the optimal robot path generated by the ACS algorithm in the previous iteration.

4.4 Updating rules of pheromone concentration

After each iteration, the pheromone concentration of each node on the best ant tour T^+ generated since the beginning of the trial needs to be updated using the following global pheromone updating rule to encourage the ants to search for the paths in the vicinity of the best tour T^+.

$$\tau_{ij}(t) \leftarrow (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\Delta\tau_{ij}(t) \tag{7}$$

where n_{ij} is a node belonging to T^+; $0<\rho<1$ is the pheromone decay parameter; and $\Delta\tau_{ij}(t)$ is computed by

$$\Delta\tau_{ij}(t) = 1/L^+ \tag{8}$$

where L^+ is the length of robot path corresponding to T^+.

When an ant passes through a node n_{ij}, the pheromone concentration of node n_{ij} needs to be updated immediately using the following local pheromone updating rule:

$$\tau_{ij}(t) \leftarrow (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\tau_0 \tag{9}$$

When a node is visited several times by ants, repeatedly applying the local updating rule will make the pheromone level of this node diminish. This has the effect of making the visited nodes less and less attractive to the ants, which indirectly favors the exploration of not yet visited nodes.

4.5 Globally optimal path searching algorithm

The ACS algorithm for finding the globally optimal path of a mobile robot between a given starting point S and a given goal T in a given environment can be summarized as follows.

Step 1. Establish the free space model of the robot using the MAKLINK graph theory, and then find a sub-optimal collision-free path, $S \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_d \rightarrow T$, in the free space using the Dijkstra algorithm.

Step 2. Define the number of ants m; specify the values of parameters β, q_0, ρ, and τ_0; for each ant k ($k = 1, 2, \ldots, m$), define a one-dimensional array $Path_k$ with d elements, in which the y coordinates of the d nodes (not including points S and T) that ant k will pass through in each iteration of the algorithm will be stored in order. Array $Path_k$ can be used to denote the moving path of ant k.

Step 3. Set the iteration counter $t = 1$ and define the maximum number of iterations NC; then place all the m ants at the starting point S.

Step 4. Set $i = 1$.

Step 5. Set $k = 1$.

Step 6. Select a node on line h_i for ant k using (4), (5), and (6), move ant k to this node, and save the y coordinate of this node into the ith element of $Path_k$; then, update the pheromone concentration of this node using the local pheromone updating rule.

Step 7. Set $k \leftarrow k+1$. If $k \leq m$, go to Step 6; otherwise, continue.

Step 8. Set $i \leftarrow i+1$. If $i \leq d$, go to Step 5; otherwise, continue.

Step 9. Move each ant from its location on line h_d to the goal T.

Step 10. For each ant k ($k = 1, 2, \ldots, m$): a) according to array $Path_k$, obtain the values of parameter set $\{h_1^k, h_2^k, \ldots, h_d^k\}$; b) according to $\{h_1^k, h_2^k, \ldots, h_d^k\}$ determine the locations of path nodes P_1^k, P_2^k, ..., and P_d^k on their corresponding free MAKLINK lines; and c) use (3) to computer the length L_k of the robot path, which is corresponding to the moving path of ant k.

Step 11. Compare the obtained m paths and find the shortest robot path T^t in the current iteration t. Compare T^t with T^+, the shortest robot path generated from the beginning of the trial till the previous iteration $t-1$ (for the first iteration, directly denote its shortest path by T^+), and denote the new shortest robot path by T^+, then save the values of h parameter set corresponding to T^+ into $\{h_1^*, h_2^*, \ldots, h_d^*\}$.

Step 12. Set each element of $Path_k$ to zero, k=1, 2, ..., m.

Step 13. Update the pheromone concentration of each node on T^+ using the global pheromone updating rule.

Step 14. Set $t \leftarrow t+1$. If $t < NC$ and all of the m ants do not make the same tour, place all the ants at point S and return to Step 4. If $t < NC$ but all of the m ants make the same tour or $t = NC$, then output the optimal (shortest) robot path$T^+$and its corresponding parameter set $\{h_1^*, h_2^*, \ldots, h_d^*\}$, and stop.

4.6 Parameter setting methods in ACS algorithm

The selections of β, q_0, and ρ influence the convergence speed of the ACS algorithm and the quality of the final solution. By observing the results of simulation experiments with different β, q_0, and ρ, their values were finally determined as β =2, q_0= 0.85, and ρ =0.1.

Using too many ants will increase the opportunities that every node on the grid graph of Fig. 4 is passed many times by the ant colony. This makes the pheromone concentration of every node basically have a uniform change. In this case, although the stochastic search ability of the ACS algorithm would be enhanced, its convergence speed may slow down. However, if we use too few ants, the colony of ants would not produce good cooperation, which may weaken the global search capability of the ACS algorithm and easily lead the algorithm to a premature convergence. For the selection of total number of ants, Dorido and Gambardella proposed the following formula[12]:

$$m = \frac{\log(\varphi_1 - 1) - \log(\varphi_2 - 1)}{q_0 \cdot \log(1-\rho)} \tag{10}$$

where φ_1 and φ_2 are two parameters such that $\varphi_1\tau_0$ and $\varphi_2\tau_0$ denote the average values of pheromone concentrations of all nodes on the best ant tour T^+ before and after using the global updating rule, respectively. From their experiments Dorido and Gambardella found that the ACS algorithm works well when the ratio $(\varphi_1-1)/(\varphi_2-1)\approx 0.4$. Substituting q_0=0.85 and ρ=0.1 into (10), we obtain m=10.

The selection of τ_0 is related to $\Delta\tau_{ij}(t)$. In order to implement the pheromone concentration-based positive feedback search mechanism, it must be guaranteed that the pheromone concentration of each node on the best ant tour T^+ should be reinforced after the pheromone concentrations of these nodes are updated by the global updating rule. Based on this consideration, according to (7), we have

$$(1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\Delta\tau_{ij}(t) > \tau_{ij}(t), \quad 1 \leq t \leq NC \tag{11}$$

where $\tau_{ij}(t)$ is the pheromone concentration of node n_{ij} before using the global updating rule and NC is the defined maximum number of iterations. Substituting $\Delta\tau_{ij}(t) = 1/L^+$ into (11), we obtain

$$\tau_{ij}(t) < 1/L^+, \quad 1 \leq t \leq NC \tag{12}$$

When $t = 1$, because $\tau_{ij}(1) = \tau_0$, we have

$$\tau_0 < 1/L^+ \tag{13}$$

Formula (13) just gives the rough principle of determining τ_0. For the path planning problem considered here, the results of many simulation experiments show that the ACS algorithm has better performance when setting $\tau_0 = (m \times L_{sub})^{-1}$, where m is the number of ants and L_{sub} is the length of the sub-optimal robot path. For the example given in Fig.1, $m = 10$ and $L_{sub} = 507.692$, so we obtain $\tau_0 = 0.0002$.

5 Simulation results and comparison of ACS algorithm and GA with elitist model

5.1 Simulation result based on ACS algorithm

In order to examine the performance of the proposed ACS algorithm, for the example given in Fig. 1, a simulation experiment was executed on a personal computer with 2.66-GHz CPU and 256-MB RAM. The parameters were set as $\beta = 2$, $q_0 = 0.85$, $\rho = 0.1$, $\tau_0 = 0.0002$, $m = 10$. The maximum number of iterations NC was set to 200. The simulation result is shown in Fig. 5.

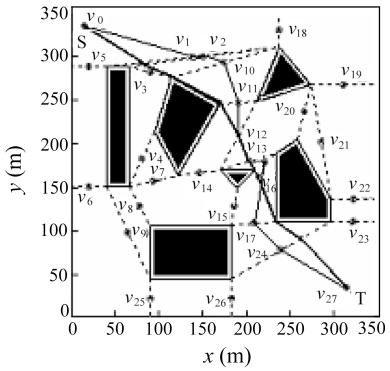


Fig. 5 Computer simulation result using ACS algorithm

In this figure, the thin solid line denotes the sub-optimal robot path with the length of 507.692 meters; the bold solid line denotes the globally optimal robot path with the length of 440.233 meters, which was obtained from the ACS algorithm. In this example, d is equal to 9, and the path nodes needed to be adjusted, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, and P_9, are located respectively on the free MAKLINK lines with the points v_1, v_2, v_{10}, v_{11}, v_{12}, v_{13}, v_{16}, v_{17}, and v_{24}. The optimal h parameter set for this example is $\{h_1^*,\ h_2^*,\ h_3^*,\ h_4^*,\ h_5^*,\ h_6^*,\ h_7^*,\ h_8^*,\ h_9^*\} = \{0.2, 0.1, 0.0, 0.0, 0.5, 0.5, 0.0, 1.0, 0.7\}$.

It should be pointed out the globally optimal path generated by the ACS algorithm is closely related to the number of dividing portions of each of the free MAKLINK lines which the optimal path passes through. The more the portions, the better the obtained result. For example, for the above example, if each of the free MAKLINK lines is divided into twenty equal portions, the length of the optimal robot path generated by the ACS algorithm is 439.372 meters and the corresponding optimal h parameter set $\{h_1^*,\ h_2^*,\ h_3^*,\ h_4^*,\ h_5^*,\ h_6^*,\ h_7^*,\ h_8^*,\ h_9^*\} = \{0.20, 0.10, 0.00, 0.00, 0.50, 0.45, 0.00, 1.00, 0.65\}$. Obviously, this result is better than 440.233 meters, the result with only ten equal portions on each of the free MAKLINK lines. This is because if each of the free MAKLINK lines is divided into more portions, the obtained optimal robot path will be closer to its extreme location.

5.2 Comparison of ACS algorithm and real-coded GA with elitist model

As a comparing object, we also performed a simulation experiment for the given example using the real-coded genetic algorithm (real-coded GA) with elitist model[15]. In the GA, an individual was encoded as $\{h_1 \ldots h_i \ldots h_d\}$, where $h_i \in [0, 1]$ and $i = 1, 2, \ldots, d$. The parameters of the GA were specified as: population size $m = 50$, crossover probability $P_c = 0.6$, crossover constant $a = 0.5$, and mutation probability $P_m = 0.05$.

1) Comparison of convergence speed

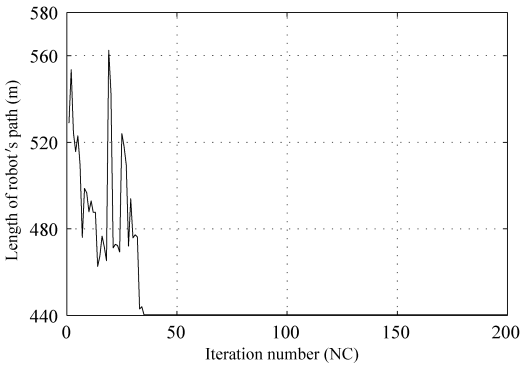


Fig. 6 Convergence tendency of the ACS algorithm

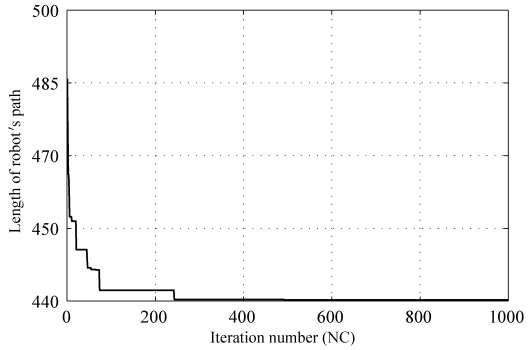


Fig. 7 Convergence tendency of the real-coded GA

Fig. 6 and Fig. 7 show the convergence processes of the optimal solutions generated respectively by the two algorithms. As can be seen, the ACS algorithm generated the very stable optimal solution (440.233 meters) just in around the thirty-fifth iteration, whereas the real-coded GA needed to execute about two hundred and forty iterations to generate the stable optimal solution (440.233 meters). Obviously, the ACS algorithm is much faster than the real-coded GA in convergence speed.

2) Comparison of solution variation

In order to observe the variation in their optimal solutions, we also performed 100 simulation trials for the two methods with different random number. The result is shown in Fig. 8, in which the solid line and the dotted line represent respectively the variations in the optimal so-

lutions of the ACS algorithm and the real-coded GA.

We introduce the following evaluation factor δ, called sample standard deviation:

$$\delta = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(L_i - \overline{L})^2} \tag{14}$$

where L_i is the length of the robot path obtained in the ith trial, and $\bar{L}$ is the average length of the robot paths obtained in N trials.

In $N = 100$ trials, the maximum and minimum robot path lengths obtained from the ACS algorithm are 447.020 meters and 440.233 meters respectively, and the sample standard deviation δ is 1.5644; the maximum and minimum robot path lengths obtained from the real-coded GA are 452.743 meters and 440.233 meters respectively, and the sample standard deviation δ is 1.5227. Although the two methods have almost the same δ value, the maximum robot path length of the real-coded GA is larger than that of the ACS algorithm.

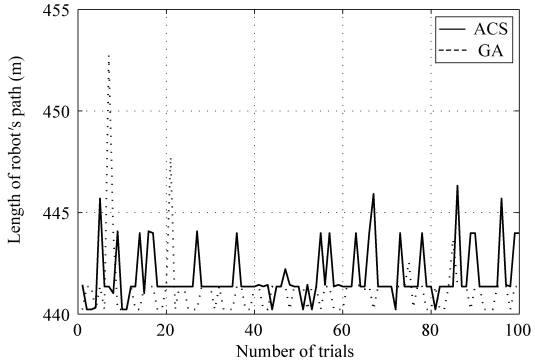


Fig. 8 Comparison of variations in the optimal solutions of both methods (100 trials)

3) Comparison of dynamic convergence behavior

In order to observe the dynamic convergence behavior of the two algorithms during their evolutionary processes, two statistic indexes were used in the simulation programs, that is, the mean value μ and the standard deviation σ. They are defined as

$$\mu = \frac{\sum_{k=1}^{m} L_k}{m} \tag{15}$$

$$\sigma = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(L_k - \mu)^2} \tag{16}$$

where L_k represents the solution, *i.e.*, the robot path length, generated by the k-th individual after an algorithm completes an iteration, and m denotes the total number of individuals in the population.

The mean value μ is the average value of the solutions generated by all individuals in the population after an iteration. The less the μ value, the better the solution generated by each individual. So, μ reflects the accuracy of an algorithm. The standard deviation σ represents the centrality of the solutions generated by all individuals after an iteration. The less the σ value, the better the centrality of the solutions generated by all individuals. So, σ reflects the convergence speed of an algorithm. During the evolutionary processes of the ACS algorithm and the real-coded GA, after each iteration we recorded their mean value μ and the standard deviation σ in the current iteration. Fig. 9 and Fig. 10 show the convergence tendency of μ and σ of the two algorithms during 200 iterations.

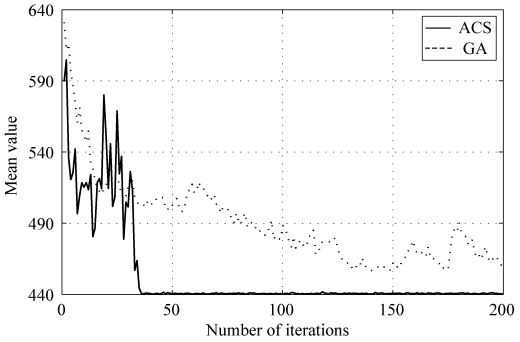


Fig. 9 Convergence tendency of mean value μ using both algorithms

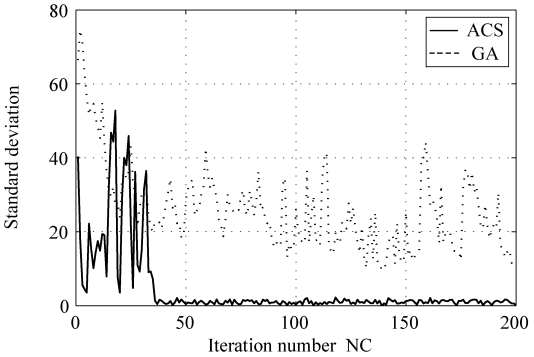


Fig. 10 Convergence tendency of standard deviation σ using both algorithms

As can be seen, during 200 iterations, a) in most iterations, the mean value μ of the ACS algorithm is smaller than that of the real-coded GA, which indicates that the solutions generated by the ACS algorithm are better than that of the real-coded GA; b) in most iterations, the standard deviation σ of the ACS algorithm is smaller than that of the real-coded GA, which indicates that the centrality of the solutions generated by the ACS algorithm in each iteration is better than that of the real-coded GA; c) the mean value μ and the standard deviation σ of the ACS algorithm become very stable after the thirty-fifth iteration, whereas the mean value μ and the standard deviation σ of the real-coded GA still oscillate when achieving the 200th iteration, which indicates that the convergence speed of the ACS algorithm is faster than that of the real-coded GA.

4) Comparison of computational efficiency

Table 1 shows the comparison data on computational efficiency of the two algorithms. As can be seen, the computation efficiency of the ACS algorithm is much higher than that of the real-coded GA. This is because the ACS algorithm has faster convergence speed. For the ACS algorithm, the average CPU time needed for generating the optimal solution is only around one tenth of a second.

Table 1 Comparison of computation efficiency of both algorithms

Algorithm	Average CPU time per iteration (sec.)	Average number of iterations needed for convergence	Average CPU time needed for obtaining optimal solution (sec.)
ACS algorithm	0.00059	175	0.1033
Real-coded GA	0.00067	912	0.6110

6 Conclusion

This paper presents a new globally optimal path planning method for mobile robots based on the ACS algorithm. The results of computer simulation experiments show that this method is effective. The search time needed for generating the globally optimal path is just one tenth of a second, which indicates that this method can be used in the real-time path planning of mobile robots. It has been verified that the proposed ACS algorithm has better performance in convergence speed, solution variation, dynamic convergence behavior, and computational efficiency than the path planning method based on the real-coded genetic algorithm with elitist model.

Acknowledgement

The authors gratefully acknowledge the reviewers for their comments and suggestions which have led to the significant improvements of this paper.

References

1 Ge S S, Cui Y J. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation* , 2000, **16**(5): 615∼620

2 Li L, Ye T, Tan M. Present state and future development of mobile robot technology research. *Robot* , 2002, **24**(5): 475∼480 (in Chinese)

3 Boschian V, Pruski A. Grid modeling of robot cells: a memory-efficient approach. *Journal of Intelligent and Robotic Systems*, 1993, **8**(2): 201∼223

4 Yung N H C, Cang Y. An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 1999, **29**(2): 314∼321

5 Lebedev D. Neural network model for robot path planning in dynamically changing environment. *Modeling and Analysis of Information Systems*, 2001, **18**(1): 12∼18

6 Liu C H, Hu J Q, Qi X N. Path design of robot with continuous space based on hybrid genetic algorithm. *Journal of Wuhan University of Technology* (*Transportation Science & Engineering*), 2003, **27**(6): 819∼821 (in Chinese)

7 Qin Y Q, Sun D B, Li N, Ma Q. Path planning for mobile robot based on particle swarm optimization. *Robot* , 2004, **26**(3): 222∼225 (in Chinese)

8 Dorigo M, Bonabeau E, Theraulaz G. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 2000, **16**(8): 851∼871

9 Maniezzo V, Colorni A. The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge Data Engineering*, 1999, **11**(5): 769∼778

10 Dorigo M, Di Caro G, Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, **5**(2): 137∼172

11 Tan G Z, Zeng Q D, Li W B. Design of PID controller with incomplete derivation based on ant system algorithm. *Journal of Control Theory and Applications*, 2004, **2**(3): 246∼252

12 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997, **1**(1): 53∼66

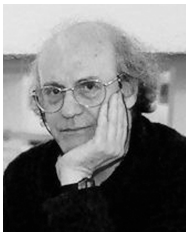
13 Habib M K, Asama H. Efficient method to generate collision free paths for autonomous mobile robot based on new free space structuring approach. In: Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japan, 1991. 563∼567

14 Yan W M, Wu W M. *Data Structure*. Beijing: Tsinghua University Press, 1997. 187∼192 (in Chinese)

15 Bhandari D, Murthy C A, Pal S K. Genetic algorithm with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, 1996, **10**(6): 731∼747

TAN Guan-Zheng Ph.D., professor of Central South University and Visiting Professor of the University of Birmingham. His research interest covers intelligent robotic systems, artificial intelligence, cognitive science, and evolutionary computation. Corresponding author of this paper. E-mail: tgz_csu@yahoo.com.cn

HE Huan Ph.D. candidate at Center for Space Science and Applied Research, Chinese Academy of Sciences. Her research interest covers intelligent robotics systems, artificial intelligence, and space science.



SLOMAN Aaron Ph.D., professor in the School of Computer Science, the University of Birmingham, UK. His research interest covers artificial intelligence, cognitive science, and intelligent robotic systems.