Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

PATH PLANNING FOR MOBILE ROBOT USING THE PARTICLE SWARM OPTIMIZATION WITH MUTATION OPERATOR

YUAN-QING QIN[1], DE-BAO SUN[1], NING LI[1,2], YI-GANG CEN[1]

[1]Department of Control Science and Engineering, Huazhong University of Science and Technology
[2] School of Computer Science & Technology, WuHan University of Technology, Wuhan 430074, China
E-MAIL: yuan_qing@163.com, sundebao@21cn.com, zidong@tom.com,cenyigang1@etang.com

Abstract:

Path planning is one of the most important technologies in the navigation of the mobile robot, which should meet the optimization and real-time requests. This paper presents a novel approach of path planning. First the MAKLINK graph is built to describe the working space of the mobile robot; then the Dijkstra algorithm is used to obtain the shortest path from the start point to the goal point in the graph, finally the particle swarm optimization algorithm is adopted to get the optimal path. Aiming at the shortcoming of the PSO algorithm, that is, easily plunging into the local minimum, this paper puts forward an advanced PSO algorithm with the mutation operator. By adding a mutation operator to the algorithm, it can not only escape the attraction of the local minimum in the later convergence phase, but also maintain the characteristic of fast speed in the early phase. The results of the simulation demonstrate the effectiveness of the proposed method, which can meet the real-time requests of the mobile robot's navigation.

Keywords:

Mobile robot; path planning; MAKLINK graph; Dijkstra algorithm; particle swarm optimization; mutation operator

1. Introduction

Path Planning is one of the most vital issues in the navigation of mobile robot, which means to find out an optimized collision free path from the start state to the goal state according to some performance merits. It can be classified into two categories: global path planning with all the information of the robot's known environment and local path planning in a partly or totally unknown environment[1]. There have been many algorithms for global path planning, such as artificial potential field, visibility graph, cell decomposition etc. Potential field has been used widely due to its simple structure and easy implementation, yet it still has some shortcomings[2,3]. For instance, traps exist in potential field, the path between two close obstacles cannot be found, and the oscillation before obstacles will be caused. Visibility graph algorithm demands that the control accuracy of the robot is high and its efficiency of searching path is low.

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. Compared to GA, the advantages of PSO are that PSO is easier to implement and there are fewer parameters to be adjusted. PSO has been successfully applied in many areas: such as function optimization, artificial neural network training, fuzzy system control and other areas where GA can not be used[4]. In this paper a new method, which uses free space algorithm to build the robot's environment model, is developed for mobile robot path planning. This algorithm divides the planning into two stages, that is, first finds out the shortest path from the start point to the goal point in the MAKLINK graph using Dijkstra algorithm, then adopts PSO algorithm to optimize the shortest path and get the best global path. The simulation results show the effectiveness of the proposed method. Some improved methods are also discussed to solve the local minimum problem of PSO.

2. Question description and modeling

Because of the direct effect on the simplicity and computational efficiency of the path-planning strategy, modeling robot's environment is the key issue of the mobile navigation planning. There are many methods for modeling, such as grid, vertex graph and generalized wimble etc. All these methods can obtain the accurate solutions of the path planning, but the construction and update of the models are very complicated. What's more, it is required that the sensors should possess very good accuracy, which brings on the difficulties for practical application.

In this paper, the authors adopt a simple model of robot's environment called MAKLINK graph[5], which can reduce the complicacy of the model and get the optimized path. To realize the proposed path-planning algorithm, some assumptions are made as the following:

1) The mobile robot moves in a two-dimension space;
2) There are several static obstacles in the space and these obstacles can be represented as polyhedral obstacles whose heights are all paralleled to z axis, so we can ignore the height information and represent them only using X axis and Y axis;
3) To ensure that the path is not too close to any of the obstacles, the dimension of the robot is represented by a point, and the boundaries of the obstacles are expanded according to the plus of the maximum distance occupied by robot's cross and the minimum distance required for proper sensing

For the purpose of developing an efficient path-planning algorithm, free space within the robot's environment is structured in terms of free convex area using free link approach, in which a free link represents:

(a) A line whose two ends are either corners of two polygonal obstacles or one of them is a polygonal corner and the other is a point on a working space boundary wall. A connection among the corners of the same obstacle is also considered for this purpose;
(b) Each free link is an open edge between two adjacent free convex areas;
(c) A free link should not intersect with any edge of the obstacles within the robot's environment;
(d) The boundary of each free convex area to be constructed will have at least two links as a part of its total number of edges. The structuring of free space within the robot's environment is as the following steps:
    1) Find all the lines that connect one of the corners that belong to a polygonal obstacle, with all the other obstacles' corners including the corners of the current obstacle. The perpendicular lines from the current corner to the working space boundary walls are considered for this purpose;
    2) Delete the redundant free links to make every free space, of which the edges are free links, obstacle edges and boundary walls, be a convex polygon and its area be largest;
    3) Find the midpoint of the remained free links and take them as the path nodes, labeling orderly as 1, 2, ⋯,n. The connections among the midpoints (nodes) that belong to the same convex area compose a network in which the robot can move freely.

An example is shown in Figure 1. The black polygons are static obstacles within the robot's environment, the dot lines are free links and the broken lines are connections among the midpoints of the free links which form the collision free path network.

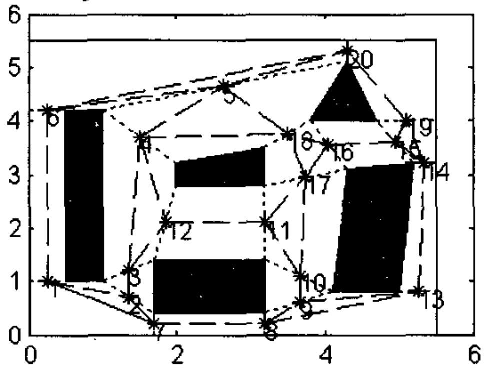

Figure 1. MAKLINK Graph Based on Free Convex Polygon

3. Mobile robot global path planning

This paper proposes a two-step path planning strategy. After modeling the robot's working space, the path-planning problem translates into a shortest path-finding problem in the MAKLINK graph, which can be solved by the mature algorithm in graph theory. But the robot can walk along the edges of obstacles rather than must along the network path. Therefore, the shortest path of the MAKLINK graph is not the exact optimized path of the whole planning space. This paper uses PSO algorithm to optimize the obtained shortest path and get the best global path.

3.1 Using Dijkstra algorithm to get the shortest path

Take the label sequence of the path nodes as a path code. If the start or goal point is not in the MAKLINK graph, connect this point with the path nodes that belong to the same free convex space and take it as a new path node. The lengths of the arcs in the graph are their weights and the definition of the adjust matrix is as the following:

$$weight\,(i,j)=\begin{cases} w_{ij} & v_i \neq v_j, and < v_i, v_j > \in E(G) \\ 0 & v_i = v_j \\ \infty & others \end{cases} \quad (1)$$

Where v_i and v_j are any two of the nodes in the graph E(G), $<v_i, v_j>$ represents an arc in the graph and w_{ij} is its weight. Suppose the coordinate of the start point is (0, 5.5), and the goal point is (5.5, 0), we can use Dijkstra algorithm to get the shortest path:

Start->5->18->17->10->9->goal, and the length of the path is 9.1566. The simulation result is shown in Fig. 2 and the solid line represents the shortest path we got.

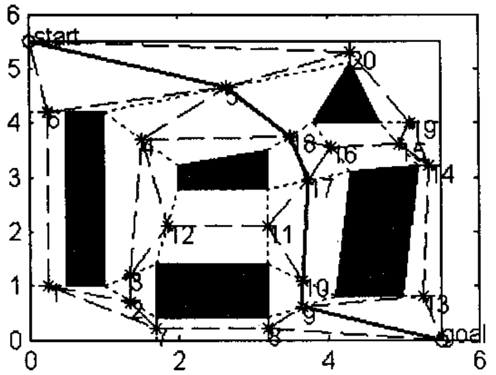

Figure 2. The Shortest Path Obtained by Dijkstra Algorithm

3.2 Particle Swarm Optimization

PSO is an algorithm proposed by Kennedy and Eberhart in 1995. Social behavior of organisms such as bird flocking and fish schooling motivated them to look into the effect of collaboration of species when achieving their goals as a group. Years of study on the dynamics of bird flocking resulted in the possibilities of utilizing this behavior as an optimization tool. They named it PSO. In a PSO system, multiple candidate solutions coexist and collaborate simultaneously. Each solution candidate, called a "particle", flies in the problem space (similar to the search process for food of a bird swarm) looking for the optimal position to land. A particle, as time passes through, adjusts its position according to its own "experience", as well as according to the experience of neighboring particles.

The mathematic description of PSO is as the following [6,7].

Suppose the dimension of the searching space is D, the number of the particles is n. Vector $X_i=(x_{i1}, x_{i2}, \dots, x_{iD})$ represents the position of the i^{th} particle and $pBest_i=(p_{i1}, p_{i2}, \dots, p_{iD})$ is its best position searched by now, and the whole particle swarm's best position is represented as $gBest=(g_1, g_2, \dots, g_D)$. Vector $V_i=(v_{i1}, v_{i2}, \dots, v_{iD})$ is the position change rate of the i^{th} particle. Each particle updates its position ("flies") according to the following formulas:

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times rand() \times [p_{id}(t) - x_{id}(t)] + c_2 \times rand() \times [g_d(t) - x_{id}(t)] \quad (2)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), \quad 1 \le i \le n \quad 1 \le d \le D \quad (3)$$

Where c_1 and c_2 are positive constant parameters called acceleration coefficients (which control the maximum step size the particle can do). *Rand()* is a random function with the range [0, 1]. The inertia weight, w is a user-specified parameter that controls, with c_1 and c_2, the impact of previous historical values of particle velocities on its current one. A larger inertia weight pressures towards global exploration (searching new area) while a smaller inertia weight pressures towards fine-tuning the current search area. Proper selection of the inertia weight and acceleration coefficients can provide a balance between the global and the local search. Eq. (2) is used to calculate the particle's new velocity according to its previous velocity and to the distances of its current position from its own best historical position and its neighbors' best position, then the particle flies toward a new position according to Eq. (3). The various range of the d^{th} position is $[XMINX_d, XMAX_d]$ and the various range of the d^{th} velocity is $[-VMAX_d, VMAX_d]$. If the value calculated by Eq. (2) and Eq. (3) exceeds the range, set it as the boundary value. The performance of each particle is measured according to a predefined fitness function, which is usually proportional to the cost function associated with the problem. This process is repeated until user-defined stopping criteria are satisfied.

Maurice Clerc[8] and Fvan den Bergh[9] analyzed the Convergence and Stability of PSO and put forward some parametric conditions that can ensure the convergence of PSO.

3.3 The realization of PSO for Path Planning

Suppose the shortest path of the MAKLINK graph that we get by Dijkstra algorithm is $P_0, P_1, P_2, \cdots, P_D, P_{D+1}$, where P_0=start is the start point and P_{D+1}=goal is the goal point. $P_i (i=1, 2, \cdots, D)$ is the midpoint of the free link. The optimization task is to adjust the position of P_i to shorten the length of path and get the optimized (or acceptable) path in the planning space. The adjust process of P_i is shown in Figure 3: Any P_i can slide freely along the free link that it lies on (Note: the free space in the model consists of convex polygon, which ensures the new path formed by the slide of P_i on line $P_{i1}P_{i2}$ will not intersect with the obstacles). The position of P_i can be decided by the following parametric equation:

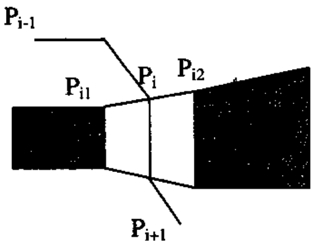

Figure 3. Path Coding Method

$$P_i = P_{i1} + (P_{i2} - P_{i1}) \times t_i, \quad t_i \in [0,1] \quad i = 1,2,\cdots D \quad (4)$$

After processing every P_i, i=1, 2, ⋯, D, the new path nodes sequence forms a new collision free path for the robot, which can be determined with D values in range[0,1]. The path code for the robot can be represented as $P = t_1 t_2 \cdots t_D$.

Suppose the shortest path got by Dijkstra algorithm has D free path nodes, which also means the particle has D dimension and each dimension's value range is [0,1]. Take the path length from the start point to the goal point as the particle's fitness value. The smaller the fitness value, the better the solution. The definition of the i^{th} particle's fitness value is

$$f(X_i) = \sum_{k=1}^{D+1} P_{k-1}P_k \quad i=1,2,\cdots,n \quad (5)$$

Where $P_{k-1}P_k$ is the distance between the two points and P_k can be calculated according to Eq. (4). The realization process of PSO is as the following:

I. Initialize particle X_i (randomly initialize each dimension of the particle's position and velocity in the solution space), each particle's historic optimal position *pBest* is itself. Calculate each particle's fitness value according to Eq. (5) and label the particle with the minimum fitness value as *gBest*;

II. Update particle's velocity according to Eq. (2), if $v_{id} < -VMAXX_d$, set $v_{id} = -VMAXX_d$; if $v_{id} > VMAXX_d$, set $v_{id} = VMAXX_d$;

III. Update particle's position according to Eq. (3), if $x_{id} < XMINX_d$, set $x_{id} = XMINX_d$, if $x_{id} > XMAXX_d$, set $x_{id} = XMAXX_d$;

IV. Calculate each particle's fitness value according to Eq. (5). If the i^{th} particle's fitness value is less than the $pBest_i$'s fitness value, set $pBest_i = X_i$; find out $pBest_k$ ($1 \le k \le n$) whose fitness value is the minimum of $pBest_i$'s fitness value ($i=1,2,\cdots,n$) and label it as *gBest* particle;

V. Turn to step II and iterate until the user-defined stopping criteria are satisfied.

3.4 PSO with Mutation Operator

To solve the local minimum problem of PSO, this paper proposes an improved PSO with mutation operator. As we can see in Eq. (2), if the historic optimal position of the particle swarm P_l has been unchanged for a while, once the swarm approaches P_l, the last two terms of Eq. (2) are very small and the velocity is mainly determined by $w*v_{id}(t)$. The velocity will be smaller and smaller, at that time, the particle swarm shows great "convergence", which means, at the angle of optimization performance, that the velocity of convergence is fast but easily traps into the local minimum, especially when the solution space becomes a non-convex set because of the restriction of the problem. Bibliography [10] proposed Multi-start PSO, which maintains the historic optimal position of the particle swarm and reinitializes all the particles after having iterated for certain times in order to improve the particles' variety and enlarge the searching space. This algorithm can escape from the local minimum's basin of attraction and ensure the search converge to the global optimization, but it destroys the structure of the current particles with total re-initialization, which results in the slowing down of the convergence speed and the great decrease of the searching accuracy.

Realizing this point, this paper introduces a mutation operator to PSO, something like GA. If the historic optimal position P_l keeps continuously unchanged or has changed extremely a little, in other word, the particle swarm aggregates heavily, we will keep P_l and re-random several dimension of particles according to certain probability to improve global searching ability without decreasing the convergent speed and search accuracy. The definition of PSO with mutation is as the following:

```
While k<MaxIterationTime & fitnessValue>acceptedAccurary
  If LogjamStep>=MaxStep
    If SwarmDist<BorderDist
       Re-random the position and velocity of several
         particles according to certain probability $\rho$;
       LogjamStep=0;
    Else
       LogjamStep=LogjamStep+1;
    End
  End
  Update the position and velocity according to Eq. (2) and
  Eq. (3)
End
```

Where k is iteration time, *LogjamStep* is the times that the historic optimal position P_l keeps continuously unchanged or has changed extremely a little; *MaxStep* is the threshold value of the unchanged times, *SwarmDist* is the distance between all the particles and the historic optimal position, which can be the mean aggregate distance or the max aggregate distance; *BorderDsit* is the threshold value to determine whether the historic optimal position has changed or not.

4. Simulation Result

As shown in Figure 2, the shortest path has 5 path nodes that can slide freely on its free link, which means

D=5. Set the parameters of PSO as the following:

$c_1 = c_2 = 1.49$, $w=0.729$, *MaxIteration=500;*

The simulation environment is: Pentium IV 1.7GHz, 256M, Windows 2000, Matlab 6.1. Run the basic PSO algorithm 30 times, we can get the optimal solution 24 times and gets into the local minimum 6 times. The length of the optimal path is 8.2329, much shorter than the length of the previous path 9.1566. The result is shown in Figure 4, in which the bold solid line is the optimal path of PSO and the thin solid line is the shortest path obtained by Dijkstra algorithm.

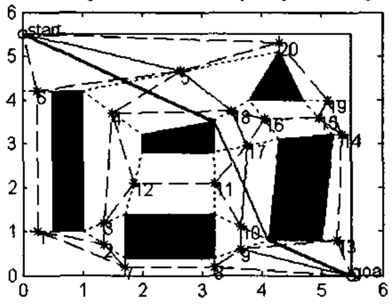

Figure 4. 5 Dimension PSO Simulation Result

In this optimal problem, each dimension of the particles has a strict range ($x_i \in [0,1]$, i=1,2,⋯,n) and the searching space is no longer continuous at the terminals of the interval, which causes the PSO algorithm to get into the local minimum easily. Aiming at this problem, the paper improves the range limitation in step Ⅲ as the following:

if $x_{id} < XMINX_d$, set $x_{id} = XMINX_d + rand()*0.01$, if $x_{id} > XMAXX_d$, set $x_{id} = XMAXX_d - rand()*0.01$;

In another 30 times running, the searching success rate is 96.6% and the mean success time does not increase greatly.

Rewrite the algorithm according to the advanced PSO proposed in section 3.4, and run it 30 times. The parameters are as the flowing: mutation operator ρ =0.033, *MaxStep*=10, *BoardDist* is 1/1000 of the optimal fitness value, that is to say, if the change rate of *gBestFitness* between two generations is less than 1/1000, we will regard it as unchanged. The results are much better than both the basic algorithm and the improved algorithm above. The contrast is shown in Table 1.

Table 1. Performance Contrast of Three PSO Algorithms

Example	Basic PSO		Improved PSO		PSO with mutation operator	
	Success rate (%)	Mean success time (s)	Success rate (%)	Mean success time (s)	Success rate (%)	Mean success time (s)
5D PSO	80	0.315	96.6	2.156	100	1.432
8D PSO	20	3.35	80	6.841	100	5.474

Figure 5 is another example with 8 dimensions. The shortest path of the MAKLINK graph is: start->2->3->12->11->17->18->15->19->goal and its length is 8.0524. The PSO optimization result is shown in Table 1 and the length of the optimal path is 7.1065.

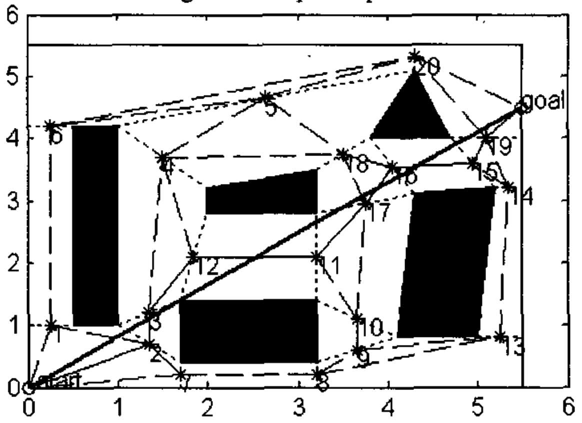

Figure 5. 8 Dimension PSO Simulation Result

5. Conclusion

In this paper, the authors propose a novel method of fractional steps to solve the global path-planning problem for the mobile robot. First the MAKLINK graph is built to describe the working space of the mobile robot; then the Dijkstra algorithm, a graph theory method, is used to obtain the shortest path from the start point to the goal point in the graph; finally the particle swarm optimization algorithm is adopted to get the global optimal path. By this method, it is easy to build a model and meet the real-time demand for mobile robot's navigation with the simple algorithm, which results in a certain practical value.

As a new optimization algorithm, PSO has shown its advantages and attracted more researchers' interests, but the theory's foundation is not perfect and the realization still needs much improvement to overcome its shortcomings such as local minimum problem. This paper propose an advanced PSO with the mutation operator, which can solve the local minimum problem to a certain degree, and the performance in our application is very good. However, there is still a lot of work to do: the performance is very sensitive to the choice of parameter ρ, *MaxStep* and *BoardDist*. Too big ρ and *BoardDist* with too small *MaxStep* will greatly decrease the convergence speed and the searching accuracy; on the contrary, too small ρ and *BoardDist* with too big *MaxStep* will affect the global searching ability. Besides, the analysis is quantitative. Finding an adaptive method that can adjust the parameters in the process of optimization is our next plan to improve the PSO algorithm.

References

[1] Li Lei, Ye Tao, Tan Min, Chen xi-jun. "Present state and future development of mobile robot technology research", Robot, Vol. 24, No. 5, pp. 475-480, Sept. 2002.

[2] Yoram Keron, Johann Borenstein. "Potential field methods and their inherent limitations for mobile robot navigation", Conf Robot Automation, California, pp. 1398-1404, April 1991.

[3] Ma Zhao-qing, Yuan Zen-ren. "Real time navigation and obstaticle avoidance based on grids method for fast mobile robot", Robot, Vol. 18, No. 6, pp. 344-348, Nov. 1996.

[4] J.Kennedy and Eberhart, R.C. "Particle swarm optimization", Proc. IEEE International Conference on Neural Networks, IV, Piscataway, NJ, pp. 1942-1948, 1995.

[5] Habib M K, Asama H. "Efficient method to generate collision free paths for autonomous mobile robot based on new free space structuring approach". IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91.Osaka, Japan, pp. 563-567, 1991

[6] Shi, Y. and Eberhart, R.C. "Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway NJ, pp. 1945-1950, 1999

[7] Eberhart, R.C. and Shi, Y. Particle swarm optimization: developments, applications and resources [A]. Proc. Congress on Evolutionary Computation 2001[C]. Piscataway, NJ:IEEE Press, 2001: 81-86

[8] Maurice Clerc, James Kennedy. "The particle swarm—Explosion, stability and convergence in a multidimensional complex space". IEEE Transaction on Evolutionary Computation, Vol. 6. No. 1, pp. 58-73, 2002

[9] van den Bergh F, Engelbrecht AP. "A new locally convergent particle swarm optimizer". IEEE Conference on Systems, Man, and Cybernetics, 2002

[10] F van den Bergh. "An Analysis of Particle Swarm Optimizers". PhD thesis, Department of Computer Science, University of Pretoria, South Africa,2002