Path Planning Based on Improved Binary Particle Swarm Optimization Algorithm

Zhang Qiaorong Gu Guochang
College of Computer Science and Technology
Harbin Engineering University
Harbin, China
zhangqiaorongs@sina.com

Zhang Qiaorong
College of Information
Henan University of Finance and Economics
Zhengzhou, China

***Abstract*—A new global path planning approach based on binary particle swarm optimization algorithm (BPSO) for a mobile robot is presented. The detailed realization of the approach is illustrated. The obstacles in the robot's environment are described as polygons and the vertexes of obstacles are numbered from 1 to n. Binary particle swarm optimization is used to plan the path. The length of the particle is the number of the vertexes. Every bit in the particle may be 1 or 0 which represents whether the vertex is in the path or not. To avoid converging too fast (the algorithm stops when the optimal path is not found), the algorithm is improved and the mutation operation is used. Simulation results are provided to verify the effectiveness and practicability of this approach.**

***Keywords*—Binary Particle Swarm Optimization Algorithm, Path Planning, Mobile Robots**

I. INTRODUCTION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr.Eberhart and Dr. Kennedy in 1995[1], inspired by social behavior of bird flocking or fish schooling. In past several years, PSO has been successfully applied in many research and application areas such as function optimization, pattern classification and training neural network because PSO gets better results in a faster, cheaper way compared with other methods and PSO has few parameters to adjust. The traditional particle swarm optimization is mainly used to solve problems in continuous space and it can not solve problems in discrete space well. In 1997, James Kennedy and Russ Eberhart proposed binary particle swarm optimization (BPSO) and used it to solve problems in discrete space successfully [2].

Path planning is a important field in mobile robots research and it contains global path planning and local path planning. Global path planning can be considered as a optimization problem with restriction. The restriction is that the path can not pass through the obstacles in the workspace. The optimization aim is that the length of the path is shortest. Using particle swarm optimization in path planning is a novel attempt [3-5]. Qin *et al.* described the environment using mark-link graph and planned the path using Dijkstra algorithm [3], then optimized the path using particle swarm optimization algorithm. So the path was not planned by particle swarm optimization algorithm directly. Sun *et al.* built a new map by coordinate switch and considered the length of the path as the optimization function [4], and then converted the path into the set of points of original coordinates. Lei *et al.* divided the workspace into several parts according to the abscissa of the obstacle vertexes and described the length of the path as the function of the ordinate of the vertexes. So the path planning problem was converted into a optimization problem of the high dimension function with restriction. All the methods mentioned above consider path planning as an optimization problem in continuous space and the traditional particle swarm optimization algorithm is used.

Global path planning includes two sub-problems which are environment representation and search strategy. If the appropriate environment representation method is used such as vertex-graph and cell decomposition, the workspace of the robot is described as a discrete space. So the binary particle swarm optimization can be used.

This paper proposes a novel global path planning approach based on binary particle optimization algorithm. The obstacles in the workspace of the robot are described as polygons and the vertexes of the obstacles are numbered from 1 to n which is the amount of the vertexes. Then the binary particle optimization algorithm is used to plan the path. To overcome the disadvantage of the binary particle optimization algorithm, we improve the algorithm by using double-structure particle coding and adding mutation operation into the algorithm.

The rest of the paper is organized as follows. Section II provides the representation of the environment. The improved binary particle swarm optimization algorithm is given in section III. Section IV provides the key technologies in the implement of the algorithm. Simulation results and analysis are given in section V. Conclusions are found in section VI.

II. ENVIRONMENT REPRESENTATION

Path planning is to find a set of points passed through when the robot moves in the workspace, and the path which is composed of these points can not pass though any obstacle. This paper uses vertex-graph to represent the environment, and the path is composed of the start, vertexes of the obstacles and the destination. Without loss of generality, it is supposed that the robot moves in the workspace as described below.

- The robot moves in a limited two-dimensional space;
- The robot can be considered as a particle if the boundary of each obstacle is extended by the half size of the

robot's maximal dimension in length or width direction;

- The obstacles in the robot's workspace are described as polygons. The vertexes of each obstacle in the workspace are numbered from 1 to n (n is the amount of the obstacle's vertexes in the workspace).

Fig.1 shows a representation of a sample environment. The vertexes are numbered from 1 to 20 according to their x-coordinate values and y-coordinate values.

III. IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM

The particle swarm optimization algorithm was first reported in 1995 by James Kennedy and Russell C. Eberhart. In 1997, James Kennedy and Russ Eberhart proposed binary particle swarm optimization (BPSO).

In BPSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. This value is called *pBest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lBest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gBest*. The particle swarm optimization concept consists of, at each time step, changing the velocity of each particle toward its *pBest* and *gBest* locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pBest* and *gBest* locations.

Assuming that the particle swarm contains N particles and the dimension of each particle is D. Each particle can be described as (X_i,V_i,P_i). $Xi=(x_{i1},x_{i2},\ldots x_{id})$ represents the location of the ith particle. $Vi=(v_{i1},v_{i2},\ldots v_{id})$ represents the velocity of the ith particle and $Pi=(p_{i1},p_{i2},p_{i3}\ldots p_{id})$ represents the *pBest* value of the ith particle. The *gBest* value, the best value obtained by any particle in the particle swarm so far, is described as gBest=(g1,g2,g3….gd). Each particle changes its velocity and location in terms of (1) and (2).

$$v_{id}(t+1)=w\times v_{id}(t)+c1\times rand()\times(p_{id}(t)-x_{id}(t)) + c2\times rand()\times(g_d(t)-x_{id}(t)) \quad (1)$$

$$x_{id}(t+1)=\begin{cases}0 & sign(v_{id}(t+1))\le r3\\ 1 & sign(v_{id}(t+1))>r3\end{cases} \quad (2)$$

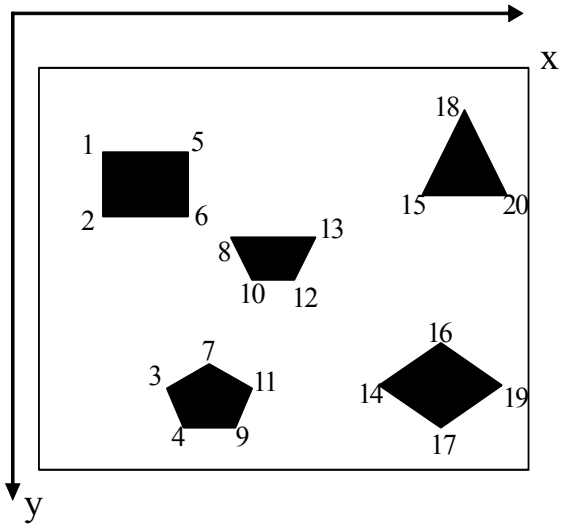

Figure 1. Environment representation

Where c1 and c2 are positive constants; rand() is a random variable; w is the inertial weight; p_{id} represents the best location that the *i*th particle has found so far and g_d represents the best location that the particle swarm has found so far. sign() is a sigmoid function which can be defined as $sign(x)=1/(1+e^{-x})$. The initial location and velocity of the particles are generated randomly and updated using (1) and (2) until the best solution is obtained.

Equation (1) can be modified as (3) if we define V_1 and V_2 as (4) and (5).

$$v_{id}(t+1)=w\times v_{id}(t)+V1+V2 \quad (3)$$

$$V1=c1\times rand()\times(p_{id}(t)-x_{id}(t)) \quad (4)$$

$$V2=c2\times rand()\times(g_d(t)-x_{id}(t)) \quad (5)$$

Where $w\times v_{id}(t)$ represents the velocity of the particles in the generation t. V1 represents the influence on the current velocity of this particle's best location in the first t generations. V2 represents the influence on the current velocity of all the particles' best location in the first t generations. In binary particle swarm optimization algorithm, $p_{id}(t)$, $x_{id}(t)$ and $g_d(t)$ are 0 or 1. So there are many zeroes in V1 and V2. If there are too many zero elements in V_1 and V_2, the effect of V1 and V2 on the current velocity will be slight. So the state of the particles can not be adjusted sufficiently and the diversity and convergence of the particles are reduced. So the particle swarm runs into the premature convergence easily and the performance of the algorithm is reduced. To avoid this case, we use double-structure particle coding [6] and add mutation operation to the algorithm to improve the binary particle swarm optimization algorithm.

In the improved binary particle swarm optimization algorithm, each particle can be denoted as $(X_i,\overline{X_i},V_i,P_i,\overline{P_i})$. $\overline{X_i}=(\overline{x_{i1}},\overline{x_{i2}},\ldots,\overline{x_{id}})$ represents the assistant location of the *i*th particle. $\overline{P_i}=(\overline{p_{i1}},\overline{p_{i2}},\ldots,\overline{p_{id}})$

represents the assistant best location that the ith particle has searched so far. The assistant best location that all the particle have searched can be described as $\overline{gBest} = (\overline{g_1}, \overline{g_2}, ..., \overline{g_d})$. The assistant locations are not composed of only 0 and 1 but continuous values. So using the assistant locations to update the velocity can reduce the amount of zero in V1 and V2, and the state of the particles can be adjusted sufficiently. (6) and (7) are used to update the location and the velocity of the particles.

$$v_{id}(t+1) = w \times v_{id}(t) + c1 \times rand() \times (\overline{p_{id}}(t) - \overline{x_{id}}(t)) + c2 \times rand() \times (g_d(t) - \overline{x_{id}}(t)) \quad (6)$$

$$\overline{x_{id}}(t+1) = \overline{x_{id}}(t) + v_{id}(t+1)$$

$$x_{id}(t+1) = \begin{cases} 0 & sign(\overline{x_{id}}(t+1)) \le r3 \\ 1 & sign(\overline{x_{id}}(t+1)) > r3 \end{cases} \quad (7)$$

The path planning approach based on improved binary particle swarm optimization algorithm can be described as follows.

- **Step1:** Initialize the velocity and location of each particle randomly.
- **Step2:** Calculate the fitness value of each particle and the pBest value the particle has searched so far.
- **Step3:** Choose the location of the particle with the best fitness value of all the particles as the gBest.
- **Step4:** Calculate the velocity of each particle using (6).
- **Step5:** Update each particle's location according to (7), and the next generation is obtained.
- **Step6:** Do the mutation operation.
- **Step7:** Goto step2 until the maximum iterations is attained or the fitness is small enough (the algorithm converges).

IV. KEY TECHNOLOGIES

A. Particle Representation

According to the environment representation method mentioned above, the obstacles in the workspace are described as polygons and the vertexes of the obstacles are numbered from 1 to n which is the amount of the vertexes. Each particle can be encoded as a binary vector with the dimension n. The 0 value of a bit represents the path does not pass through this vertex and the 1 value represents the path passes through the vertex. For example, the current location of a particle in Fig.2 is denoted as (0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0), which represents a path: start-→vertex 2→vertex 10→vertex 16→destination.

B. Initialization

- Initialize each particle's velocity randomly.
- Decide each element is zero or nonzero at the probability 0.5.

C. Inertial Weight

The inertial weight w in (1) is to make the particles maintain inertia and have the ability to explore new areas. If w is large, the algorithm has a strong global search capability, and if w is small, the algorithm has a strong local search capability. Generally, (8) is used to adjust the inertial weight *w*.

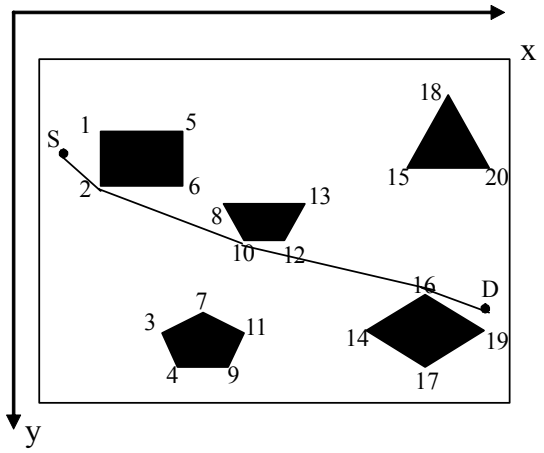

Figure 2. Example of the particle representation

$$w = w_{\max} - iter * \frac{w_{\max} - w_{\min}}{iter_{\max}} \quad (8)$$

Where $w_{\min} = 0.9$ is the largest inertial weight $w_{\min} = 0.4$, which is the smallest inertial weight. *iter* is the current iteration and $iter_{Max}$ is the total iteration number.

D. Fitness Function

The fitness function is an important factor for the convergence and the stability of particle swarm optimization (PSO). The collision avoidance and the shortest distance should be considered in path planning. In this paper, the collision avoidance can be considered as a restriction which can be solved by using penalty function [7]. The length of the path is to be optimized, and then the fitness function is defined as (9).

$$f = \begin{cases} \sum_{i=1}^{d} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} & \text{If constraints are satisfied} \\ \sum_{i=1}^{d} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} + N & \text{otherwise} \end{cases} \quad (9)$$

Where $\sum_{i=1}^{d} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$ represents the sum of the distance between the two adjacent vertexes in the path. (x_i, y_i) represents the coordinate of the vertex in the path. N is the penalty and its value is very large. When the path which is denoted by the particle does not satisfy the restriction (the path passes through the obstacles), the particle can be punished by the penalty N. So the problem with restriction can

be converted into a problem without restriction and the algorithm can be simplified.

E. Mutation

Generally the reason why the particle swarm optimization algorithm gets into the local extremum easily is that the diversity of the particle swarm decreases in the search space. In the evolutionary algorithms, mutation operation can increase the diversity of the particle swarm, and then the early convergence can be avoided. In this paper, mutation is applied in the algorithm to overcome the demerit mentioned above.

For each bit in the particle, a number between 0 and 1 is generated randomly. If the number is less than the mutation probability, the mutation operation is done to this bit. If the original value of the bit is 0, it is changed to 1. If the original value of the bit is 1, it is changed to 0. Although the convergence speed is decreased because of the mutation, the diversity of the particle swarm is increased. So the global search ability is improved and the local extremum can be avoided.

V. SIMULATION RESULTS AND DISCUSSION

This algorithm was implemented to validate the correctness and practicability. The size of the particle swarm is as 1.5 times as the quantity of the vertexes of the obstacles. The values of c1 and c2 in (6) are equal to 2. The mutation probability is 0.1 and the total iteration number is 100. Fig.3 shows the simulation results of the algorithm in different environments.

In this paper, the length of the particle is the number of the obstacles' vertexes and the quantity of the particles in the swarm is not defined as a constant but defined as the multiple of the number of the vertexes. In this way, with the number of the obstacles increasing the quantity of the particles also increases. So the initial particles can distribute in the whole search space. Table 1 shows the impact of the quantity of the particles on the results of the algorithm in different environment (the environment differ in the number of the obstacles' vertexes). From Table 1 we can see with the size of the swarm increases the average fitness value decreases slightly but the time consumption is increased. Taking the average fitness value and the time consumption into consideration, we let the size of the swarm be 1.5 times as the number of the obstacles' vertexes.

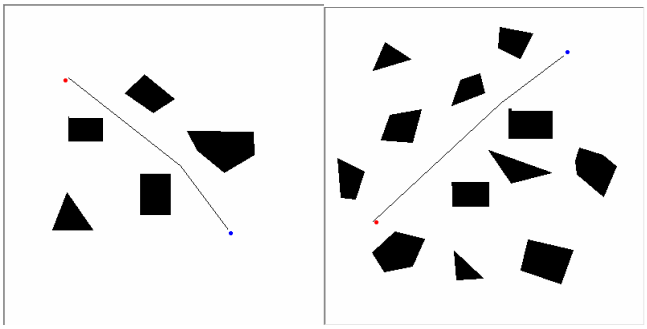

Figure 3. Simulation results

TABLE I. THE IMPACT OF THE SIZE OF THE SWARM ON THE RESULTS

The number of the vertexes D	The size of the swarm N	The average fitness	The time consumption
8	N=D	362.615	0.062s
	N=1.5D	362.615	0.094s
	N=2D	360.652	0.1250s
16	N=D	446.136	0.312s
	N=1.5D	420.847	0.436s
	N=2D	415.104	0.625s
24	N=1D	436.234	0.797s
	N=1.5D	429.251	1.187s
	N=2D	412.406	1.578s
32	N=D	337.640	1.593s
	N=1.5D	329.207	2.468s
	N=2D	321.113	3.202s

The mutation probability has great effect on the performance of the algorithm. If the mutation probability is small, it will be difficult to generate new particles and the diversity of the swarm will decrease. If the mutation probability is large, the algorithm will become the random search and the individual experience and global experience obtained will not be saved. The algorithm was implemented 30 times with different mutation probability. Fig.4 shows the comparison of the average fitness values obtained in these 30 experiments. Form Fig.4, we can see that with the mutation probability increases, the average fitness value decrease at first and then increases gradually. So 0.1 is a better choice for the mutation probability.

Fig.5 shows the comparison of the convergence process of three algorithms which are genetic algorithm, BPSO without mutation and BPSO with mutation. From Fig.5 we can see that with the mutation operation the convergence speed of the BPSO decreases (the algorithm with mutation converges near the 80th generation and it converges near the 40th generation without mutation), but the average fitness value also decreases. This phenomenon means that the algorithm converges rapidly without mutation but it probably traps into the local extremum. Although using mutation in the algorithm lowers the convergence speed but it can increase the diversity of the swarm and overcome the early convergence. The convergence speed of BPSO with mutation is close to the convergence speed of genetic algorithm, but it is simpler to implement BPSO than to implement genetic algorithm because there are no selection operation and crossover operation in BPSO.

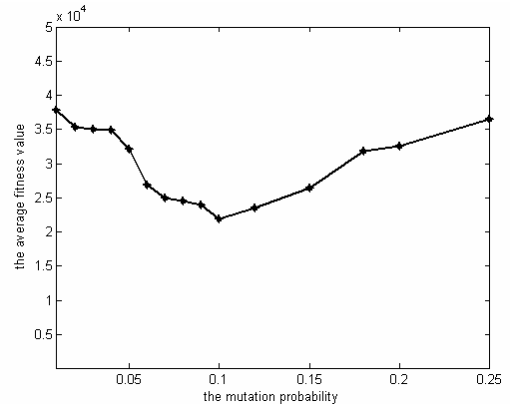

Figure 4. Effect of the mutation probability on the result

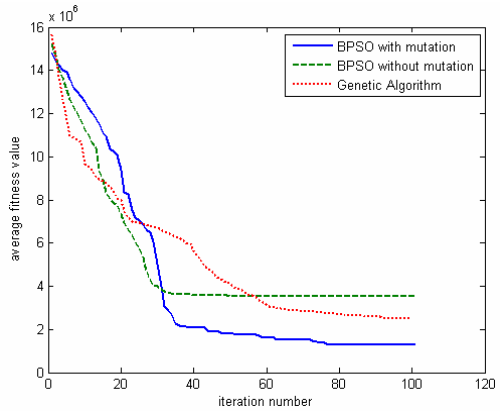

Figure 5. Convergence of three algorithms

VI. CONCLUSION

Global path planning is one of the key technologies of mobile robots, and it represents the intelligence level of mobile robots in some ways. This paper presents a new global path planning approach based on improved binary particle swarm optimization algorithm. In this approach, path planning is considered as a optimization problem with constraint and the binary particle swarm optimization algorithm is used. The length of the particle is equal to the number of the obstacles' vertexes in the environment and the particles are encoded as binary vectors. To overcome the demerit of the traditional binary particle swarm optimization algorithm, double-structure particle encoding and mutation operation are used in the algorithm. The simulation results show that compared with traditional binary particle swarm optimization algorithm and genetic algorithm, the algorithm presented in this paper has a better performance. This algorithm overcomes the problem of early convergence well and the length of the path obtained is shorter and the convergence speed is close to genetic algorithm.

REFERENCES

[1] R.C.Eberhart , J.Kennedy, "A New Optimizer Using Particle Swarm Theory," in Proc. of the 6th Int. Symp. On Micro Machine and Human Science, Nogoya, Japan, pp. 1995:39-43, April 1995.

[2] J.Kenedy, R.C.Eberhart, "A Discrete Binary Particle Swarm Optimization," in Proc. of Conf. on System, Man and Cybernetices, Piseateway,NJ, 1997, pp. 4104-4109.

[3] Qin Yuanqing, Sun Debao and Li Ning, "Path Planning for Mobile Robot Based on Particle Swarm Optimization," Journal of Robot, vol. 26, June 1963, pp. 222-225.

[4] Sun Bo, Chen Weidong and Xi Yugeng, "Particle Swarm Optimization Based Global Path Planning for Mobile Robots," Journal of Control and Decision,, vol. 20, September 2005, pp. 1052-1055.

[5] Lei Kaiyou, Qiu Yuhui, "A Modified Particle Swarm Optimizer for Mobile Robot Global Path Planning," Journal of China Southwest Normal University, vol. 31, April 2006, pp. 103-106

[6] He Yizhao, Wang Yanqi, "A New Particle Swarm Optimization for Solving Discrete Problems," Journal of Computer Applications and software, vol. 24, January 2007, pp. 157-159

[7] Zhang Gibiao, Zhou Chunguang, "A novel evolutionary algorithm for solving constrained optimization problem," Journal of Jilin University, vol. 42, August 2004, pp. 534-539