Robot Path Planning Based on Random Coding Particle Swarm Optimization

Kun Su
College of Electronic and Electrical Engineering
Shanghai University of Engineering Science
Shanghai, P.R. China

YuJia Wang
College of Electronic and Electrical Engineering
Shanghai University of Engineering Science
Shanghai, P.R. China

XinNan Hu
College of Electronic and Electrical Engineering
Shanghai University of Engineering Science
Shanghai, P.R. China

***Abstract*—Mobile robot navigation is to find an optimal path to guide the movement of the robot, so path planning is guaranteed to find a feasible optimal path. However, the path planning problem must be solve two problems, i.e., the path must be kept away from obstacles or avoid the collision with obstacles and the length of path should be minimized. In this paper, a path planning algorithm based on random coding particle swarm optimization (RCPSO) algorithm is proposed to get the optimal collision-free path. Dijstra algorithm is applied to search a sub-optimal collision-free path in our algorithm; then the RCPSO algorithm is developed to tackle this optimal path planning problem in order to generate the global optimal path. The crossover operator of genetic algorithm and random coding are introduced into the particle swarm optimization to optimize the location of the sub-optimal path. The experiment results show that the proposed method is effective and feasible compared with different algorithms.**

Keywords—robot path planning; Dijsktra algorithm; random coding; particle swarm optimization

I. INTRODUCTION

Since mobile robot is widely used in manufacturing, service and aviation field. There is still a challenging and interesting subject for research in mobile robotics field. The optimal path planning is the primary task problem of navigation, which is to search a collision-free optimal path from a starting point to the target. The path planning problem is launched at the late 1960s. Lozano-perez and Wesley firstly proposed the free C-space method in path planning [1]. So much focus has been given to the path planning problem. Now there are many methods to solve the question, according to the similarities and differences in environmental modeling and search strategy. Path planning can be classified into two main categories, i.e., the classical and heuristic [2, 3, 9]. The classical approach such as visibility graph methods [4], which requires a huge amount of computation when the number of vertices of polygonal obstacles are increasing, also its computation complexity is $O\left(n^2\right)$ in 2-D, where n is the number of the vertices. The main problem of the grid methods [5] is how to determine the size of the grids, as to the size of grids have great influence in accuracy of the described environment, memory space and search efficiency. Dijkstra algorithm [6] is widely used in search for the shortest path problem and can find the shortest path in the explicit figure. The artificial potential field method [7] has some inherent limitations that may lose the global optimization and be trapped in the local minimum. In order to improve the efficiency of classical approach, the heuristic approach has been developed to overcome the complex nature of the NP-complete problem, such as randomized potential planner (RPP) [8], probabilistic road map (PRM) [10] and rapidly-exploring random tree (RRT) [11]. There is lots of interest in the field such as genetic algorithm [12] which has strong optimization ability. However the genetic algorithm may fall into local optimum and premature problem. The swarm intelligence algorithm [13] is simple, less control parameters and the fast convergence such as the ant colony optimization [20], which has demonstrated to outperform classical approach. But the main disadvantage of the ant colony optimization is difficult to determine its parameters, which may be not conducive to obtaining the quick solution convergence, and so on.

Particle swarm optimization (PSO) is a population algorithm based stochastic optimization technique that inspired by social behavior of bird flocking [14]. Moreover, existing studies has shown that the competence of PSO to tackle the path planning problem with different map [15, 16, 17]. However, those works are easy to trap into local minimum or cannot be rapid convergence under a complex map.

In this paper, we present a RCPSO algorithm, which is applied for the global optimal path planning of mobile robot. The path planning problem includes two sub-problems: the free space modeling and searching for the feasible optimal path. The first step, establishing the free space modeling in the globe environment with the MAKLINK graph; the second step, making use of the Dijkstra algorithm to search a sub-optimal collision-free path; in order to generate the global optimal path, finally, utilizing the RCPSO algorithm with constant weight to optimize and adjust the location of the sub-optimal, in this algorithm, a global optimal path is described as a set of discrete point on MAKLINK graph. Moreover, there is the crossover operator of genetic algorithm introduced into the RCPSO algorithm to improve the feasible path.

The remainder of this paper is organized as follows. In Section II, a brief review of PSO is given; the free space is

This work was supported by the National Natural Science Foundation of China under Grant no. 61403249, and also was supported by the Innovation Program of Shanghai University of Engineering Science under Grant No.E1-0903-14-01040

established for mobile robot and the performance criterions are presented in Section III; Section IV gives the process of path planning, the Dijkstra algorithm is described and the RCPSO algorithm is applied for the MAKLINK graph. Section V describes the experiments and results. Section VI gives our conclusions finally.

II. Particle Swarm Optimization

PSO is a population based stochastic optimization technique that inspired by social behavior of bird flocking and it is to use the information sharing mechanism, particle learn from each experience and promote population growth. PSO on behalf of a group of particles through evolution candidate solution to solve the problem, each particle preserve their positions after all the best as individuals adjust to the optimal value. Save personal best group of all the particles in the optimal fitness value as a global optimum. Through the personal best and global best fly expect particles to converge to the optimal solution. Considering the i-th particle in the swarm, its position and velocity is $\vec{x}_i(t)=\left(x_{i,1}(t),x_{i,2}(t),\cdots,x_{i,\mathrm{n}}(t)\right)$ and $\vec{v}_i(t)=\left(v_{i,1}(t),v_{i,2}(t),\cdots,v_{i,\mathrm{n}}(t)\right)$ in the t-th iteration, and then in the iteration $(t+1)$-th will be updated by the following equations:

$$\begin{cases} v_{i,j}(t+1)=\omega v_{i,j}(t)+c_1r_1\left(pb_{i,j}(t)-x_{i,j}(t)\right)+c_2r_2\left(gb_j(t)-x_{i,j}(t)\right) \\ x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \end{cases} \quad (1)$$

Where $pb_{i,j}(t)$ denotes the personal best particle and $gb_j(t)$ denotes the global best particle the gene j of particle i in the t-th iteration, and the acceleration coefficients c_1 and c_2 are non-negative constant, r_1 and r_2 are two random number between 0 and 1, and the inertia weight ω is defined as controlling the exploration of particle in space.

III. Problem Formulation

There are two issues to solve in path planning: how to prevent collision from the obstacles and how to search an optimal path. To tackle the problem, the workspace is established for robot.

A. Modeling of the mobile workspace

The method proposed by Tan et al. [19] is adopted to establish the free space for mobile robot because of its small sensitivity to obstacles shape, the result is shown in Fig. 1 and Fig .2.

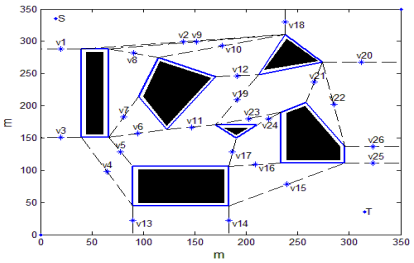


Fig.1. Map with MAKLINK graph

In Fig. 1, there are vertices of grown obstacles, free links, start point and target, the (x, y) coordinates of the obstacle vertices are (40, 288), (40, 151), (66, 151), (66, 288), (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 start point S(15, 335) and target T(315 35), respectively.

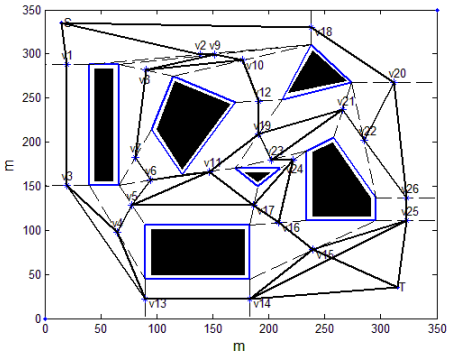


Fig.2. Network graph for mobile robot

As depicted in Fig. 2, v_i denotes the middle point of each free link, respectively, $i=1,2,\cdots,l$ where the symbol l denotes the sum of free link number. If two adjacent middle point are connected together to form a line that cannot intersect with any obstacles, moreover, point S and T are also connected to adjacent point make an integrated network graph, which make a possible feasible path for the mobile robot in such environment.

Fig. 2 is a weight undirected graph that denoted by $G=\left(\omega_{i,j},V,E\right)$, where $\omega_{i,j}$ is the distance between two adjacent middle points, which can also be defined as:

$$\omega_{i,j}=\begin{cases} length\left(v_i,v_j\right), & if\ edge\left(v_i,v_j\right)\in E \\ 0, & if\ i=j \\ \infty, & others \end{cases} \qquad (i,j)\in(0,1,\cdots,l+1) \quad (2)$$

Where $V=v_i, i=0,1,\cdots,l,l+1$ is a set of points, v_0 denotes respectively the start point S and the target point T is denoted by v_{l+1}, respectively. E is a set of the lines that includes: the lines those the middle points of two adjacent free links connect to each other, the lines connect point S with the middle points on the free links adjacent to S, and the lines connect the target T with the middle points on the free links adjacent to T. By utilizing the weight undirected graph G, the free space model can be established.

B. Performance criterions

A sub-optimal path can be searched and denoted by $P_0,P_1,\cdots,P_g,P_{g+1}$ in network graph for mobile robot. The points $P_1,\mathrm{P}_2,\cdots,P_g$ are the middle points of free links and P_0,P_{g+1} are the start point S and target T respectively. There are two points P_{i1} and P_{i2} on the corresponding free link for each P_i, i.e., the start point and the end point. The location of

free link needs to adjust in order to find an optimal path. The parameter ph_i is introduced that can be defined as:

$$ph_i = P_{i1} + (P_{i2} - P_{i1}) \times p_i,\ \ p_i \in [0,1], (i = 1,2,\cdots,g) \tag{3}$$

Where p_i is a parameter between zero and one. A complete path can be found, which is constructed by $ph_i\ (i = 1,2,\cdots,g)$, the start point S and target T, therefore, the path is transform to be optimize the following set of points $P = (S, ph_1, ph_2, \cdots, ph_g, T)$.

For the performance criterion, the length of the path, assuming that the start point S and the target T are ph_0 and ph_{g+1}, so the path can be described as:

$$L(P) = \sum_{i=0}^{g} D(ph_i, ph_{i+1}) \tag{4}$$

Where $D(ph_i, ph_{i+1})$ denotes the distance between ph_i and ph_{i+1}, so the value of $D(ph_i, ph_{i+1})$ can be calculated as follows:

$$D(ph_i, ph_{i+1}) = \sqrt{\left(x_{ph_i} - x_{ph_{i+1}}\right)^2 + \left(y_{ph_i} - y_{ph_{i+1}}\right)^2} \tag{5}$$

In accordance with the performance criterion, we need optimize the points ph_i respectively, which is mean to adjust the parameter p_i to find the optimal path.

For the purpose of dynamic convergence behavior of algorithm in the iteration process, two statistic index is introduced into the experiment, namely, the mean value E and the standard deviation D. they can be define by the Eq.(6) and Eq.(7).

$$E = \frac{\sum_{i=1}^{n} L_i}{n} \tag{6}$$

$$D = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (L_i - E)^2} \tag{7}$$

Where L_i denotes length of a feasible path of mobile robot, which generate by the $i-$th iteration, and n represents the total number of current iteration. The mean value E is the average value of a feasible path generated by the all current iteration. The less the E value, the better the solution generated by each iteration. So E represents the accuracy of an algorithm. The standard deviation D reflects the discrete degree of feasible optimized path. The smaller the D value, the better the centrality of the solution generated by all current iteration.

IV. THE PATH PLANNING

To reduce the computational complexity of path planning problem, firstly, a sub-optimal path is searched through the Dijkstra algorithm, then utilizes the RCPSO algorithm to find an optimal path.

A. Dijkstra algorithm

In order to search a feasible collision-free path from the start point S to the target T in a free space, we take the Dijkstra algorithm to solve the problem. As the weight undirected graph $G = (\omega_{i,j}, V, E)$ has been known that is a n-order graph, to calculate the sub-optimal collision-free path in graph G, it requires some assumption[21] as follow:

(1) Assume $l_{P(i)}(n)$ is equal to the weight $\omega_{0,i}$ that is the shortest length from v_0 to v_i, if v_i get the label $l_{P(i)}(n)$ then v_i obtain the permanent label in step n.

(2) Assume $l_{T(j)}(n)$ is equal to the weight $\omega_{0,j}$ that is the upper shortest length from v_0 to v_j, if v_j get the label $l_{T(j)}(n)$ then v_i obtain the permanent label in step n.

(3) Set $P_n = \{v \mid v\ has\ obtained\ permantent\ label\}$ to be "the passed the vertex" in step n.

(4) Set $T_n = V - P_n$ to be "the vertex set that hasn't been passed" in step n, then the Dijkstra algorithm is following these steps:

Step 1: Initialization
$$\begin{cases} n = 0 \\ l_{P(0)}(0) = 0 \\ P_0 = \{v_0\} \\ T_0 = V - P_0 \\ l_{T(j)}(0) = \omega_{i,j}; (j \neq 0) \end{cases}$$

Step 2: Traverse, WHILE the maximum number of the vertices of the set V has been reached, i.e, $n \le l+1$, DO

Step 2.1 Search next permanent vertex

(1) Set $l_{P(i)}(n) = \min\{l_{P(i)}(n-1)\}, (n \ge 1)$,

(2) Update the vertex set P_n and T_n by the Eq.(8)

$$P_n = P_{n-1} \cup \{v_i\}, T_n = V - P_n \tag{8}$$

(3) Check T_n: if $T_n = \varnothing$ then the algorithm end, else jump step 2.2

Step 2.2 Update the vertex set in T_n

Set $l_{T(j)}(n) = \min\{l_{T(j)}(n-1), l_{P(j)}(n) + \omega_{i,j}\}$

Step 2.3 Increment the loop counter, $n = n+1$.

Step 3: output the points of optimal path

Applying the Dijkstra algorithm to the Fig. 2, the optimal path can get as shown in Fig. 3, and the length of this path is 507.692 meters.

B. The RCPSO algorithm

By applying the Dijkstra algorithm can find a sub-optimal collide-free path, however, in order to search a global optimal feasible path for mobile robot from the start point S to the target T, it is still to adjust and optimize the path. So as to satisfy the performance criterion, this section represents RCPSO algorithm, where the crossover operator of genetic algorithm and random coding are introduced into the PSO algorithm to improve the capability of the PSO algorithm.

1) Particle coding

The section 3.1 indicates that a point set of $ph_1, ph_2, \cdots, ph_g$ determines path $L(P)$, and according to the Eq. (3), a constant set of $p_1, p_2, \cdots, p_g$ is the decision variables of the path planning problem. And the $p_1, p_2, \cdots, p_g$ can be selected as a particle which represents a global path points on the corresponding free link, respectively, so the particles can be based on a random sampling to create.

2) Updating the personal best positions

In this section, the crossover operator of genetic algorithm introduced to update the personal best position (*pbest*), crossover operator is applying the crossover operator has two advantages: on the one hand, it can escape from the local optima, on the other hand, it improves the performance of convergence speed, hence, we adopt the crossover operator to update *pbest* .

3) Updating the particle position

In the RCPSO algorithm, a particle represents a global path points on the corresponding free link, so in this section a new method based on crossover operator and random sampling to update the particle's position.

The $i-$th particle $\vec{x}_i(t) = (x_{i,1}(t), x_{i,2}(t), \cdots, x_{i,n}(t))$ was considered as an example, and the proposed approach is following these steps:

Step 1: Initialize, set $t = 1$, $i = 1$;

Step 2: Get *pbest* from particles, as for definiteness and without loss the generality, each *pbest* particle applies the crossover operator in the $t-$th iteration.

Step 3: Update the value of position $x_{i,j}$ by the Eq. (1), and *t*++.

Step 4: In case of the particles maintain the boundary of position $x_{i,j}$, we confine the particle in a range, if the particle is out of boundary, then updating $x_{i,j}$ by

$$\begin{cases} range = upbound - lowbound \\ x_{i,j} = range \times rand \end{cases} \quad (9)$$

Step 5: Let t =t+1, if $t > iter_{\max}$, jump out of the iteration loop process, else return to Step 2.

In those steps, $iter_{\max}$ refers to the maximal number of iteration, Fig. 3 also shows the flowchart of the RCPSO algorithm.

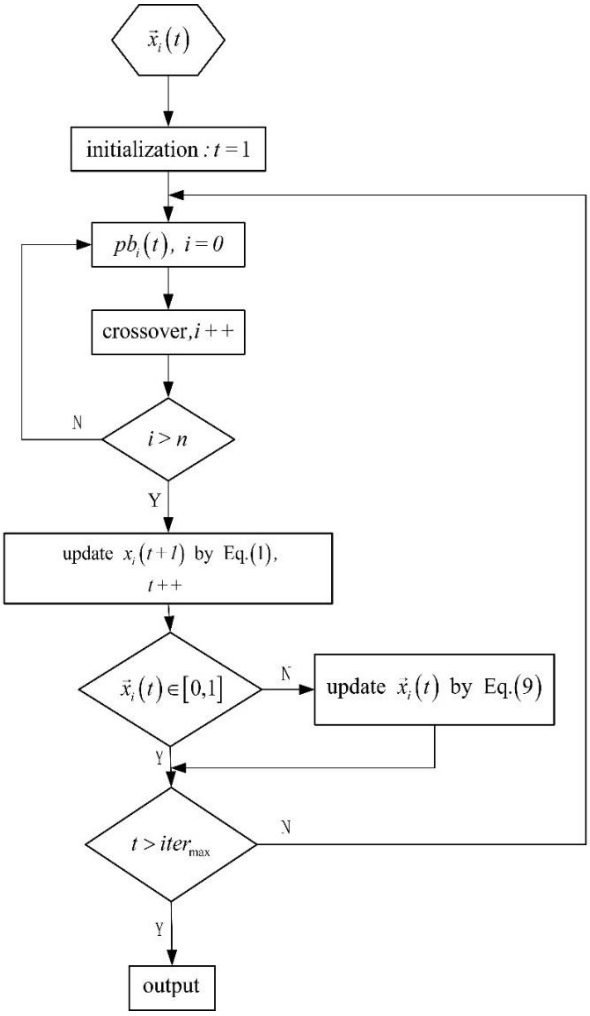


Fig.3. Flowchart of the RCPSO algorithm

V. EXPERIMENTS AND RESULTS

In this section, the experiment results validate the performance of RCPSO algorithm, and in the following experiments, the RCPSO algorithm was running the following parameters: simulation experiments were executed on a personal computer with 3.20-GHz and 4.00GB RAM, the size of swarm is $n = 60$, the maximum number of iteration $iter_{\max} = 200$, the acceleration coefficients c_1=c_2=0.2, and the inertia weight ω has three scenarios to identify the performance of algorithm, and the result is shown in Fig. 4.

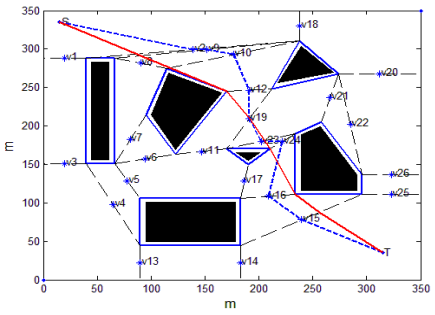


Fig.4. Results of experiment using proposed PSO algorithm

In Fig. 4, the blue dotted line denotes the sub-optimal mobile robot path with the length of 507.6919 meters, and the red solid line denotes the optimal path for mobile robot, without loss the generality, the inertia weight ω represents three scenarios, which are constant weight $\omega_1 = 0.28$, linear variable weight and exponential variable weight, which shows in Fig. 5.

$$\omega_2 = \omega_{\min} + (\omega_{\max} - \omega_{\min}) * t/iter_{\max} \quad (10)$$

$$\omega_3 = \omega_{\min} * \left(\frac{\omega_{\max}}{\omega_{\min}}\right)^{\frac{1}{1+10t/\text{iter}_{\max}}} \quad (11)$$

For three different inertia weight, the algorithm receive differences in the results, which shows in table I and Fig. 6.

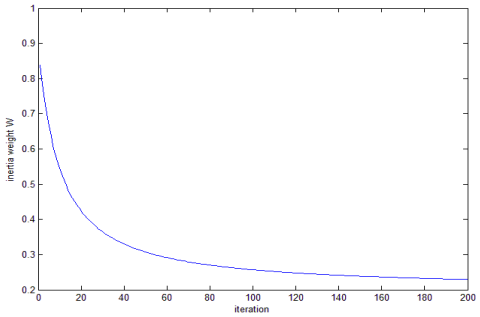
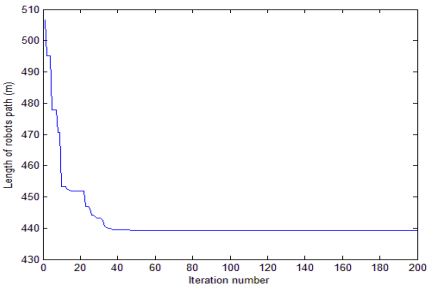


Fig.5. exponential variable weight

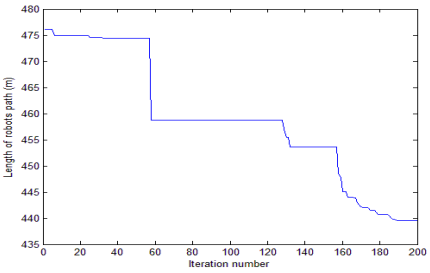
TABLE I. EXPERIMENT RESULTS OF THREE WEIGHT

Weight	Length(m)	ph_1	ph_2	ph_3	ph_4	ph_5	ph_6	ph_7	ph_8	ph_9
ω_1	439.249	0.237	0.143	0.018	0	0.489	0.445	0.999	1	0.646
ω_2	439.486	0.254	0.164	0.029	0	0.498	0.441	0.999	0.997	0.636
ω_3	439.824	0.279	0.182	0.071	0.007	0.549	0.457	0.999	1	0.664

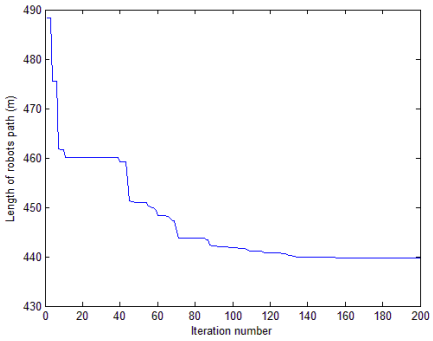
As we can see, the RCPSO algorithm with three inertia weights have a subtle differences about the results. However, there are huge difference about the convergence, which is shown in Fig.6.



(a) RCPSO ($\omega_1 = 0.28$)



(b) RCPSO (linear ω_2)



(c) RCPSO (exponential ω_3)

Fig.6. Convergence tendency of RCPSO

In Fig. 6, which reveals the convergence processes under three weights station of RCPSO algorithm, which (a) is under condition of constant weight ω_1, and (b) is under condition of linear variable weight ω_2, and (c) is under condition of exponential variable weight ω_3. As is obvious, the constant weight ω_1 generates the stable solution (439.249 meters) is around the fortieth iterations, and linear variable weight ω_2 gets the optimal stable solution (439.486 meters) until the iteration of 190, and under condition of exponential variable weight ω_3 requires to execute about one hundred and fifty

iterations to get stable optimal solution (439.824 meters). Apparently, the constant weight is much better than the other two weights in convergence speed.

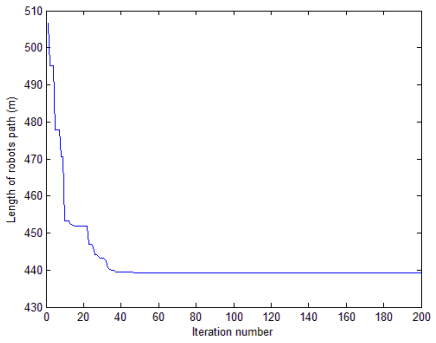
In order to enhance generality of the algorithm, we can contrast with basic PSO, Dijkstra algorithm and the results of literature [18], which utilizes the Ant Colony System (ACS) algorithm to optimize the path, which shows in table II.

TABLE II. EXPERIMENT RESULTS OF DIFFERENT ALGORITHMS

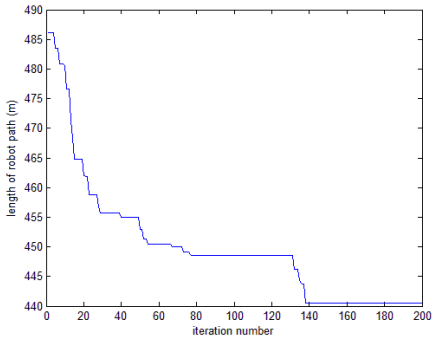
Algorithm	Length(m)	ph_1	ph_2	ph_3	ph_4	ph_5	ph_6	ph_7	ph_8	ph_9
Dijkstra	507.692	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Basic PSO	460.180	0.330	0.247	0.182	0.555	0.364	0.229	0.458	0.865	0.609
ACS	440.233	0.2	0.1	0	0	0.5	0.5	0	1	0.7
RCPSO	**439.248**	**0.237**	**0.143**	**0.018**	**0**	**0.489**	**0.445**	**1**	**1**	**0.646**

In order to compare more performances of the algorithm, we compare convergence speed, mean value and standard deviation.

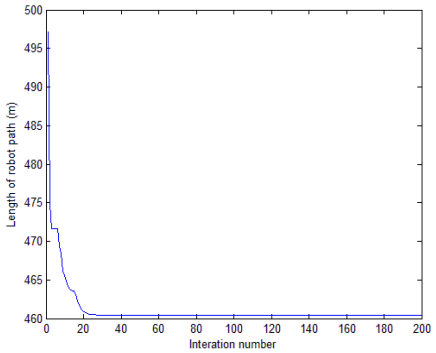
1) Convergence speed



(a) RCPSO



(b) ACS



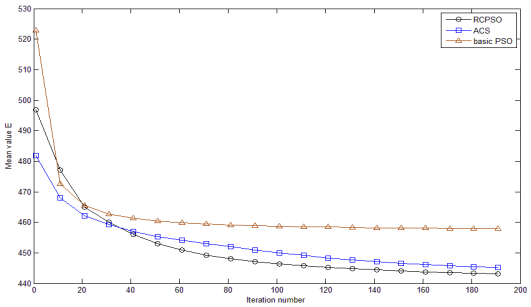
(c) Basic PSO

Fig.7. Convergence tendency of different algorithms

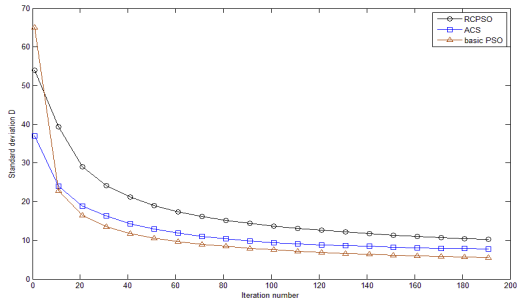
In Fig. 7, which reveals the convergence processes of RCPSO algorithms and ACS algorithm, which (a) is under condition of constant weight $\omega_1 = 0.28$, and (b) is the ACS algorithm, (c) is a basic PSO algorithm with the same constant weight. As is obvious, the constant weight ω_1 generates the stable solution (439.2488 meters) is around the fortieth iterations. As for the ACS algorithm, which can generate a stable optimal solution (440.233 meters) about 140 iterations. It is easy to find that the basic PSO algorithm has a fast convergence, it gets a stable value about 25-th, however, in this case, the algorithm is "premature", so the stable solution is 460.180 meters. Apparently, the RCPSO is much better than ACS in convergence speed, and it has a more stable value than basic PSO algorithm.

2) Dynamic convergence

Fig. 7 and Fig. 8 reveal the convergence tendency of E and D under three weights station of proposed PSO algorithm.



(a) Mean value E



(b) Standard deviation D

Fig.8. Convergence tendency of using RCPSO algorithm

In Fig. 8, which reveals the dynamic convergence processes of different algorithms, which (a) is the mean value of the RCPSO algorithm, and (b) is standard deviation. As can be seen, in the process of iteration, the mean value E and the standard deviation D are all small and smooth. However, the RCPSO algorithm has a rapid convergence and optimal solution in Fig.8 (a), also we can find that RCPSO is with wide feasible solution in Fig. 8 (b), which reveal the solution generated by RCPSO are much better than other algorithms.

VI. CONCLUSIONS

In order to solve robot path planning problem, we utilize RCPSO algorithm for mobile robot path planning in this paper. Matlab software is used to simulate the process of path planning and environment, then the random coding and crossover operation is introduced into particle swarm optimization to search the optimal path for guiding robot. The results of experiments demonstrate the proposed algorithm is effective, also it has been assured that the proposed algorithm under constant weight has better performance in convergence speed, dynamic convergence and can be a viable alternative for solving the robot path planning problem.

However there are still some limits to environment, such as how to solve the situation with the concavity or convexity polygons environment, how to get away from the local minimum in a more obstacles with complex environment, for example, dynamic obstacles. All the above problems will be discussed in our future work.

REFERENCES

[1] T. Lozano-Perez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” Communications of the ACM, Vol. 22, No. 10, 1979, pp. 560~570.

[2] E. Masehian and D. Sedighizadeh. “Classic and heuristic approaches in robot motion planning- a chronological review,” Industrial and Mechatronics Engineering, Vol. 1, No. 5, 2007, pp. 249-254

[3] P. Raja and S. Pugazhenthi, “Optimal path planning of mobile robots: A review,” International Journal of Physical Sciences, Vol. 7, 2012, pp. 1314-1320.

[4] J. Kim, M. Kim and D. Kim, “Variants of the Quantized Visibility Graph for Efficient Path Planning,” Advanced Robotics, Vol. 25, 2011, pp. 2341-2360.

[5] J. Gu and Q. Cao, “Path Planning for mobile robot in a 2.5-dimensional grid-based map,” Industrial Robot: An International Journal, Vol. 38, 2011, pp. 315-321.

[6] Y. Q. Qin, D. B. Sun, N. Li and et al, “Path Planning for Mobile Robot using Particle Swarm Optimization with Mutation Operator,” Proceedings of Third International Conference on Machine Learning and Cybernetics, 2004, pp. 2473-2478.

[7] J. Y. Zhang, Z. P. Zhao and D. Liu, “A path planning method mobile robot based artificial potential field,” Journal of Harbin of Technology, Vol. 38, No. 8, 2006, pp. 1306-1309.

[8] S. M. LaValle. “Planning algorithms,” University of Illinois, 2006

[9] H. J. Liu, J. Y. Yang, J. F. Lu and et al, “Overview of mobile robot motion planning study,” Engineering Science, Vol. 8, No. 1, 2006, pp. 85-94.

[10] C. M. Clark, “Probabilistic Road Map sampling strategies for multi-robot planning,” Robotics and Autonomous System, Vol. 53, 2005, pp. 244-264.

[11] S. Rodriguez, X. Tang, J. Lien and et al, “An Obstacle-Based Rapidly-Exploring Random Tree,” Proceedings of IEEE International Conference on Robotics and Automation, 2006, pp. 895-900.

[12] S. C. Yun, V. Ganapathy and L. O. Chong, “Improved Genetic Algorithms based Optimum Path Planning for Mobile Robot,” Int. Conf. Control, Automation, Robotics and Vision, 2010, pp. 1565-1570.

[13] C. A. Sierakowski and L. S. Coelho, “Study of two swarm intelligence techniques for path planning of mobile robots,” IFAC World Congress, 2005.

[14] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” Proc. IEEE Int. Conf. Neural Network, 1995, pp. 1942-1948.

[15] W. Huang, D. B. Sun and Y. Q. Qin, “Path Planning of Mobile Robot Based on Particle Swarm Optimization Algorithm,” Measurement and Control Technique, Vol. 25, No. 4, 2006, pp. 49-61.

[16] G. Z. Tan and G. J. Liu, “Global optimal path planning for mobile robots based on Particle Swarm Optimization,” Application Research of Computers, Vol. 24, No. 11, 2007, pp. 210-212.

[17] B. Sun, W. D. Chen and Y. G. Xi, “Particle Swarm Optimization Based Global Path Planning for Mobile Robots,” Control and Decision, Vol. 20, No. 9, 2005, pp. 1052-1060.

[18] G. Z. Tan, H. He and S. Aaron, “Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots,” Acta Automatica Sinica, Vol. 33, No. 3, 2007, pp. 279-285.

[19] G. Z. Tan and D. Mamady, “Real-Time Global Optimal Path Planning of Mobile Robots Based on Modified Ant System Algorithm,” Internal Conference on Natural Computation, 2006, pp. 204-214.

[20] M. A. P. Garcia, O. Montiel, O. Castillo and et al, “Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation,” Applied Soft Computing, Vol. 9, 2009, pp. 1102-1110.

[21] Wang S. X, “The Improved Dijkstra's Shortest Path Algorithm and Its application,” Procedia Engineering, Vol.29,2012, pp.1186-1190.