International Journal on
"Technical and Physical Problems of Engineering"
(IJTPE)
Published by International Organization of IOTPE

ISSN 2077-3528
IJTPE Journal
www.iotpe.com
ijtpe@iotpe.com

September 2014 | Issue 20 | Volume 6 | Number 3 | Pages 38-43

A MULTI-OBJECTIVE PSO BASED ALGORITHM FOR A VEHICLE ROUTING

D. Sedighizadeh

Engineering Department, Saveh Branch, Islamic Azad University, Saveh, Iran, davoud.sedighizadeh@gmail.com

Abstract- In this paper a novel method is presented for robot motion planning with respect to two objectives, the shortest and smoothest path criteria. A Particle Swarm Optimization (PSO) algorithm is employed for global path planning, while the Probabilistic Roadmap method (PRM) is used for obstacle avoidance (local planning). The two objective functions are incorporated in the PSO equations in which the path smoothness is measured by the difference of the angles of the hypothetical lines connecting the robot's two successive positions to its goal. The PSO and PRM are combined by adding good PSO particles as auxiliary nodes to the random nodes generated by the PRM. The proposed algorithm is compared in path length and runtime with the mere PRM method searched by Dijkstra's algorithm, and the results showed that the generated paths are shorter and smoother and are calculated in less time.

Keywords: Particle Swarm Optimization (PSO), Routing.

I. INTRODUCTION

Starting from mid 1970's, the Robot Motion Planning (RMP) problem, which in its most elementary form is to find a collision-free start-to-goal path for robots moving amid obstacles, has been actively researched, and many classic methods have been proposed for solving it [1]. Existing classic methods are variations of a few general approaches: Roadmap, Cell Decomposition, Potential fields, Mathematical programming, although these approaches are not essentially mutually exclusive, and combinations of them are often utilized in developing motion planners.

Because of NP-Hardness of the RMP problem, heuristic methods have been developed increasingly over the past two decades to cope with high computational costs and complexities of classic methods, especially for high degrees of freedom. When using heuristic algorithms, it is not guaranteed to find a solution, but if a solution is found, it will be done much faster than deterministic methods. The main metaheuristic approaches employed in RMP are Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony (ACO), and Tabu Search (TS). A chronological review of applications of classic and heuristic algorithms in RMP is presented in [2].

Being inspired form natural phenomena, both PSO and GA algorithms are evolutionary, population-based metaheuristic algorithms, and have proved to be very efficient and powerful in a wide range of optimization problems. The PSO has found applications in robot motion planning quite recently. To name just some of them, an algorithm for path planning of mobile robots using PSO with mutation operator is developed in [3]. An obstacle avoidance path planning for soccer robots using PSO is proposed in [4], and a smooth path planning of a mobile robot using Stochastic PSO is implemented in [5].

Among other successful heuristics for robot motion planning are the probabilistic algorithms, including the Probabilistic Roadmap Method (PRM) and rapidly exploring Random Trees (RRT) [6].

In most of the abovementioned methods, the robot motion-planning problem has been solved with respect to a single objective function, mainly the path length. However, the paths generated by these methods are generally non-smooth and their practicality is questionable in most of the cases, since mobile robots lose considerable energy and time when changing their course of motion abruptly. Therefore, in this paper we have worked out a planning algorithm to apply the two objectives simultaneously, which are the shortest and smoothest criteria. This is done through an aggregative weighting multi-objective approach incorporated in a PSO context.

The reasons for implementing the PSO method in our planner are that PSO is versatile enough to accommodate multiple objectives, and as shown in [7], PSO is more efficient and faster than the GA. On the other hand, considering the success of the PRM method and its speed in especially high dimensional spaces, the PRM is found suitable for obstacle avoidance in our algorithm. In fact, we have selected the PSO as the global planner of our algorithm, while the PRM is utilized as a local planner.

The combination and interaction of the PSO and PRM methods is done for the first time in the field of motion planning, and as computational results have shown, they act very coherently since both of these methods have probabilistic elements and parameters. More specifically, the notions of particles in the PSO and random nodes generated in the PRM roadmap complement each other and unify these methods.

In this paper the robot motion planning is done for a point robot navigating among static 2D obstacles with known vertices. The proposed new method iteratively shifts from PSO to PRM until the goal is reached.

The overall steps of the algorithm are as follows:

1. A preset number of particles are generated around the robot's initial position and within its sensing range.
2. Each particle takes a new velocity and position based on the constantly updated PSO equations. A candidate for the robot's next position is determined by the position of the best particle (i.e. the one nearest to the goal).
3. If the robot's current position can be directly connected to the candidate best particle obtained in Step 3, it is set as the robot's next position. Go to Step 2.
4. If the candidate best particle is located beyond an obstacle (i.e. the line connecting the best position to the robot's current position intersects an obstacle), a probabilistic roadmap is formed and searched for the shortest path. As a result, the current position of the robot is connected to a node of the PRM, which is nearest to the goal through their shortest path.
5. Steps 2 to 4 are executed until the goal is within the robot's sensing range and can be accessed via a straight line.

In the next section of the paper, the PSO component is introduced and the processes of generating the particles' initial population, applying multiple objectives, and parameter tuning are explained in detail. The PRM component is described in section 3, and in section 4 the experimental results obtained from simulations are presented. Conclusions and future research directions are presented in the last section.

II. PSO COMPONENT: THE GENERAL PLANNER

In this algorithm the Particle Swarm Optimization method is employed as the global motion planner; that is, it is used for planning the large-scale, 'gross' motions of the robot. In this section an overview of the basic PSO algorithm is presented, after which its implementation in the new algorithm is described.

The Particle Swarm Optimization (PSO) algorithm was first introduced in Nov. 1995 by Kennedy and Eberhart [8]. They used the idea of swarms in the nature such as birds, fish, etc. and proposed an algorithm called PSO. The PSO has particles driven from natural swarms, with communications based on evolutionary computations. The PSO combines the particles' self-experiences with their social experiences. In this algorithm, a candidate solution is presented as a particle. The algorithm utilizes a collection of flying particles (changing solutions) in a search space (current and possible solutions) and moves towards a promising area to get to a global optimum.

Taxonomy of the PSO algorithm has been presented in [9], which categorizes the elements of the PSO algorithm into four main aspects: variables, particles, swarm, and process. In PSO the particles showing the solution candidates start their fly from random positions in a search area. In each iteration, particles update their position according to (1) and (2) and move to another position. Flying is affected by a fitness function that assesses the quality of each solution.

$$prtpos_j^i = prtpos_j^{i-1} + prtvel_j^i \tag{1}$$

$$prtvel_j^i = \chi \begin{bmatrix} w \times prtvel_j^{i-1} + \\ c_1 r_1 (pbest_j^{i-1} - prtpos_j^{i-1}) + \\ c_2 r_2 (gbest^{i-1} - prtpos_j^{i-1}) \end{bmatrix} \tag{2}$$

where

$prtpos^i{}_j$ = the position of the jth particle in ith iteration,
$prtvel^i$ = the velocity of the jth particle in ith iteration,
$pbest^i{}_j$ = the best position of the jth particle,
$gbest^i{}_j$ = the best position within the swarm,

$$\chi = 2 \Big/ \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|, \quad \varphi = \varphi_1 + \varphi_2 > 4 .$$

The PSO has some dependent parameters: c_1 and c_2 are factors balancing the effect of self-knowledge and social knowledge when moving the particle towards the target, and are usually set to a value of 2, although good results have been also produced with $c_1 = c_2 = 4$ [10]. r_1 and r_2 are random numbers between 0 and 1, different at each iteration, and χ is a constriction factor to limit the velocity.

The w regulates the global search behavior, set to a large value in the beginning of the searching process and dynamically reduced during the optimization (which emulates a more local search behavior). Its range is suggested to be $0.2 \le w \le 0.4$. Dynamic adjustment of w has several advantages: first, it causes faster convergence to an optimal solution, and second, it controls the effect of previous part velocities on current velocities, hence, adjusting the tradeoff between the capability of swarms in local and global exploration. Figure 1 illustrates a schematic view of updating the position of a particle in two successive iterations.

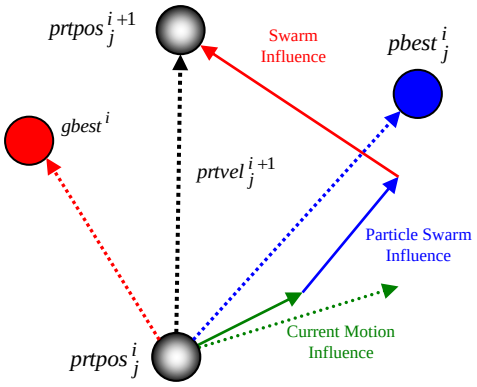

Figure 1. Depiction of a particle's position update in PSO

The procedure of the PSO algorithm is presented in Figure 2. The algorithm has a main nested loop terminated when the total number of iterations exceeds a certain limit or a minimum error threshold is achieved. In each iteration, particles are generated and best fitness values for each particle (*pbest*) and for the whole swarm (*gbest*) are calculated. Particles' positions and velocities are then updated in (1) and (2).

A. Generating Particles Initial Population

In the basic PSO algorithm a number of particles are required to be created and positioned randomly in the search space. In our proposed method, the particles are generated with respect to the robot's initial position and regarding its sensing range.

The initial population is generated such that along each sensing direction, a particle is created at a certain distance from the robot, determined by the range of the used sensor. If any obstacle point is within the sensing range at that direction, a point near the obstacle's border is selected as the particle at that direction. Thus, the number of created particles depends on the number of sensors (or in a virtual space, the number of divisions on the circumferential circle). Figure 3 illustrates the creation of 36 particles around the robot's starting point. The larger the number of divisions on the circle is, the larger the number of particles would be, and therefore the planning accuracy would be higher.

This innovative procedure has the advantage that the initial particles are generated around the robot's start point such that the movement from the start position to the next best position can be made through a fast, straightforward and safe connection within the sensing range.

Procedure Basic PSO
while maximum iterations or minimum error criteria is not attained
do
  for each particle **do**
    Initialize particle
  end
  for each particle **do**
    Calculate the fitness value
  If the fitness value is better than the best fitness value in history (*pbest*)
  then Set current value as the new *pbest*
  end
  end
  for each particle **do**
    Find in the particle neighborhood the particle with the best fitness (*gbest*)
    Calculate particle velocity $prtvel^i_j$ according to the velocity equation (2)
      Apply the velocity constriction
    Update the particle position $prtpos^i_j$ according to the position equation (1)
    Apply the position constriction
  end
end

Figure 2. Pseudo code of the basic PSO

In existing PSO-based approaches, the initial positions are generated randomly, whereas in the presented approach, while maintaining the centralization of the robot's start point, the obstacles' distribution around it is also considered.

B. Multiple Objective Fitness Function

Most path planners aim to generate an optimal path considering a single criterion like path travel time or path length. However, in practice, a path is feasible if it meet several conditions, such as safety, estimated needed time for navigation, energy consumption, etc.

A path that is considered as optimal in terms of a single criterion may not essentially satisfy other criteria all together [11]. For instance, a shortest path is not required at the expense of safety along the path.

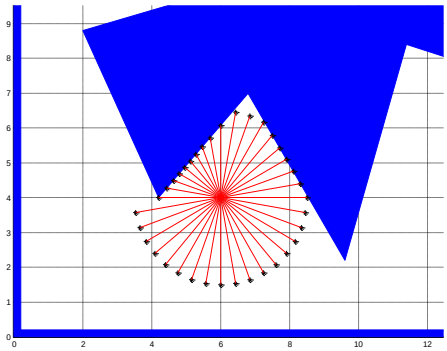

Figure 3. The particles initial population is generated based on the borders of the robot's sensed area

Some works exist in the field of multi-objective RMP, including an approach for obstacle avoidance with multi-objective optimization by PSO in dynamic environment in [12], and multi-objective optimal trajectory planning of a space robot using PSO in [13].

Path planning in dynamic environments epitomizes the necessity of considering multiple objects in path planning: when the environment is time varying, the minimum length path and minimum delay path are usually different issues. Delay is defined as the time needed for traveling from start to goal, whereas length is the distance actually traveled by the robot along the path. For robots needing to reach their destination as early as possible, a minimum-time path might seem desirable, but it may require a lot of time to be traversed due to uneasy terrain. Surely, there exist various feasible paths between start and goal points being neither short nor fast but providing reasonable tradeoffs between shortness and fastness. These are generally desirable paths, while a path optimal for a single criterion without considering other equally important criteria is not desirable [11]. This is just one type of problem for which our multi-objective search is designed.

A common method for enforcing multiple objectives is the Simple Additive Weighting (SAW) method, in which a weighted sum of multiple objectives is expressed as a conventional single-objective function in the form of $Total\ Cost = w_1c_1 + w_2c_2 + \ldots$, where c_i is the ith cost and w_i is its weight. By selecting proper weights, a path with desirable property can be obtained by planning with a single objective [14].

In the proposed method, the criterion for path shortness is defined as the Euclidean distance between each particle and the goal point in each iteration, and the criterion for path smoothness is defined as the angle between the two hypothetical lines connecting the goal point to the robot's two successive positions in each iteration, i.e. $gbest^i$ and $gbest^{i-1}$, in which i is the iteration number. The definition of path smoothness in this way is a novel idea.

The first objective function, the shortest path, is defined as:

$$fitness(1)^i_j = \sqrt{(prtpos^i_j(x)-xgoal)^2+(prtpos^i_j(y)-ygoal)^2} \quad (3)$$

and the second objective function, the smoothest path, is mathematically expressed as:

$$fitness(2)^i_j = \cos^{-1}\left[\frac{(prtpos^i_j(x)-xgoal)\times(gbest^{i-1}_x-xgoal)+(prtpos^i_j(y)-ygoal)\times(gbest^{i-1}_y-ygoal)}{\sqrt{(prtpos^i_j(x)-xgoal)^2+(prtpos^i_j(y)-ygoal)^2}\times\sqrt{(gbest^{i-1}_x-xgoal)^2+(gbest^{i-1}_y-ygoal)^2}}\right] \quad (4)$$

The overall fitness (or objective) function is obtained by the weighted sum of these two shortest and smoothest objectives:

$$fitness^i_j = \lambda_1 \times fitness(1)^i_j + \lambda_2 \times fitness(2)^i_j \quad (5)$$

By minimizing the overall fitness function regarding the assigned weights of each criterion, a suitable path is obtained. The weights of the shortest and smoothest fitness functions, λ_1 and λ_2 respectively, are tuned through extensive simulation and try and errors, with best-found values $\lambda_1 = 1$ and $\lambda_2 = 0.25$.

C. Parameter Tuning

For tuning the parameters used in the proposed algorithm, it is tried to specify acceptable and practical limits for each as presented in Table 1, which are obtained after numerous runs.

After incorporating the developed fitness function in the general PSO algorithm (Figure 2), and applying the parameters tuned properly, the positions of the generated particles are updated using (1) and (2). When a particle is generated or updated in each iteration, it is verified whether it lies inside an obstacles or not. If it is inside an obstacle, it will be omitted from the swarm. Furthermore, after calculating the position of $gbest^i$ in iteration i, it is checked if the line connecting the $gbest^i$ and $gbest^{i-1}$ is entirely in free space. This connecting line is actually a segment of the final path, which the robot travels in each iteration, and so its freeness is vital. If the line is not free (i.e. intersects an obstacle), the local planner component, which is based on the RPM method, is invoked to find a path between the two successive *gbests* by detouring the obstructing obstacle(s).

III. PRM COMPONENT: THE LOCAL PLANNER

Due to its ease of implementation and ability to plan in high dimensional configuration spaces, the Probabilistic Roadmap method (PRM) has drawn considerable attention in recent motion planning works. Initial PRMs succeeded in solving a number of complex problems with high-dimensional configuration spaces, which had not been solved efficiently until that time [6]. The PRM was enhanced later into some variant forms like MAPRM, OBPRM, and Visibility-based PRM, improve the process of random node generation and make it more effective.

The PRM has three phases: (1) generating random nodes in free configuration space, (2) connecting the nodes via some edges such that the edges lie in the free space and the nodes are connected through a single graph, and (3) searching the graph to find the shortest path between the start and goal nodes.

In the second phase, an edge is generated between two nodes by first trying to connect them via a straight line, and if this fails, a simple local planner is employed to connect them through a few intermediate newly generated nodes (Figure 4). The path planning is done by searching this graph.

Table 1. Guidelines for tuning the parameters of the algorithm

Title and symbol	Function	Suggested range	Conditions for increase (▲) and decrease (▼)
Acceleration constant c_1	Attracts the particle's position towards its *pbest*	[1.5, 4]	▲ High velocity and acceleration of particle movements are required ▼ Low velocity and acceleration of particle movements are required
Acceleration constant c_2	Attracts the particle's position towards the *gbest*		
Inertia weight w	Maintains the particle's current velocity	[0.4, 0.9]	▲ Local exploitation is emphasized ▼ Global exploration is emphasized
Visibility range of the goal R_{goal}	Controls the scope of particles' accessibility to the goal	[0, 4]	▲ Greedy convergence with less accuracy ▼ Cautious convergence with more accuracy
weight factor λ_1	Controls the weight of the shortest path in fitness function	[0.1, 2]	▲ More emphasis on the shortest path ▼ More emphasis on the smoothest path
weight factor λ_2	Controls the weight of the smoothest path in fitness function	[0.1, 2]	▲ More emphasis on the smoothest path ▼ More emphasis on the shortest path
Population size n	Number of particles	[10, 10000]	▲ More accuracy in reaching the goal in more time ▼ Less accuracy in reaching the goal in less time
Maximum position (x_{max}, y_{max})	The maximum position a particle can be located on	–	
Minimum position (x_{min}, y_{min})	The minimum position a particle can be located on	–	
Maximum iterations $iter_{max}$	Maximum iterations of procedure	[10, 10000]	▲ More accuracy in optimizing the selected function spending more time ▼ Less accuracy in optimizing the selected function but with higher speed

In our version of PRM, four groups of nodes lying in free space are considered as the set of probabilistic roadmap nodes:

(a) A number of randomly generated nodes,
(b) The robot's current position,
(c) The best particles generated in the PSO,
(d) Two points around each corner of the obstructing obstacle.

The above combination of nodes is proposed for the first time in the literature and secures a subtle intertwining of the PSO and PRM methods. In addition to randomly generated nodes (item (a) above) which are typical in the PRM method, about %30-40 of PSO particles with highest

fitness values (*pbests*) are also integrated in the PRM graph. The item (d) helps in circumnavigating obstacle vertices naturally and easily.

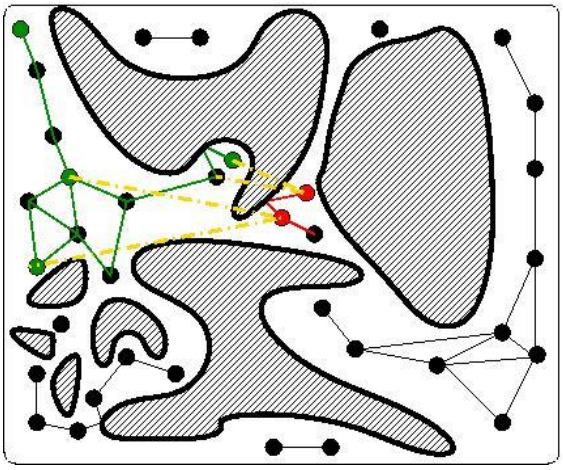

Figure 4. A snapshot of the edge building process using the PRM. Roadmap edges are generated such that they lie entirely in the free space

After creating the necessary nodes, new edges are generated in the second phase of the PRM by connecting nodes to each other and deleting invalid edges (i.e., those intersecting with obstacles). The shortest path between the robot's current position and the point *gbest* (calculated based on the best position among particles) is then found using the Dijkstra search algorithm. As a result, the robot can move from $gbest^{i-1}$ to $gbest^{i}$ and get closer to the goal, while avoiding the obstacles locally intercepting its path to the goal. Once the robot is located on its new position, the PSO particles' velocity and position updating is performed again, as described earlier.

IV. EXPERIMENTAL RESULTS

In order to analyze the function of the proposed new algorithm, numerous simulations were run through which the algorithm's parameters were tuned to their best values. A few graphical results of running the algorithm on problems with simple to complex obstacles are illustrated in Figure 5.

On the other hand, for comparing the algorithm's performance with another efficient algorithm, the standard PRM method was selected. After constructing the probabilistic roadmap, it was searched by the Dijkstra's method to yield a start-to goal shortest path on it. For comparison, 35 problems with different obstacles sizes and shapes (both convex and concave) were designed and solved by the new multi objective PSO and PRM + Dijkstra methods. Both methods were coded in Matlab and run on an Intel 3.0 GHz processor.

The number of obstacle vertices in the sample problems was designed to increase from 17 to 414, as depicted in Figure 6. The runtime and path lengths obtained by both methods are superimposed in Figures 7 and 8, respectively. The comparison results are summarized in Table 2, in which the mean and standard deviation of runtimes and path lengths of the 35 problems are calculated for each method.

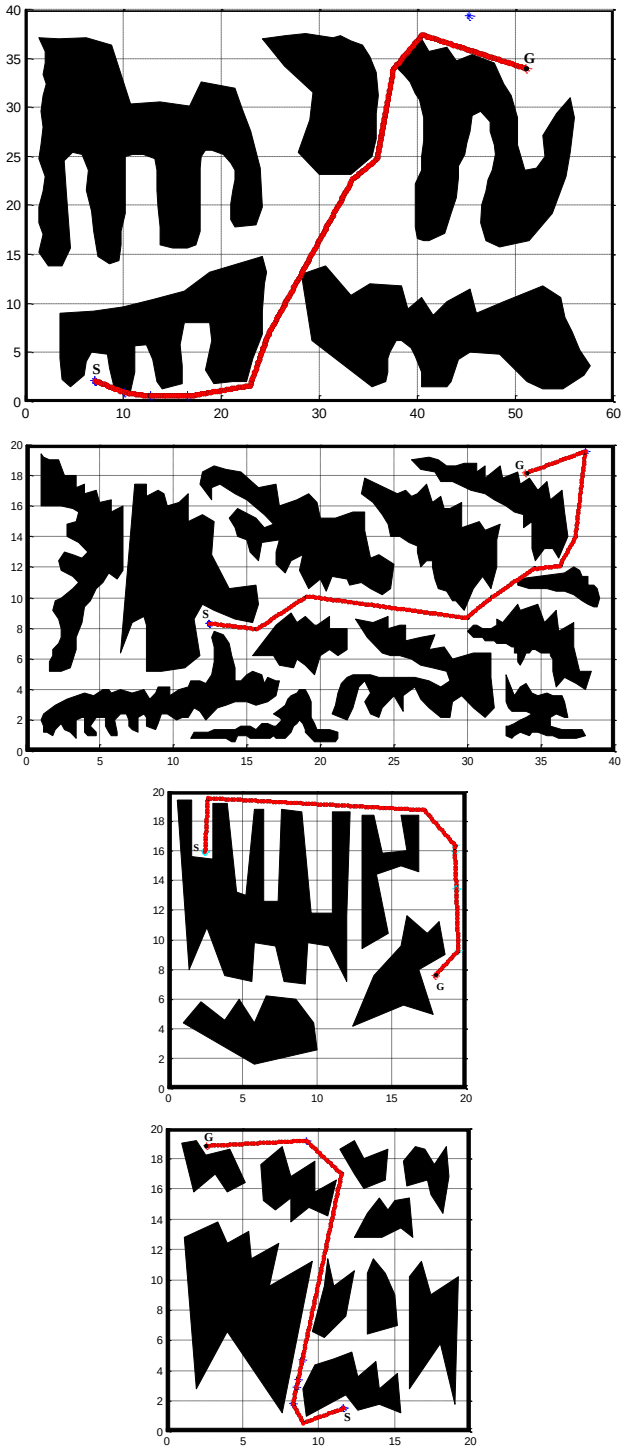

Figure 5. Some simulations of the developed method

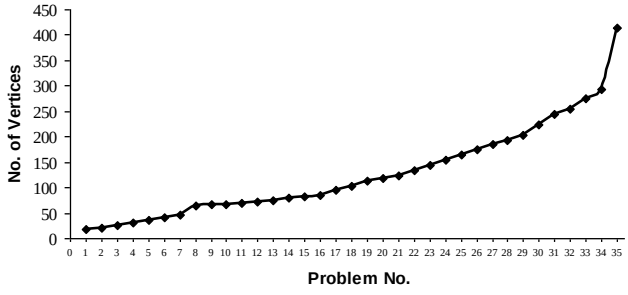

Figure 6. Number of vertices in various test problems

The results demonstrate that the developed method is nearly %60 faster than the PRM + Dijkstra method, while path lengths generally do not differ significantly. In addition, the standard deviation of the new method was far less than that of the PRM method, which exposes its more robust and reliable nature.

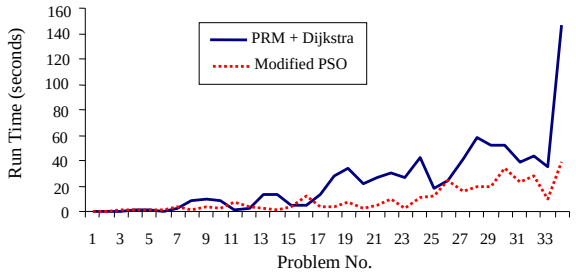

Figure 7. Comparison of Dijkstra and PSO algorithms' run times

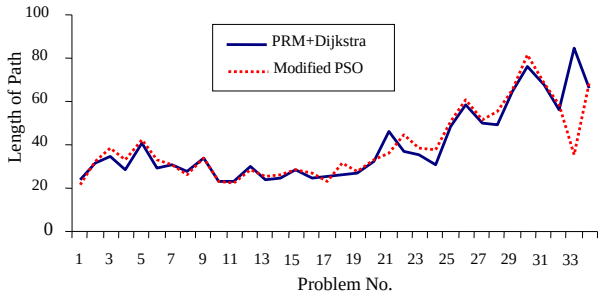

Figure 8. Comparison of lengths of path by Dijkstra and PSO algorithms

Table 2. Means and standard deviations of the solved problems

Algorithm	Time		Distance	
	Mean	STD	Mean	STD
RPM + Dijkstra	23.0	27.9	37.9	14.8
Multi-objective PSO	9.0	10.1	39.2	15.4

V. CONCLUSIONS

In this paper a new PSO-based robot motion planning algorithm is presented which handles two objectives simultaneously: the shortest path and the smoothest path. The algorithm has two main components: a PSO component used as the global planner, and a modified PRM component employed as the local planner.

The algorithm provides a unique and novel method to combine and unify the PSO and PRM components by integrating four groups of nodes into one single population: the best PSO particles, randomly generated PRM nodes, the robot's current and succeeding candidate positions, and a pair of nodes around each obstacle vertex. This node population is then connected via straight edges according to the PRM procedure and searched to find the shortest path between the robot's two successive positions.

By this, the free space around obstacles is efficiently searched in much less time than the classic PRM. Experiments and comparisons showed that the new algorithm is considerably faster than the classic PRM method (about %60), while being competitive in terms of path length. A direction for future research, could be adding another objective criterion, the safest path, to the algorithm. For this purpose, the Voronoi diagram can be utilized. In addition, this method can be generalized for motion planning of multiple robots, when they are considered disk-shaped or polygonal.

REFERENCES

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, "Principle of Robot Motion: Theory, Algorithms, and Application", MIT Press, Cambridge, 2005.

[2] E. Masehian, D. Sedighizadeh, "Classic and Heuristic Approaches in Robot Motion Planning - A Chronological Review", World Academy of Science, Engineering and Technology, Vol. 23, pp. 101-106, 2007.

[3] Q. Yuan-Qing, S. De-Bao, L. Ning, C. Yi-Gang, "Path Planning for Mobile Robot Using the Particle Swarm Optimization with Mutation Operator", Int. Conf. on Machine Learning and Cyber., pp. 2473-2478, 2004.

[4] W. Li, L. Yushu, D. Hongbin, X. Yuanqing, "Obstacle Avoidance Path Planning for Soccer Robots Using Particle Swarm Optimization", IEEE Int. Conf. on Rob. and Biomimetics (ROBIO), pp. 1233-1238, 2006.

[5] C. Xin, L. Yangmin, "Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization", IEEE on Mechatronics and Automation, pp. 1722-1727, 2006.

[6] L. Kavraki, P. Svestka, J.C. Latombe, M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", IEEE Trans. Robot. Autom., Vol. 12, No. 4, pp. 566-580, 1996.

[7] R. Hassan, B. Cohanim, O. De Weck, "A Comparison of Particle Swarm Optimization and the Genetic Algorithm", American Institute of Aeronautics and Astronautics, 2004.

[8] J. Kennedy, R.C. Eberhart, "Particle Swarm Optimization", IEEE Int. Conf. on Neural Networks, pp. 1942-1948, 1995.

[9] D. Sedighizadeh, E. Masehian, "A New Taxonomy for Particle Swarm Optimization (PSO)", 10th International Conference on Automation Technology, National Cheng Kung University, Tainan, Taiwan, pp. 317-322, 2009.

[10] Y. Shi, R. Eberhart, "Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight", Workshop on Particle Swarm Optimization, Indianapolis, 2001.

[11] K., Fujimura, "Path Planning with Multiple Objectives", J. of IEEE Robotics and Automation Society, Vol. 3, No. 1, pp. 33-38, 1996.

[12] H.Q. Min, J.H. Zhu, X.J. Zheng, "Obstacle Avoidance with Multi-Objective Optimization by PSO in Dynamic Environment", IEEE Int. Conf. Machine Learning and Cyber., Vol. 5, pp. 2950-2956, 2005.

[13] L. Gang, Y. Jianping, X. Yangsheng, "Multi-Objective Optimal Trajectory Planning of Space Robot Using Particle Swarm Optimization", Int. Symp. on Neural Networks, Vol. 5264, pp. 171-179, 2008.

BIOGRAPHIES

Davoud Sedighizadeh received the B.Sc. degree in Industrial Engineering from Amir Kabir University, Tehran, Iran in 2000, and the M.Sc. form Islamic Azad University, South Tehran Branch in 2003 and Ph.D. degrees in Industrial Engineering from Tarbiat Modares University, Tehran, Iran, in 2011, he is an Assistant Professor in Faculty of Engineering, Islamic Azad University, Saveh Branch, Saveh, Iran. His research interests are Optimization and modeling, Robotics, artificial intelligence, application of optimization in Industrial Engineering.