Multi-Objective PSO- and NPSO-based Algorithms for Robot Path Planning

Ellips MASEHIAN, Davoud SEDIGHIZADEH
Faculty of Engineering, Tarbiat Modares University, Tehran, P.O. Box 14115-143, Iran
masehian@modares.ac.ir, sedighizadeh@modares.ac.ir

***Abstract*—In this paper two novel Particle Swarm Optimization (PSO)-based algorithms are presented for robot path planning with respect to two objectives, the shortest and smoothest path criteria. The first algorithm is a hybrid of the PSO and the Probabilistic Roadmap (PRM) methods, in which the PSO serves as the global planner whereas the PRM performs the local planning task. The second algorithm is a combination of the New or Negative PSO (NPSO) and the PRM methods. Contrary to the basic PSO in which the best position of all particles up to the current iteration is used as a guide, the NPSO determines the most promising direction based on the negative of the worst particle position. The two objective functions are incorporated in the PSO equations, and the PSO and PRM are combined by adding good PSO particles as auxiliary nodes to the random nodes generated by the PRM. Both the PSO+PRM and NPSO+PRM algorithms are compared with the pure PRM method in path length and runtime. The results showed that the NPSO has a slight advantage over the PSO, and the generated paths are shorter and smoother than those of the PRM and are calculated in less time.**

***Index Terms*—Heuristic algorithms, Mobile robots, Particle swarm optimization, Path planning, Robot motion.**

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]. The existing classic methods are variations of a few general approaches: Roadmap, Cell Decomposition, Potential fields, and Mathematical programming, although these approaches are not essentially mutually exclusive, and combinations of them are often utilized in developing motion planners.

As a result of the 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 the 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 the Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony (ACO), and Tabu Search (TS) methods. A chronological review of the 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].

PSO has been extensively researched since its inception, and many variations of it have been proposed. The New or Negative PSO is one of them, which was first introduced in [6]. In this paper, we have employed the NPSO algorithm for the first time for robot motion planning, and have compared its efficiency with the basic PSO algorithm in this context.

Among other successful heuristics for robot motion planning are the probabilistic algorithms, including the Probabilistic Roadmap Method (PRM) and the Rapidly-exploring Random Trees (RRT) [7].

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 PSO and NPSO contexts.

The Basic PSO is motivated by the social behavior of organisms, such as bird flocking and fish schooling. Each particle studies its own previous best solution to the optimization problem, and its group's previous best, and then adjusts its position (solution) accordingly. The optimal value will be found by repeating this process. In the NPSO approach, each particle adjusts its position according to its own previous worst solution and its group's previous worst to find the optimal value. The strategy here is to avoid a particle's previous worst solution and its group's previous worst based on similar formulae of the Basic PSO [6].

The reasons for implementing the PSO method in our planner are that PSO is versatile enough to accommodate multiple objectives, and as shown in [8], it 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 and NPSO as the global planners, while the PRM is utilized as a local planner.

The combination and interaction of the NPSO and PRM methods is done for the first time in the field of motion planning, and as computational results have shown, they act

Digital Object Identifier 10.4316/AECE.2010.04011

very coherently since each of three method have probabilistic elements and parameters. More specifically, the notions of particles in the PSO, NPSO 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 methods iteratively shift from PSO (or NPSO) to PRM until the goal is reached.

The overall steps of the PSO-based 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.

The procedure of NPSO is almost the above except that instead of determining the best direction based on the best position of all particles (*gbest*), the worst position of all particles (*gworst*) is found and avoided.

In the next section of the paper the PSO and NPSO are introduced consecutively, 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 III, and in section IV 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 [9]. 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 [10] which categorize 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.

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

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

in which

$prtpos^i_j$= the position of the particle j in iteration i,
$prtvel^i_j$ = the velocity of the particle j in iteration i,
$pbest^i_j$ = the best position of the particle j in iteration i,
$gbest^i$ = the best position in the swarm till iteration i,
$\chi = 2 \Big/ \left| 2-\varphi-\sqrt{\varphi^2-4\varphi} \right|, \quad \varphi = \varphi_1 + \varphi_2 > 4.$

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 $c_1 = c_2 = 4$ [11]. 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.

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. Fig. 1 illustrates a schematic view of updating the position of a particle in two successive iterations in PSO algorithm.

The procedure of the PSO algorithm is presented in Fig. 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. The Negative PSO: Another Global Planner

In our second proposed model, the Negative Particle Swarm Optimization method is employed as the global motion planner. Similar to the pervious method (basic PSO), it is used for planning the large-scale, 'gross' motions of the robot.

In NPSO, which is a modified form of the basic PSO and was introduced in 2005 in [6], each particle adjusts its position according to its own previous worst solution and its group's previous worst to find the optimal value. The strategy here is to avoid a particle's previous worst solution and its group's previous worst based on similar formulae of the basic PSO.

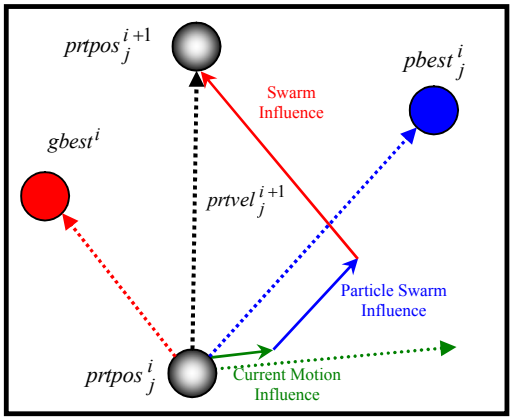

Fig. 1. Depiction of a particle's position update in PSO.

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

Fig. 2. Pseudocode of the basic PSO.

In NPSO 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 (3) and (4) and move to another position. Flying is affected by a fitness function that assesses the quality of each solution. Note the difference of equations (4) and (2).

$$prtpos_j^i = prtpos_j^{i-1} + prtvel_j^i, \tag{3}$$

$$prtvel_j^i = \chi \begin{bmatrix} w \times prtvel_j^{i-1} + \\ c_1 r_1 (prtpos_j^{i-1} - pworst_j^{i-1}) + \\ c_2 r_2 (prtpos_j^{i-1} - gworst^{i-1}) \end{bmatrix}, \tag{4}$$

in which
$prtpos^i_j$ = the position of particle j in iteration i,
$prtvel^i_j$ = the velocity of the particle j in iteration i,
$pworst^i_j$ = the worst position of particle j in iteration i,
$gworst^i$ = the worst position in the swarm till iteration i,
$\chi = 2 \Big/ \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|, \quad \varphi = \varphi_1 + \varphi_2 > 4$

Fig. 3 illustrates a schematic view of updating the position of a particle in two successive iterations in NPSO algorithm. Note the negative impact of *gworst* and *pworst*, represented by arrows. The procedure of the NPSO algorithm is presented in Fig. 4. 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 worst fitness values for each particle (*pworst*) and for the whole swarm (*gworst*) are calculated. Particles' positions and velocities are then updated in (3) and (4). Compare this procedure with the one shown in Fig. 2.

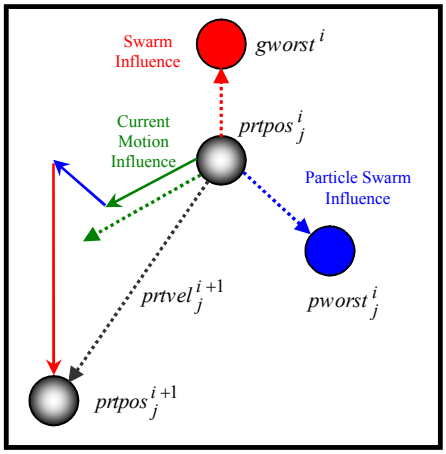

Fig. 3. Depiction of a particle's position update in NPSO.

Procedure Negative 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 worse than the worst fitness value in history (*pworst*)
      then Set current value as the new *pworst*
    end
  end
  for each particle **do**
    Find in the particle neighborhood the particle with the worst fitness (*gworst*)
    Calculate particle velocity $prtvel^i_j$ according to the velocity equation (4)
    Apply the velocity constriction
    Update the particle position $prtpos^i_j$ according to the position equation (3)
    Apply the position constriction
  end
end

Fig. 4. Pseudocode of the basic NPSO.

B. 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). Fig. 5 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. 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. This procedure is applied on both of PSO and NPSO algorithms.

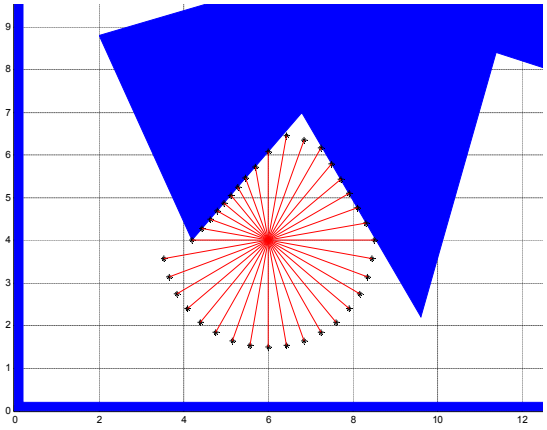

Fig. 5. The particles initial population is generated based on the borders of the robot's sensed area.

C. 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 which is considered as optimal in terms of a single criterion may not essentially satisfy other criteria all together [12]. For instance, a shortest path is not required at the expense of safety along the path.

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 [13], and multi-objective optimal trajectory planning of a space robot using PSO in [14].

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 [12]. 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 i-th 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 [15].

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. $gbes_i$ 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:

$$fitness(1)_j^i = \sqrt{(prtpos_j^i(x)-xgoal)^2+(prtpos_j^i(y)-ygoal)^2} \quad (5)$$

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

$$fitness(2)_j^i = \frac{\cos^{-1}\left[(prtpos_j^i(x)-xgoal)\times(gbest_x^{i-1}-xgoal)+(prtpos_j^i(y)-ygoal)\times(gbest_y^{i-1}-ygoal)\right]}{\left[\sqrt{(prtpos_j^i(x)-xgoal)^2+(prtpos_j^i(y)-ygoal)^2}\times\sqrt{(gbest_x^{i-1}-xgoal)^2+(gbest_y^{i-1}-ygoal)^2}\right]} \quad (6)$$

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

$$fitness_j^i = \lambda_1 \times fitness(1)_j^i + \lambda_2 \times fitness(2)_j^i \quad (7)$$

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 $\lambda_2 = 0.25$.

D. 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 I, which are obtained after numerous runs.

After incorporating the developed fitness function in the general PSO algorithm (Fig. 2), and applying the parameters tuned properly, the positions of the generated particles are updated using (1) and (2). This work is accomplished for NPSO also. 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.

TABLE I.
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
Acceleration constant c_2	Attracts the particle's position towards the *gbest*		▼Low velocity and acceleration of particle movements are required
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
Max position (x_{max}, y_{max})	The maximum position on which a particle can be located		
Min position (x_{min}, y_{min})	The minimum position on which a particle can be located		
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

Initial PRMs succeeded in solving a number of complex problems with high-dimensional configuration spaces which had not been solved efficiently until that time [7]. 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 (Fig. 6). The path planning is done by searching this graph.

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 and NPSO,
(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 and 30%-35% of NPSO particles with highest fitness values (*pbests*) are also integrated in the PRM graph. The item (d) helps in circumnavigating obstacle vertices naturally and easily.

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.

The procedure of the NPSO algorithm in RMP is to update the positions in the NPSO based on the worst positions of particles. The best position of each particle and the best position among all particles are simultaneously calculated in every stage, and the robot position in each stage is the best global position of all particles (*gbest*). In other words and based on Equations (3) and (4), avoiding from the worst positions specifies the next positions of the particles and the best position achieved in each stage shows the best position of robot.

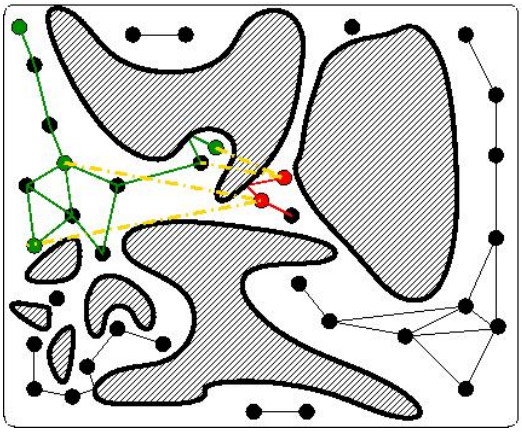

Fig. 6. A snapshot of the edge building process using the PRM. Roadmap edges are generated such that they lie entirely in the free space.

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 RMP based on PSO and NPSO on problems with simple to complex obstacles are illustrated consecutively in Figs. 5 and 6.

As a competing method for comparing both PSO and NPSO algorithms' performances, the efficient 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. In order to avoid obstacles corners, their vertices were incorporated in the network of random nodes.

For comparison, 35 problems with different obstacles sizes and shapes (both convex and concave) were designed and solved by the new multi objective PSO, new multi objective NPSO, and PRM + Dijkstra methods. Each of the three methods was 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 Fig. 7. Regarding the probabilistic nature of the PSO algorithms, each problem was run 5 times and their average runtime and path length were calculated for making comparisons. In total, 5×35×3 = 525 experiments were simulated. The runtime and path lengths obtained by each of the three methods are graphically superimposed in Figs. 8 and 9, respectively.

Fig. 5. Some simulations of the developed method based on PSO.

The path lengths and runtimes of the solved problems are presented in Table II in detail, and the comparison results are summarized in Table III, in which the mean and standard deviation of runtimes and path lengths of the 35 problems are calculated for each method.

The results demonstrate that the PSO- and NPSO-based models are about %45 and %52 faster than the PRM + Dijkstra method respectively, while the path lengths do not differ significantly.

In order to test the efficiency of the algorithm, an experimental robot was run in a sample workspace based on the path generated by the algorithm. The robot's size is 12×12 cm, which is controlled by an embedded micro-

TABLE II.
PATH LENGTHS AND RUNTIMES OF THE SOLVED PROBLEMS IN DETAIL

No. of Vertices	Criteria	Method		
		PRM+ Dijkstra	BAPSO	NPSO
17	Distance	34.19	34.92	35.77
	Time	0.04	1.01	0.36
20	Distance	23.61	25.39	24.01
	Time	0.20	1.55	0.46
26	Distance	31.58	33.17	33.17
	Time	0.25	0.80	0.42
31	Distance	34.48	44.72	42.71
	Time	0.38	1.93	1.57
37	Distance	28.44	19.88	32.81
	Time	0.62	0.98	0.78
40	Distance	40.86	67.98	45.30
	Time	1.57	12.21	2.75
46	Distance	29.53	32.85	32.93
	Time	0.44	2.06	1.56
64	Distance	30.39	30.53	29.64
	Time	2.28	4.27	3.09
67	Distance	27.44	32.14	29.58
	Time	8.28	2.86	2.20
68	Distance	33.60	40.13	33.99
	Time	9.22	6.09	4.01
69	Distance	22.82	27.18	23.41
	Time	8.31	2.77	2.27
72	Distance	23.17	24.60	22.36
	Time	1.16	2.03	1.72
74	Distance	29.97	42.09	30.16
	Time	2.39	10.92	4.60
80	Distance	23.56	26.56	25.79
	Time	13.12	3.33	1.91
83	Distance	24.85	31.71	28.14
	Time	13.69	2.62	3.89
84	Distance	28.13	28.43	26.33
	Time	4.63	5.02	4.15
94	Distance	24.72	28.42	26.90
	Time	4.53	17.78	15.63
104	Distance	25.34	23.44	22.74
	Time	12.91	3.26	3.42
114	Distance	26.51	36.14	36.05
	Time	28.44	22.11	5.66
118	Distance	27.19	30.19	30.39
	Time	34.42	11.84	8.52
124	Distance	32.11	35.26	32.90
	Time	21.34	5.06	2.48
134	Distance	45.80	48.61	45.90
	Time	26.49	19.68	7.85
144	Distance	36.97	45.84	40.16
	Time	30.08	8.34	7.22
154	Distance	35.09	37.21	38.98
	Time	26.12	7.56	7.74
164	Distance	30.72	48.65	40.26
	Time	42.78	19.64	7.98
174	Distance	48.82	48.82	48.82
	Time	17.68	12.27	11.78
184	Distance	58.40	60.08	58.55
	Time	24.55	25.09	30.75
194	Distance	50.08	66.29	51.87
	Time	41.81	4.86	12.01
204	Distance	49.35	58.72	57.88
	Time	58.04	26.71	20.11
224	Distance	64.97	66.85	67.30
	Time	51.79	32.22	30.30
244	Distance	76.08	81.54	81.49
	Time	52.28	35.96	34.00
255	Distance	67.93	69.91	68.08
	Time	38.82	19.74	34.31
274	Distance	56.23	58.52	105.61
	Time	43.15	28.31	24.87
294	Distance	84.74	36.24	37.81
	Time	34.97	0.39	10.11
414	Distance	66.49	70.56	70.26
	Time	146.43	76.97	75.01

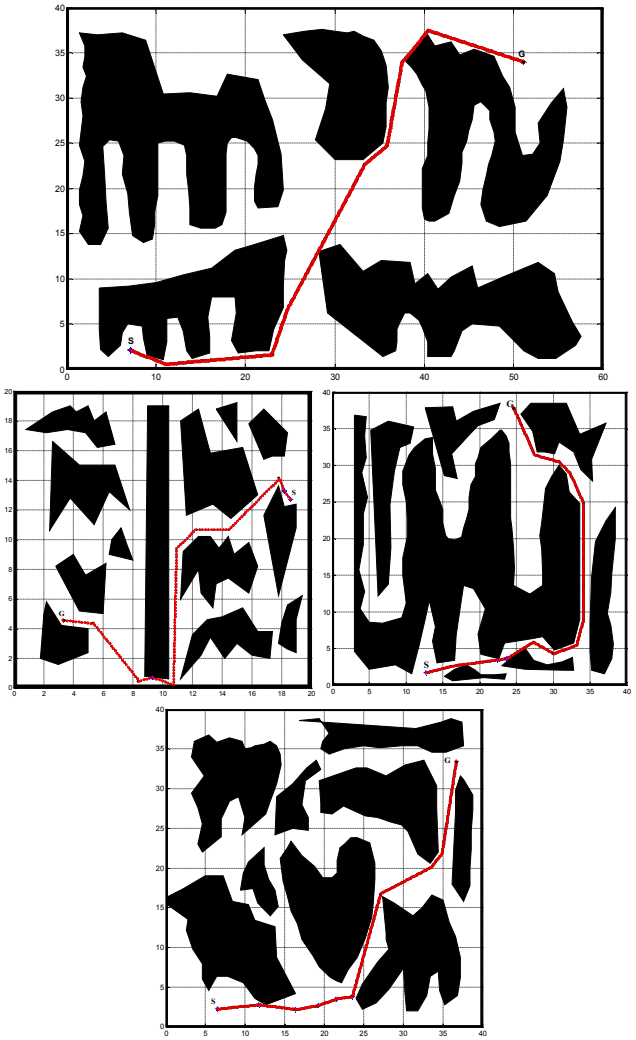

Fig. 6. Some simulations of solved problems by the NPSO.

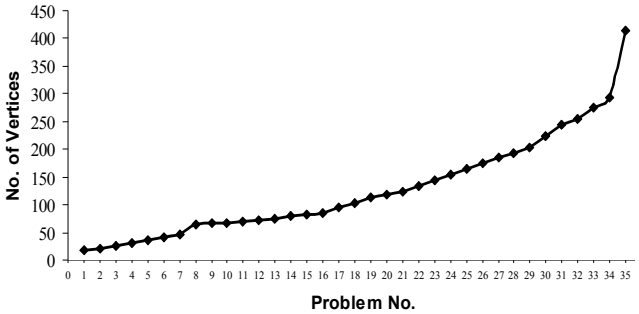

Fig. 7. Number of vertices in various test problems.

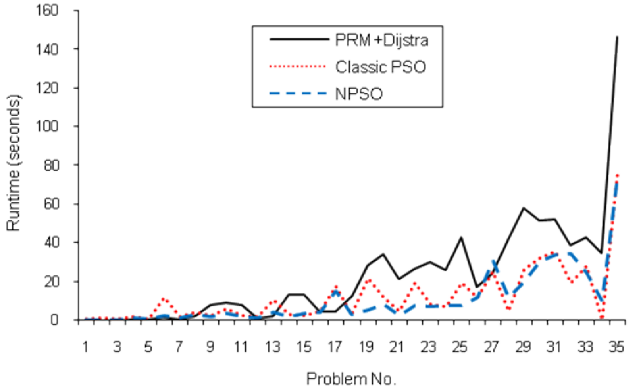

Fig. 8. Comparison of the runtimes of the PRM+Dijkstra, Classic PSO and NPSO algorithms.

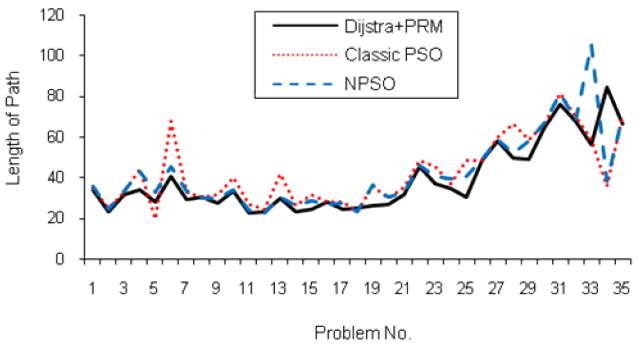

Fig. 9. Comparison of the path lengths produced by PRM+Dijkstra, Classic PSO and NPSO algorithms.

TABLE III.
MEANS AND STANDARD DEVIATIONS OF THE SOLVED PROBLEMS

Algorithm	Runtime		Path length	
	Mean	STD	Mean	STD
RPM + Dijkstra	23.0	27.9	38.28	16.74
Classic PSO	12.5	15.1	41.68	16.38
NPSO	11	15.0	40.68	18.67

controller (Fig. 10) and the workspace size about 100×130 cm (Fig. 11). The robot successfully navigated in the workspace and managed to reach the goal in a few seconds.

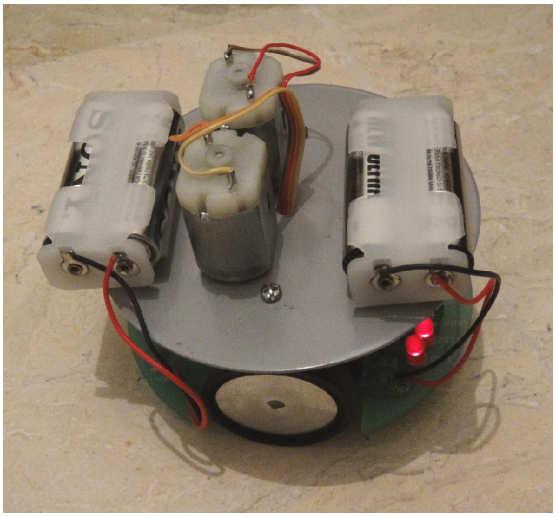

Fig. 8. The experimental robot.

Figures 12(a)-(f) show the successive movements of the robot at each iteration toward the *gbest* of each iteration. The robot collects information about the environment obstacles via its sensors and avoids them by finding its next best point until the goal is reached.

V. CONCLUSION

In this paper two PSO- and NPSO-based robot motion planning algorithms are presented which handle two objectives simultaneously: the shortest and the smoothest path criteria. Each algorithm has two main components: a PSO-based component used as global planner, and a modified PRM component employed as the local planner.

Fig. 11. The workspace, obstacles, and Start and Goal positions.

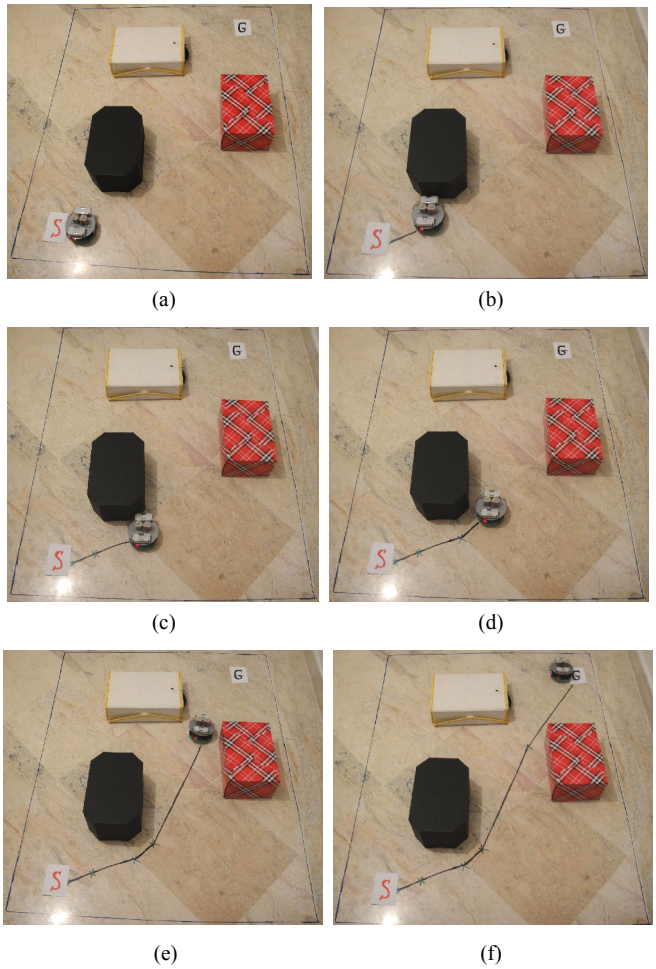

Fig. 12. (a) to (f) The robot reaches the Goal point in 6 iterations. Each stopping point represents a *gbest* in each iteration.

The algorithm provides a unique and novel method to combine and unify the PSO and PRM components also NPSO and PRM components by integrating four groups of nodes into one single population: the best PSO and NPSO particles separately, 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 algorithms is considerably faster than the classic PRM method (about %45 for PSO-based algorithm and 52% for NPSO-based algorithm), while being competitive in terms of path length.

As a direction for future research, another objective criterion such as the safest path could be added to the algorithm. For this purpose, the Voronoi diagram can be utilized. Also, this method can be generalized for motion planning of multiple robots, when they are considered to be disk-shaped or polygonal.

REFERENCES

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thrun, *Principle of Robot Motion: Theory, Algorithms, and Application*, MIT Press, Cambridge, 2005, ISBN 0-262-03327-5.

[2] E. Masehian, and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning–a chronological review", Proc. of *World Academy of Science, Engineering and Technology*, Vol. 23, pp. 101-106, 2007.

[3] Q. Yuan-Qing, S. De-Bao, L. Ning and C. Yi-Gang, "Path planning for mobile robot using the particle swarm optimization with mutation operator", in Proc. *Int. Conf. on Machine Learning and Cybernetics*, pp. 2473 – 2478, 2004.

[4] W. Li, L. Yushu, D. Hongbin, and X. Yuanqing, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization", in Proc. IEEE *Int. Conf. on Rob. and Biomimetics* (ROBIO) pp. 1233-1238, 2006. DOI: 10.1109/ROBIO.2006.340104.

[5] C. Xin and L. Yangmin, "Smooth path planning of a mobile robot using stochastic particle swarm optimization" in Proc. *IEEE on Mechatronics and Automation*, pp. 1722-1727, 2006. DOI: 10.1109/ICMA.2006.257474.

[6] Yang, C. and Simon, D., "A new particle swarm optimization technique", in Proc. *IEEE Int. Conf. on Systems Engineering*, pp. 164-169, 2005. DOI: 10.1109/ICSENG.2005.9.

[7] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *IEEE Trans. Robot. Autom.* Vol 12, No. 4, pp. 566-580, 1996. DOI: 10.1109/70.508439.

[8] R. Hassan; B. Cohanim, and O. de Weck, "A comparison of particle swarm optimization and the genetic algorithm", *American Institute of Aeronautics and Astronautics*, 2004.

[9] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization", in Proc. *IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995. DOI: 10.1109/ICNN.1995.488968.

[10] D. Sedighizadeh and E. Masehian, "A new taxonomy for particle swarm optimization (PSO)", in Proc. *10th International Conference on Automation Technology*, National Cheng Kung University, Tainan, Taiwan, pp. 317-322, 2009.

[11] Y. Shi and R. Eberhart, "Particle swarm optimization with fuzzy adaptive inertia weight", in Proc. *Workshop on Particle Swarm Optimization*, Indianapolis, 2001.

[12] K., Fujimura, "Path planning with multiple objectives", J. of *IEEE Robotics and Automation* Society, vol.3. No.1, pp. 33-38, 1996. DOI: 10.1109/100.486659.

[13] H.Q. Min, J.H. Zhu, and X.J. Zheng, "Obstacle avoidance with multi-objective optimization by PSO in dynamic environment", in Proc. *IEEE Int. Conf. Machine Learning and Cyber*., Vol. 5, pp. 2950-2956, 2005. DOI: 10.1109/ICMLC.2005.1527447.

[14] L. Gang, Y. Jianping, and X. Yangsheng, "Multi-objective optimal trajectory planning of space robot using particle swarm optimization", vol. 5264, Proc. of *Int. symp. On Neural Networks*, pp. 171-179, 2008. DOI: 10.1007/978-3-540-87734-9_20.

[15] D.O. Kang, S.H. Kim, H. Lee, and Z. Bien, "Multi objective navigation of a guide mobile robot for the visually impaired based on intention inference of obstacles", *J. of Autonomous Robots*, Vol. 10, No. 2, 2001. DOI: 10.1023/A:1008990105090.