An Improved Particle Swarm Optimization Method for Motion Planning of Multiple Robots

Ellips Masehian and Davoud Sedighizadeh

Abstract. Multi robot motion planning is a challenging problem in the robotics field due to its complexity and high computational costs induced by the number of robots. In this paper a new heuristic method is presented for solving this problem through a decentralized approach with global coordination. The method is based on a new improved variant of the Particle Swarm Optimization (PSO) metaheuristic, which serves as a global planner. Alternatively, for local planning and avoiding obstacles in narrow passages, the Probabilistic Roadmap Method (PRM) is employed. The global and local planners act sequentially until all robots reach their goals. The algorithm iteratively and simultaneously minimizes two main objectives, shortness and smoothness of the paths. The proposed algorithm is simulated and compared with the standard (basic) PSO, as well as the standard Probabilistic Roadmap methods. The experimental results show a meaningful advantage of the new method regarding computational time and path quality.

1 Introduction

The robot motion planning discipline experienced a boost specifically after the advent of the Configuration Space (C-space) notion by Lozano-Pérez and Wesley in the mid 70's [1]. While early motion planning algorithms were mainly developed for single robots, the multi robot motion planning problem remained untackled until a decade later, when the prioritization and coordination concepts were developed, as in [2].

The general single robot motion planning problem is defined as the problem of finding a collision-free path for a robot navigating among various obstacles, and is classified as a PSPACE-hard and NP-hard problem [3]. This complexity is further increased for multi-robot motion planning as the larger number of robots creates difficult problems regarding their coordination, cooperation and obstacle avoidance. Thus, as a challenging problem, the multi robot motion planning problem increasingly attracts the attention of roboticists and researchers.

Ellips Masehian · Davoud Sedighizadeh
Faculty of Engineering, Tarbiat Modares University, Tehran, Iran

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 175–188.
springerlink.com

The primary approaches for path planning of single and multiple robots were generally based on computational geometry and handled deterministic low-dimensional problems. These methods, also known as classic methods, are variations of a few general techniques: Roadmaps (including Visibility Graph, Voronoi diagrams, and Silhouette), Cell Decomposition, Potential Fields, and Mathematical programming (including operations research and game theory models) [4].

Due to the complexities of the motion planning problem and its progressive increase for the multi-robot case, many heuristic and metaheuristic methods have been developed or applied extensively over the recent years, generally showing better performance than the classic methods in terms of computational burden. However, it should be noticed that heuristic methods do not guarantee to find a solution, but if a solution is found, it is done in much shorter time than exact methods.

1.1 Multi Robot Motion Planning

The Multi Robot Motion Planning (MRMP) problem has been solved through two main approaches: *centralized* and *decentralized* (or *decoupled*) [5].

The centralized planning considers all of the robots concurrently; that is, paths for all robots are planned simultaneously by searching the C-space of a hypothetical multi-arm robot consisted of all the robots, in which collisions between robots are considered as self-collisions of the multi-arm robot. The degrees of freedom (dof) of this hypothetical robot equals to the sum of the dof's of all individual robots. The main advantage of the centralized planning is that it is complete; i.e., it is guaranteed to find a solution if one exists. However, it is potentially expensive and typically requires searching high-dimensional spaces and the knowledge of goals and states of all robots, meaning that it hardly can be applied for online situations.

The decoupled planning performs the motion planning of each robot independently and sequentially, and has two phases: first a collision-free path τ_1 is generated for each robot considering only obstacles (ignoring other robots) in its space, and then, in order to prevent collisions between the robots, the robots' motions along their pre-generated paths are coordinated via two main techniques, namely *prioritization*, and velocity tuning. Each robot is restricted to move along its previously-generated path, although it may stop, retreat or change velocity to allow coordination with other robots [5].

The two main coordination approaches are *pairwise* and *global* coordination. In the pairwise coordination, the paths τ_1 and τ_2 of the first two robots are coordinated in their 2-dimensional coordination space. The process is repeated for paths $\tau_{1,\,2}$ and τ_3 resulting in a coordinated path $\tau_{1,\,2,\,3}$. Eventually, a collision-free coordinate path $\tau_{1,\,2,\,\ldots,\,m}$ is generated that defines a valid coordination of all m robots. In global coordination, the paths of all m robots are coordinated in an m-dimensional coordination space, resulting in a collision-free path $\tau_{1,\,2,\,\ldots,\,m}$.

The decoupled planning is generally less computationally expensive than the centralized planning since lower dimensional spaces are searched [6]. However, it is not complete, and failures usually occur in the second phase as it might not be possible to coordinate the paths generated in the first phase without collision

between robots [7]. Nevertheless, some attempts have been made to combine the centralized and decoupled approaches [8].

A trend of applying metaheuristic algorithms such as simulated annealing (SA), genetic algorithms (GA), and ant colony optimization (ACO) to the MRMP problem is noticeable especially among more recent contributions, as in [9], [10], and [11], respectively. Also, the particle swarm optimization (PSO) algorithm has found some applications MRMP. The first work in this regard is due to [12] in which the PSO is used for single and multiple target tracing applications for multiple robots. In [13] obstacle avoidance is done for a single robot in dynamic environments, in [14] bio-inspired group behaviors for deployment of a swarm of robots to multiple destinations are proposed. Other fresh works in this regard are [15], [16], and [17]. In [18] a PSO-inspired algorithm is proposed as a framework for robots to work together to find their targets. In [19] an asynchronous mechanism is proposed for information exchange and position update of small robots with limited sensing capabilities. In [20] a PSO-based method is developed for searching operations by a large number of mobile robots, with small inter-robot communications.

In this paper, a new PSO-based algorithm is developed for MRMP. The reason of employing the PSO is that as a population-based metaheuristic, it is very consistent with the distributed nature of multi robot systems. Moreover, although both the PSO and GA are population-based metaheuristics, the PSO proved to be more efficient and faster than the GA algorithm as reported in [21], after they were analyzed and compared statistically from both efficiency (speed) and effectiveness (quality) perspectives for eight optimization functions. The advantage of PSO over GA is also mentioned in [22].

A distinctive feature of the presented work, as compared to the previous works, is that the PSO is combined with a well-known and fast motion planning technique, called Probabilistic Road Map method (PRM), to produce obstacle-free paths in shorter times. Also, in order to enhance the quality of the produced paths, a multi-objective fitness function has been developed to minimize the path length while discouraging the robot to make sharp and abrupt turns, thus maintaining its smoothness.

2 Overview of the New Method

After analyzing many PSO-based algorithms and examining their components, it was found out that PSO is more successful in *diversification* rather than *intensification* due to high distribution of the particles in the space [23]. Intensification forces the algorithm to search a specific area with more depth and within a local scope, while diversification forces exploration of completely new regions, acting in a global scope. Therefore, the PSO component of the proposed algorithm was considered as a global planner, with the responsibility of searching and exploring new areas. This idea was first used in our previous work for a single robot [24].

Our analysis also showed that the PSO is not sufficiently efficient in obstacle avoiding, especially when a large number of obstacles populate the workspace densely, or there are narrow passageways in the workspace. Although thanks to the probabilistic nature of the PSO it can eventually find a collision-free path from the

robot's start to goal, this usually happens after so many unsuccessful attempts and thus takes considerable time. To remedy this drawback, we took advantage of a fast planner, namely, the *Probabilistic Road Map* (PRM) method, which is based on searching a graph with randomly-generated nodes and edges and is more powerful in local search (i.e. intensification). This component is described in section 4.

In addition to the abovementioned speed issue, the quality of the paths is also of great importance. Considering that the two major attributes of a high-quality path are its shortness and smoothness, we tried to incorporate these dual objectives in the fitness function, and concurrently minimize the length and maximize the smoothness of the path. This issue is addressed in section 3.

As the speed and efficiency are of specific importance in this work, the decentralized approach was employed: in fact, for m robots, m PSO algorithms are performed sequentially but independently in each iteration to determine the positions of the robots. This process is iterated until all robots reach their goals.

The combination and interaction of the PSO and PRM methods is a new concept in the field of multi-robot motion planning. As computational results have shown, PSO and PRM act very coherently since both have probabilistic elements and parameters. More specifically, the notions of particles in the PSO and random nodes generated in the PRM complement each other and unify these methods. The algorithm iteratively shifts from PSO to PRM until all robots' goals are reached.

For each robot, the following steps are executed:

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 improved 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 2, then set it as the robot's next position and go to Step 2, otherwise continue with Step 4.
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 collision-free line.

The above steps are executed for every robot separately and concurrently until the last robot reaches its goal.

As mentioned earlier, the decentralized planning consists of two phases: the first phase specifies a collision-free start-to-goal path for each robot without considering other robots, and the second phase deals with velocity tuning, in which the robots' velocities along their generated paths are coordinated in order to avoid collision among the robots. In the proposed algorithm the *global coordination* approach is implemented for the velocity tuning, in which the paths of all m robots are coordinated in an m-dimensional space. Each robot is limited to move along its

previously generated path although it may stop and vary its speed for coordination with other robots. More precisely, whenever two robots get closer than a limit to each other, moving priorities are assigned to them at random, after which the robot with lower priority reduces speed to let the robot with higher priority pass.

Depending on the robot's start and goal positions, each robot reaches its goal at different times and after different number of iterations, and since the algorithm runs for each robot in parallel with others, at a specific moment, the planning phases underway for each robot might be different from other robots. Therefore, another factor called *action mode* was introduced to precisely describe the situation of each robot at a given time. This concept facilitates the robots' coordination and increases the algorithm's speed and efficiency.

There are five modes for each robot as explained below, in which $gbest^i$ is the position of the robot at the i-th iteration:

Mode 1 is for when a robot can move from $gbest^i$ to $gbest^{i+1}$ via a straight line without colliding with any obstacle. In other words, the robot's motion is planned by the global algorithm (PSO).
Mode 2 is for when Mode 1 does not hold due to collision with obstacles. As a result, the robot moves from $gbest^i$ to the nearest node in the PRM network.
Mode 3 is for when robot moves between two nodes of the PRM network. In other words, the robot's motion is totally planned by the local algorithm (PRM).
Mode 4 is for when the robot abandons the PRM network and moves to $gbest^{i+1}$ straightforwardly.
Mode 5 is for when the robot's goal is within its line of sight and can be reached directly without collision with obstacles.

It should be noted that the robots traverse the lines between two successive points with regard to their own speed and step size, and sequentially move to the intermediate points obtained from interpolating the line. For any robot, if after taking a step towards its desired point, it is estimated that a collision with another robot is imminent, the robot will automatically reduce its step size such that the collision is avoided. Therefore, a global coordination is performed at each iteration.

Also, at the end of each action mode an attempt is made to connect the robot's current position to its goal via a straight line. If it fails, the robot will continue moving to the $gbest^{i+1}$ directly or through a PRM network. If the attempt is successful, then the robot reaches its goal and the algorithm terminates unconditionally for that robot. However, since the robots do not essentially reach their goal at the same time, the termination criterion of the algorithm is satisfied whenever the last robot gets to its goal.

In the following two sections the details of the PSO and PRM components are described in detail.

3 PSO: The Global Planner

In the proposed MRMP method, 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 robots. In this section, an overview of the basic (standard) PSO algorithm is presented.

The basic Particle Swarm Optimization algorithm was proposed by Kennedy and Eberhart in 1995 [25], inspired by the collective behavior of swarms of fish, birds, etc. Each member of the swarm is denoted by a particle, which shows a solution candidate. The particles start their fly from random positions in a search area, and in each iteration, they update their positions and velocities according to equations (1) and (2) below, and move to another position. Flying is affected by a fitness function that assesses the quality of each solution.

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

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

in which:

$prtpos^i_j$ = the position of the j-th particle in j-th iteration,
$prtvel^i_j$ = the velocity of the j-th particle in j-th iteration,
$pbest^{i-1}_j$ = the best position of the j-th particle at the end of $(i-1)$-th iteration,
$gbest^{i-1}$ = the best position in the swarm at the end of $(i-1)$-th iteration,

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

The PSO has some dependent parameters: c_1 is a constant called *cognitive acceleration coefficient*, and c_2 is another constant named *collective acceleration coefficient*. These factors balance the effect of self-knowledge and social knowledge when particles move towards the target, and are usually set to a value of 2, although good results have been also produced with $c_1 = c_2 = 4$ [26]. r_1 and r_2 are random numbers between 0 and 1, different at each iteration, and χ is the *constriction factor*, which limits the velocity. w is a weight that regulates the global search behavior, set to an upper bound w_{max} in the beginning of the searching process and dynamically reduced during the optimization to a lower bound w_{min}, (which emulates a deeper local search behavior). Its range is suggested to be [0.2, 0.4].

The first term of equation (2), i.e. ($prtvel^{i-1}_j$), considers the velocity of the particle in the previous iteration, which produces a momentum needed for particles to fly all over the search space. The second term, ($pbest^{i-1}_j - prtpos^{i-1}_j$), which is known as the cognitive part, simulates the 'personal memory' of a particle: it encourages the particles to fly towards the best position they have found till now (i.e. *pbest*). Finally the third term, ($gbest^{i-1} - prtpos^{i-1}_j$), called the *collective part*, presents the effect of the particles' cooperation in finding the global optimum: it always directs the particles towards the best position ever found among all the members of the swarm.

The overall procedure of the PSO method 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 based on (1) and (2).

To improve the performance of the basic PSO and increase its efficiency, we propose a modified, improved variant of the PSO algorithm. The new variant incorporates two new criteria for the particles' velocity updating equation, as shown by equation (3). The particles' positions are still updated by equation (1).

$$prtvel_j^i = \chi \cdot \begin{bmatrix} w_1 \cdot (prtvel_j^{i-1}) + \\ w_2 \cdot c_1 \cdot r_1 \cdot (pbest_j^{i-1} - prtpos_j^{i-1}) + \\ w_3 \cdot c_2 \cdot r_2 \cdot \alpha_1 \cdot (gbest^{i-1} - prtpos_j^{i-1}) + \\ w_4 \cdot c_3 \cdot r_3 \cdot \alpha_2 \cdot (pbest_{rand}^{i-1} - prtpos_j^{i-1}) + \\ w_5 \cdot c_4 \cdot r_4 \cdot \alpha_3 \cdot (prtvel_{rand}) \end{bmatrix} \tag{3}$$

In this equation, $pbest^i_{rand}$ is the best position of a randomly selected particle in the i-th iteration, $prtvel_{rand}$ is a random velocity vector with a size between $V_{\min}$ and $V_{\max}$, w_1 is the inertia weight, w_2–w_5 are control weights within [0.4, 0.9], c_1–c_4 are acceleration constants within [1.5, 4], r_1–r_4 are random numbers different at each iteration and in the range of [0, 1], and α_1–α_3 are respectively the influence factors of *gbest*, *pbest*, and $prtvel_{rand}$, in the ranges of [0, 10], [0, 20], and [0, 1].

We added two new terms based on the following logic: the fourth term of the velocity update equation, which we call the '*random self-cognition part*', sends the particles towards one of the best positions found randomly by particles ($pbest^{i-1}_{rand}$). This scheme gives an opportunity for reasonably good local positions of other randomly selected particles in the swarm to influence the velocities of other particles. The fifth term, which is enforced through the random velocity parameter $prtvel_{rand}$, increases the variety in the swarm and leads to a better and more effective movement of the swarm in narrow and complicated search areas.

All but the first term of equation (3) contribute to the overall velocity updating process in random proportions at each iteration. Consequently, the particles' positions spread all over the search space and the goal is reached quickly. If a particle lies inside an obstacle, it is simply deleted from the swarm and replaced by random particles in the free space.

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, a set of particles are generated for each robot with respect to its 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, on the number of divisions of the circumferential circle). Fig. 1 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 our method, while maintaining the centralization of the robot's start point, the obstacles' distribution around it is also considered.

3.1 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 meets several conditions, such as safety, estimated needed time for navigation, energy consumption, etc.

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. Categorically, 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 [27]. This is just one type of problems for which our multi-objective approach has been designed. In the developed method, the Simple Additive Weighting (SAW) technique is employed, in which a weighted sum of multiple objectives is expressed as a conventional single-objective function.

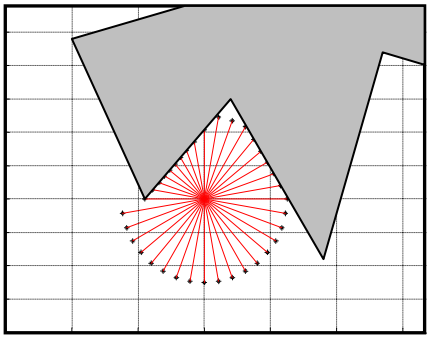

Fig. 1 The particles' initial population is generated based on the borders of the robot's sensed area

In our 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 two hypothetical lines connecting the goal point to the robot's positions in two successive iterations, 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, i.e. the shortest path for the k-th robot, and the second objective function, the smoothest path for the k-th robot, are mathematically expressed in equations (4) and (5),

$$F_{\text{short}}(k)_j^i = \left\| prtpos(k)_j^i - goal(k) \right\| \tag{4}$$

$$F_{\text{smooth}}(k)_j^i = \frac{\cos^{-1}\left[(x_{prtpos(k)_j^i} - x_{goal(k)}) \cdot (x_{gbest(k)^{i-1}} - x_{goal(k)}) + (y_{prtpos(k)_j^i} - y_{goal(k)}) \cdot (y_{gbest(k)^{i-1}} - y_{goal(k)}) \right]}{\sqrt{(x_{prtpos(k)_j^i} - x_{goal(k)})^2 + (y_{prtpos(k)_j^i} - y_{goal(k)})^2} \times \sqrt{(x_{gbest(k)^{i-1}} - x_{goal(k)})^2 + (y_{gbest(k)^{i-1}} - y_{goal(k)})^2}} \tag{5}$$

in which (4) shows the distance of the particles' position to the goal point, and $k = 1, \ldots, m$ is the number of robot. The overall fitness (or objective) function is obtained by the weighted sum of the above shortest and smoothest objectives:

$$Fitness_j^i = \lambda_1 \cdot F_{\text{short}}{}_j^i + \lambda_2 \cdot F_{\text{smooth}}{}_j^i \tag{6}$$

By minimizing the overall fitness function with the assigned weights of each criterion, a shortest path with the least oscillations is obtained. The weights of the shortest and smoothest fitness functions, λ_1 and λ_2, are tuned through extensive simulation and try and errors, with best found values of $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$.

4 PRM: 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 [28]. The PRM was enhanced later into some variant forms like Medial Axis PRM (MAPRM), Obstacle-Based PRM (OBPRM), and Visibility-based PRM, which improved the process of random node generation and made it more effective [29]. The PRM has also been applied in the multi robot systems [30].

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. 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 PRM nodes:

(*i*) a number of randomly generated nodes,
(*ii*) the robot's current position,
(*iii*) the best particles generated in the PSO,
(*iv*) two points around each corner of the obstructing obstacle.

The above combination of nodes is proposed for the first time in the literature, and provides a subtle intertwining of the PSO and PRM methods. In addition to the randomly generated nodes (group (*i*)) 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 group (*iv*) helps in circumnavigating obstacle vertices naturally and easily by creating nodes at safe clearances from both sides of a vertex.

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's search algorithm. As a result, the robot can move from $gbest^{i}$ to $gbest^{i+1}$ and get closer to the goal, while avoiding the obstacles that locally intercept its path to the goal. Once the robot is located on its new position, the PSO particles' velocities and positions are updated again, as described in equations (1) and (2).

5 Experimental Results

In order to analyze the function of the proposed new algorithm, numerous simulations were run for 2-, 3-, 4-, and 5-robot problems through which the algorithm's parameters were tuned to their best values. A few simulations for problems with simple to complex obstacles are illustrated in Fig. 2.

For comparing the algorithm's performance with other efficient and well-known algorithms, the standard PRM method was selected. Ten sample problems with 20 to 414 vertices were designed and solved for 2, 3, 4, and 5 robots using the proposed Improved PSO+PRM, Standard PSO+PRM, and standard PRM methods. Regarding that all these algorithms are heuristic and incorporate random parameters, we solved each problem set 5 times and calculated the mean value of runtimes. Note that the runtime is calculated based on the time needed for the last robot to reach its goal. In total, (10 problems) × (4 sets of robots) × (3 methods) × (5 times each) = 600 instances were run on an Intel 3.0 GHz processor.

The standard PSO against which we tested our algorithm was coded based on the basic PSO algorithm proposed in [25], combined with the PRM method. Also, the standard PRM was coded according to the explanations in section 4. In the above three methods, whenever a probabilistic roadmap was constructed (either in combination or stand-alone), it was searched by the Dijkstra's method to yield a shortest path on the roadmap. The results of solving the test problems are shown in Fig. 3 for 2, 3, 4, and 5 robots (from left to right, up to dawn), summarized in Table 1.

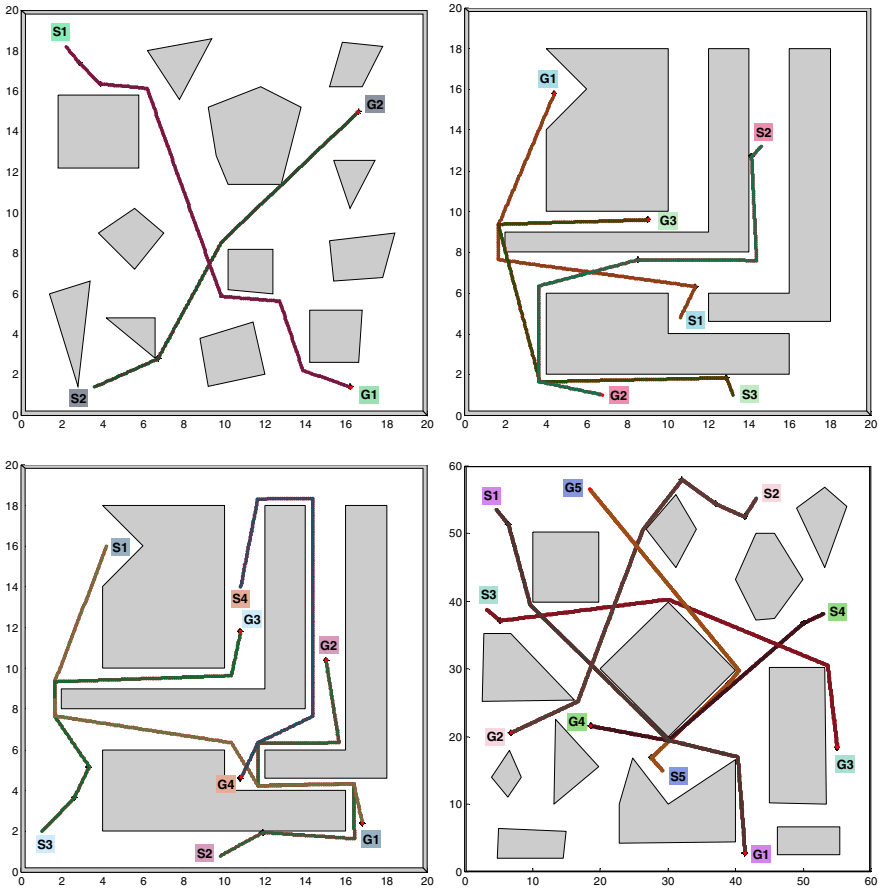

Fig. 2 Some simulations for 2-, 3-, 4-, and 5-robot motion planning. S_i and G_i indicate the start and goal of the i-th robot, respectively.

Table 1 Comparison of the average runtimes of the three methods and their standard deviations

	2 robots		3 robots		4 robots		5 robots		Total
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.
Improved PSO + PRM	22.28	30.76	28.95	36.59	34.65	41.58	38.77	43.69	31.16
Standard PSO + PRM	31.27	41.35	34.36	43.26	40.23	47.53	43.71	47.54	37.39
Standard PRM	40.86	49.77	44.16	54.43	49.17	56.33	53.71	56.75	46.98

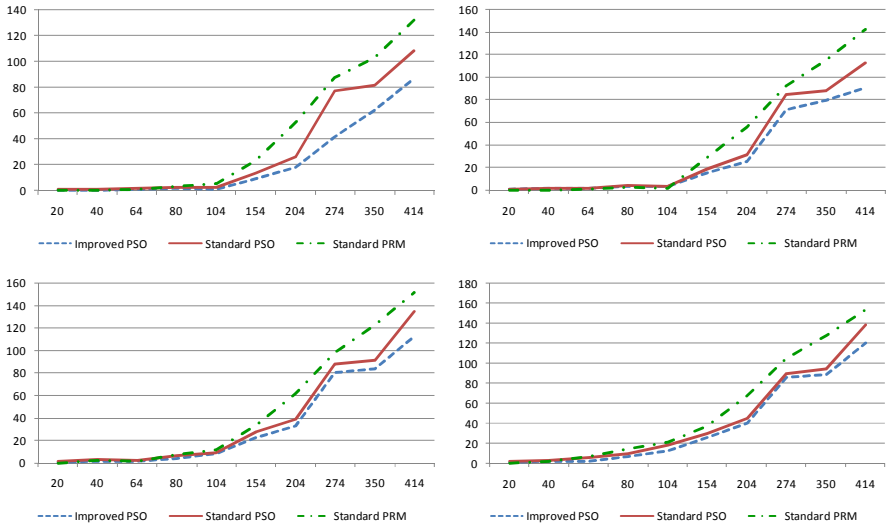

Fig. 3 Average runtime (s) vs. number of obstacle vertices for 2-, 3-, 4-, and 5-robot problems

The results of 600 solutions show that the proposed Improved PSO+PRM compound method was averagely about 17% and 34% faster than the Standard PSO+PRM and PRM methods, respectively, with considerably smaller standard deviation. Furthermore, the PSO+PRM method satisfied a bi-objective fitness function while in the PRM such a possibility is absent. Also, it is observed that runtime differences in the three methods increase as the number of vertices grows, showing the success of the new method in this type of problems.

6 Conclusions

In this paper, a new Improved PSO-based heuristic method is presented for multi-robot motion planning, which satisfies shortest and smoothest path objectives. The algorithm consists of a global planner (PSO) as well as a local planner (PRM). The multi robot motion planning problem is solved by this algorithm through decentralized planning with a global coordination model. For each robot, the proposed algorithm is run separately and then their motion coordination is performed all together and online. Also, five action modes were defined to describe the accurate situation of each robot at a given time. As a result, each robot moves one step toward its goal in each iteration.

The algorithm provides a novel and unique method to combine and coordinate the PSO and PRM algorithms by incorporating four groups of nodes within a single population. These nodes include: the best particles of the PSO, random nodes generated by PRM, current and next positions of the robot, and a pair of particles around each obstacle vertex. The set of these nodes form a network by being connected through straight edges. The shortest path between two consecutive robot

positions is then searched using a graph searching algorithm like the Dijkstra's method. As a result the free spaces around the obstacles can be searched in much less time than in the classic PRM algorithm.

After running and simulating 600 problem instances, the results showed that the proposed algorithm runs about 17% and 34% faster than the standard PSO+PRM and PRM methods, respectively, while two objectives are also optimized.

Considering the possibility of extending the PSO algorithm to high-dimensional spaces, we believe that the proposed method can be used for motion planning in high dimensional spaces provided that a proper distance metric is used.

References

[1] Lozano-Perez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM 22, 560–570 (1979)

[2] Warren, C.W.: Multiple robot path coordination using artificial potential fields. In: Proc. IEEE Int. Conf. Robotics and Automation, pp. 500–505 (1990)

[3] Canny, J.F.: The Complexity of Robot Motion Planning. The MIT press, Cambridge (1988)

[4] Hwang, Y.K., Ahuja, N.: Gross motion planning – A survey. ACM Computing Surveys 24(3), 219–291 (1992)

[5] Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, London (1991)

[6] Chun, L., Zheng, Z., Chang, W.: A decentralized approach to the conflict-free motion planning for multiple mobile robots. In: Proc. IEEE Int. Conf. Rob. Autom., vol. 2, pp. 1544–1549 (1999)

[7] Fujimura, K.: Motion Planning in Dynamic Environments. Springer, New York (1991)

[8] Arai, T., Ota, J.: Motion planning of multiple robots. In: Proc. IEEE Int. Conf. on Intelligent and Robotic Systems, pp. 1761–1768 (1992)

[9] Sanchez-Ante, G., Ramos, F., Frausto, J.: Cooperative Simulated Annealing for Path Planning in Multi-Robot Systems. In: Cairó, O., Cantú, F.J. (eds.) MICAI 2000. LNCS, vol. 1793, pp. 148–157. Springer, Heidelberg (2000)

[10] Sheng, G., Jie, Z., Hegao, C.: Genetic algorithm based path planning of coordinated multi-robot manipulators. In: Proc. IEEE Int. Conf. on Rob. Intell. Sys. & Signal Proc., pp. 763–767 (2003)

[11] Liu, S., Mao, L., Yu, J.: Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems. In: Proc. IEEE Int. Conf. on Mech. and Autom., pp. 1733–1738 (2006)

[12] Doctor, S., Venayagamoorthy, G.K., Gudise, V.G.: Optimal PSO for collective robotic search applications. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1390–1395 (2004)

[13] Min, H.Q., Zhu, J.H., Zheng, X.J.: Obstacle avoidance with multi-objective optimization by PSO in dynamic environment. In: Proc. IEEE Int. Conf. on Mach. Learning and Cyber., pp. 2950–2956 (2005)

[14] Berman, S., Halasz, A., Kumar, V., Pratt, S.: Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In: Proc. IEEE Int. Conf. Rob. and Autom., pp. 2318–2323 (2007)

[15] Rigatos, G.G.: Distributed gradient and particle swarm optimization for multi-robot motion planning. Robotica 26(3), 357–370 (2008)
[16] Parhi, D.R., Pothal, J.K., Singh, M.K.: Navigation of multiple mobile robots using swarm intelligence". In: World Congress on Nature and Biologically Inspired Computing, pp. 1145–1149 (2009)
[17] Kim, S.H., Lee, G., Hong, I., Kim, Y.J., Kim, D.: New potential functions for multi robot path planning: SWARM or SPREAD. In: Proc. IEEE/ICCAE, vol. 2, pp. 557–561 (2010)
[18] Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with particle swarm optimization. In: Proc. IEEE Swarm Intelligence Symp., pp. 332–339 (2007)
[19] Akat, S.B., Gazi, V., Marques, L.: Asynchronous particle swarm optimization-based search with a multi-robot system: simulation and implementation on a real robotic system. Turkish Journal of Electrical Engineering & Computer Science 18(5), 749–764 (2010)
[20] Hereford, J.M.: A distributed particle swarm optimization algorithm for swarm robotic applications. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1678–1685 (2006)
[21] Hassan, R., Cohanim, B., de Weck, O.: A comparison of particle swarm optimization and the genetic algorithm. In: Proc. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference (2004)
[22] Matsui, T., Kato, K., Sakawa, M., Uno, T., Matsumoto, K.: Particle swarm optimization for nonlinear integer programming problems. In: Proc. International Multi-Conference of Engineers and Computer Scientists, pp. 1874–1877 (2008)
[23] Sedighizadeh, D., Masehian, E.: Particle swarm optimization methods, taxonomy and applications. International Journal of Computer Theory and Engineering 1(5), 482–499 (2009)
[24] Masehian, E., Sedighizadeh, D.: Multi-objective robot motion planning using a particle swarm optimization model. Journal of Zhejiang University–Science C 11(8), 607–619 (2010)
[25] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Int. Conf. on Neural Networks, pp. 1942–1948 (1995)
[26] Shi, Y., Eberhart, R.C.: Particle swarm optimization with fuzzy adaptive inertia weight. In: Proc. Workshop on Particle Swarm Optimization, Indianapolis, pp. 101–106 (2001)
[27] Fujimura, K.: Path planning with multiple objectives. Journal of IEEE Robotics and Automation Society 3(1), 33–38 (1996)
[28] Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic Roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
[29] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston (2005)
[30] Sanchez, G., Latombe, J.C.: Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: Proc. IEEE Int. Conf. on Rob. and Automation, pp. 2112–2119 (2002)