Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Multi-objective robot motion planning using a particle swarm optimization model

Ellips MASEHIAN, Davoud SEDIGHIZADEH

(*Faculty of Engineering, Tarbiat Modares University, Tehran 14115-143, Iran*)

E-mail: {masehian, sedighizadeh}@modares.ac.ir

Received Aug. 23, 2009; Revision accepted Jan. 7, 2010; Crosschecked June 9, 2010; Published online July 23, 2010

Abstract: Two new heuristic models are developed for motion planning of point robots in known environments. The first model is a combination of an improved particle swarm optimization (PSO) algorithm used as a global planner and the probabilistic roadmap (PRM) method acting as a local obstacle avoidance planner. For the PSO component, new improvements are proposed in initial particle generation, the weighting mechanism, and position- and velocity-updating processes. Moreover, two objective functions which aim to minimize the path length and oscillations, govern the robot's movements towards its goal. The PSO and PRM components are further intertwined by incorporating the best PSO particles into the randomly generated PRM. The second model combines a genetic algorithm component with the PRM method. In this model, new specific selection, mutation, and crossover operators are designed to evolve the population of discrete particles located in continuous space. Thorough comparisons of the developed models with each other, and against the standard PRM method, show the advantages of the PSO method.

Key words: Robot motion planning, Particle swarm optimization, Probabilistic roadmap, Genetic algorithm
doi:10.1631/jzus.C0910525 **Document code:** A **CLC number:** TP242

1 Introduction

The robot motion planning (RMP) problem, in its most elementary form, is to find a collision-free start-to-goal path for a robot moving amid obstacles. Since the mid-1970's, and particularly after the important contribution of Lozano-Pérez and Wesley (1979), this problem has attracted the attention of many researchers, and many methods have been actively proposed for solving it (Choset *et al*., 2005).

The general motion planning problem was shown to be PSPACE-hard using an example of a many-jointed object constrained to move within a complex system of narrow channels (Hwang and Ahuja, 1992). The exponential space requirement stems from the vast number of different configurations with which these planners have to cope.

It has also been shown that the path planning problem is NP-complete (Canny, 1988). This means that a path planner's running time is exponential in degrees of freedom. The exponential time and storage requirements together make the RMP a challenging research problem.

Early approaches for solving the problem were generally based on computational geometry, and were limited to deterministic, low-dimensional problems. These classic methods are variations of a few general techniques: (1) roadmaps, including the visibility graph (Asano *et al.*, 1985), Voronoi diagrams (Canny, 1985), and silhouette (Canny, 1987); (2) cell decomposition (Keil and Sack, 1985); (3) potential fields (Khatib, 1986); (4) mathematical programming, including operations research models, game theory, and spline functions (Latombe, 1991).

These classic methods are not mutually exclusive, and combinations of them are often used for developing hybrid or compound motion planners.

For example, Masehian and Amin-Naseri (2004) combined the three methods of visibility graph, Voronoi diagrams, and potential fields into a single path planner. Also, Bhattacharya and Gavrilova (2008) proposed a Voronoi-based method for finding minimum-clearance paths, induced by the visibility graph. Kim *et al.* (2003) presented a version of Dijkstra's shortest path algorithm which satisfies kinematic or dynamic constraints while maintaining collision avoidance and minimum clearance from obstacles.

As a result of the NP-hardness of the RMP problem, classic methods proved to be inefficient for high-dimensional configuration space, requiring considerably long time and huge storage memory. Consequently, heuristic methods have been developed increasingly over the past two decades to cope with the high computational costs and complexities of classic methods, especially for high degrees of freedom. When using heuristic algorithms, a solution is not guaranteed, but if a solution is found, it will be achieved much faster than with deterministic methods.

The main metaheuristic approaches employed in RMP are simulated annealing (SA), genetic algorithms (GAs), particle swarm optimization (PSO), ant colony optimization (ACO), and tabu search (TS). These metaheuristics have also appeared in combination with a classic method; e.g., an SA and PF (potential field) compound technique has been applied by Janabi-Sharifi and Vinke (1993).

A chronological review of the applications of classic and heuristic algorithms in RMP is presented by Masehian and Sedighizadeh (2007). Some heuristic and metaheuristic methods used or referred to later in this paper are discussed in Section 2.

2 Related works

Inspired by natural phenomena, both GA and PSO algorithms are evolutionary, population-based metaheuristic algorithms, and have proved to be very efficient and powerful in a wide range of optimization problems.

The first application of GA in RMP dates back to Davidor (1990), where the environment was assumed to be static and known. GA requires that a suitable chromosome representation of the path be defined. Also, appropriate mechanisms for path guidance, obstacle avoidance, and constrained minimization of path distance should be designed for planning smooth paths. In the study by Solano and Jones (1993), the working area was represented by a discretized bitmap image with obstacles being colored to aid obstacle avoidance. Shibata and Fukuda (1993) combined GA with the fuzzy critic concept for use in intelligent motion planning.

The implementation of GA in motion planning has continued actively over the past decade. Parallel GA for searching and constrained multi-objective optimization was proposed by Wilson *et al.* (2004). Path planning for mobile robots based on a hybrid GA was developed by Li *et al.* (2006). A real-time obstacle avoidance strategy for mobile robots based on improved coordinated potential fields with GA was proposed by Cen *et al.* (2007), and a path planning method for mobile robots based on chaos GA was proposed by Gao *et al.* (2008).

Recently, optimal trajectories for mobile robots were found by implementing a genetic-based fuzzy logic controller (Rekik *et al.*, 2009), and mobile robot path planning using GA was performed by Ghorbani *et al.* (2009). Similarly, efficient and safe path planning for a mobile robot using GA was proposed by Naderan-Tahan and Manzuri-Shalmani (2009). Chang and Liu (2009) used a multi-objective island parallel GA with migration for collision-free curvature-derivative continuous path planning of autonomous unicycle wheeled mobile robots.

The PSO algorithm was first introduced by Kennedy and Eberhart (1995). It was based on the idea of swarms in nature such as those of birds and fish. PSO has particles driven from natural swarms, with communications based on evolutionary computations. PSO combines the particles' self experiences with their social experiences. In this algorithm, a candidate solution is presented as a particle. The algorithm uses a collection of flying particles (changing solutions) in a search space (current and possible solutions) and moves towards a promising area to reach a global optimum.

A taxonomy of the PSO algorithm has been recently presented by Sedighizadeh and Masehian (2009), in which the elements of the PSO-based algorithm are categorized into four main aspects, variables, particles, swarm, and process, and 22 subclasses.

The PSO method has found applications in RMP quite recently, and on a much more limited scale compared with GA. An algorithm for path planning of mobile robots using PSO with a mutation operator was developed by Qin *et al.* (2004). Min *et al.* (2005) presented a multi-objective optimization with a PSO approach for obstacle avoidance in dynamic environments. Also, Saska *et al.* (2006) developed a robot path planning method using a PSO of Ferguson splines. Li *et al.* (2006) proposed an obstacle avoidance path planning method for soccer robots using PSO, and smooth path planning of a mobile robot using stochastic PSO was implemented by Chen and Li (2006).

Li and Shi (2008) generated a graph named 'MAKLINK' in the workspace and then searched using Dijkstra's algorithm. The generated robot's path was then optimized by PSO.

Raja and Pugazhenthi (2009) developed mobile robot navigation paths in dynamic environments using PSO, in which obstacles can be convex or concave, and the velocity is variable. Gong *et al.* (2009) developed a method based on PSO for robot path planning in uncertain environments, where the information about some of the obstacles in the workspace was assumed to be uncertain.

Among other successful heuristics for RMP are the probabilistic algorithms, including the probabilistic roadmap (PRM) method developed by Kavraki *et al.* (1996), and rapidly-exploring random trees (RRT) proposed by LaValle (1998). Both of these methods work by forming random graphs (in PRM) or trees (in RRT) to be searched for start-to-goal paths. The nodes of these graphs are created by sampling random configurations, and their edges are generated by simple local planners (e.g., connecting via straight lines). These sampling-based methods have proved to be very efficient in searching and path planning in high-dimensional configuration spaces.

3 Overview of the new method

In most of the above methods, the RMP 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 in most cases their practicality is questionable, since mobile robots lose considerable energy and time when changing their course of motion abruptly. Therefore, in this paper we have designed a planning algorithm to apply two objectives simultaneously, i.e., the shortest and the smoothest criteria. This is done through an aggregative weighting multi-objective approach incorporated in the context of an improved PSO (IPSO).

The reason for implementing the PSO method in our planner is that PSO is versatile enough to accommodate multiple objectives, and is more efficient and faster than GA (Hassan *et al.*, 2004). However, given the success of PRM and its speed, especially in high dimensional spaces, PRM was considered suitable for obstacle avoidance in our algorithm. Thus, we have selected PSO as the global planner of our algorithm, while PRM is used as a local planner.

In addition to proposing a new and improved variant of PSO, the combination and interaction of the PSO and PRM methods is unique in the field of 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 PSO and random nodes generated in PRM complement each other and unify these methods.

In this paper, RMP is designed 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:

Step 1: A preset number of particles are generated around the robot's initial position and within its sensing range.

Step 2: Each particle takes a new velocity and position based on the constantly updated IPSO 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).

Step 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.

Step 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 its shortest path.

Step 5: Execute Steps 2–4 until the goal is within the robot's sensing range and can be accessed via a straight line.

4 Improved particle swarm optimization: the general planner

In this work, an improved variant of the basic PSO method is proposed to accommodate more control parameters and considerations. This method is then used as the global motion planner of our model; i.e., 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, and then the improved variant and its implementation in the new algorithm are described.

In the basic PSO proposed by Kennedy and Eberhart (1995), the particles showing the solution candidates start their fly from random positions in a search area. In each iteration, the particles update their positions and velocities according to Eqs. (1) and (2) and move to another position. Flying is affected by a fitness function that assesses the quality of each solution.

$$p_{\mathrm{p}j}^{\,i} = p_{\mathrm{p}j}^{\,i-1} + v_{\mathrm{p}j}^{\,i}, \tag{1}$$

$$v_{\mathrm{p}j}^{\,i} = \chi\left(wv_{\mathrm{p}j}^{\,i-1} + c_1 r_1 (p_{\mathrm{b}j}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1}) + c_2 r_2 (g_{\mathrm{b}}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1})\right), \tag{2}$$

where $p_{\mathrm{p}j}^{\,i}$ is the position of the jth particle in the ith iteration, $v_{\mathrm{p}j}^{\,i}$ is the velocity of the jth particle in the ith iteration, $p_{\mathrm{b}j}^{\,i-1}$ is the best position of the jth particle so far, $g_{\mathrm{b}}^{\,i-1}$ is the best position in the swarm in the ith iteration so far, $g_{\mathrm{b}}^{\,i}$ is the best position in the swarm in the ith iteration, and $\chi = 2\Big/\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|$, $\varphi > 4$.

PSO has some dependent parameters: c_1 is a constant called the 'cognitive acceleration coefficient', and c_2 is another constant named the '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 also been produced with $c_1=c_2=4$ (Shi and Eberhart, 2001). r_1 and r_2 are random numbers between 0 and 1, different at each iteration. χ is the constriction factor, which limits the velocity. w is a weight that regulates the global search behavior, set to an upper bound $w_{\max}$ at the beginning of the searching process and dynamically reduced during optimization to a lower bound $w_{\min}$ (which emulates a more local search behavior). Its range is suggested to be $0.2 \le w \le 0.4$.

The first term of Eq. (2), $wv_{\mathrm{p}j}^{\,i-1}$, 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 $c_1 r_1 (p_{\mathrm{b}j}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1})$, 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., p_{b}). The third term $c_2 r_2 (g_{\mathrm{b}}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1})$, 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.

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

$$\begin{aligned} v_{\mathrm{p}j}^{\,i} = \chi\Big(& w_1 v_{\mathrm{p}j}^{\,i-1} + w_2 c_1 r_1 (p_{\mathrm{b}j}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1}) \\ & + w_3 c_2 r_2 \alpha_1 (g_{\mathrm{b}}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1}) \\ & + w_4 c_3 r_3 \alpha_2 (p_{\text{b-ran}}^{\,i-1} - p_{\mathrm{p}j}^{\,i-1}) + w_5 c_4 r_4 \alpha_3 v_{\text{p-ran}}\Big), \end{aligned} \tag{3}$$

where $p_{\text{b-ran}}^{i}$ is the best position of a randomly selected particle in the ith iteration, $v_{\text{p-ran}}$ is a random velocity vector between $V_{\min}$ and $V_{\max}$, w_1 is the inertia weight, w_i (i=2, 3, 4, 5) are control weights, c_i (i=1, 2, 3, 4) are acceleration constants, r_i (i=1, 2, 3, 4) are random numbers, different at each iteration and between 0 and 1, and α_i (i=1, 2, 3) are convergence factors.

In this variant, the fourth term of Eq. (3), $w_4c_3r_3\alpha_2(p_{\text{b-ran}}^{i-1}-p_{\text{p}j}^{i-1})$, which we call the 'random self-cognition part', sends the particles towards one of the best positions found randomly by particles ($p_{\text{b-ran}}^{i-1}$). 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 $w_5c_4r_4\alpha_3v_{\text{p-ran}}$, which is enforced through the random velocity parameter $v_{\text{p-ran}}$, increases the variety in the swarm and leads to a better and more effective movement of the swarm in narrow and completed search areas (Mishra, 2006).

All but the first term of Eq. (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.

The developed PSO variant also has a novel dynamic and sophisticated mechanism for setting the value of the inertia weight w_1, which controls the algorithm's global search behavior. Eqs. (4)–(6) give the details of this mechanism:

$$w_1 = w\cdot f(\text{iter}^i), \tag{4}$$

$$w = w_{\max}-\left(\frac{w_{\max}-w_{\min}}{\text{iter}_{\max}}\right)\cdot\text{iter}^i, \tag{5}$$

$$f(\text{iter}^i)=\mu\cdot f(\text{iter}^{i-1})\left(1-f(\text{iter}^{i-1})\right), \tag{6}$$

where iter^i denotes the ith iteration, $\text{iter}_{\max}$ the maximum number of iterations (explained later in Table 1), and μ the control parameter for f, set to μ=0.4. For the first iteration we set $f(\text{iter}^1)$=0.63, which is found empirically after trial and error.

This mechanism was inspired by the velocity updating procedures presented by Caponetto *et al.* (2003) and Park *et al.* (2005), which are modified and combined here for the first time. The constriction parameter in Eq. (7) is also introduced as a novel contribution:

$$\varphi = 2\Big/\left|2-5(c_2+c_3)-(c_2+c_3)^2\right|. \tag{7}$$

The proposed weighting mechanism works in such a way that not only does the swarm converge rapidly towards a goal, but also the generated path is near optimal.

The overall procedure of the IPSO method is shown in Fig. 1. The algorithm has a main nested loop which is 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 the best fitness values for each particle ($p_{\text{b}j}^{i-1}$, or pbest) and for the whole swarm (g_{b}^{i-1}, or gbest) up to that iteration are calculated. Particles' positions and velocities are then updated based on Eqs. (1) and (3). Note that this procedure differs from that of the basic PSO in the way in which the velocities and inertia weights are updated.

```
While iter_max is not reached or minimum error criterion is not met 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)
        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);
    Select randomly a velocity for the particle (v_p-ran);
    Calculate particle velocity v_pj^i according to Eq. (3);
    Apply the velocity constriction;
    Update the particle position p_pj^i according to Eq. (1);
    Apply the position constriction;
  End
End
```

Fig. 1 Pseudocode of the improved particle swarm optimization (IPSO)

4.1 Generating an initial population of particles

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 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 in that direction, a point near the obstacle's border is selected as the particle in 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. 2 shows the creation of 36 particles around a robot's starting point. The larger is the number of divisions on the circle, the larger would be the number of particles, and therefore the higher would be the planning accuracy.

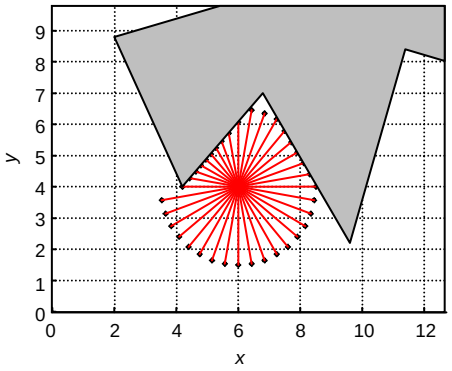

Fig. 2 The generation of the initial population of particles based on the borders of the robot's sensed area
Robot's starting point is (6, 4). The robot emanates 36 radial rays to measure its distance from the obstacle, which is shown as a gray polygon. The black dots at the end of rays represent particles

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 approach, while maintaining the centralization of the robot's start point, the obstacles' distribution around it is also considered.

4.2 Multi-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 only if it meets several conditions, such as safety, estimated time needed for navigation, and energy consumption.

A path which is considered as optimal in terms of a single criterion may not satisfy all other criteria (Fujimura, 1996). For instance, a shortest path is not required at the expense of safety along the path.

Some studies have been published in the field of multi-objective robot motion planning, including an approach for obstacle avoidance with multi-objective optimization by PSO in dynamic environments (Min *et al.*, 2005), and multi-objective optimal trajectory planning of a space robot using PSO (Liu *et al.*, 2008).

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 to travel from start to goal, whereas the length is the distance actually traveled by the robot along the path.

For robots needing to reach their destinations as soon as possible, a minimum-time path might seem desirable, but it may require a lot of time to be traversed because of uneasy terrain. There must 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 that is optimal for a single criterion without considering other equally important criteria is not desirable (Fujimura, 1996). 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

$$\text{Total cost}=w_1z_1+w_2z_2+\ldots,$$

where z_i is the ith cost and w_i is its weight. By selecting proper weights, a path with the desirable property can be obtained by planning with a single objective (Kang *et al.*, 2001).

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., $g_b^{\,i}$ and $g_b^{\,i-1}$ (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

$$F^i_{\text{short}\,j}=\sqrt{(x_{p^i_{pj}}-x_{\text{goal}})^2+(y_{p^i_{pj}}-y_{\text{goal}})^2}.$$

The second objective function, the smoothest path, is mathematically expressed as

$$F^{i}_{\text{smooth } j} = \arccos\Big((x_{p_{\text{p}j}^{\,i}} - x_{\text{goal}})(x_{g_{\text{b}}^{\,i-1}} - x_{\text{goal}}) + (y_{p_{\text{p}j}^{\,i}} - y_{\text{goal}})(y_{g_{\text{b}}^{\,i-1}} - y_{\text{goal}}) \Big) \cdot \Big(\sqrt{(x_{p_{\text{p}j}^{\,i}} - x_{\text{goal}})^2 + (y_{p_{\text{p}j}^{\,i}} - y_{\text{goal}})^2} \cdot \sqrt{(x_{g_{\text{b}}^{\,i-1}} - x_{\text{goal}})^2 + (y_{g_{\text{b}}^{\,i-1}} - y_{\text{goal}})^2} \Big)^{-1}.$$

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

$$\text{Fitness}^{i}_{j} = \lambda_1 F^{i}_{\text{short } j} + \lambda_2 F^{i}_{\text{smooth } j}. \tag{8}$$

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 trial and error, with best found values λ_1=1 and λ_2=0.25.

4.3 Parameter tuning

For tuning the parameters used in the proposed algorithm, the aim is to specify acceptable and practical limits for each parameter (Table 1) after numerous runs and simulations.

After incorporating the developed fitness function in the basic PSO algorithm (Fig. 1), and applying the properly tuned parameters, the positions of the generated particles are updated using Eqs. (1) and (3). When a particle is generated or updated in each iteration, the algorithm determines whether it lies inside an obstacle. If it is inside an obstacle, it will be omitted from the swarm. Furthermore, after calculating the $g_{\text{b}}{}^{i}$ in iteration i, it is checked if the line connecting the $g_{\text{b}}{}^{i}$ and $g_{\text{b}}{}^{i-1}$ is entirely in free space. This connecting line is actually a segment of the final path which the robot travels in each iteration, and thus its freeness is vital. If the line is not free (i.e., it intersects an obstacle), the local planner component, which is based on the RPM method, is invoked to find a path between the two successive g_{b}'s by detouring the obstructing obstacles.

5 Probabilistic roadmap components: the local planner

Owing to its ease of implementation and ability to plan in high dimensional configuration spaces, PRM has attracted considerable attention in recent motion planning studies. Initial PRMs succeeded in solving a number of complex problems with high-dimensional configuration spaces which had previously not been solved efficiently (Kavraki *et al*., 1996). PRM was enhanced later in 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 (Choset *et al*., 2005).

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. 3). 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: (1) a number of randomly generated nodes, (2) the robot's current position, (3) the best particles generated in PSO, and (4) two points around each corner of the obstructing obstacle.

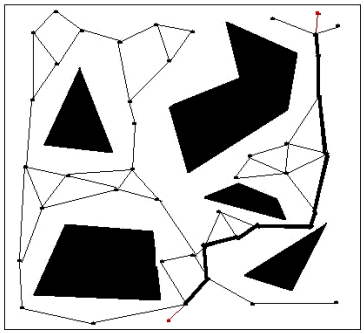

Fig. 3 A probabilistic roadmap (PRM) consisting of random nodes connected by straight edges

Roadmap edges are generated by local planners such that they lie entirely in the free space. The thick line shows the shortest path on the roadmap from robot's start to goal

Table 1 Guidelines for tuning the algorithm's parameters

Parameter	Function/Description	Suggested range	Conditions for increase (▲) and decrease (▼)
c_1, acceleration constant	Attracts the particle's position towards its pbest	[1.5, 4]	▲High velocity and acceleration of particle movements ▼Low velocity and acceleration of particle movements
c_2, acceleration constant	Attracts the particle's position towards a random particle's pbest	[1.5, 4]	As above
c_3, acceleration constant	Attracts the particle's position towards the gbest	[1.5, 4]	As above
c_4, acceleration constant	Controls the randomness of the particle's velocity (v_p)	[1.5, 4]	As above
w_1, inertia weight	Maintains the particle's current velocity	[0.4, 0.9]	▲Global exploration (search in breadth) is emphasized ▼Local exploitation (search in depth) is emphasized
w_2, w_3, w_4, w_5, control weights	Control the effects of their multiplicands in the overall new v_p	[0.4, 0.9]	▲Local exploitation (search in depth) is emphasized ▼Global exploration (search in breadth) is emphasized
α_1, gbest factor	Controls the influence of the gbest	[0, 10]	▲Greedy convergence for solving non-complicated objective functions ▼Cautious convergence when solving complicated objective functions
α_2, random pbest factor	Controls the influence of the pbest of a randomly selected particle	[0, 20]	▲Cautious convergence is required when solving complicated objective functions ▼Greedy convergence for solving non-complicated objective functions
α_3, random v_p factor	Controls the influence of the randomly selected particle velocity	[0, 1]	▲Moving to farther distances from the primary position of particles (i.e., risky motions) is encouraged ▼Moving to closer distances to the primary position of particles (i.e., cautious motions) is encouraged
λ_1, weighting factor	Controls the weight of the shortest path in the fitness function	[0.1, 2]	▲More emphasis on the shortest path ▼More emphasis on the smoothest path
λ_2, weighting factor	Controls the weight of the smoothest path in the fitness function	[0.1, 2]	▲More emphasis on the smoothest path ▼More emphasis on the shortest path
R_{goal}, radius of the goal	Controls the scope of particles' visibility for the goal	[0, 4]	▲Greedy convergence with less accuracy is emphasized ▼Cautious convergence with more accuracy is emphasized
n, population size	Total number of particles	[10, 10 000]	▲More accuracy in reaching the goal in more time ▼Less accuracy in reaching the goal in less time
(x_{max}, y_{max}), maximum position	The maximum position on which a particle can be located	–	As above
(x_{min}, y_{min}), minimum position	The minimum position on which a particle can be located	–	As above
$iter_{max}$, maximum number of iterations	Maximum number of iterations in the procedure	[10, 10 000]	▲More accuracy in optimizing the selected function spending more time ▼Less accuracy in optimizing the selected function but with higher speed

The above combination of nodes is new and involves a subtle intertwining of the PSO and PRM methods. In addition to the randomly generated nodes (group 1) which are typical of the PRM method, about 30%–40% of the PSO particles with the highest fitness values (p_b's) are also integrated in the PRM graph. Group 4 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 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 g_b (calculated based on the best position among particles) is then found using Dijkstra's search algorithm. As a result, the robot can move from g_b^{i-1} to g_b^{i} and get closer to the goal, while

avoiding the obstacles that locally intercept its path to the goal. Once the robot is located in its new position, the PSO particles' velocities and positions are updated again, as described earlier.

6 Experimental results

To analyze the function of the proposed new algorithm, numerous simulations were run in which the algorithm's parameters were tuned to their best values. A few graphical results from running the algorithm on problems with simple to complex obstacles are shown in Fig. 4.

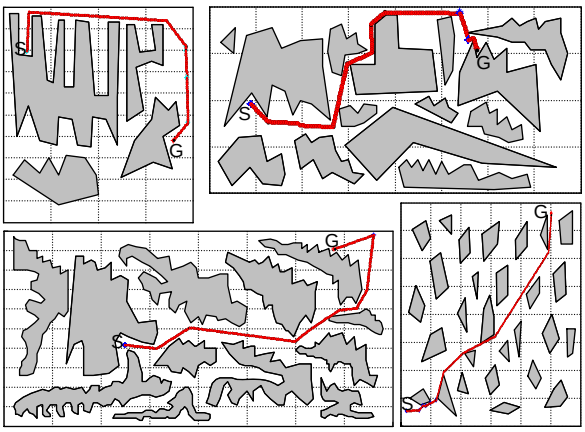

Fig. 4 Some simulations of the developed method
The calculated path connects the robot's start point (S) to its goal point (G)

For comparing the algorithm's performance with other efficient and well-known algorithms, the standard PRM and GA methods were selected. Thirty-five test problems with different obstacle sizes and shapes (both convex and concave) were designed, with the total number of obstacle vertices ranging from 17 to 414 (Fig. 5).

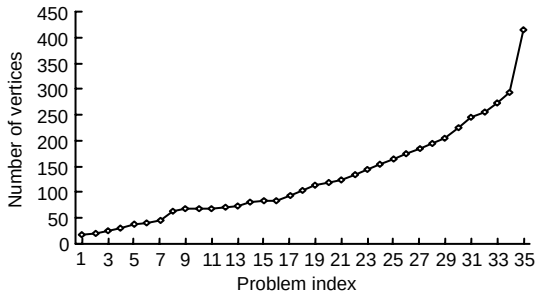

Fig. 5 Number of vertices for the 35 test problems with different obstacle sizes and shapes

The test problems were solved by all three methods: (1) the multi-objective IPSO+PRM; (2) the classic PRM; (3) GA+PRM. All methods were coded in Matlab and run on an Intel 3.0 GHz processor.

6.1 Competing methods

The classic PRM against which we tested our algorithm was coded according to the explanations in Section 4. Each test problem, after constructing the probabilistic roadmap, was searched using Dijkstra's method to yield a start-to-goal shortest path on the roadmap.

However, the GA+PRM algorithm was devised to evaluate the specific effect of the IPSO+PRM compared to the performance of the GA component in the same settings and conditions. Thus, to perform a fair comparison between the GA and PSO components, all parts of the motion planning algorithm need to be maintained, except for the PSO updating equations, which are replaced by GA operators for updating the population.

In this version of GA, developed specifically for this study, instead of describing the population of chromosomes as discrete strings of genes, the population is defined as a collection of particles. Also, the GA operators are modified to meet the continuous space conditions.

First, a random population (e.g., with a size of N) is generated in the obstacle-free space. Each member of this population is in fact a position in the workspace. Then, using GA operators, the initial population is evolved stage by stage.

The operators are designed as follows:

1. Selection operator: A part of the population (sized N_1) is randomly selected and passed to the next generation.

2. Mutation operator: A small part of the population (sized N_2) is replaced by randomly generated new particles of the same size, which are passed to the next generation.

3. Crossover operator: The remaining part of the population (sized $N_3=N-N_1-N_2$) is randomly divided into two equal portions (note that N_1 and N_2 should be set such that N_3 is even). Afterwards, random pairs of 'parents' are formed by selecting two particles from each portion, and their arithmetic averages are calculated. This will generate $N_3/2$ 'children', which are in fact new positions in the space, and

which shall be passed to the next generation. Another $N_3/2$ particles will be selected from the parents in N_3 having better fitness function values (i.e., closer to the goal) and will be passed to the next generation. This way, the population size N remains constant.

At each stage, the best individual of the population is considered as the robot's destination in that stage, which is then connected to the robot's position in the previous position via a straight line. If the line intersects an obstacle, the PRM module is evoked, which acts as described previously and connects the two positions via intermediate probabilistic nodes using Dijkstra's search. The population evolution continues till the goal is reached.

6.2 Comparison results

The results of solving the 35 test problems by the three methods are shown in Table 2 and Figs. 6 and 7, where they are compared in terms of path length and runtime.

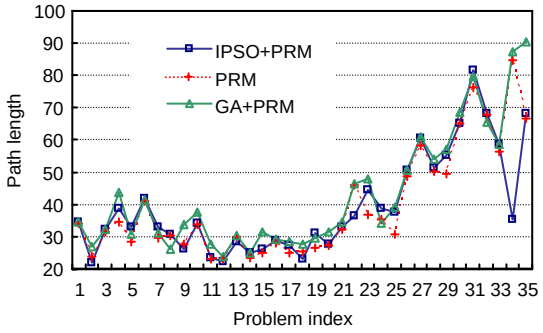

Fig. 6 Path lengths of the three competing methods
IPSO: improved particle swarm optimization; PRM: probabilistic roadmap; GA: genetic algorithm

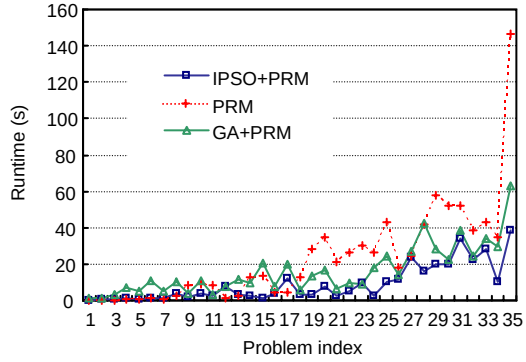

Fig. 7 Runtime of the three competing methods
IPSO: improved particle swarm optimization; PRM: probabilistic roadmap; GA: genetic algorithm

Table 2 Comparison results for the three competing methods

Vertice No.	Runtime (s)			Path length		
	IPSO +PRM	PRM	GA +PRM	IPSO +PRM	PRM	GA +PRM
17	0.32	0.04	1.11	34.54	34.19	34.60
20	0.36	0.20	0.98	21.88	23.61	26.71
26	0.38	0.25	3.27	32.00	31.58	32.27
31	1.17	0.38	7.11	38.75	34.48	43.78
37	0.85	0.62	5.44	33.07	28.44	30.51
40	1.56	1.57	11.19	41.95	40.86	41.20
46	1.56	0.44	5.34	32.86	29.53	31.02
64	3.79	2.28	10.12	30.78	30.39	26.01
67	1.70	8.28	3.73	26.18	27.44	33.70
68	3.70	9.22	10.92	33.99	33.60	37.58
69	2.38	8.31	3.34	23.38	22.82	27.59
72	7.64	1.16	7.47	22.27	23.17	23.57
74	3.26	2.39	11.61	28.41	29.97	30.42
80	2.26	13.12	9.38	25.03	23.56	24.91
83	1.25	13.69	20.26	26.18	24.85	31.57
84	4.16	4.63	8.03	28.74	28.13	29.00
94	12.05	4.53	19.64	27.19	24.72	28.51
104	3.50	12.91	6.08	22.97	25.34	27.55
114	3.43	28.44	13.32	31.18	26.51	29.39
118	7.63	34.42	16.59	27.76	27.19	31.56
124	2.41	21.34	6.66	32.82	32.11	34.42
134	5.14	26.49	9.89	36.27	45.80	46.50
144	9.68	30.08	8.88	44.43	36.97	47.84
154	2.87	26.12	18.14	38.74	35.09	33.97
164	10.48	42.78	24.61	37.54	30.72	39.24
174	11.74	17.68	14.05	50.45	48.82	50.59
184	23.76	24.55	27.06	60.60	58.40	60.77
194	15.78	41.81	42.45	51.51	50.08	54.07
204	19.77	58.04	28.34	55.03	49.35	57.01
224	19.90	51.79	22.58	65.18	64.97	68.73
244	34.05	52.28	38.67	81.53	76.08	79.60
255	22.44	38.82	24.51	68.09	67.93	65.47
274	28.27	43.15	34.13	58.52	56.23	58.59
294	10.06	34.97	29.44	35.08	84.74	87.39
414	38.78	146.43	63.04	68.34	66.49	90.51
Mean	9.08	22.95	16.18	39.23	38.26	42.75
SD	10.18	27.88	13.53	15.41	16.74	18.17

IPSO: improved particle swarm optimization; PRM: probabilistic roadmap; GA: genetic algorithm. SD: standard deviation

The results showed that our new method exceeded the GA+PRM method in path length by 9% (with a smaller standard deviation), while the path lengths produced by the IPSO and standard PRM methods were not significantly different. However,

our IPSO+PRM method proved to be about 60% and 44% faster than the standard PRM and GA+PRM methods, respectively. Also, the lower standard deviations of the PSO method showed its relatively great robustness over the other two methods.

These figures suggest that the runtime of the new method is significantly shorter than those of the other two methods.

To confirm this and to test if the results taken from these sample runs can be generalized to the whole populations of their respective algorithms, a right-sided mean difference *t*-test was conducted. We considered this test appropriate and valid as the number of samples exceeded 30.

The null and alternative hypotheses and the *t*-statistic were defined as follows:

$$\mathrm{H0}:\ \mu_{T_i}=\mu_{T_j},\quad \mathrm{H1}:\ \mu_{T_i}<\mu_{T_j},$$

$$t=\left(\overline{T}_i-\overline{T}_j\right)\Big/\left(\overline{S}\sqrt{\frac{1}{n_i}+\frac{1}{n_j}}\right),$$

$$\overline{S}=\sqrt{\left((n_i-1)S_{T_i}^2+(n_j-1)S_{T_j}^2\right)\Big/(n_i+n_j-2)},$$

where indices i and j refer to two of the three competing algorithms, n_i=n_j=35 denote their sample sizes, μ_{T_i} and $\overline{T}_i$ are the mean runtime of the whole population and the sample, respectively, and $S_{T_i}^2$ is the variance of the runtime of the sample. The degrees of freedom of the *t*-test were v=n_i+n_j−2=68.

The following three possible mean difference tests were conducted, and the results are summarized in Table 3: (1) IPSO+PRM vs. the standard PRM; (2) IPSO+PRM vs. GA+PRM; (3) GA+PRM vs. the standard PRM.

Table 3 *t*-test results for mean runtime of the three methods

H1: $\mu_{T_i}<\mu_{T_j}$	t	Maximum confidence level (%)
$\mu_{\mathrm{IPSO+PRM}}<\mu_{\mathrm{GA+PRM}}$	2.4884	99.24
$\mu_{\mathrm{IPSO+PRM}}<\mu_{\mathrm{PRM}}$	2.7631	99.63
$\mu_{\mathrm{GA+PRM}}<\mu_{\mathrm{PRM}}$	1.2863	89.87

In Table 3 each row represents a *t*-test for the mean difference of two competing algorithms. The *t*-statistic was calculated in the second column and the maximum confidence level for which the alternative hypothesis holds was computed in the third column. The runtime efficiency of the new improved PSO method was very significantly superior to those of the standard PRM and GA+PRM methods.

The newly developed GA+PRM method (for continuous spaces) also outperformed the standard PRM method with a confidence level of around 90%. These results showed that decomposing the searching process into global and local path planning modules is very effective in terms of both path length and especially runtime.

7 Conclusions and future research

In this paper a new heuristic particle swarm optimization (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 probabilistic roadmap (PRM) method component used as the local planner.

The algorithm provides a unique and novel method for combining and unifying 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. In this way, the free space around obstacles is efficiently searched in much less time than with the classic PRM. Experiments and comparisons showed that the new algorithm is considerably (about 60%) faster than the classic PRM method, while being competitive in terms of path length.

Another contribution of this paper is the development of a new GA-based path planning model which incorporates PRM as a local planner. The selection, mutation, and crossover operators of this algorithm are tailored in such a way that discrete particles located in continuous spaces are manipulated as genetic populations. Although the GA-based method has a shorter runtime than the standard PRM

method, the PSO-based method still outperforms the GA by about 44%. All comparisons were supported by results from *t*-tests with a sample size of 35 test problems.

As a direction for future research, another objective criterion could be incorporated in the algorithm, aiming to enhance the path safety (i.e., its clearance from obstacles). For this purpose, the Voronoi diagram can be effectively used. Also, this method can be generalized for motion planning of multiple robots, when they are considered to be disc-shaped or polygonal.

References

Asano, T., Asano, T., Guibas, L., Hershberger, J., Imai, H., 1985. Visibility-Polygon Search and Euclidean Shortest Path. Proc. 26th Symp. on Foundations of Computer Science, p.155-164.

Bhattacharya, P., Gavrilova, M., 2008. Path planning with the required minimum clearance using the Voronoi diagram methodology. *IEEE Rob. Autom. Mag.*, **15**(2):58-66. [doi:10.1109/MRA.2008.921540]

Canny, J.F., 1985. A Voronoi Method for the Piano-Movers Problem. Proc. IEEE Int. Conf. on Robotics and Automation, **2**:530-535.

Canny, J.F., 1987. A New Algebraic Method for Robot Motion Planning and Real Geometry. Proc. 28th IEEE Annual Symp. on Foundations of Computer Science, p.39-48.

Canny, J.F., 1988. The Complexity of Robot Motion Planning. MIT Press, Cambridge, MA, USA.

Caponetto, R., Fortuna, L., Fazzino, S., Xibilia, M.G., 2003. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Trans. Evol. Comput.*, **7**(3):289-304. [doi:10.1109/TEVC.2003.810069]

Cen, Y., Wang, L., Zhang, H., 2007. Real-Time Obstacle Avoidance Strategy for Mobile Robot Based on Improved Coordinating Potential Field with Genetic Algorithm. IEEE Int. Conf. on Control Applications, p.415-419. [doi:10.1109/CCA.2007.4389266]

Chang, H.C., Liu, J.S., 2009. High-Quality Path Planning for Autonomous Mobile Robots with η_3-Splines and Parallel Genetic Algorithms. IEEE Int. Conf. on Robotics and Biomimetics, p.1671-1677. [doi:10.1109/ROBIO.2009.4913252]

Chen, X., Li, Y., 2006. Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization. Proc. IEEE Int. Conf. on Mechatronics and Automation, p.1722-1727.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S., 2005. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston.

Davidor, Y., 1990. Robot Programming with a Genetic Algorithm. Proc. IEEE Int. Conf. on Computer Systems and Software Engineering, p.186-191.

Fujimura, K., 1996. Path planning with multiple objectives. *IEEE Rob. Autom. Mag.*, **3**(1):33-38. [doi:10.1109/100.486659]

Gao, M., Xu, J., Tian, J., Wu, H., 2008. Path Planning for Mobile Robot Based on Chaos Genetic Algorithm. Proc. Int. Conf. on Natural Computation, p.409-413. [doi:10.1109/ICNC.2008.627]

Ghorbani, A., Shiry, S., Nodehi, A., 2009. Using Genetic Algorithm for a Mobile Robot Path Planning. Proc. Int. Conf. on Future Computer and Communication, p.164-166. [doi:10.1109/ICFCC.2009.28]

Gong, D., Lu, L., Li, M., 2009. Robot Path Planning in Uncertain Environments Based on Particle Swarm Optimization. Proc. IEEE Congress on Evolutionary Computation, p.2127-2134. [doi:10.1109/CEC.2009.4983204]

Hassan, R., Cohanim, B., de Weck, O., 2004. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. Proc. 46th Structures, Structural Dynamics and Materials Conf., p.1-13.

Hwang, Y.K., Ahuja, N., 1992. Gross motion planning—a survey. *ACM Comput. Surv.*, **24**(3):219-291. [doi:10.1145/136035.136037]

Janabi-Sharifi, F., Vinke, D., 1993. Integration of the Artificial Potential Field Approach with Simulated Annealing for Robot Path Planning. Proc. IEEE Int. Symp. on Intelligent Control, p.536-541.

Kang, D.O., Kim, S.H., Lee, H., Bien, Z., 2001. Multiobjective navigation of a guide mobile robot for the visually impaired based on intention inference of obstacles. *Auton. Rob.*, **10**(2):213-230. [doi:10.1023/A:1008990105090]

Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Rob. Autom.*, **12**(4):566-580. [doi:10.1109/70.508439]

Keil, J.M., Sack, J.R., 1985. Minimum Decomposition of Polygonal Objects. *In*: Toussaint, G.T. (Ed.), Computational Geometry. North-Holland, Amsterdam, p.197-216.

Kennedy, J., Eberhart, R.C., 1995. Particle Swarm Optimization. Proc. Int. Conf. on Neural Networks, p.1942-1948. [doi:10.1109/ICNN.1995.488968]

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, **5**(1):90-99. [doi:10.1177/027836498600500106]

Kim, J., Pearce, R.A., Amato, N.M., 2003. Extracting Optimal Paths from Roadmaps for Motion Planning. Proc. IEEE Int. Conf. on Robotics and Automation, p.2424-2429.

Latombe, J.C., 1991. Robot Motion Planning. Kluwer Academic Publishers, Boston.

LaValle, S.M., 1998. Rapidly-Exploring Random Trees: a New Tool for Path Planning. Technical Report, TR 98-11, Computer Science Department, Iowa State University, USA.

Li, G., Shi, H., 2008. Path Planning for Mobile Robot Based on Particle Swarm Optimization. Proc. Control and Decision Conf., p.3290-3294. [doi:10.1109/CCDC.2008.4597938]

Li, Q., Tong, X., Xie, S., Zhang, Y., 2006. Optimum Path Planning for Mobile Robots Based on a Hybrid Genetic Algorithm. Proc. 6th Int. Conf. on Hybrid Intelligent Systems, p.53-58.

Liu, G., Yuan, J.P., Xu, Y.S., 2008. Multi-Objective Optimal Trajectory Planning of Space Robot Using Particle Swarm Optimization. Proc. Int. Symp. on Neural Networks, p.171-179.

Lozano-Pérez, T., Wesley, M.A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, **22**(10):560-570. [doi:10.1145/359156.359164]

Masehian, E., Amin-Naseri, M.R., 2004. A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *J. Rob. Syst.*, **21**(6):275-300. [doi:10.1002/rob.20014]

Masehian, E., Sedighizadeh, D., 2007. Classic and Heuristic Approaches in Robot Motion Planning: a Chronological Review. Proc. World Academy of Science, Engineering and Technology, p.101-106.

Min, H.Q., Zhu, J.H., Zheng, X.J., 2005. Obstacle Avoidance with Multiobjective Optimization by PSO in Dynamic Environment. Proc. Int. Conf. on Machine Learning and Cybernetics, p.2950-2956. [doi:10.1109/ICMLC.2005.1527447]

Mishra, S.K., 2006. Repulsive Particle Swarm Method on Some Difficult Test Problems of Global Optimization. Available from http://mpra.ub.uni-muenchen.de/1742/ [Accessed on July 16, 2009].

Naderan-Tahan, M., Manzuri-Shalmani, M.T., 2009. Efficient and Safe Path Planning for a Mobile Robot Using Genetic Algorithm. Proc. IEEE Congress on Evolutionary Computation, p.2091-2097. [doi:10.1109/CEC.2009.4983199]

Park, J.B., Lee, K.S., Shin, J.R., Lee, K.Y., 2005. A particle swarm optimization for economic dispatch with no smooth cost functions particle swarm optimization for economic dispatch with any smooth cost functions. *IEEE Trans. Power Syst.*, **20**(1):34-42. [doi:10.1109/TPWRS.2004.831275]

Qin, Y.Q., Sun, D.B., Li, N., Cen, Y.G., 2004. Path Planning for Mobile Robot Using the Particle Swarm Optimization with Mutation Operator. Proc. Int. Conf. on Machine Learning and Cybernetics, p.2473-2478.

Raja, P., Pugazhenthi, S., 2009. Path Planning for Mobile Robots in Dynamic Environments Using Particle Swarm Optimization. Proc. Int. Conf. on Advances in Recent Technologies in Communication and Computing, p.401-405. [doi:10.1109/ARTCom.2009.24]

Rekik, C., Jallouli, M., Derbel, N., 2009. Optimal Trajectory of a Mobile Robot by a Genetic Design Fuzzy Logic Controller. Proc. Int. Conf. on Advances in Computational Tools for Engineering and Applications, p.107-111. [doi:10.1109/ACTEA.2009.5227926]

Saska, M., Macas, M., Preucil, L., Lhotska, L., 2006. Robot Path Planning Using Particle Swarm Optimization of Ferguson Splines. Proc. IEEE Conf. on Emerging Technologies and Factory Automation, p.833-839. [doi:10.1109/ETFA.2006.355416]

Sedighizadeh, D., Masehian, E., 2009. A New Taxonomy for Particle Swarm Optimization (PSO). Proc. 10th Int. Conf. on Automation Technology, p.317-322.

Shi, Y., Eberhart, R., 2001. Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight. Proc. Workshop on Particle Swarm Optimization, p.101-106.

Shibata, T., Fukuda, T., 1993. Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic. Proc. IEEE Int. Conf. on Intelligent Control, p.565-570.

Solano, J., Jones, D.I., 1993. Generation of Collision-Free Paths: a Genetic Approach. Proc. IEEE Colloquium on Genetic Algorithms for Control Systems Engineering, p.5/1-5/6.

Wilson, L.A., Moore, M.D., Picarazzi, J.P., Miquel, S.D.S., 2004. Parallel Genetic Algorithm for Search and Constrained Multi-Objective Optimization. Proc. 18th Int. Parallel and Distributed Processing Symp., p.165. [doi:10.1109/IPDPS.2004.1303161]