An Optimal Two-Stage Path Planner

Xuyue Deng, Jianqiang Yi, Dongbin Zhao

Key Laboratory of Complex Systems and Intelligence Science
Institute of Automation
Chinese Academy of Sciences

{xuyue.deng,jianqiang.yi,dongbin.zhao}@ia.ac.cn

Abstract

Probabilistic roadmap (PRM) planner applied to the robot with multiple degrees of freedom moving in static environment with static obstacles is proved to be efficient. But it is clumsy to environment changes, and the achieved path is usually not optimal and not smooth enough. In order to make the planner optimal and more flexible, this paper integrates the PRM planner with simulated annealing (SA) to constitute a two-stage path planner: at first a path is generated by PRM planner, then this original path is optimized by SA. If the original path becomes infeasible because of environment changes, a feasible path can also be obtained from the original path by SA. The simulations are carried out to verify the efficiency of the proposed planner.

Keywords: Path Planning; Probabilistic Roadmap; Simulated Annealing

I. Introduction

Robot path planning is a basic and long standing problem of robotics, which is required to seek for a collision-free path that brings the robot from its initial configuration to the specified goal configuration. In practice, a robot may have many degrees of freedom (DOF), which is a great challenge for robot path planning. Many deterministic path planners especially the complete deterministic planners applied for robot with many DOF are short of efficiency if not infeasible at all, for the limit computation resource. Therefore the researchers turn to the probabilistic complete path planners. Two of such successful planners are the randomized path planner (RPP) and probabilistic path planner (PPP) [21]. The former is actually a potential field planner, which escapes from local minima by performing random walks [2]. The latter finds the required path according to a probabilistic roadmap that is constructed by randomly selecting configurations from the free configuration space (*C*-space) and interconnecting certain pairs by simple local powerful planners. The data structures of the roadmap can be a graph [10] or a tree [15]. Two popular PPPs are PRM planner [10] and RRT planner [13,15]. The theoretic analysis of PRM is present in [11,14], and it is proved that the PRM planner is probabilistic complete.

This work was partly supported by NSFC Projects (No. 60334020, 60440420130, 60475030, and 60575047), MOST Projects (No. 2003CB517106 and 2004DFB02100), and the Outstanding Overseas Chinese Scholars Fund of Chinese Academy of Sciences (No. 2005-1-11), China.

As a PPP, PRM planner is applied successfully from robot path planning to protein folding [7,10,20,21]. Two phases compose general PRM algorithm [10], named roadmap construction phase and query phase. In the roadmap construction phase, a graph *G(V, E)* is constructed which represents the feasible paths in the free *C*-space, and it is used as the data base to find a feasible path from the starting configuration to the goal configuration of the robot. *V* is the node set, and *E* denotes the edge set. In the set of *V*, each node is an admissible configuration of the robot. The nodes can be selected from the whole free *C*-space randomly or by a deterministic technique. In the set of *E*, every edge denotes a feasible path between the two corresponding end nodes. The feasible path is generated by a simple local path planner, which tries connecting one configuration node to its neighbors or some candidates selected by some criterions. In the query phase, firstly the starting and the goal configuration nodes are attached to *G(V, E)* at q_s and q_g, respectively. Then some graph search algorithm is invoked to try searching feasible path between q_s and q_g.

Since then, a lot of PRM variations are presented and the tricks for node sampling, collision checking, local path planning and so on are exploited. A comparative study of PRM planners is given in [7] in order to help future users to choose proper techniques. Here some not all typical variations are briefly reviewed. Amato et al [1] proposed an obstacle-base PRM method (OBPRM) in 3D workspace. In OBPRM planner, the sample nodes are near or on the surface of obstacle in *C*-space. Bekris et al [3] integrated PRM and RRT, and used the bidirectional-growing RRT (BI-RRT) as the local path planner for PRM. In fact, the BI-RRT has its own local planner too. Branicky et al [6] proposed a quasi-random PRM method (Q-PRM), and Sanchez et al [19] applied Q-PRM to nonholonomic robot path planning. In Q-PRM, a deterministic sample technique instead of a random sample technique is used to choose the configuration node from the free *C*-space. Usually the common PRM algorithm constructs a graph containing only feasible paths, whereas lazy-PRM algorithm [5] doesn't. The lazy-PRM planner generates a roadmap while doesn't check whether its edges are collision-free or not and leaves this work to the query phase.

Although the advantages of PRM for high dimensional planning problems are exploited and verified by many researchers [7], one pitfall must be made clear that this type of approach is short of efficiency when it is applied in a changing environment. The probabilistic roadmap is constructed according to static workspace before any feasible path is generated and usually it is a tree or a low-connectivity graph and there is only one or a few paths between any pair nodes. If there are some newly-emerged obstacles in the robot workspace, maybe they block probabilistic roadmap and no feasible path can be achieved by the corrupted roadmap. To handle this situation, one option is to construct a high-connectivity graph roadmap, i.e., find feasible paths between any pair nodes as much as possible. The tradeoff is that collision-check will be performed many more times, and it will be heavy time-consumed. It also contravenes the original intention of PRM.

Another pitfall of PRM is that the generated path is always curling and not optimal. The character of the resulted roadmap, i.e., low-connectivity, usually yields curling path, and the nonoptimal path should be ascribed to that there are no global optimization for obtained paths when the local planner finds them. Some smoothing techniques or optimization methods can be applied to improve the paths. One smoothening technique is repeatedly picking a pair of random nodes in the resulting path and trying to connect each other by the local path planner [21]. If successful, the path will be updated and shortened. Isto [8] introduced a polygonal optimizer [4] to optimize the resulting path. But all these techniques affect the path locally and global optimization goal is not achieved yet.

This paper applies the simulated annealing algorithm to optimize the path generated by PRM. If the original path is feasible, the SA optimizer will produce an optimal one according to the specified global criterions. If the original path is infeasible, the SA optimizer with a higher initial

"temperature" will still produce a feasible and optimal path based on the original one. In the optimization, the smoothness of the path is taken into account, and the path will be smoothed simultaneously.

The remainder parts of this paper are arranged as following: The 2nd section describes the related works, which are general PRM planner and SA algorithm. The 3rd section describes the presented SA optimizer in details, including how the new state (path) is generated, and how the state (path) is evaluated. Some simulations results in 2D workspace are reported in the 4th section and some conclusions are drawn in the 5th section.

II. Related Works

A. PRM and its variations

In this paper, the general PRM algorithm [10] is applied and every configuration node is sampled uniformly in the free *C*-space. See Algorithm 1 for the details of the general PRM planner's roadmap construction phase.

Algorithm 1

Probabilistic Roadmap Construction

1. Initialization: $V \leftarrow 0$, $E \leftarrow 0$
2. Repeat the following loop until determine criteria is satisfied
3. Select randomly a new node q from the free configuration space
4. Get the neighbors of q, which is $V_q \subset V$
5. $V \leftarrow V \cup \{q\}$
6. For all $q' \in V_q$, if q' and q are not in the same connected component
7. If local path planner can find a path between q' and q
8. $E \leftarrow E \cup e(q, q')$
9. Repeat the following loop for *N* times
10. Get the nodes with low-connectivity, which is $V_b \subset V$
11. For all $q^* \in V_b$
12. Select randomly a new node q from the free neighbor space of q^*
13. Get the neighbors of q, which is $V_q' \subset V$
14. $V \leftarrow V \cup \{q\}$
15. For all $q'' \in V_q$, if q'' and q are not in the same connected component
16. If local path planner can find a path between q'' and q
17. $E \leftarrow E \cup e(q, q')$

The neighbors of one node are determined by the distance between the neighbor candidate nodes and the node. If the distance is shorter than some threshold D_N, which is usually the maximal radii of the local path planner's detecting range, the candidate can be regarded as its neighbor. The distance between nodes q_1 and q_2 is defined as

$$d(q_1, q_2) = \|q_1 - q_2\|_W = (q_1 - q_2)^T W (q_1 - q_2) \quad (1)$$

Here W is positive diagonal constant matrix with proper dimension.

In some difficult areas, e.g., the narrow passages of the robot workspace, the roadmap maybe disconnects. Therefore *Step 9→17* are performed in order to increase the connectivity of the graph *G(V,E)*. (The connectivity of each node is defined to be proportional to its degree, i.e., the number of edges connected with it. The low-connectivity node is the node whose connectivity is lower than some threshold, e.g. 40% of the average connectivity.)

Step 6 & 15 forbid the new edges between the nodes in the same connected component. At the end of the preprocess phase, a graph *G(V,E)* without containing any cycle (in fact it is a tree if the workspace has no separate room) is constructed to denote the connectivity of the free *C*-space. This nice property of *G(V,E)* makes it easy to query a path from the roadmap.

In the query phase, ordinary graph search algorithm is implemented to find a path between the starting configuration and the goal configuration, or declare there is no such a path between them.

B. Simulated Annealing

Simulated annealing, which is introduced by Kirkpatrick et al [12], is a popular probabilistic technique for combinational optimization problems. Let L denotes some kind of state variable. It may be a scalar, a vector with constant or varying dimension, or a combination of some scalars and some vectors. The optimization goal is to seek the global minimal of a scalar function $E(L)$. Therefore the optimization implemented by SA can be stated as the following Algorithm 2.

Algorithm 2

Basic Simulated Annealing

1. Initialization: T , L_0, $E_0=E(L_0)$
2. Repeat the following loop until determine criteria is satisfied
3. Generate a new state L_n, get its energy $E_n=E(L_n)$
4. If $E_n<E_0$
5. Accept the new state L_n, and assign $L_0 \leftarrow L_n$, $E_0 \leftarrow E_n$
6. Else
7. Generate a random number *rand*
8. If $rand < e^{-(E_n-E_0)/T}$, accept the new state as the former Step 5
9. Decrease the temperature T

In fact, *Step 3→8* can be performed more times in one loop, or more new states can be generated in *Step 3*.

Of course the techniques of new state generation and energy evaluation are diverse with various applications. As for this paper, the details of them are described in the 3rd section.

As [18] said, SA is well suited to tackling such "problems best described as 'dirty', i.e., problems with either numerous contradictory constraints, or complex baroque cost functions". Obviously the robot path planning problem is one of them. In the literature, SA is always integrated with the artificial potential field approach to avoid the local minima problem of the latter [9,16,17].

The contribution of this paper is attempting to integrate SA with PRM planner. In our opinion, SA is suitable to optimize the original path obtained by PRM. In fact the *probabilistic* roadmap is inherently compatible with this *probabilistic* optimization technique. It is convenient to implement random operations on the probabilistic roadmap in the optimization process. The original path, no matter whether it is feasible or not, is a proper initial state of SA. The initial temperature of SA is determined by the energy of the original path (see Section 3 for path evaluation). Infeasible path will have high energy and the high initial temperature will permit great modification on the original path, while feasible path and then low initial temperature will prohibit the great modification.

III. SA Optimizer

The framework of SA applied here is the same as Algorithm 1. Here $L = \{q_1 \quad q_1 \quad \cdots \quad q_n\}$, $q_k \in V$ denotes the robot path. Let L_0 denotes the original path that is achieved in the query phase of PRM planner, or the old path in the iterations of the optimization algorithm. And L_n denotes the new path. At different temperature a unique path is generated from the old one and evaluated to get its energy and then determined to be accepted or rejected. The details of new path generation and path evaluation in SA Optimizers are described as following.

A. New Path Generation

Three kinds of operations constitute the new path generation.

The first is named "Path_Shortening": A pair of nodes—q_l and q_m—in the old path L_0 are selected randomly, the distance between them should be shorter than some threshold D_L. Then an approximal "line"(it is a line in hyperspace) constituted by the nodes in the node set V is made to connect the two nodes q_l, q_m. Of course not all the nodes are in the exact line. The corresponding motion in practice to the line connection between two nodes is that every entity of q_l increases (or decreases) linearly to the corresponding entity of q_m. If the line section is feasible, the path is shortened. The threshold D_L is assigned deliberately to prohibit many line connections, which are probably infeasible because of too long distance between the two nodes.

The second is named "Path_Neighbor": Every node q except for the starting and the goal node is examined; meanwhile a random number $p \in [0,1]$ is uniformly generated. If $p<Q(q)$, q is substituted by one node which is randomly selected from its neighbor nodes. Here $Q(q)$ is a value associated with the quality of q (in the following part of this paper $Q(q)$ is called quality function, in fact it is inverse-quality function more than quality function), i.e., the node q has larger $Q(q)$, if q stays closer to the obstacles, or the path turns sharper in the position denoted by q. Q(q) should be defined properly and can vary within [0,r]. Here a number r with $1.5 \geq r \geq 1.0$ may be adequate. Therefore a node with $Q(q) \geq 1.0$ will be definitely substituted by its neighbor. In fact $Q(q)$ can be obtained as the byproduct of the path evaluation. Note that one or more nodes should be added between the sequential pair nodes, because sometime they are too far to reach each other for the local path planner.

The third is named "Path_Repairing": For a blocked path section $\overline{q_k q_{k+1}}$, choose a node $q_{k+1/2}$, which stays from q_k and q_{k+1} as far as possible, but still lies in the detecting range of local path planner when it works at q_k and q_{k+1}. Then the path section $\overline{q_k q_{k+1}}$ in L_0 becomes

$\overline{q_k q_{k+1/2} q_{k+1}}$ in L_n. This operation will be performed for all the blocked path sections in the whole old path.

*B. **Path Evaluation***

When the path is evaluated, three kinds of factors are taken into consideration: the length, the smoothness and the clearance of the path, as Xiao et al did in [22]. In this paper, the energy of the path L is expressed as

$$E(L) = \alpha \cdot Length(L) + \beta \cdot Smooth(L) + \eta \cdot Clear(L) + \Gamma \cdot (e^{\frac{Length(L')}{Length(L)}} - 1) \quad (2)$$

All the paths, feasible paths or infeasible paths, are evaluated by the same measurement. If the path is feasible, the fourth term, penalty term, in right side of (2) will be zero (for feasible path, $Length(L') = 0$, else it will not. In (2), the coefficient constants α, β and η are the weights of three corresponding terms in the energy function, and Γ is the penalty constant for infeasible path. All the constants should be empirically assigned according to path requirements.

The function of every term in right side of (2) is computed as:

- $Length(L) = \sum_{k=1}^{n-1} d_k$, which is the total length of the path. Here d_k is the distance between q_k and q_{k+1}, i.e., $d_k = \|q_{k+1} - q_k\|_W$.

- $Smooth(L) = \sum_{k=2}^{n-2} s_k$. Here $s_k = \left\| \frac{q_k - q_{k-1}}{d_{k-1}} - \frac{q_{k+1} - q_k}{d_k} \right\|_W$. s_k depicts how sharp the path turns at q_k: the sharper the path turns at q_k, larger s_k is.

- $Clear(L) = \max_{k=1}^{n-1} \begin{cases} D_1 - g_k & g_i > D_0 \\ e^{\gamma(D_0 - g_k)} & D_0 \geq g_i > 0 \end{cases}$. Here g_k is the smallest distance between $\overline{q_k q_{k+1}}$ and the obstacles around it. D_0 is the safe distance and D_1 is the upper bound of the distance between every $\overline{q_k q_{k+1}}$ and the obstacles around it.

- $Length(L') = \sum_{g_k = 0} d_k$, which is the total length of the blocked path sections.

The quality function $Q(q_k)$ can be obtained when the path is evaluated.

$$Q(q_k) = 0.5 \begin{cases} S_2 & s_k \geq S_2 \\ s_k & S_2 > s_k > S_1 \\ 0 & s_{k \leq S_1} \end{cases} + 0.5 \begin{cases} \frac{D_0 - g_k}{D_0} & g_k \leq D_0 \\ 0 & g_k > D_0 \end{cases}$$

Here S_1 is the threshold assigned to indicate how sharp the path turns can be accepted, and S_2 limits the greatest volume in $Q(q_k)$ introduced by path smoothness at q_k.

IV. Simulations

The simulations are carried out as following in order to validate the proposed planner. The robot moves in a plane and is regarded as a moving point, and the local planner attempts to connect its

neighbor nodes with a line path section. The simplified scenario makes it easy to test the feasibility of the proposed planner. Of course the planner can be implemented in higher dimension space.

Table 1. Parameters' Table

Name	N	α	β	η	Γ	γ	S_1	S_2	D_1	D_2	D_L	D_N
Value	20	1.0	1.0	10.0	500.0	10.0	0.6	1.6	0.1	5.0	5.0	0.2

The workspace is 10×10. W in (1) is a 2D identity matrix. The parameters of PRM and SA are in Table 1, they mainly appear in Algorithm 1 & 2 and (2). Finally the free C-space is denoted by a tree with 5000 nodes.

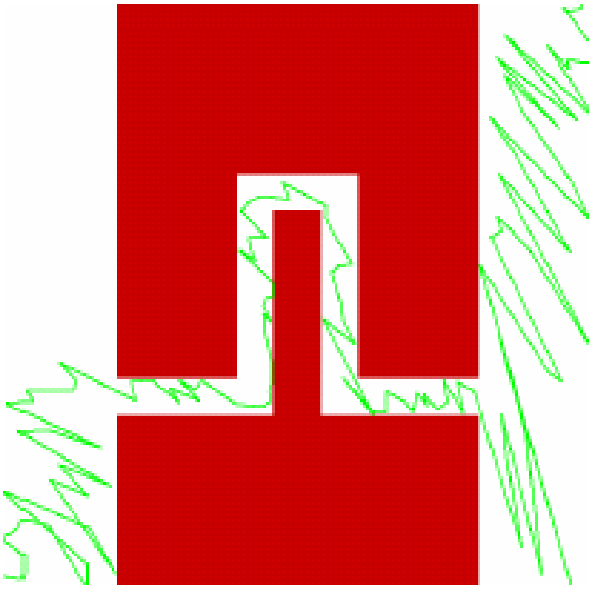

Fig.1. Original Path

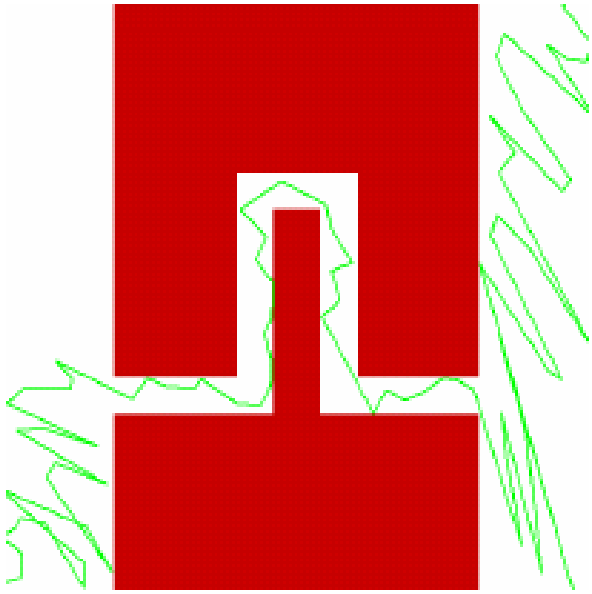

Fig.2. Smooth Path

Fig.1 shows the path generated by the general PRM planner without any path smoothing techniques applied. The path starts from the left-bottom corner and arrives at the right-top corner. Obviously it is curling and unnecessary long. With path smoothing techniques of [21], the path becomes shorter and smoother (see Fig.2). Besides, increasing the total number of the nodes of $G(V,E)$ will certainly shorten and smooth the path, and improve the quality of the path, no matter the aforementioned smoothing techniques are applied or not.

The path shown in Fig.1 is optimized by the SA optimizer and the achieved path is shown in Fig.3. In the iterations of SA, the energy of the path decreases as one expects (see Fig.4). From Fig.3, one can see that the path is more smooth and shorter than the path in Fig.2, and safer too. In another word, the path in Fig.3 is an optimal one.

If there are some additional obstacles that block the original path generated by the PRM planner, the SA optimizer gets a new feasible path according to the original path (see Fig.5).

It is sound that the final energy of Fig.6 is higher than that of Fig.4 because of the additional obstacles. Unfortunately the energy of the path doesn't decrease uniformly, which is mainly caused by the operation "Path_Neighbor" when new path is generated (see Section 3). The nodes of V are discrete in the free C-space and the energy of path neighbor produced by ``Path_Neighbor'' may vary relatively great. Therefore more node samples and then shorter distance of node neighbors will improve the quality of the optimized path too.

Fig.3. Optimal Path

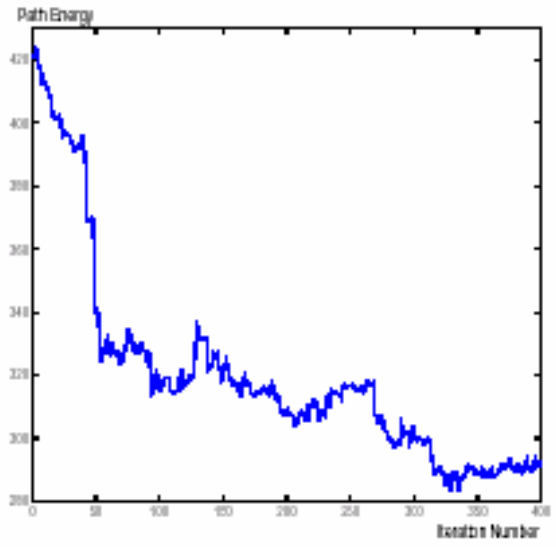

Fig.4. Path Energy When Getting Left Path

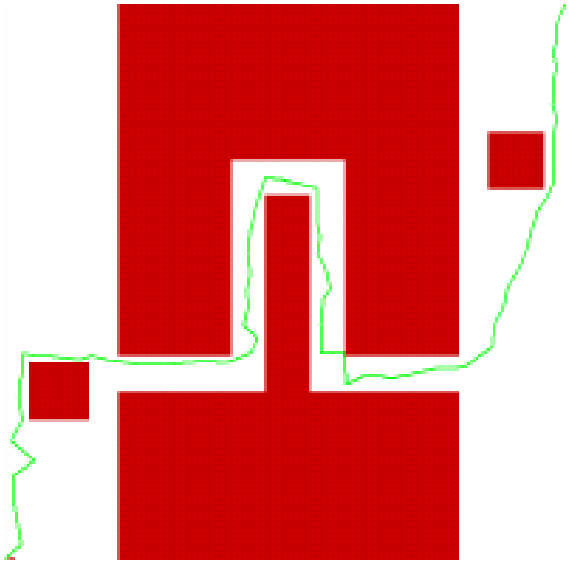

Fig.5. Path Changes According to Additional Obstacles

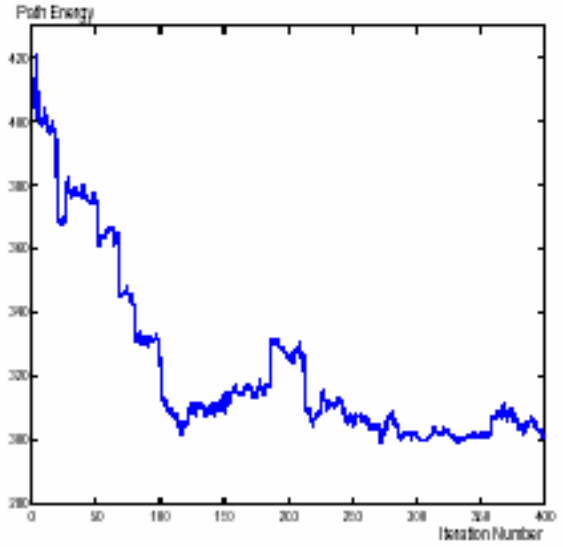

Fig.6. Path Energy When Getting Left Path

V. Conclusions

In this paper, the simulated annealing algorithm is applied to optimize the path obtained by PRM planner. At first, a probabilistic roadmap is constructed for the static environment, and then multiple queries can be made to generate paths between starting configurations and goal configurations (or declare there are no such paths). The paths generated by PRM planner are usually curling. SA optimizes these original paths and smoother and optimal paths can be obtained. If the environments have been changed and maybe the paths generated based on the original roadmap are blocked, SA optimizer can process the infeasible paths and new feasible paths can also be achieved based on the infeasible paths. Therefore, the PRM planner and the SA optimizer constitute a two-stage path planner, which is optimal and more flexible than the PRM planner. The simulations carried out according to simplified case validate the proposed planner's efficiency.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspace", Proceedings of the International Workshop on Algorithmic Foundation of Robotics, 1998, p155-168.

[2] J. Barraquand, B. Langlois, J.-C. Latombe, "Numerical potential field techniques for robot path planning", IEEE Transactions on System, Man and Cybernectics, Vol.22, No.2, 1992, p224-241.

[3] K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, L. E. Kavraki, "Multiple Query Probabilistic Roadmap Planning using Single Query Planning Primitives", Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003, p656-661.

[4] S. Berchtold, B. Glavina, "A scalable optimizer for automatically generated manipulator motions", Proceedings of IEEE/RSI/GI International Conference on Intelligent Robots and Systems, 1994, p1796-1802.

[5] R. Bohlin, L. E. Kavaraki, "Path Planning Using Lazy PRM", Proceedings of The 2000 IEEE International Conference on Robotics and Automation, Vol.1, 2000, p521-528.

[6] M. S. Branicky, S. M. LaValle, K. Olson, L. Yang, "Quasi-randomized path planning", Proceedings of IEEE International Conference on Robotics and Automation, 2001, p1481-1487.

[7] R.Geraerts, M. H. Overmars, ``A Comparative Study of Probabilistic Roadmap Planners'', in Algorithmic Foundation of Robotics V, Start 7, J.-D. Boissonnat et al. (Eds.) Springer-Verlag, Berlin Heidelberg, 2004, p43-57.

[8] P. Isto, "Constructing Probabilistic Roadmaps with Powerful Local Path Planning and Path Optimization", Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, p2323-2328.

[9] F. Janabi-Sharifi, D. Vinke, "Robot path planning by integrating the artificial potential field approach with simulated annealing", Proceedings of the International Conference on Systems, Man and Cybernetics, 1993, p282-287.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", IEEE Transactions On Robotics and Automation, vol.12, no.4, 1996, p566-580.

[11] L. E. Kavraki, M. N. Kolountzakis, J.-C. Latombe, "Analysis of Probabilistic Roadmaps for Path Planning", IEEE Transactions on Robotics and Automation, vol.14, no.1, 1998, p166-171.

[12] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing", Science, Vol.220, No.4598, 1983, p671-680.

[13] J. J. Kuffner, Jr., S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning", Proceedings of The 2000 IEEE International Conference on Robotics and Automation, Vol.2, 2000, p995-1001.

[14] A. M. Ladd, L. E. Kavraki, "Measure theoretic analysis of probabilistic path planning", IEEE Transactions on Robotics and Automation, Vol.20, No.2, 2004, p229-242.

[15] S. M. LaValle, "Rapid-exploring random trees: A new tool for path planning", TR 98-11, Computer Science Dept., Iowa State Univ., Oct. 1998.

[16] M. G. Park, J. H. Jeon, M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing", Proceedings of the 2001 IEEE International Symposium on Industrial Electronics, 2001, p1530-1535.

[17] S. Piao, B. Hong, "Path planning of robot using genetic annealing algorithm", Proceedings of the 4th World Congress on Intelligent Control and Automation, 2002, p493-495.

[18] R. A. Rutenbar, "Simulated annealing algorithms: an overview", IEEE Circuits and Devices Magazine, Vol.5, No.1, 1989, p19-26.
[19] A. Sanchez, J. A. Arenas, R. Zapata, "Nonholonomic Path Planning using a Quasi-Random PRM Approach", Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, p2305-2310.
[20] G. Song, N. M. Amato, "A Motion-Planning Approach to Folding: From Paper Craft to Protein Folding", IEEE Transactions on Robotics and Automation, vol.20, no.1, 2004, p60-71.
[21] P. Svestka, M. H. Overmars, "Probabilistic path planning", *Robot Motion Planning and Control*, Lectures Notes in Control and Information Sciences 229, J.-P. Laumond(eds), Springer, 1998, p255-304.
[22] J. Xiao, Z, Michalewicz, L. Zhang, K. Trojanowski, "Adaptive Evolutionary Planner/Navigator for Mobile Robots", IEEE Transactions on Evolutionary Computation, Vol.1, No.1, 1997, p18-28.

X. Y. Deng received his Bachelor degree in 2001 from Nanjing University, China, and Master degree in 2003 from Graduate School of Chinese Academy of Sciences. And currently is a Doctor Candidate in Graduate University of Chinese Academy of Sciences. His main research interests include robotics, nonlinear control, etc.

D. B. Zhao received his Ph.D degree in 2000 from Harbin Institute of Technology, China. He is now an associate professor in Institute of Automation, Chinese Academy of Sciences. His research interests include intelligent, robotics, and mechatronics, etc.