Proc. IASTED Robotics and Applications Conference, Santa Barbara, USA, October 1999.

LOCALLY-OPTIMAL PATH PLANNING BY USING PROBABILISTIC ROADMAPS AND SIMULATED ANNEALING

GILDARDO SÁNCHEZ, FERNANDO RAMOS, JUAN FRAUSTO
Computer Science Department
ITESM Campus Morelos
Av. Paseo de la Reforma 182-A
Col. Lomas de Cuernavaca, 62589
Cuernavaca, Morelos, MEXICO

{gisanche, framos, jfrausto}@campus.mor.itesm.mx

Abstract:
It is presented an algorithm for locally-optimal path planning which is based on Kavraki and Latombe's Probabilistic Roadmaps (PRM) and Simulated Annealing (SA). The PRM method is able to generate collision-free paths in a very short time (once that the roadmap has been constructed). However, when generating the paths, it is not taken into consideration any optimization criteria. It means that it would be interesting trying to reduce the cost of the paths. We propose in this work a method based on a simulated annealing to solve this optimization problem. SA is a method that has been successfully applied in complex optimization problems, and it seems straightforward to think of its application for optimizing a path provided by PRM. In other words, PRM are used to generate one valid solution, and SA is used to optimize it. The results show that the method can be applied for off-line planning (the SA took between 15 and 30 seconds in most of the cases), and we established a set of results that can be used in order to compare the performance of SA against other optimization algorithms.

Keywords: path planning, optimization, randomized algorithms, motion planning, manipulator robotics.

1. INTRODUCTION

Many efforts have been conducted in robotics research for solving the basic problem of motion planning, which consists on generating a collision-free path between an initial and a goal position for one robot in a static and completely known environment, where there could be obstacles [1]. There is strong evidence that a complete planner (i.e., one that finds a path whenever one exists and indicates that no one exists otherwise), will take time exponential in the number of degrees of freedom (dof) of the robot (i.e, that the algorithm belongs to a class of problem known as NP-Complete) [2].

The computational complexity of the problem suggests the application of randomized techniques, in which case there is a tradeoff between completeness and velocity of the algorithm. Generally speaking, the algorithms for motion planning can be broadly classified in any of two possibilities: *i*) **global** planners, and *ii*) **local** planners. In the first case it is required a complete model of the free space, and in the second case the knowledge is bounded to a certain neighborhood of the current position of the robot. In practice, the global approach is well suited to generate collision-free paths in the case of the basic problem. It is the case of Probabilistic Roadmaps (PRM), a randomized method proposed by Kavraki and Latombe which has been successfully applied for solving motion planning problems in static and completely known environments [3], [4].

The original PRMs find paths quickly (in the order of seconds), considering that the graph used to represent the free-space has already been constructed. The generated paths are only guaranteed to be free of collision. However, we think that it is possible to obtain better paths, considering that a better path is one than uses more efficiently the resources. For instance, by requiring less time to complete the execution of the path or by demanding less energy. In practice, finding better paths could imply reducing cycle times for task execution, or reducing operational costs of the robots.

This paper presents an algorithm for finding locally-optimal paths[1]. The main idea is to obtain any path between a pair of configurations, and optimizing it by using a well known non-deterministic optimization technique called

[1] The paths are named *locally-optimal* because of the method we used for improving the path. It works on a certain neigborhood of an initially proposed path, as a consequence, we can only guarantee finding an optimum in that restricted search space.

Simulated Annealing (SA) [5]. The structure of this paper is as follows: section two describes related works concerning optimal path planning, section three presents the statement of the problem to be tackled, section four describes the algorithm proposed, section five shows the results obtained and finally, section six presents conclusions and future work.

2. RELATED WORK

One of the optimal motion problems that have received more attention is the unconstrained shortest-path problem in two dimensions. The problem consists of computing, for a point robot, the shortest path between two given points, such that the path does not intersect the interior of a given set of polygons (considered as obstacles). The algorithms developed for solving this problem can be classified as visibility-graph methods and shortest-path-map methods [6].

The *visibility-graph* methods are based on the construction of a weighted graph and the application of any Dijkstra-type algorithm for search. The critical step is the construction of the visibility graph. The time required grow very fast with the number of dof of the robot and the degree of the algebraic equations used to describe the obstacles, making this approach impractical for robots with more than four degrees of freedom [1]. Any shortest path algorithm that depends on an explicit construction of the visibility graph will have a worst-case running time of $\Omega(n^2)$ [2].

The *shortest-path-map* method constructs a planar subdivision of the plane with respect to the start point *s*, so that all points in the same region of the subdivision have the same vertex sequence in their shortest paths from *s*. Many works have been focused on very specific instances of the problem, giving place to different algorithms, for instance, Mitchell and Papadimitriou [7] presented an algorithm that operates on planar subdivisions and which runs in $O(n^7L)$, where *n* is the number of edges in the subdivision and *L* is the precision of weights associated with the graph. The shortest-path problem in three dimensions is substantially more difficult that the 2D one; in fact, it was proved to be NP-hard by Canny and Reif [8]. Most of the optimal motion planning problems are intractable.

Other cases, in optimal motion planning include the kinodynamic motion planning, which studies the problem of computing collision-free, minimum-time trajectories for a robot whose motion is governed by Newtonian dynamics and whose accelerations and velocities are bounded [6]. Donald and Xavier studied the following problem: given a robot system, find a minimal-time trajectory from a start position and velocity to a goal position and velocity, while avoiding obstacles and respecting dynamic constraints on velocity and acceleration [9]. Recently, Steven LaValle *et al*. presented an approach for optimal motion planning for multiple robots. Their work is based on multi-objective optimization and game theory [10]. Besides, Hsu and Latombe presented an algorithm for optimizing the base location of a manipulator in an environment with obstacles. Our method resembles the one of Hsu in the sense that both methods try to optimize the execution of tasks. However, in our case, we optimize paths, while Hsu's method optimizes robot location [11].

3. PROBLEM DEFINITION

The problem consists of finding a path **p** that connects the initial and final configurations with the lowest cost. The cost associated to a path can be established considering parameters such as the time required, the energy demanded or the smoothness of the path, among the most important. In our particular case, we considered as a first attempt the optimization of the path in terms of the time required executing the path.

Assuming that the controller of the robot uses a *joint-interpolated motion strategy*[2], the time required to follow a path can be computed as follows:

1) Having two consecutive configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, obtain the maximum displacement between pairs of corresponding joints. For $\mathbf{q} = \{\theta_1, \ldots, \theta_n\}$, where *n* is the number of dof of the robot, find $\varphi = max_j |\theta_{i,j} - \theta_{i+1,j}|$, for all $j = 1,\ldots, n$, where $\theta_{i,j}$ denotes element *j* of configuration *i*. Considering the joint-interpolated motion strategy, we need the maximum displacement to compute the time required to accomplish one movement of the robot.

2) Considering a maximum velocity κ for the joints, obtain the time τ for the movement from configuration $\mathbf{q}_i$ to configuration $\mathbf{q}_{i+1}$.
$$\tau = \varphi / \kappa$$

3) To obtain the total time Γ for a path to be executed.
$$\Gamma = \sum_{i=1}^{m} \tau_i$$
where *m* is the number of configurations in the path.

It is important to point out that it is possible to add more terms to the cost function, in which case care must be taken in order to maintain the homogeneity of units.

---

[2] It means that the robot controller has to calculate the amount of time it will take each joint to reach its destination at the commanded speed. It selects the maximum time among these, and uses this value as the time for all the axes [16].

4. THE ALGORITHM

The algorithm that we propose proceeds as follows:

a) Applies the PRM algorithm [3] for preprocessing the free-space (it is necessary to have a description of the obstacles and of the robot).
b) Given an initial and final configuration, it lets the PRM generate any valid path.
c) Takes the path and uses Simulated Annealing [5][12] to improve the path until an optimum is found or until certain stopping criteria is met.

There are two components of the process that deserve a more detailed description: Probabilistic Roadmaps and Simulated Annealing.

Probabilistic Roadmaps:
PRM proceeds by a two-stage algorithm:

- **Preprocessing**: a set of collision-free configurations is generated and their nodes are interconnected into a network using very simple and fast planning techniques applied to pairs of neighboring nodes. The network produced has a large number of nodes (order of thousands). It may contain one or more components, depending on the robot's free space and the time spent on preprocessing. The network obtained represents the free-space connectivity. Nodes in that network are valid (i.e. without collision) configurations of the robot, and edges between nodes indicate that it is possible for the robot to travel in a straight-line segment from one to another configuration without colliding.

- **Query**: after preprocessing, planning a path between any two configurations is solved by connecting both configurations to some two nodes A and B in the network, and searching the network for a sequence of edges connecting A and B. The planner fails if it cannot connect any of the two input configurations to the network, or if A and B lie in two different connected components of the network.

Simulated Annealing:

It emulates a physical process that proceeds in two steps: a) increases the temperature of a solid until it gets liquefied, b) slowly decreases the temperature until the solid crystallizes. Its purpose is to generate solids at its lowest energy level. The computational algorithm can be summarized as follows: given a candidate solution **c** to the problem, with an associated cost c_d, generates a modification of the solution **t** in the neighborhood of the current solution, with cost c_t. If $c_t<c_d$ then takes the modification as the better solution and iterates again. If $c_t>c_d$ generates a random number x ($0<x<1$) and sees if $x< \exp[(c_t - c_d)/T]$, in which case, takes the modification, otherwise rejects it and iterates again. This step is used to escape local minima, by allowing the algorithm to explore alternatives that seem to be worst. The value of T is used as a parameter for regulating the acceptance ratio of solutions[13], [12]. It will be explained later in this paper.

The following is the algorithm proposed for obtaining a locally optimal path. It receives q_i and q_g, the initial and final configurations, respectively.

```
Algorithm locally-optimal path
   1. Construct the roadmap for the environment
   and the robot.
   2. Find a collision-free path for qi and qg and
   call it p.
   2. If a collision-free path is found (p ≠∅)
   3.    Evaluate its cost: p_cost
   4.    do {
   5.      Propose a new path t, modifying path p
   6.      Evaluate the cost of t: trial_cost
   7.      if (trial_cost < p_cost)
             Accept path t as the best so far
             goto 15
   9.      else {
   10.      Let x be a random number
              between 0 and 1
   11.      if (x<exp((p_cost-trial_cost)/T))
   12.         Accept trial path, goto 15
   13.      else rejects trial path
   15.     iter++;
   16.   }
   17.   while (iter<MAX_ITER or STOP <> TRUE)
   18. return (p, p_cost)
```

There are some points regarding the algorithm that require more attention. We have not mentioned how the cost of a path is evaluated (lines 3 and 6), neither have we explained what it means to accept a trial even when its cost is worst that the one we have as optimum at that moment (lines 11 and 12). In the same way, we should mention how the algorithm is considered to stop (line 16), and of course, how a given path is modified (line 5).

In our case, the cost of a path is given by the function described in section 3. It considers that the time that a robot requires executing the path is the parameter to be optimized. It means that we would like a robot that moves on the shortest time. In order to avoid demanding dangerous accelerations, we bounded the maximum velocity allowed for moving the joints. It should be pointed out that it does not imply any loose in generality. Other parameters could be considered.

At a first glance, a simulated annealing algorithm looks like a greedy algorithm, and it is well known that these kind of algorithms have a common problem: they easily get stuck into local minima. However, SA has a mechanism to escape from traps: it accepts solutions not as good as the one it has at a given time, but it maintains the possibility of using it to move around. In our case, if by modifying a path we found another one that is better (i.e., cheaper); we keep it as the best solution so far. In case that the trial path is worst, there is a chance (represented by the Boltzmann function (exp (p_cost - trial_cost/T)), where T is a parameter that represents the temperature. It is very important to mention that the performance of a SA algorithm relies heavily on the adequate selection of some parameters, we will describe some of them now. The initial value of T, the way in which T is decreased (note that at a higher value of T almost any solution is accepted, without regarding its cost, and as T decreases, the algorithm gets more and more greedy), and the number of iterations we would like to do.

The modification of a given path is as follows: Consider a function *d*, which receives two configurations *x* and *y* of the robot as parameters, and returns the longest Euclidean distance of *x* and *y* in $\mathbf{R}^d$. It means that we want to obtain the maximum displacement that any point in the robot will suffer when traveling from configuration *x* to configuration *y*.

$$d(x,y) = \max_{z \in \text{robot}} |z(x) - z(y)|$$

z denotes a point on the robot (which can be located at the joints or the end-effector), *z(x)* is the position of *z* in the workspace when the robot is at configuration *x*, and $|z(x) - z(y)|$ is the Euclidean distance between *z(x)* and *z(y)*. Figure 1 illustrates the meaning of function *d(x,y)*.

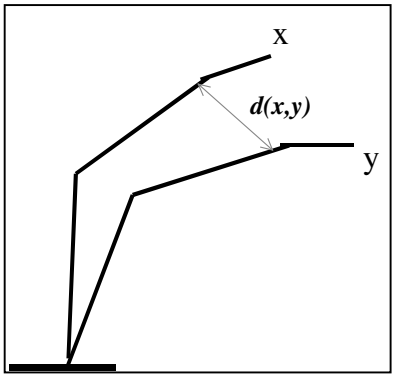

Figure 1. Function d(x,y), which returns the longest Euclidean distance among two configurations.

We define a neighborhood space by using a value ε, which is the maximum Euclidean distance allowed to be traveled by any point in the robot. What the algorithm does is: given a path, choose randomly any configuration, and any joint belonging to it. Randomly propose a new value for the joint, and see if the distance between the original and the modified configuration (distance measured according to the *d* function) does not exceed ε, in which case the proposed path is checked for collision, and if it does not collide with anything, it is a good one, and it can be used in the rest of the algorithm. In other case, repeat the process for proposing a new configuration. The reason behind limiting the distance that is permitted to displace the robot is simple: we would like to use the PRM's local planner in order to save time.

Finally, the criteria for stopping can be established in different fashions. In our case, we considered that the algorithm should stop by one of two causes: a maximum number of iterations have been reached, or the algorithm has made a certain number of iterations not finding any different solution. Of course that the last condition can be reached in the case of a local minimum. However, in practice what can be done is to re-run the complete algorithm (in such a case, it is said that the algorithm is a multiple-restarting one). The idea behind that action is to initiate the process from other initial point in order to have more chances of going on the right way.

5. RESULTS

Some of the results obtained are summarized in the tables I, II and III. All tests were run on a RS6000 250 Power Station, considering a 5 dof planar robot. The obstacles are represented by convex polygons in the working space. The joints were discretized for having up to 128 values each. In all cases the number of configurations that conformed a path is 1500. We run several experiments varying the following parameters: initial temperature T, temperature decrease ratio (α), and maximum number of iterations. The time required to construct the PRM and to search for the path is not reported in the tables, since the graph that represents the C_{free} space was constructed once for a given environment and it was not changed.

Exp No	Initial Cost	Final Cost	No of Iterations	Running time (sec)	Time reduction (sec)
1	12.8	8.7	3560	12	4.1
2	14.5	13.2	5000	17	1.3
3	15.3	10.3	4092	14	5
4	10.5	6.2	5000	19	4.3

Table I. T_0=5000, MAX_ITER= 1000, $\alpha = 0.95$

Exp No	Initial Cost	Final Cost	No of Iterations	Running time (sec)	Time reduction (sec)
1	13.5	11.6	5242	23	1.9
2	14.2	12.3	8890	56	1.9
3	19.6	17.3	8721	57	2.3
4	11.3	8.8	9029	68	2.5

Table II. T_0= 10000, MAX_ITER= 5000, $\alpha = 0.97$

Exp No	Initial Cost	Final Cost	No of Iterations	Running time (sec)	Time reduction (sec)
1	15.5	8.6	10273	85	6.9
2	18.4	18.2	11927	93	0.2
3	19.2	17.2	8927	62	2
4	10.2	10.2	5424	65	0

Table III. T_0= 100000, MAX_ITER=10000, $\alpha = 0.5$

From the tables shown, we can extract some interesting observations: The time invested for reducing execution time ranges from few seconds (12 in the best case) to almost one minute and a half (in the worst case). The save of execution time goes from 0 (in the worst case) to 5 seconds (in the best). Now, what it means in practical terms? Lets suppose that a robot executes only task A, which takes 19.6 s (case 3 in table II), all 24 hours, 365 days, which means executing 4408 times the task in one day. Reducing 2.3 seconds in the task, means 4994 repetitions, i.e., 586 more than before. It could be significantly enough to invest one minute (really more, because of the construction of the PRM, which in our case took almost 4 minutes).

We can also observe that when the decrease of the temperature was faster, the reduction was lower, perhaps because the algorithm got "frozen" too early.

By running more experiments, we would be able of tuning up the algorithm, finding a trade off between the value of α and MAX_ITER. Some other thing that could be important to emphasize is that in almost all cases the PRM+SA obtained a better path that the one proposed by PRM alone.

6. CONCLUSIONS AND FUTURE WORK

From the experiments run, we show that simulated annealing could improve the solutions found by the probabilistic roadmap method. It means that it would be interesting doing further research on the coupling of optimization techniques and motion planning methods.

Concerning the application of SA, an additional effort can be conducted for tuning the algorithm. Besides, some SA-like algorithms can be tested searching for an improvement in the performance of the algorithm (for instance, threshold algorithms).

Other research line that can be developed is the modification of the objective function. It is possible to add constraints (or terms) in order to get paths that are smooth (with bounded velocity and acceleration), short and cheap (considering energy consumption).

In the same way, it would be a good idea to apply this algorithm in cases where traditional PRM fail or require great effort to find a solution. It is the case of problems in which the space has too many narrow passages. Other possibility could be trying this scheme in problems in which a complete network is not constructed. In particular we are thinking in the work of David Hsu and Jean-Claude Latombe on planning for expansive configuration spaces [14], [15]. In that approach, instead of computing the whole network, a couple of trees rooted at the initial and final configurations are grown until they get in a certain vicinity among them. In this case, we could let the trees find one path, and then use a SA in order to improve the path.

REFERENCES

[1] J-C Latombe, *Robot Motion Planning*, Kluwer Academic Pub., Boston, MA, 1991.

[2] J.F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[3] L. Kavraki, *Random Networks in Configuration Space for Fast Path Planning*, PhD Thesis, Stanford University, 1995.

[4] L. Kavraki, J-C Latombe, *Probabilistic Roadmaps for Robot Path Planning*, In Practical Motion Planning in Robotics: Current and Future Directions", K. Gupta & A. del Pobil Eds., Addison-Wesley, 1998.

[5] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing No. 13, May 1983, Vol. 2220, Nr. 4598, pp. 671-680.

[6] H. Wang, *Approximate and Adaptive Algorithms for some Optimal Motion-Planning Problems*, PhD Thesis, Duke University, 1996

[7] J.S.B Mitchell, C. Papadimitriou, The weighted region problem: finding shortest paths through a weighted planar subdivision, Tech. Rep. Cornell University, 1985.

[8] J. Canny, and J. Reif, New lower bound techniques for robot motion planning. Proc. of the IEEE Symp. On the Foundations of Computer Science, Los Angeles, CA, 1987.

[9] B. Donald, P. Xavier, J. Canny, and J. Reif. *Kinodynamic Motion Planning*, J. ACM, 40(5):1048-1066,1993.

[10] S. LaValle, S. A. Hutchinson, *Optimal Motion Planning for Multiple Robots Having Independent Goals*, IEEE Trans. On Robotics and Automation, Vol. 14, No. 6, Dec. 1998.

[11] D. Hsu, J-C Latombe, S. Sorkin*, Placing a Robot Manipulator Amid Obstacles for Optimized Execution*, Submited to IEEE Int. Symp. On Assembly and Task Planning, 1999.

[12] V. Cerny, *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, J. Of Optimization Theory and Applications, Vol. 45, No. 1, pp. 41-51, 1985.

[13] E. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, 1989.

[14] D. Hsu, J-C Latombe, R. Motwani, *Path Planning in Expansive Configuration Spaces*, Proc. IEEE Int. Conf. On Robotics and Automation, pp. 2719-2726, 1997.

[15] D. Hsu, *On Finding Narrow Passages with Probabilistic Roadmap Planners*, Proc. 1998 Workshop on Algorithmic Foundations of Robots, 1998

[16] M. Groover, M. Weiss, R. Nagel, N. Odrey, Industrial Robotics, McGraw-Hill Int. Editions, 1986