Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Generalized Sampling based Motion Planners with Application to Nonholonomic Systems

Suman Chakravorty, S. Kumar

***Abstract*—In this paper, generalized versions of the probabilistic sampling based planners, Probabilisitic Road Maps (PRM) and Rapidly exploring Random Tree (RRT), are presented. The generalized planners, Generalized Proabilistic Road Map (GPRM) and the Generalized Rapidly Exploring Random Tree (GRRT), are designed to account for uncertainties in the robot motion model as well as uncertainties in the robot map/ workspace. The proposed planners are analyzed and shown to be probabilistically complete. The algorithms are tested by solving the motion planning problem of a nonholonomic unicycle robot in several maps of varying degrees of difficulty and results show that the generalized methods have excellent performance in such situations.**

I. Introduction

In this paper, generalized versions of the traditional probabilistic sampling based planners, the probabilistic roadmap (PRM) and the rapidly exploring random tree (RRT), are presented. The traditional techniques are modified to take into account uncertainties in the robot motion model as well as uncertainties in the obstacle locations in the map. The algorithms are analyzed to show that they are probabilistically complete. Experiments are performed on an idealized planar holonomic point robot and the initial results show that the performance of the generalized planners, in terms of their probability of success, is significantly improved when compared to the traditional techniques.

Motion Planning of robots while avoiding obstacles in the workspace has been an active area of research for the past several decades. Classical motion planning can be roughly divided into three different deterministic approaches [1]: cell decomposition, roadmaps and potential field methods. The cell decomposition and roadmap techniques are deterministic methods in the sense that the environment of the robot is sampled/ discretized in a deterministic fashion. However, the problems are PSPACE-Hard [2] and in order to circumvent this computational complexity, randomized sampling based methods known as probabilistic roadmaps (PRM) were introduced [3], [4]. In recent years, there has been a lot of research into accounting for robot dynamics in motion planning, also known as kinodynamic planning. The rapidly exploring random tree (RRT) is a randomized sampling based planners that take into account the dynamics of the mobile robot [2], [5] while building a tree of dynamically feasible trajectories in the free space of the robot.

Suman Chakravorty is an Assistant Professor, Department of Aerospace Engineering, Texas A&M University, College Station, schakrav@aero.tamu.edu
S. Kumar is a Graduate Research Assistant, Department of Aerospace Engineering, Texas A&M University, College Station

All the methods mentioned so far assume that a perfect model of the robot as well as the environment is available. However, this is generally never true. If the uncertainties in the robot and the environment can be modeled probabilistically, the robot motion planning problem can be formulated as a Markov decision problem (MDP), or more generally as a partially observed Markov Decision Problem (POMDP) [2] under sensing uncertainty. However, MDPs/ POMDPs are virtually intractable for anything but small to moderate state spaces. One approach to resolving the issue is through the use of hierarchical methods, an approach that is seen in most biological systems. In recent years, a variety of methods to solve such large MDPs in a hierarchical model-free fashion have been developed, and the field of research is known as hierarchical Reinforcement Learning (Hierarchical RL) [6], [7]. These methods, instead of taking actions at every time step, allow for temporally abstraction policies. Moreover, if it is assumed that these temporally abstract policies can terminate only at one of a few "distinguished" states, then the original large MDP can be transformed into a significantly smaller semi Markov Decision Problem (SMDP) that needs to be solved only at the distinguished states and thus, drastically reduces the computational burden of the Dynamic Programming algorithms used to solve the problem. However, three issues are key in the formulation and solution of any SMDP: a) how to choose the landmark states, b) how to design the local options and c) how to estimate the cost of operation and probability of success of the options? We answer the questions above by proposing the generalized probabilistic roadmap (GPRM) which: a) randomizes the selection of the landmark states, b) designs the local options using traditional feedback control system design techniques and c) evaluates the cost of operation and probability of success of the local options through Monte Carlo simulations. Further, we propose the generalized RRT (GRRT) algorithm, which though it does not fit in the SMDP framework, generalizes the RRT algorithm to robustly handle uncertain robot motion models as well as map uncertainty. This method can be used to rapidly plan dynamically feasible "safe" trajectories, and for extremely large maps, can be used as the options in the GPRM framework.

The methodology advocated in this paper for robot motion planning is an integrated planning and control technique and as such, is related to the sequential composition methods [8]–[10] for deterministic robotic systems. In these methods

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

a global control policy, (for stabilization, tracking etc.), is designed by concatenating local policies that have smaller (local) domains of operation. These papers thus advocate the design of local planners using traditional control techniques and stitching them together using a higher level graph that shows the interconnection of these local policies to form a global policy. Our method takes uncertainty into account and the construction of the higher level graph required for planning is entirely different in our approach in that it is based on the construction of an SMDP.

The rest of the paper is organized as follows. In section II, we present the GPRM and the GRRT algorithms. In section III, the methodology is applied to the motion planning of nonholonomic unicycle robot in several cluttered maps with uncertainty in the motion model as well as in the map. Owing to the paucity of space the analysis of the algorithms is left out of this paper, the interested reader can see the complete analysis in the complete technical report on the first author's webpage at $http : //dnc.tamu.edu/wiki/index.php/Suman_Chakravorty$.

II. Generalized Sampling based Motion Planners

In this section, we present the generalized sampling based motion planners, the generalized probabilistic roadmap (GPRM) and the generalized rapidly exploring random tree (GRRT), which extend the traditional PRM and RRT algorithms to systems with uncertainty.

A. Uncertainty Model

The generalized sampling based algorithms require an uncertainty model for both the motion of the robot as well as a model for the map uncertainty. In the following, we outline the models that are used in this paper.

We assume that the dynamics of the mobile robot are specified by the following white noise perturbed stochastic differential equation:

$$\dot{x} = f(x) + g(x)u + h(x)w, \tag{1}$$

where x represents the state of the robot, w represents the white noise perturbation and u represents the control input to the robot. The above is a non-parametric model of uncertainty in the robot motion model and will be used throughout this paper for the lower level control law designs.

We assume that the uncertainty in the map is specified through a binary occupancy value, i.e., $p(O/y)$, the probability that there is an obstacle at the point y in the map. The occupancy values in the map can be considered to be the output from a mapping algorithm. However, in this paper we shall not consider the mapping algorithm and assume that a map with binary occupancy values is provided to the planner by some suitable mapping algorithm. The state of the robot consists of $x = (q, \dot{q})$ where q represents the configuration of the robot and $\dot{q}$ represents the generalized velocities. The free region in the map induces a free region in the configuration space, say $\mathcal{C}_{free}$. This means that any state whose configuration is in $\mathcal{C}_{free}$ is safe. This in turn induces a free space in the state space of the robot, say $\mathcal{X}_{free}$. From now on, we will assume that for GPRM and GRRT, we are sampling equilibrium states, i.e., states wherein the velocities are zero, in the free state space $\mathcal{X}_{free}$.

Further, we shall assume in this paper that the state of the robot is known perfectly. The case of imperfect state observation will be considered in future research.

B. Generalized Probabilistic Roadmap (GPRM)

In motion planning, the objective is to plan the path of a robot from a start state to an end state. The Probabilistic RoadMap (PRM) attempts to accomplish this by a) randomly sampling the state space of the robot, b) connecting every sampled point with its k-nearest neighbours using some local open loop planner such as a straight line planner while checking for collisions. The end result of the PRM is a graph or roadmap on the workspace of the robot that contains the feasible connections between the sampled points in the state space. The problem is solved if there exists a path on the graph that connects the start and the goal states. In the case of systems with uncertainty, it may be impossible to find a path that succeeds with probability one and hence, we are interested in finding paths that have a success probability above a pre-specified minimum threshold p_{min}.

The pseudo-code for the generalized probabilistic roadmap (GPRM) algorithm is shown below. As can be seen from the pseudo-code, Steps 2-3 and 5 are different from the traditional probabilistic roadmap (PRM) algorithm. In the following, we discuss these steps of the algorithm in detail.

Algorithm 1 Algorithm GPRM

- Given x_0, the starting point, x_g, the goal point of the robot and p_{min}, the minimum probability of success
- Initialize the GPRM with the nodes x_0 and x_g
  1) Sample the equilibrium states in $\mathcal{X}_{free}$ probabilistically using a uniform distribution
  2) Grow the GPRM by connecting every sampled point in the domain with its k-nearest neighbours using suitable obstacle-free feedback controllers
  3) Evaluate the cost of every connection in the resulting graph using Monte Carlo simulations
  4) Plan on the resulting graph using the evaluated edge costs from step 3
  5) Evaluate the probability of success, p_s, of the resulting path from step 4. If $p_s > p_{min}$, end; else go to step 1
- End

First, we discuss Step 2 in detail.

Given the robot dynamics as defined in the previous section and some equilibrium point x_g in the state space of the robot, there exists a feedback controller $u(., x_g)$ such that the robot can be controlled into a neighbourhood of the point x_g with some (high) probability, in the presence of the stochastic disturbance forces and in the absence of any obstacles in the

map. Note here that the equilibrium point of the robot x_g corresponds to some location in the map that the robot needs to reach. Let Ω_{x_g} denote a neighbourhood of the point x_g, the above then implies that the probability of the state of the robot, $p(x(t))$, is concentrated mostly in the region Ω_{x_g} as $t \to \infty$.
The following notes are in order here.

- Due to the stochasticity of the system, it is impossible to control the robot exactly to the point x_g even in the absence of obstacles.
- The use of a feedback controller to connect nodes is the analogue of the step in traditional PRM wherein near by nodes are connected using some local open loop planner, such as a straight line planner. In the case of stochastic systems, a feedback controller is necessary because the uncertainty can carry the robot away from its prescribed path in which case the open loop planner breaks down as it no longer has a plan for the deviated path. In fact, it is the entire premise of the field of feedback control that the use of feedback grants robustness to such uncertainty.
- The feedback controller is designed for a workspace without any obstacles as otherwise the controller design is quite complicated owing to the constraints imposed on the robotic system by its workspace.

Next, we detail step 3.
The feedback controller that we design for controlling the robot from one node to another is for an obstacle free map and hence, there is no guarantee that the controller will actually succeed in connecting the two nodes in the presence of obstacles. Thus, we need to test the controller through repeated simulations to evaluate its probability of success. This can be stated precisely as follows.
Given a start node x_i and a target node x_j, we may evaluate the probability of success of the local controller, $u(:, x_j)$, in connecting the nodes as follows. Recall that we are never sure to be at either landmark x_i or x_j due to the uncertainty in the system. Hence, the feedback controller to control the system from $x_i \to x_j$ is turned on when the state of the robot enters some pre-specified neighbourhood of x_i, say Ω_i, and turned off when the state of the robot enters some neighbourhood of the node x_j, say Ω_j, at which time the feedback controller to get it to one of the neighbouring nodes of x_j is switched on. Let one particular instance of a trajectory, say the N^{th}, that goes from $\Omega_i \to \Omega_j$ under the feedback controller $u(:, x_j)$ be $x_0^{(N)}, \cdots, x_{t(N)}^{(N)}$, where $t(N)$ denotes the time the controller terminates. The time $t(N)$ is called a stopping time and is also random since it depends on the particular realization. The probability of success of the N^{th} realization is given by

$$p_s^{(N)} = (1 - p(O/x_0^{(N)})) \cdots (1 - p(O/x_{t(N)}^{(N)})), \tag{2}$$

where recall that $p(O/x)$ is the occupancy probability that there is an obstacle at the point x in the state space of the robot. In addition, we can find the cost of the plan from x_i to x_j, $c_{ij}^{(N)}$, in terms of physical variables such as fuel, time etc. Then, if we do repeated simulations, the probability of success and cost of the controller $u(:, x_j)$ in controlling the robot from x_i to x_j can be approximated as

$$p_s^{ij} \approx \frac{1}{T} \sum_{N=1}^{T} p_s^{(N)}, \tag{3}$$

$$c_s^{ij} \approx \frac{1}{T} \sum_{N=1}^{T} c_{ij}^{(N)}, \tag{4}$$

and due to the law of large numbers, it follows that as $T \to \infty$, the above estimates converge to the true values of the parameters.
Given the probability of success of a controller in connecting nodes x_i and x_j and the cost in successfully connecting them, the cost of the edge connecting the nodes x_i and x_j in the graph is given by

$$c^{ij} = p_s^{ij} c_s^{ij} + (1 - p_s^{ij}) c_F, \tag{5}$$

where c_F is some heuristically defined, suitably high cost of failure. The above equation allows us to evaluate the edge costs in the graph that is formed by connecting any node to its k-nearest neighbours.

Next, we explain the rationale behind step 5. In traditional PRM, if we find a path from the start node to the goal node, the planning problem is solved. However, in the presence of uncertainty, we have to make sure that the probability of success of the path planned on the graph is above the minimum threshold value of p_{min}. Thus, it is not necessary that if there is a path from the start node to the goal node, it has the minimum required probability of success. This has to be tested. Thus, once a minimum cost path is found on the graph according to the edge costs as defined above, the probability of success of the path is given by the product of the success probabilities of the individual segments of the path, which in turn is known from step 3. Thus, if the success probability is higher than the threshold, the planning problem is solved; else, more points have to be sampled in the state space, or the nodes connected to more of its neighbours, in order to make sure that the success probability is higher than the given threshold. The central reason that we need to do this is because the probability of success of a path is given by the product of the probabilities of the individual segments of the path whereas the cost of a path on the graph, using which we search on the graph, is the sum of the costs of the individual segments. That this method is probabilistically complete, i.e., it converges to a solution, give that one exists, as the number of sampled nodes go to infinity, is shown in the analysis section of this paper

C. Generalized Rapidly Exploring Random Trees (GRRT)

The traditional Rapidly Exploring Random Tree (RRT) algorithm attempts to connect a start point and an end point in the workspace of a robot by growing a tree using random sampling as follows: a) randomly pick a point in the state space of the robot, b) find the nearest node on the tree according to some pre-specified metric, c) connect the nearest node on the tree to the sampled node using some local planner while checking for collision, and d) add the new node to the tree if the robot does not collide with obstacles.

The tree is grown in this fashion till a feasible path is found from the start point till the goal point. Due to uncertainty it might not be possible to find a path that succeeds in connecting two points with probability one. Hence, in the current scenario we require that the path have a minimum pre-specified probability of success p_{min}.
We now present the generalized version of the Rapidly Exploring Random Tree (RRT) algorithm, the GRRT. The pseudo-code for the algorithm is presented below: Steps 2,

Algorithm 2 Algorithm GRRT

- Given start state x_0 and goal node x_g. Initialize tree with x_0, set $p(x_0) = 1$.
- Repeat until the tree reaches goal node x_g:
  1) Generate node x at random (x is an equilibrium state in X_{free})
  2) Connect x to the node x^* on the tree, using local feedback control, that satisfies:
  $$x^* = arg\max_i p(x_i)p(x_i x), \tag{6}$$
  where $p(x_i x)$ is the probability of successfully transitioning from $x_i \rightarrow x$ under the local feedback law.
  3) Set $p(x) := p(x^*)p(x^* x)$.
  4) If $p(x) > p_{min}$, then add node x to the tree with label $p(x)$, else go to Step 1.
- End.

3 and 4 in the algorithm are different from the traditional RRT algorithm. In the following, we discuss the details about these differences starting with Step 2 in the algorithm.

Step 2 is common with the GPRM algorithm. The nodes (or more precisely, the neighbourhoods of the nodes) are connected by local feedback controllers that have been designed using control techniques, and the probability of success of the controller evaluated as in the GPRM algorithm. The reason we chose the node as in Eq. 6 has to do with the proof of completeness of the resulting algorithm. In fact, the choice can be thought of as the "nearest node" metric that is used to select the node in the tree that is connected to the newly generated node. Hence, x^* as defined in Eq. 6 is the best node in terms of the probability of success of transitioning from $x_0 \rightarrow x^* \rightarrow x$, where recall that x_0 is the root node.

Next, we detail steps 3 and 4. Step 3 labels any newly generated node with the "probability of success" of the robot moving from the root node to that particular node. We only want to keep nodes in the tree that have a success probability (of transitioning to it from the root node) more than the threshold of p_{min}. Hence, we have included the tree pruning in step 4. Note that given the probability of success, $p(x_i, x_j)$ of transitioning from any parent node x_i to its children x_j in the tree, the probability of success of a path $x_0, x_1, \cdots, x_N$ is given by $p(x_0, x_1)p(x_1, x_2)\cdots p(x_{N-1}, x_N) = p(x_{N-1})p(x_{N-1}, x_N)$, according to the labeling convention that we have used, where, recall that $p(x)$ represents the probability of successfully transitioning form the root node to node x.

D. Connection to Markov Decision Processes (MDP)

It is well known that most sequential decision making problems under uncertainty may be posed as Markov Decision Problems (MDP). Thus, it would behoove us to know what connection, if any, exists between the Genralized smapling based motion planners presented above and MDPs.
First, we provide a very brief overview of Markov decision processes (MDP) and semi-Markov decision processes (SMDP). More comprehensive treatments of these topics can be found in [11], [12]. We shall only consider discrete-time finite state MDPs here. Let s denote the state of a finite MDP, $s \in S$, S being a finite set. Let u denote the control action which can take a discrete number of values. The MDP is characterized by a transition probability function, $p(r/s, u)$, which is the probability that the system will transition from state s to r under control u, at the end of one time step. The goal of the MDP is to solve the infinite horizon discounted optimal control problem given by

$$\mu^*(s_0) = arg\min_{\mu=\{u_0,u_1,\cdots\}} E[\sum_{t=0}^{\infty} \beta^t c(s_t, u_t)/s_0)], \tag{7}$$

where $\mu = \{u_o, u_1, \cdots\}$ is an infinite horizon control policy, $c(s, u)$ is the cost that the system incurs in taking control action u at state s and $\beta < 1$ is a given discount factor. It is well known that the optimal control policy corresponding to the problem posed above is stationary and is given by [11], [12]

$$u^*(s) = arg\min_u \{c(s, u) + \beta \sum_r p(r/s, u)J^*(r)\}, \tag{8}$$

where $J^*(s)$ is the optimal cost-to-go function and is found as the solution of the Bellman fixed point equation/ Dynamic Programming equation:

$$J^*(s) = arg\min_u \{c(s, u) + \beta \sum_r p(r/s, u)J^*(r)\}. \tag{9}$$

It is very well known that the (Bellman) operator underlying the above equation is a contraction operator with contraction factor β and thus, the solution can be found through successive approximations [12] which is the basic principle behind value and policy iteration [12].
Semi-Markov Decision Processes (SMDP) offer a method for temporal abstraction in MDPs. In addition to the set of primitive controls, u, available at every state s, assume that there are available also a set of options/ policies Π_s which can execute for variable periods of time before terminating stochastically at some state r with probability distribution $\beta(.)$. Note that the controls (or the primitive controls) are also options that execute for one step and then, necessarily terminate. The optimality equation for SMDPs may be shown to be

$$J^*(s) = \min_{\pi \in \Pi_s} \{c^\pi(s) + \sum_r p^\pi(r/s)J^*(r)\}, \tag{10}$$

where $c^{\pi}(s)$ is the expected discounted cost of executing policy π starting at state s, and $p^{\pi}(r/s)$ is the generalized discounted transition probability function of an MDP and is given by

$$p^{\pi}(r/s) = \sum_{N} p^{\pi}(r, N/s)\beta^{N}, \tag{11}$$

where $p^{\pi}(r, N/s)$ denotes the probability that the policy π terminates after exactly N steps at state r, given that the policy started executing at the state s. The above optimality equation is a generalization of the Bellman equation to the case of SMDPs and inherits the contraction property of the Bellman operator. Thus, policy and value iterations may be used to solve the optimality equation for SMDPs.

Thus, although most problems of sequential decision making under uncertainty are well-posed as a Markov Decision Problem (MDP), the solutions techniques for MDPs suffer from the curse of dimensionality and thus, are computationally intractable for anything but small maps. The SMDP framework described above is a hierarchical approach to the solution of MDPs. In this approach, at every state, a set of options are defined. Using the terminology of Sutton et. al. [6], [7], [13], [14], options are stationary policies (or local controllers) on the state space of the problem that can execute for varying lengths of time, and thus, provide a means for temporal abstraction in the problem. We further assume that the options, once executed, can only terminate in the neighbourhood of one of the landmark states. The situation is shown in Fig. 1. Under the framework of options, the Markov Decision Process (MDP) on the primitive states underlying the planning problem is transformed into a semi-Markov Decision Problem (SMDP) on the landmark states. Since the number of landmark states is orders of magnitude smaller when compared to the number of "primitive" states in the MDP, the computational complexity of SMDP (hierarchical) planning is greatly reduced when compared to the complexity of the original "primitive" planning. The hierarchical planning formulation requires answers to the following questions: a) a scheme for the selection of the landmark states in the state space, b) a design scheme for the local controllers/ options and c) knowledge of the generalized cost and transition functions associated with the options/ local controllers in order to solve the higher level SMDP.

To the perceptive reader, it will be clear by now that the GPRM methodology provides the answers to the above three questions by: a) picking the landmarks at random (random sampling of the state space), b) utilizing feedback control techniques to design local controllers and c) evaluating the local options for their costs and success probabilities using Monte Carlo simulation. In fact, it is clear that the traditional PRM methodology is a method of constructing SMDPs for purely deterministic systems. We believe that randomization is the key to breaking the curse of dimensionality in SMDPs. There is ample evidence of this in the robotic motion planning literature, in the deterministic setting, through the empirical success of the PRM and RRT techniques on very high dimensional planning problems. Also, it has been shown that naive randomization does break the curse of dimensionality in discrete decision processes (DDPs) [15], which are MDPs with a continuous state space but finite set of actions. It can be seen that the GPRMs proposed in this paper are more sophisticated cousins of the DDPs and hence, we could expect that the GPRMs also break the curse of dimensioanlity, in some sense, while having better performance than DDPs in practice. Of course, this is conjecture and a rigorous proof of these statements will be the course of our future research.

III. Nonholonomic Unicycle Robot

In this section, we apply the sampling based motion planners to the motion planning of a unicycle model whose equations of motion are given by

$$\dot{x} = vcos(\theta), \tag{12}$$

$$\dot{y} = vsin(\theta), \tag{13}$$

$$\dot{\theta} = \omega, \tag{14}$$

where (x, y, θ) represents the pose of the robot and the velocity v and the angular velocity ω represent the control inputs to the problem. In this case, our sampled poses are in the (x, y, θ) spaces and the job of the local feedback controllers is to stabilize the robot about any of these equilibrium configurations. Due to the nonholonomy of the sytem, linear control techniques are not applicable to the stabilization problem. Thus, suitable nonlinear control techniques are used to design feedback laws. We chose a dynamic feedback linearization based controller design that has been treated in detail in the reference [16].

Uncertainty was added to the robot motion model by adding white noise to the robot dynamics equations above with the intensity of the white noise being approximately 30% of the maximum allowable vehicle linear/ angular speed, i.e., the noise in the x, y equations had intensity equal to 30% of the maximum allowable linear speed while the noise in the θ equation had intensity equal to 30% of the maximum allowable angular speed.

The results of our numerical simulations on a candidate map is shown in Fig. 3. We have tested our algorithms on several other maps of varying degrees of difficulty but cannot show here due to the paucity of space. These results are available in the extended technical report on the first author's webpage. In the Figure, subfig. (a) represents the tree of feasible trajectories built by the GRRT algorithm, the feasible trajectories shown without any noise in the system. Subfig. (b) shows the final bundle of trajectories from the start state to the goal state under the sequence of feedback controllers encoded in the tree. Note that there are multiple trajectories because of the uncertainty in the motion model. Similarly, Subfig. (c) shows the graph built by the GPRM algorithm, however, the edges between the nodes on the graph are only virtual, i..e, they are not the actual trajectories. Subfig. (d) represents the final bundle of trajectories form the start to the goal point encoded in the GPRM graph. It can be seen from these plots that the performance of the method is excellent and it successfully navigates quite complicated maps under

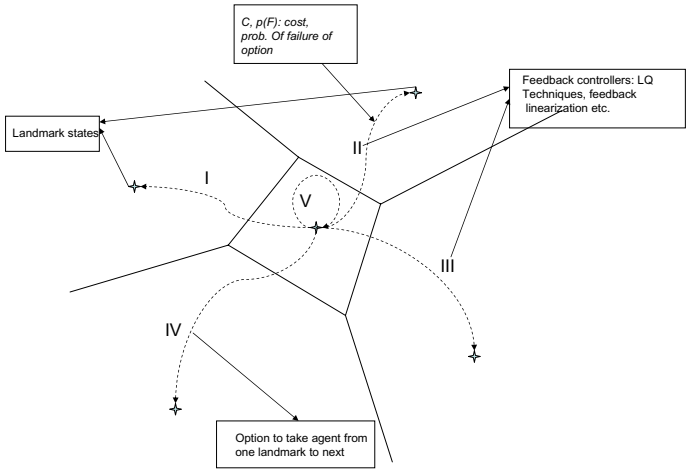

Fig. 1. Schematic of Hierarchical Planning

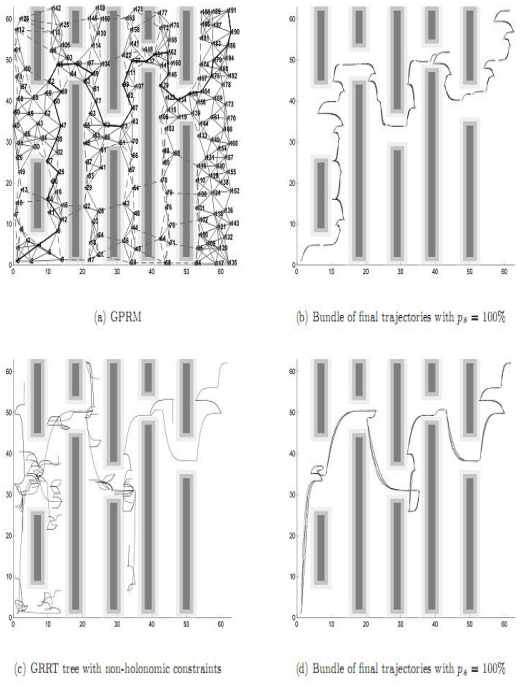

Fig. 2. Performance of GRRT/ GPRM on unicycle robot: Map4

motion as well as map uncertainty.

Thus, in this section, we have shown the application of the generalized sampling based motion planner to an under-actuated nonholonomic robotic system. As can be seen from the results, the planners have excellent performance in quite complicated maps, in the presence of motion uncertainty as well as uncertainty in the map. the next step would be to test the planners on larger maps as well as include dynamics into the planning equations. Also, we would like to experiment with the planners on more generic robotic systems than the mobile robot systems considered in this paper.

REFERENCES

[1] S. M. LaValle, "Robot motion planning: A game-theoretic foundation," *Algorithmica*, vol. 26, pp. 430–465, 2000.
[2] ——, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2005.
[3] L. E. Kavraki, M. N. Koulountzakis, and J. C. Latombe, "Analysis of probabilistic roadmaps for path planning," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996.
[4] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings IEEE International Conference on Robotics and Automation*, 1996.
[5] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 1999.
[6] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, pp. 181–211.
[7] R. Parr, *Hierarchical Control and Learning from Markov Decision Processes, PhD Thesis*. Berkeley, CA: University of California, 1998.
[8] R. R. Burridge, A. A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviour," *Int. J. Rob. Res.*, vol. 18, pp. 534–555, 1999.
[9] D. C. Conner, *Integrating Planning and Control for Constrained Dynamical Systems, PhD Thesis*. Pittsburgh, PA: Robotics Institute, Carnegie Mellon University, 2007.
[10] D. C. Conner, H. Kress-Gazit, h. Choset, A. Rizzi, and G. J. Pappas, "Valet parking wihout a valet," in *Proc. RSJ/ IEEE Int. Conf. on itell. rob. and syst. (IROS)*, 2007.
[11] M. Puterman, *Markov Decision Process*. Hoboken, NJ: Wiley-Interscience, 1994.
[12] D. P. Bertsekas, *Dynamic Programming and Optimal Control, vols I and II*. Cambridge: Athena Scientific, 2000.
[13] D. Precup and R. Sutton, "Multi-time modeling for temporally abstract planning," in *Advances in Neural Information Processing Systems*, 1997.
[14] R. Parr and S. Russell, "Reinforcement learning with hierarchy of machines," in *Advances in Neural Information Processing Systems*, 1997.
[15] J. Rust, "Using randomization to break the curse of dimensionality," *Econometrica*, vol. 65, pp. 487–516, 1997.
[16] G. Oriolo, A. D. Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: Design, implementation and experimental validation," *IEEE Transactions on Control System Technology*, vol. 10, pp. 835–852, 2002.