GENERALIZED SAMPLING-BASED FEEDBACK MOTION PLANNERS

A Dissertation

by

SANDIP KUMAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2011

Major Subject: Aerospace Engineering

GENERALIZED SAMPLING-BASED FEEDBACK MOTION PLANNERS

A Dissertation

by

SANDIP KUMAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Suman Chakravorty
Committee Members,	Nancy Amato
	Raktim Bhattacharya
	John Junkins
	John Valasek
Head of Department,	Dimitris Lagoudas

December 2011

Major Subject: Aerospace Engineering

ABSTRACT

Generalized Sampling-Based Feedback Motion Planners. (December 2011)
Sandip Kumar, B.Tech, Indian Institute of Technology, Kharagpur, India
Chair of Advisory Committee: Dr. Suman Chakravorty

The motion planning problem can be formulated as a Markov decision process (MDP), if the uncertainties in the robot motion and environments can be modeled probabilistically. The complexity of solving these MDPs grow exponentially as the dimension of the problem increases and hence, it is nearly impossible to solve the problem even without constraints. Using hierarchical methods, these MDPs can be transformed into a semi-Markov decision process (SMDP) which only needs to be solved at certain landmark states. In the deterministic robotics motion planning community, sampling based algorithms like probabilistic roadmaps (PRM) and rapidly exploring random trees (RRTs) have been successful in solving very high dimensional deterministic problem. However they are not robust to system with uncertainties in the system dynamics and hence, one of the primary objective of this work is to generalize PRM/RRT to solve motion planning with uncertainty.

We first present generalizations of randomized sampling based algorithms PRM and RRT, to incorporate the process uncertainty, and obstacle location uncertainty, termed as "generalized PRM" (GPRM) and "generalized RRT" (GRRT). The controllers used at the lower level of these planners are feedback controllers which ensure convergence of trajectories while mitigating the effects of process uncertainty. The results indicate that the algorithms solve the motion planning problem for a single agent in continuous state/control spaces in the presence of process uncertainty, and constraints such as obstacles and other state/input constraints.

Secondly, a novel adaptive sampling technique, termed as "adaptive GPRM" (AGPRM), is proposed for these generalized planners to increase the efficiency and overall success probability of these planners. It was implemented on high-dimensional robot n-link manipulators, with up to 8 links, i.e. in a 16-dimensional state-space. The results demonstrate the ability of the proposed algorithm to handle the motion planning problem for highly non-linear systems in very high-dimensional state space.

Finally, a solution methodology, termed the "multi-agent AGPRM" (MAGPRM), is proposed to solve the multi-agent motion planning problem under uncertainty. The technique uses a existing solution technique to the multiple traveling salesman problem (MTSP) in conjunction with GPRM. For real-time implementation, an "inter-agent collision detection and avoidance" module was designed which ensures that no two agents collide at any time-step. Algorithm was tested on teams of homogeneous and heterogeneous agents in cluttered obstacle space and the algorithm demonstrate the ability to handle such problems in continuous state/control spaces in presence of process uncertainty.

DEDICATION

To My Parents and My Wife

ACKNOWLEDGMENTS

After working for three years in industry, and living with the feeling that a graduate degree is a must, I landed in College Station four and a half years ago. I met Dr. Suman Chakravorty and expressed my interest and since then have never looked back. It has been an awesome journey, with finally the sense of satisfaction, in which I have grown both on personal and professional front. I would like to gratefully and sincerely thank my adviser, Dr. Suman Chakravorty, for his continuing guidance, understanding, and patience during my graduate studies at Texas A&M University. There were days when I was clueless, days when I was stuck, days when I was down, days when I got scoldings, but Suman was always there to talk it through and had continued trust in me that I would come this far. I would really like to thank him for all the in-depth discussions, those frivolous talks, immensely scientific insights which has made this dissertation possible today.

I would sincerely like to thank Dr. John Valasek, who has always been there to monitor the progress and motivate me to work even harder to meet goals and deadlines. His valuable comments during meetings have been deep rooted in this work. I would like to thank Dr. Raktim Bhattacharya, who has been an inspiration and taught me to enjoy my work and research. He was the goto person when I was in need of ideas or directions beyond research and those discussions were really helpful. I would like to thank Dr. Nancy Amato and Dr. John Junkins for their critical comments and guidance. I would also like to thank Dr. Swaroop Darbha and Dr. Sivakumar Rathinam in Mechanical Engineering, who helped me understand the working and intricacies of multiple traveling salesman problem in a short duration. I would also like to thank sponsors of this project, AFOSR.

Two people who has always been there not only as my research group mates but also as friends and people who have taught me concepts which have been instrumental throughout this journey, are Roshmik and Ali and I would like to extend my sincere

thanks to them too. I would also like to thank my AFOSR meeting members, Mrinal, Anshu, Elizabeth, Kenton and Kiron, who have always provided valuable inputs to my work.

I would also like to thank my friends Gaurav, Nikhil, GV who have made this journey memorable, enthusiastic and fun.

It has been amazing past one and a half year of marriage and past approximately six years of knowing my lovely wife and one of my best friend, Vinny, who has shared her life with me. She has been there always with her immense love and faith in me. She has tolerated me and loved me even more, I would like to thank her for being there always. And, last but I guess the most important, I take this opportunity to thank my lovely and proud mummy and papa, who have always supported me with their love, patience, guidance throughout my journey. I just want to say to them that now you can be even more proud as I have finally done it.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

1. INTRODUCTION

In recent years, considerable interest has been shown in, and relevant resources have been devoted to, the design, development and operation of unmanned aerial, underwater, and ground vehicle. The purposes of such unmanned vehicles are extremely diverse, ranging from scientific exploration and data collection, to commercial services, and military reconnaissance and intelligence gathering. Unmanned vehicles make it possible to perform critical tasks without endangering the life of human pilots. There is a strong perceived need for an increased level of automation, in order to improve the system's efficiency, reliability, and safety, and decrease cost. Some successful examples are, NASA's Spirit and Opportunity rovers which allow humans to see and explore surface of Mars, autonomous vehicles that complete the DARPA "Grand Challenge", robots on the assembly floor that assemble everything from automobiles to mp3 players, thereby increasing productivity and decreasing costs. For helping in mundane tasks such as vacuuming floors, there is the Roomba robot.

A basic problem which has to be faced and solved by autonomous vehicles, on which this dissertation will focus, is the problem of *motion planning*. It involves generation and execution of a plan for moving around an environment towards a designated goal, or to accomplish a desired task avoiding collisions with obstacles in the environment. Moreover, it is desirable to optimally use the available resources to achieve the goal, thereby optimizing some "cost" measure.

There are many established techniques to solve motion planning problem in a deterministic framework ranging from optimal control method [1], grid world approachs [2] to randomized sampling based motion planners [2] [3, 4]. However real world systems are not deterministic and there evolution involves uncertainty due to surrounding environments or internal parameter variance. Hence, in the real world,

---

This dissertation follows the style of *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*.

these deterministic algorithms are applied along with some trajectory tracking techniques, that accounts for the uncertainty in the system [5].

Another approach to solving motion planning problem is to take uncertainty into account while solving the problem. Introduction of uncertainty in the motion planning problem increases the complexity of the problem. Uncertainty in the system can be due to two scenarios, one due to sensing uncertainty and the other due to process uncertainty. Sensing uncertainty arrises from sensor noise during measurements or an uncertain environment, i.e. partial knowledge of obstacle locations in the given environment. The process uncertainty is the motion uncertainty due to presence of stochastic forcing in the system dynamics and controls.

Dynamical systems with uncertainty evolves as a stochastic process which has the Markovian property, i.e the evolution is memoryless and the future and past states do not influence the system evolution given the current state. The problem of solving an optimization problem for such a Markovian stochastic process can be modelled in a mathematical framework termed *Markov Decision Process* (MDP) [6]. Researchers have attempted to solve MDPs using *Dynamic Programming* (DP) [7,8] and *Reinforcement Learning* (RL) techniques [9]. The DP methods are model-based methods while RL is a model-free fashion of approaching the same problem.

Introduction of sensing uncertainty in the system, transforms the MDP into a so-called *Partially Observed Markov Decision Process* (POMDP) [7,8]. An autonomous robot, starting in an unknown region leads to the problems of localization (knowing the current position with respect to the surrounding), mapping (knowing the map, given the location) and planning (given the current state and surrounding map, plan over it). The *Simultaneous Localization and Mapping* (SLAM) research community attempts to solve the localization and mapping problem simultaneously. The researchers in planning community, attempt to solve the planning problem, i.e. achieving a desired goal given an optimizing cost criteria, and a map of the surround-

ing environment. The map can be a deterministic or a stochastic one. The stochastic map would be an output from a mapping algorithm.

In this dissertation we assume the robot state is known (or can be precisely determined by sensors, i.e. there is no sensing uncertainty) and we shall focus on the problem of solving motion planning in presence of process uncertainty over a given stochastic map (knowledge of obstacle locations on an environment are probabilistic but stationary over time).

1.1 Motivation

The motion planning problem is a sequential decision making problem and optimal control is the most general framework for solving such a problem. We wish to solve the motion planning problem involving uncertainty in the form of process uncertainty and stationary stochastic maps. The motion planning problem can be formulated as a Markov decsion process (MDP), if the uncertainties in the robot motion and environments can be modeled probabilistically. However MDPs are virtually intractable for anything but small to moderate state spaces as they suffer from "curse of dimensionality" [10]. It means the complexity of solving MDPs grows exponentially as the dimensionality of the problem increase and hence it is nearly impossible to solve even without constraints. Introduction of constraints (which in robotic motion planning problem framework means obstacles, velocity and acceleration rates, torque and force saturation, and limited domain), there is no structured technique to accomplish the planning. Hierarchical methods [11, 12] have tried to resolve the issue of dimensionality by introducing "distinguished states" (we will call them "landmarks") and invoking options/ policies at these states that can only terminate at these states. Using this transformation the original large MDP can be transformed into a semi Markov decision process (SMDP) which needs to be solved only at these states, and hence, the computational burden is reduced drastically on the stochastic optimization algorithm.

In parallel, deterministic robotic motion planning problems have been solved using randomized sampling based algorithms [2] (like probabilistic roadmaps (PRM) [3] and rapidly exploring random trees (RRT) [4]). The essential idea behind these techniques is to sample configurations of the robot in the free configuration space ($\mathcal{C}_{free}$) and attempt to connect them using local planners (like straight line planners) and hence generate a topological graph $\mathcal{G}(V, E)$ with vertices V being the sampled configurations and edges E of the graph being the connecting path between vertices developed using the local planners. Given the graph, the solution to the motion planning problem involves discrete search on the developed graph (as in PRM) or biasing the graph towards the goal configuration (as in RRT). They have proved extremely powerful in solving high dimensional problems that were previously unsolvable using traditional deterministic planning techniques. Unfortunately these techniques were designed for a deterministic framework, and are not robust to systems with uncertainty. Various attempts have been made to incorporate uncertainty in the problem : stochastic maps, process uncertainty and sensing uncertainty. These research will be discussed in the related work section, further in this section.

The motivation of the work in this dissertation is to generalize the PRM method to sequential decision making problems with process uncertainty and stochastic maps such that the benefits of PRM may be realized for the robotic motion planning problem under process and map uncertainty. The solution methodology will involve design which scales for continuous state space and continuous control spaces, and will handle constraints in the state space of the robot. Our technique will utilize feedback solutions to the lowest level local planner of the hierarchical SMDP based solution technique.

The use of feedback controllers in the presence of uncertainty ensure that the trajectories converge to the goal location, thus mitigating the effects of process uncertainty.

Our technique will decompose the problem into a two-layer hierarchical SMDP. The "lowest level" will consist of feedback solutions between landmark states, and the "top level" consists of a topological graph, with landmarks as vertices, on which solution is searched (we will be using stochastic DP [7, 8]). Feedback planners of *lowest level* provide information (e.g. transition probabilities and costs) to the *top level*, and with probabilities involved they induce an MDP at the *top level* on these landmark states, resulting in a SMDP, the solution to which can be found using *stochastic DP*. Solution to this stochastic DP results in a hybrid feedback solution to the motion planning problem in presence of process uncertainty and stochastic maps.

Finally, we would also like to provide the formal analysis of the proposed algorithm in terms of probabilistic completeness, as this being a sampling based motion planner.

1.2 Problem Statement

In this dissertation, we focus on solving the robotic motion planning problem in presence of uncertainty, where uncertainty is in the form of process uncertainty and stochastic maps. Our main goal is to develop a solution which will scale to continuous state/ control spaces in particular, be able to solve the motion planning problem under uncertainty for agents/ robots having non-linear dynamics and high dimensional state spaces.

1.2.1 Problem 1

Given a stochastic map of the environment, where obstacles location probabilities are static, a robot equipped with a perfect state sensors (i.e. no sensing uncertainty in robot state), the initial configuration (q_I) of the robot, a desired goal configuration

(q_G), and a minimum required success probability (p_{min}) in the domain, to solve, the motion planning problem for the robot in presence of process uncertainty.

In order to solve the motion planning problem, we solve for the control policy π, i.e. a sequence of control inputs $\pi = \{u(:, q_1), \ldots, u(:, q_n)\}$, where u denote a feedback control law which is a function of $q_i, i = 1, \ldots, n$ sampled configurations (*landmarks states*) and $q_n = q_G$, i.e. the final desired configuration, which will take the robot from q_I to q_G through a path which will have a probability of success, p_s, associated with it such that $p_s > p_{min}$, where p_{min} is the minimum required probability of success. The notion of probability of success associated with the path is important because in presence of process uncertainty, there is an ensemble of paths between q_I and q_G and the robot may not succeed with probability 1.

The proposed solution methodology should be applicable to continuous state/ control spaces, high dimensional configuration spaces, and to any general dynamical system given by:

$$\dot{x} = f(x) + g(x)u + g'(x)w; \tag{1.1}$$

where u represents the control and w the "process noise/ uncertainty" due to incorrect modeling of the system dynamics or errors in the control input.

This problem will be solved in two stages : 1) a general solution methodology will be developed to solve the motion planning problem for a general dynamical system as stated above, and 2) improve the efficiency of the proposed solution methodology in (1) to address dynamical system having high dimensional state spaces.

1.2.2 Problem 2

Given a stochastic map with static obstacle probabilities, a system of N heterogeneous robots each equipped with the perfect state sensors, the initial configurations ($\boldsymbol{q}_I$) of all the robots, a set of m final goal configurations ($\boldsymbol{q}_G$), and a priori specified

minimum required success probability (p_{min}) in the domain, to solve, the motion planning problem for the set of robots in the presence of process uncertainty such that at least one robot visits each of the goal locations.

The aim is to generalize the solution of **Problem 1** to **Problem 2**. In this multi-agent scenario solving the problem involves generating control sequences, for each of the agent in the multi-agent system (MAS), $\boldsymbol{\pi} = \{\pi(1), \ldots, \pi(N)\}$, where $\pi(i), i = 1, \cdots, N$ is the control sequence for the ith agent. The control sequence for an agent will be defined as stated in the **Problem 1**. The final paths for each robot will be associated with a $p_s > p_{min}$ for the environment. This problem involves two additional sub-problems apart from the basic single agent motion planning problem : 1) *routing problem*, i.e. which agent should go to which configuration, and 2) *collision avoidance* in between agents.

The N agents considered here are teams of homogeneous and heterogeneous agents, i.e. the agents might have the different capabilities and different governing dynamics. This general scenario would incur a heavy computational burden due to reasons which will be discussed at a later stage of this dissertation.

1.3 Previous Work

Motion planning of robots while avoiding obstacles in the workspace has been an active area of research for the last several decades in the robotics and artificial intelligence community. Classical motion planning can roughly be divided into the following three different deterministic approaches [13]: 1) cell decomposition; 2) roadmaps; and 3) potential field methods. In potential field methods a collision free trajectory is generated by the robot according to "forces" defined as the negative gradient of a potential function. The cell decomposition and roadmap techniques are deterministic methods, because the environment of the robot is sampled or discretized in a deterministic manner. However, the problems are PSPACE-hard [2], and to circumvent this computational complexity, randomized sampling-based meth-

ods known as PRMs were introduced [3, 14]. PRM techniques usually do not take the dynamics of the robotic platform into account, and this case can lead to serious performance issues. To address these issues, RRT was introduced as a randomized sampling based planner that takes into account the dynamics of the mobile robot [2], [10] while building a tree of dynamically feasible trajectories in the free space of the robot.

The randomized PRM and RRT techniques have resulted in the solution of motion-planning problems in very high-dimensional state spaces, which were hitherto unsolvable using deterministic motion-planning techniques. However, both PRM and RRT are open-loop planners designed for perfectly known robot models/ workspaces, and our primary motivation in this work is to generalize these two techniques to generate feedback motion planners that are robust to uncertainties in the robot motion model and the map.

Furthermore, the robot motion planning problem can be formulated as an MDP if the uncertainties in the robot model and the environment are modeled probabilistically. However, MDPs are virtually intractable for anything but small to moderate state spaces, because they are subject to the famous "curse of dimensionality" [10]. In particular, it is nearly impossible to solve these problems in continuous state and control spaces even without constraints. In the presence of constraints, there are no structured techniques for accomplishing the planning. One approach to resolving the issue of dimensionality is through the use of hierarchical methods, an approach that is seen in most biological systems. A variety of methods for solving large MDPs in a hierarchical model-free manner have been developed, and the field of research is known as hierarchical reinforcement learning (RL) [11, 12].

These methods, instead of taking actions, invoke policies or options at each state, which continue until termination. 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 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, following issues are key in the formulation and solution of an SMDP : 1) how the landmark states are chosen; 2) how the local options are designed; and 3) how the generalized cost and transition probabilities of the options are estimated. The model-free techniques estimate the control without estimating the SMDP parameters through simulation or online training. However, questions 1 and 2 are not addressed in these techniques. In this work we will model the motion planning problem as a SMDP and will answer the questions posed above.

Further in this section we would like to discuss work closely related to solving motion planning problem in presence of stochastic maps and process uncertainty.

1.3.1 Related Work : Stochastic/ Uncertain Maps

In this section, we review research related to map uncertainty.

In [15], the need to plan on uncertain maps is addressed. They propose an *uncertainty roadmap*, where they maintain an upper and lower bound on the map probabilities and refine them incrementally as needed.

In [16], a sampling based motion planner was proposed with sensing uncertainty built into the planning process. This is an utility guided planner and they refined the uncertain map model using sensing actions.

In [17], an algorithm is proposed to compute motion plans that are robust to uncertain maps, which is an extension of PRM. The map uncertainty is evaluated using a feature-based EKF algorithm. Monte Carlo simulations are done for the open-loop local planner to detect collision and uses A^* search over the roadmap. The results shown in the work shows high failure rate for a 3-DOF mobile robot with uncertain maps.

In [18], a *particle RRT* (pRRT) algorithm is proposed which can deal with uncertainty in the model. The RRT is extended using using particle based techniques,

i.e. each extension is simulated multiple times under various likely conditions of the environment. The likelihood of a path involves simulating particles with uncertainty in domain. The work especially dealt with uncertainty in parameters used to define the workspace.

1.3.2 Related Work : Process Uncertainty in Dynamical Models

In this section, we consider research that has tried to account for process uncertainty in the planning.

In [19], a medial axis based PRM is proposed, which generates trajectories that are robust to modeling errors since samples on the medial axis of the plane maximize their distance from obstacles, however they do not explicitly consider uncertainty in the PRM.

In [20], a variant of PRM is proposed called the *belief roadmap* (BRM), which solves a POMDP and uses a Kalman filter based estimator. This work does not account for process uncertainty.

In [21], a motion planning algorithm to deal with process uncertainty for nonholonomic dynamical systems is proposed. Obstacle location uncertainty are solved using a minimum clearance approach. They pose the overall problem as an MDP, by discretizing the space as a grid and solve it using DP.

In [22], a *stochastic motion roadmap* is proposed to deal with process uncertainty. As a variant of PRM, they sample the configurations and consider discrete actions at each state, and transition probabilities are calculated. Using these transition probabilities they search the roadmap using DP and also carry the notion of a probability of success. Hence, they do not consider stochastic maps and the control space is discrete.

In [23], a planning algorithms is proposed to account for uncertainty in the dynamics of the vehicle. They propose a hierarchical planning approach, characterize noise as a function of controller type, terrain type and control input, introduce er-

ror dynamics, assume line follower, and search on the discretized free space using A* based- ARA* algorithm. The notion of motion uncertainty while following a trajectory in a corridor is calculated using the distance from the obstacles.

In [24], the problem of planning paths guaranteed to be safe in the presence of boundedness in process and sensing uncertainty is addressed. They propose a RRT-based algorithm, *set-RRT*, which uses set configurations. They attempt to solve the sensing uncertainty problem and propose to use proprioceptive sensors instead of exteroceptive sensors and rely on prediction to ensure collision avoidance. Not considering exteroceptive sensors leads to unbounded growth of belief uncertainty thereby leading to severe performance degradation and has been noted in [20].

In [25], a technique is proposed to account for uncertainties in the motion primitives used by a maneuver automaton. The framework of a maneuver automaton and a dynamic programming formulation were extended to explicitly account for uncertainty in each of the motion primitives. The motion primitives considered were trim conditions and maneuvers and uncertainty was considered in trim parameters and maneuver displacement and duration.

In [26], uncertainty was considered in sensing, localization and mapping in the motion planning problem. They propose to use RRT along with simulated particle based SLAM algorithm to solve this problem. They estimate the collision likelihood using the particle filter based framework.

It is worth mentioning that [27] introduced the notion of landmark based approach to solve motion planning problem under uncertainty. They introduced landmarks as a subset of robot's configuration space where position sensing and motion control are perfect, outside which the sensing is null and control is imperfect. They propose landmark design and an approach to solve the problem using geometrical analysis. They used grid-based motion planners.

1.3.3 Summary

The work mentioned above involved attempts to solve the motion planning problem in presence of uncertain maps, process uncertainty and sensing uncertainty. This dissertation addresses motion planning problems in presence of process uncertainty and stochastic maps.

Uncertain maps are dealt in various fashion including mapping using sensing [16], refinement of lower and upper bounds in map model [15], Monte Carlo simulations of local planner [17] and simulating multiple times under various likely condition of the map [18]. The work in this dissertation uses *occupancy grid* based stochastic map, which carry obstacle occupancy probabilities in the grids. We use the map probabilities while simulating "lowest level" controller in between states to determine the success probability of a path.

Process uncertainty in the robot motion model has been tackled in various fashion. [19]'s medial axis method, does not include process uncertainty explicitly, lacks applications to nonholonomic and non-linear dynamical systems. [21] attempts to solve the underlying MDP using grid based methods, which do not scale to large dimensional state spaces. [23] assumes an admissible path and follows it using control laws involving error dynamics and thus constitutes a localized controller. [24] ensures collision avoidance due to process uncertainty by performing prediction using box sets of configurations, which may not be easy to compute, and uses an open loop planner, which is not robust to uncertainty. [25] accounts for uncertainties in motion primitives of a maneuver automaton, which is a simplification of a complicated dynamical system and assumes a finite collection of motion primitives. [26] attempts to solve the planning problem using particle RRT, using open loop planners and particles carrying the history of motion. And finally [22] does not account for uncertain maps, and the control space is discrete and known.

The work mentioned covers the current state of work related to uncertain maps and process uncertainty in the motion model. Apart from the shortcomings men-

tioned with individual work, some of the major issues that are not addressed are as follows:

- A structured way of handling process uncertainty as well as map uncertainty has not been developed. A way of incorporating process uncertainty into the planning stage has been proposed in [22], but a robust methodology of handling process uncertainty is not demonstrated, as they used open-loop planners. They used discrete controls and map uncertainty was not addressed. Hence a robust methodology of incorporating process and map uncertainty in the planning stage of the motion planning algorithm has not been addressed.

- In order to solve motion planning problem in presence of uncertainty, an MDP has to be solved in continuous state and control spaces, none of the work mentioned has solved a suitably posed MDP in continuous state and control spaces.

- A general framework for application to robotic motion planning problem in high-dimensional state space under uncertainty has not been demonstrated.

- No performance guarantees have been provided for the proposed algorithms. In [22], the error in approximation of the generated probabilities has been addressed with reference to Voronoi cells, but no performance guarantees of the proposed algorithms is given.

This work would address the above mentioned open issues related to algorithms attempting to solve motion planning in presence of process uncertainty and map uncertainty. The proposed solution methodology will present a structured way of addressing process and map uncertainty, incorporating the uncertainties in a robust fashion in the planning stage, addressing the motion planning in continuous state and control spaces and provide performance guarantees. We also demonstrate the application of proposed methodology to high-dimensional state spaces. The following section will list the contributions of this work.

1.4 Dissertation Contributions

We have now set the basic foundation for the work in this dissertation. We seek to solve the robotic motion planning problem in presence of process uncertainty and stochastic maps with obstacles/ constraints, for a high-dimensional configuration space, non-linear governing dynamics, and continuous state/ control spaces. The issues with the current level of work with respect to motion planning under process and map uncertainty has been listed above in subsection 1.3.3, we would like to address the shortcomings in this work.

We pose the motion planning problem as an MDP. Furthermore, to incorporate continuous state/ control spaces with constraints, we convert the high dimensional MDP in a hierarchical fashion into a discrete state/ control SMDP, thereby solving the MDPs at a finite number of "distinguished states".

1.4.1 Contribution 1 : Generalized PRM (GPRM) & Generalized RRT (GRRT)

We generalize the *probabilistic roadmaps* (PRM) and *rapidly exploring random trees* (RRTs) for deterministic robotic motion planning such that the topological graph construction incorporates "process uncertainty". This topological graph construction has the randomly sampled "landmark states" as the vertices, and use feedback controllers to connect vertices resulting in transition costs/ probabilities.

Traditionally PRMs generate a nominal track which is then tracked in presence of disturbances using a local feedback control and/ or estimator. In our approach, the *generalized PRM* (GPRM), incorporates the feedback controllers into the topological graph (i.e. roadmaps in PRM) construction phase. Posing the motion planning problem as a path query, with a desired probability of success from an initial landmark to a final landmark, that is solved on the developed roadmap using stochastic DP resulting in a computationally tractable solution technique with provable performance guarantees.

The RRTs were introduced as a randomized sampling based motion planner, that takes into account the dynamics of the mobile robot while building a tree of dynamically feasible trajectories in the free configuration space of the robot. In practice, in presence of disturbances, the nominal trajectory is tracked using feedback controllers. In our approach of developing the *generalized RRT* (GRRT), the dynamically feasible trajectories incorporate the feedback controllers while expanding the tree. Incorporating the need for a trajectory to have a desired probability of success in a domain, leads to a modification of the "tree expansion step" when compared to the traditional RRTs.

A formal analysis of the generalized sampling-based planners (GPRM and GRRT), and formal proof of the probabilistic completeness of these planners is presented in this work.

A preliminary version of this work has been published in *IEEE International Conference on Systems, Man and Cybernatics*, 2009 (IEEE SMC '09) [28] and a journal version has appeared in *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 2011 [29].

1.4.2 Contribution 2 : Adaptive GPRM (AGPRM)

In order to increase the efficiency of the algorithm, an *adaptive sampling technique* is proposed for the GPRM, *adaptive GPRM* (AGPRM). Intelligent sampling in these randomized sampling-based framework can result in large speedups when compared to naive uniform sampling, while expanding the roadmap. We propose to use the information of transition probabilities, encoded in and unique to these generalized planners, and bias sampling to improve the efficiency of sampling, and increase the overall success probability of GPRM.

A preliminary version of this work has been published in the 49^{th} *IEEE Conference on Decision and Control*, 2010 (IEEE CDC '10) [30] and a journal version has

been accepted and will appear in *Journal of Control Theory and Application, Special Issue on Approximate Dynamic Programming*, 2010.

1.4.3 Contribution 3 : Multi-agent AGPRM (MAGPRM)

We propose the *multi-agent GPRM* (MAGPRM) to solve the multi-agent motion planning problem in presence of process uncertainty and stochastic maps, using the GPRM in conjunction with a *multiple traveling salesman problem* (MTSP) solution methodology. Assuming *partial observability* between agents, i.e. an agent is aware of only its neighbors, we propose to solve the *routing problem* discussed in **Problem 2** using *passive co-ordination* by a well proven MTSP solution methodology [31], wherein the costs of the MTSP problem are from the GPRM, and hence solve the multi-agent motion planning problem in the presence of uncertainty.

To summarize, we develop a hierarchical generalized sampling-based motion planners to solve the robotic motion planning problem under uncertainty, in presence of constraints, high-dimensional configuration space, and continuous state/ control spaces. Further, we will generalize our proposed solution methodology to the multi-agent scenario and propose an extended algorithm based on passive co-ordination using an existing MTSP solution methodology in conjunction with GPRM, to solve the multi-agent motion planning problem under uncertainty in continuous state/control spaces.

1.5 Dissertation Overview

The remainder of the dissertation is organized as follows:

Section 2 : In this section we present some basic background material related to *Markov decision process* (MDP), *dynamic programming* (DP) and *randomized sampling based algorithm*. We also review *sequential decision making*, and randomized algorithms : *probabilistic roadmaps* (PRM) and *rapidly exploring random*

trees (RRTs). We will needs these basic theories to build up our proposed solution methodology.

Section 3 : In this section, we discuss the need of generalized sampling based planners which can incorporate process uncertainty and stochastic maps during the design phase for solving the robotic motion planning problem under uncertainty. We develop the algorithms *generalized PRM* (GPRM) and *generalized RRT* (GRRT) building upon on the basic PRM and RRT algorithms respectively. We give the algorithms of the two proposed sampling based planners. We present the simulation results of application of these planners on idealized point robot and a nonholonomic Dubin's car model, and finally we discuss the results and what we achieved by the proposed planners.

Section 4 : In this section, we discuss the need of improving the efficiency of the proposed planners and ways of achieving it by using intelligent sampling. We present a novel adaptive sampling methodology unique to these generalized sampling-based motion planners (especially GPRM), named the *adaptive GPRM* (AGPRM). We give a detailed break down of the proposed methodology and give the algorithms. We present results of application of AGPRM on idealized point robots and high-dimensional n-link manipulators. We present a comparison AGPRM with a naive uniform sampling based GPRM and discuss the results.

Section 5 : In this section, we address the problem of solving the motion planning problem under uncertainty for a multi-agent system. We briefly present the differences involved in solving a multi-agent scenario compared to solving a single-agent scenario. We attempt to model the multi-agent scenario as a *routing problem* coupled with a single-agent motion planning problem. We pose the "routing problem" as solving a *multiple traveling salesman problem* (MTSP). We propose to solve the multi-agent system motion planning problem by solving the MTSP, using an existing solution technique, in conjunction of GPRM. Furthermore, we discuss inter-agent

collision due to moving agents, and list the *inter-agent collision avoidance* module requirement and propose a solution methodology for this problem.

Section 7 : In this section, we present the conclusion and discussions related to the problems proposed to solve in this work and the corresponding solution methodologies and their achievements. We outline the contributions and discuss possible future extensions of the work.

Appendix 1 : In this appendix, we formally analyze the generalized sampling based motion planners proposed using *Markov chains* and give a formal proof of probabilistic completeness of these planners.

Appendix 2 : In this appendix, we analyze the proposed collision detection and avoidance module for the multi-agent motion planning problem. A formal proof is given that this module will ensure the paths of the agents will be inter-agent collision free.

2. OVERVIEW OF TECHNICAL APPROACH

2.1 Markov Decision Process

A Markov decision process (MDP) is a mathematical framework for sequential decision making problems in stochastic domains [6].

2.1.1 Sequential Decision Making

A finite, discrete sequential decision-making problem can be specified using the following parameters:

- A discrete time step t
- A finite set of environment states $\boldsymbol{X}$ and a state $\boldsymbol{x}^t \in \boldsymbol{X}$ describes the state of the world at time step t
- A finite set of actions $\mathcal{A}$, and $a^t \in \mathcal{A}$
- A finite set of observations Ω and $o^t \in \Omega$ provides the agent with the information about the current state $\boldsymbol{x}^t$
- A state transition function $P : \boldsymbol{X} \times \mathcal{A} \times \boldsymbol{X} \to [0, 1]$ which gives the transition probability $p(\boldsymbol{x}^t | \boldsymbol{x}^{t-1}, a^{t-1})$ that the system moves to state $\boldsymbol{x}^t$ when the action a^{t-1} is performed in state $\boldsymbol{x}^{t-1}$.
- An observation function $O : \boldsymbol{X} \times \mathcal{A} \times \Omega \to [0, 1]$ which defines the probability $p(o^t | \boldsymbol{x}^t, a^{t-1})$ the agent perceives observation o^t in state $\boldsymbol{x}^t$ when action a^{t-1} was performed in the previous time step.
- A reward function $R : \boldsymbol{x} \times \mathcal{A} \to \mathbb{R}$, which provides the agent with a reward $r^{t+1} = R(\boldsymbol{x}^t, a^t)$ based on the action a^t taken in state $\boldsymbol{x}^t$.

A common assumption is that the environment has the Markov property, and hence transition probabilities are given by $p(\boldsymbol{x}^t | \boldsymbol{x}^{t-1}, a^{t-1})$.

2.1.2 MDP

A *Markov decision process* (MDP) is a sequential decision-making problem in which the current state is fully observable to the agent. So an addition assumption that the set of observations equals $\Omega = \boldsymbol{x}$ and the only non-zero observation probability is $p(o^t = \boldsymbol{x}^t | \boldsymbol{x}^t, a^{t-1}) = 1$. Hence formally, a *Markov Decision Process* (MDP) $\mathcal{M}$ is defined as a 4-tuple $\mathcal{M} = (\boldsymbol{X}, \mathcal{A}, R, P)$ where : $\boldsymbol{X}$ is a finite set of $|\boldsymbol{X}| = N$ states; $\mathcal{A}$ is a finite set of actions; R is a reward function $R : \boldsymbol{X} \times A \mapsto \mathbb{R}$, such that $R(\boldsymbol{x}, a)$ represents the reward obtained by the agent in state $\boldsymbol{x}$ after taking action a; and P is a *Markovian transition model* where $P(\boldsymbol{x}' | \boldsymbol{x}, a)$ represents the probability of going from state $\boldsymbol{x}$ to state $\boldsymbol{x}'$ after taking action a. We assume that the rewards are bounded, that is, there exists R_{max} such that $R_{max} \geq |R(\boldsymbol{x}, a)|$, $\forall \boldsymbol{x}, a$.

2.2 Dynamic Programming

2.2.1 The Basic Problem

Given a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N-1 \tag{2.1}$$

where the state x_k is an element of a space S_k, the control u_k is an element of a space C_k, and the random "disturbance" w_k is an element of a space D_k.

The control u_k is constrained to take values in a given nonempty subset $U_k(x_k) \subset C_k$, which depends on the current state x_k; that is, $u_k \in U_k(x_k)$ for all $x_k \in S_k$ and k.

The random disturbance w_k is characterized by a probability distribution $P(\cdot | x_k, u_k)$ that may depend explicitly on x_k and u_k but not on values of prior disturbances $w_{k-1}, \ldots, w_0$.

We consider the class of policies (also called control laws) that consist of a sequence of functions

$$\pi = \{\mu_0, \ldots, \mu_{N-1}\} \tag{2.2}$$

where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Such policies will be called *admissible*.

Given an initial state x_0 and an admissible policy $\pi = \mu_0, \ldots, \mu_{N-1}$, the states x_k and disturbances w_k are random variables with distributions defined through the system equation

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \quad k = 0, 1, \ldots, N-1. \tag{2.3}$$

Thus, for given functions g_k,$k = 0, 1, \ldots, N$, the expected cost of π starting at x_0 is

$$J_\pi(x_0) = E\Big\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\Big\} \tag{2.4}$$

where the expectation is taken over the random variables, w_k and x_k. An optimal policy π^* is one that minimizes this cost; that is,

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0), \tag{2.5}$$

where Π is the set of admissible policies.

Note that the optimal policy π^* is associated with a fixed initial state x_0. However, an interesting aspect of the basic problem and of dynamic programming is that it is typically possible to find a policy π^* that is simultaneously optimal for all initial states.

The optimal cost depends on x_0 and is denoted by $J^*(x_0)$; that is,

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0). \tag{2.6}$$

It is useful to view J^* as a function that assigns to each initial state x_0 the optimal cost $J^*(x_0)$ and call it the *optimal cost function* or *optimal value function*.

2.2.2 Dynamic Programming Algorithm

The dynamic programming (DP) technique rests on a very simple idea, the *principle of optimality* [7].

Principle of Optimality

Let $\pi^* = \mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*$ be an optimal policy for the basis problem, and assume that when using π^*, a given state x_i occurs at time i with positive probability. Consider the subproblem whereby we are at x_i at time i and wish to minimize the "cost-to-go" from time i to time N

$$E\big\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\big\}.$$

Then the truncated policy $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu^* N - 1\}$ is optimal for this sub-problem.

The intuitive justification of the principle of optimality is very simple. If the truncated policy $\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*$ were not optimal as stated, we would be able to reduce the cost further by switching to an optimal policy for the subproblem once we reach x_i.

Dynamic Programming (DP)

The principle elements of a problem in DP are:

1. A discrete-time dynamic system whose state transition depends on a control. Assume n states and at state i the control must be chosen from a given finite set $U(i)$ and the choice of control u specifies the transition probability $p_{ij}(u)$ to the next state j.

2. A cost that accumulates additively over time and depends on the visited states and the controls chosen. At the kth transition, we incur a cost $\alpha^k g(i, u, j)$, where g is a given function, and α is a scalar with $0 < \alpha \leq 1$, called the *discount factor*.

We are interested in policies, that is sequence $\pi = \{\mu_0, \mu_1, \ldots\}$ where each μ_k is a function mapping states into controls with $\mu_k(i) \in U(i)$ for all states i. Let us denote by i_k the state at time k. Once a policy π is fixed, the sequence of states i_k becomes a Markov chain with transition probabilities

$$P(i_{k+1} = j | i_k = i) = p_{ij}(\mu_k(i)). \tag{2.7}$$

We can distinguish between *finite horizon problems*, where the cost accumulates over a finite number of stages, say N, and *infinite horizon problems*, where the cost accumulates indefinitely. In N-stage problems the expected cost of a policy π, starting from an initial state i, is

$$J_N^\pi(i) = E\left[\alpha^N G(i_N) + \sum_{k=0}^{N-1} \alpha^k g(i_k, \mu_k(i_k), i_{k+1}) \middle| i_0 = i\right] \tag{2.8}$$

where $\alpha^N G(i_N)$ is a terminal cost for ending up with final state i_N, and the expected value is taken with respect to the probability distribution of the Markov chain $\{i_0, i_1, \ldots, i_N\}$. The distribution depends on the initial state i_0 and the pol-

icy π, as discussed earlier. The optimal N-stage cost-to-go starting from state i, is denoted by $J_N^*(i)$; that is,

$$J_N^*(i) = \min_{\pi} J_N^{\pi}(i). \tag{2.9}$$

The costs $J_N^*(i)$, $i = 0, \ldots, n$, can be viewed as the components of a vector J_N^*, which is referred to as the N-stage optimal cost-to-go vector.

In infinite horizon problems, the total expected cost starting from an initial state i and using a policy $\pi = \mu_0, \mu_1, \ldots$ is

$$J^{\pi}(i) = \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \alpha^k g(i_k, \mu_k(i_k), i_{k+1}) \middle| i_0 = i\right] \tag{2.10}$$

The optimal cost-to-go starting from state i is denoted by $J^*(i)$ that is,

$$J^*(i) = \min_{\pi} J^{\pi}(i). \tag{2.11}$$

The costs $J^*(i)$, $i = 0, \ldots, n$, as the components of a vector J^*, referred to as the *optimal cost-to-go vector.*

The DP algorithm states that the optimal control choice with k stages to go must minimize the sum of the expected present stage cost and expected optimal cost $J_{k-1}^*(j)$ with $k-1$ stages to go, appropriately discounted by α; that is,

$$J_k^*(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J_{k-1}^*(j)), \quad i = 1, \ldots, n. \tag{2.12}$$

Infinite horizon problems represent a reasonable approximation of problems involving a finite but very large number of stages. Since the infinite horizon cost of a given policy is the limit of the corresponding N-stage costs as $N \to \infty$, the following holds:

1. The optimal infinite horizon cost-to-go is the limit of the corresponding N-stage optimal cost-to-go as $N \to \infty$, that is

$$J^*(i) = \lim_{N\to\infty} J_N^*(i) \tag{2.13}$$

for all states i.

2. The following limiting form of the DP algorithm holds for all states i

$$J^*(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J^*(j)), \quad i = 1, \ldots, n. \tag{2.14}$$

This is a system of equations (one equation per state), which has a solution the optimal costs-to-go for all the states. This is referred to as *Bellman's equation*.

3. If $\mu(i)$ attains the minimum in the right hand side of the Bellman's equation for each i, the stationary policy μ is optimal.

2.3 Configuration Space

The state space of a robot/dynamical system for motion planning is a set of possible transformations that could be applied to it. This is referred as the *configuration space*, or *C-Space* [2]. With n degrees of freedom, the set of transformations is mostly a manifold of dimension n, and is referred to as the configuration space. Hence in order to solve a motion planning problem, algorithms must search in this C-space.

In presence of obstacles, the configurations that either collide, cause the robot to collide with obstacles or have some specified links of the robot to collide with each other, which need to be removed from the C-space. The removed part of $\mathcal{C}$ is referred to as obstacle region. A motion planning algorithm must search for a path in this remaining space from an initial configuration to a goal configuration.

Let $\mathcal{W}$ denote the workspace, which could be either 2D or 3D. Let $q \in \mathcal{C}$ denote the *configuration* of $\mathcal{A}$, where $\mathcal{A}$ represents the transformed configurations of a robot in the given workspace $\mathcal{W}$, and the obstacle region be denoted by $\mathcal{O}$, which is also $\mathcal{O} \in \mathcal{W}$. The *obstacle region*, $\mathcal{C}_{obs} \subseteq \mathcal{C}$, is given by:

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}, \tag{2.15}$$

which is the set of all configurations, q, at which $\mathcal{A}(q)$, the transformed robot intersects the obstacle region, $\mathcal{O}$.

The remaining configurations are called the *free space*, which is defined by $\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}$. Hence this region denotes all the possible configurations of the robot which are safe, and motion planning has to be solved by searching for a path in this space.

2.4 Sampling-Based Motion Planning Algorithm

In the book by Lavalle [2], sampling based motion planning algorithms were discussed in detail.

One of the major philosophies of addressing the motion planning problem, is the *sampling-based motion planning*. The main idea is to avoid the explicit construction of the obstacle space, i.e., the $\mathcal{C}_{obs}$, and instead perform a search that check/ probe the C-space with a sampling scheme. The probing is enabled using a collision detection module. This enables the development of motion planning algorithms that do not depend on any particular geometric models of the work space. This general philosophy has been instrumental in solving, in these recent years, problems ranging from robotics, manufacturing to applications in biology. Problems solved using these sampling based algorithms were practically impossible to solve, when trying to solve using techniques which explicitly represent $\mathcal{C}_{obs}$.

The algorithms involving sampling based methods suffer from the drawback that they can give weaker guarantees that the problem will be solved. An algorithm is

considered *complete* given any input it reports perfectly whether there exist is a solution in a finite amount of time. If solution exists, it must return it in finite time. Unfortunately sampling based motion planning cannot achieve completeness but they follow a weaker notion of completeness. The notion of denseness is important, which means that the samples come arbitrarily close to any configuration as the number of iterations or as the number of samples tends to infinity. A deterministic approach in which one samples densely is called *resolution complete*. Many of the sampling based techniques are based on random sampling on configurations, which is dense with probability one. This leads to algorithms that will be *probabilistically complete*, which means with enough samples, the probability that the algorithm will find an existing solution converges to one.

2.4.1 Randomized Sampling Based Motion Planners

Randomized Sampling based methods were introduced to provide approximate solutions, while avoiding the prohibitive cost of computing the exact representation of the free space. Randomized motion planners, such as the Probabilistic Roadmap (PRM) [3] and the Rapidly Exploring Randomized Tree (RRT) [4] have been very successful in solving planning problems for robots with many degress of freedom, problems that were previously considered intractable. However, these algorithms depend on having a complete and accurate model of the world.

It has been shown experimentally that when problem dimensionality increases, randomized sampling and planning methods fare better than deterministic methods with respect to computational complexity [32].

Probabilistic Roadmaps (PRM)

When a single start-goal pair is provided to the planning algorithm, it is called the *single query* version of the motion planning problem. Whereas, in a motion plan-

ning problem suppose numerous start-goal queries are provided, while keeping the robot motion model and the locations of the obstacles fixed, then this is called *multiple query* version. A general framework was introduced as *probabilistic roadmaps* (PRMs), in which the end-goal is to build/construct a topological graph called a *roadmap*, which will efficiently solve the multiple start-goal queries. The probabilistic aspect is due to the randomization of sampling configuration in the C-space.

Let $\mathcal{G}(V, E)$ represent a topological graph in which V is a set of vertices and E is the set of paths that map into $\mathcal{C}_{free}$. The algorithm has been discussed and listed in the Planning Algoirthms book by Lavalle [4]. There are two phases of computation:

Preprocessing Phase During this phase, need to build $\mathcal{G}$, which is also called a roadmap and it should be accessible from every part of $\mathcal{C}_{free}$. In this phase generate collision free samples, connect a sampled configuration to neighboring samples using a local planner.

Algorithm 2.1: PRM : Construction/ Preprocessing Phase

```
1 G.init(), i ← 0;
2 while i < N do
3     if α(i) ∈ C_free then
4         G.add_vertex(α(i)),i ← i + 1;
5         foreach q ∈ NEIGHBORHOOD(α(i), G) do
6             if not G.same_component(α(i), q) and CONNECT(α(i), q) then
7                 G.add_vertex(α(i), q);
```

Query Phase In this phase, a pair of configurations, q_I (start) and q_G (goal), is given. Each configuration must be able to connect easily to $\mathcal{G}$ using a local planner. Once connected, need to perform a discrete search over the roadmap to obtain a sequence of edges that forms a path from the start, q_I to goal, q_G. In this phase it

is assumed that $\mathcal{G}$ is complete enough to answer any number of incoming queries, i.e., the roadmap is resolution complete. After the query comes, the q_I and q_G are connected successfully to existing vertices in the existing roadmap $\mathcal{G}$, and then a search over this updated roadmap is performed for the path that connects the vertex q_I to the vertex q_G. The path in the graph/roadmap corresponds to a path in $\mathcal{C}_{free}$, which hence will be the solution to the query.

Researchers have analyzed the performance of this sampling-based roadmaps algorithm. Narrow channels in $\mathcal{C}_{free}$ poses a challenging planning problem for this sampling-based roadmap algorithms. They have provided metrics to understand the difficulty level of these planners, but these metrics are difficult to apply to any particular problem to determine whether the proposed algorithm will perform well or not.

Rapidly Exploring Random Trees(RRT)

For a single query case, the faster you search the solution the better it is. RRTs are built in an increment fashion in a way that quickly reduces the connecting distance of a randomly-chosen point (a random configuration) to the existing tree. RRTs are suited particularly for motion planning problems, that involve obstacles and differential constraints (nonholonomic and kinodynamic) [4]. The idea is to incrementally build a tree, on which a solution can be searched, that gradually improves the resolution needed for the domain but it does not need to set/provide any such resolution parameters, explicitly. As the number of such sampled random points tends to infinity, the tree densely covers the domain of work space. As given in the publication

in which RRT was introduced [4], the algorithm is :

Algorithm 2.2: RRT Algorithm

Data: initial configuration q_{init}, number of vertices to be constructed k, increment allowed Δq

```
1 G.init(q_init);
2 for i = k do
3     q_rand ← RAND_CONF();
4     q_near ← NEAREST_VERTEX(q_rand, G);
5     q_new ← NEW_CONF(q_near, Δq);
6     G.add_vertex(q_new);
7     G.add_edge(q_near, q_new);
```

The RAND_CONF() function samples random configurations in $\mathcal{C}_{free}$, by using a collision detection module to reject samples which intersect with the obstacles or in $\mathcal{C}obs$. The function NEAREST_VERTEX() gives the configuration on the tree which is near to this new random sample generated. The function NEW_CONF() generates a new configuration q_{new}, by traveling from q_{near} an incremental distance Δq, in the direction of this randomly sampled configuration, q_{rand}. If any differential constraints exists for the robot motion model, then these new configurations can be generated using numerical integration techniques. Finally in the existing tree a new vertex, q_{new} and a new edge is added from q_{near} to q_{new}. For a motion planning problem the RRT can be biased towards the goal configuration.

3. GENERALIZED SAMPLING-BASED MOTION PLANNERS*

3.1 Introduction

In this section*, 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 generalized to take into account uncertainties in the robot motion model and in the obstacle locations in the map. These techniques result in hybrid hierarchical feedback planners in the state space of the robot. The algorithms are analyzed to show that they are probabilistic complete, i.e. they generate hybrid planners with a guaranteed minimum probability of success if such a planner exists. Experiments are performed on an idealized planer holonomic point robot and on a nonholonomic unicycle robot, and results show that the performance of the generalized planners, in terms of their probability of success, is significantly improved compared to the traditional techniques.

Motion planning of robots while avoiding obstacles in the workspace has been an active area of research for the last several decades. Classical motion planning can roughly be divided into the following three different deterministic approaches [13]: 1) cell decomposition; 2) roadmaps; and 3) potential field methods. The cell decomposition and roadmap techniques are deterministic methods, because the environment of the robot is sampled or discretized in a deterministic manner. However, the problems are PSPACE-hard [2], and to circumvent this computational complexity, randomized sampling-based methods known as PRMs were introduced [3], [14]. PRM techniques usually do not take the dynamics of robotic platform into account, and this case can lead to serious performance issues. To address these issues, RRT was introduced as a randomized sampling based planner that takes into account the dynamics of the mobile robot [2], [10] while building a tree of dynamically feasible trajectories

*Reprinted with permission from "Generalized sampling-based motion planners", by S. Chakravorty and S. Kumar, 2011, *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 41(3):855 ©2011 IEEE

in the free space of the robot. The randomized PRM and RRT techniques have resulted in the solution of motion-planning problems in very high-dimensional state spaces, which were hitherto unsolvable using deterministic motion-planning techniques. However, both PRM and RRT are open-loop planners designed for perfectly known robot models/ workspaces, and our primary motivation in this dissertation is to generalize these two techniques to generate feedback motion planners that are robust to uncertainties in the robot motion model and the map. Because PRM and RRT helped solve motion planning problems in high dimensions, we expect that the generalized techniques-the generalized probabilistic roadmaps (GPRM) and the generalized rapidly exploring random tree (GRRT)-will help us solve feedback motion planning problems in high dimensional state spaces under uncertainty (in fact, using existing techniques, these problems can only be solved in low-dimensional or discrete state and control spaces). The GPRM and GRRT techniques are closely related to Markov decision process (MDPs), sequential composition (SC), and other generalized versions of the PRM. In the following discussion, we examine the relationship between our techniques and the following seemingly disparate planning techniques : 1)MDPs; 2)SC; and 3) other generalized PRMs.

The robot motion planning problem can be formulated as an MDP if the uncertainties in the robot and the environment are probabilistically modeled. However, MDPs are virtually intractable for anything but small to moderate state spaces, because they are subject to the famous "curse of dimensionality" [10]. In particular, it is nearly impossible to solve these problems in continuous state and control spaces even without constraints. In the presence of constraints, there are no well established techniques for accomplishing the planning. One approach to resolving the issue of dimensionality is through the use of hierarchical methods, an approach that is seen in most biological systems. A variety of methods for solving large MDPs in a hierarchical model-free manner have been developed, and the field of research is known as hierarchical reinforcement learning (RL) [11], [12]. These methods, in-

stead of taking actions, invoke policies or options at each state, which continue until termination. 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 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, these issues are key in the following formulation and solution of an SMDP : 1) how the landmark states are chosen; 2) how the local options are designed; and 3) how the cost of operation and transition probabilities of the options is estimated. The model-free techniques estimate the control without estimating the SMDP parameters through simulation or online training. However, questions 1 and 2 are not addressed in these techniques. We answer the aforementioned three questions by proposing GPRM, which can be interpreted as a principle technique for specifying an SMDP as follows: 1) randomizing the selection of the landmark states; 2) designing the local options using traditional feedback control system design techniques; and 3) evaluating the cost of operation and probability of success of the local options through Monte Carlo simulations. Our method is the only technique for stochastic control that is applicable to continuous state or action spaces in a computationally tractable manner compared to the majority of the existing MDP and SMDP solution techniques, which deal with finite-state and action-space problems such as grid-world navigation. Furthermore, our method can handle constraints in the state space of the robot, which, to the best of our knowledge, cannot be accomplished by any existing technique.

The methodology advocated in this dissertation for robot motion planning is related to the SC methods [33, 34, 35] for deterministic robotic systems. In these methods, a global control policy (e.g. for stabilization and tracking) is designed by concatenating local policies with smaller (local) domains of operation. Applications of these methods to robotic systems can be found in [36, 37, 38]. These papers 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. These methods form a covering of their workspace in a deterministic manner and attempt to cover the entire workspace of the robot with the domains of attractions of the local controllers. In contrast, our approach (GPRM) samples the workspace in a randomized manner and constructs an SMDP on the sampled landmarks to solve the higher level planning problem. Thus, it fundamentally differs from SC methods in the way that the higher level planner is constructed and its parameters are estimated. Moreover, our methods are particularly designed to handle systems with probabilistic models of uncertainty and proven to be robust to these uncertainties, which is not the case with SC methods.

There have been several attempts in the last few years to generalized PRM and RRT methodologies to handle robot motion and map uncertainties [17, 16, 18, 20, 15]. In [17], [16] and [15], various computationally efficient ways of generalizing PRMs to account for map uncertainty are devised. In [18] a method based on tree pruning is proposed to account for parametric uncertainty in the motion model. In [20], sensing uncertainty in linear systems is considered, and a sampling based method using covariance factorization is developed to solve the problem. All the aforementioned methods result in open-loop controllers and thus, in the presence of process uncertainty in the motion model, cannot be robust. Our generalized techniques result in hybrid feedback controllers and hence are robust to the process uncertainty in the robot motion models. Moreover, we provide performance guarantees for our techniques : if feasible, we prove that a minimum allowable probability of success is achieved as the number of samples increases.

The generalized techniques presented here can be interpreted as a unifying framework for the seemingly disparate planning techniques of PRMs, MDPs and SC. The contributions of this dissertation in this section are given as follows:

1. We develop the randomized hybrid hierarchical techniques GPRM and GRRT for the solution of constrained feedback motion planning problems with con-

tinuous state and control spaces under stochastic models of uncertainty, which generalizes the PRM and RRT algorithms for deterministic motion planning.

2. We rigorously establish performance guarantees in terms of a minimum desired probability of success of the feedback planners by analyzing the absorption probabilities of the underlying Markov chains into certain failure sets.

The techniques are extensively tested on holonomic and nonholonomic systems in several maps of varying degrees of difficulty to validate the theoretical performance guarantees. To the best of our knowledge, such solutions to constrained stochastic control problems in continuous state and action spaces is absent from the literature. A preliminary version of this work has been published in *IEEE International Conference on Systems, Man and Cybernatics*, 2009 (IEEE SMC '09) [28] and a journal version has appeared in *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 2011 [29].

3.2 Solution Approach : **Problem 1**

The robotic motion planning problem stated in **Problem 1** (refer subsection 1.2.1), can be formulated as a Markov decision process (MDP), if the uncertainties in the robot motion and environments can be modeled probabilistically. This MDP becomes infinite dimensional if considered in continuous state and control spaces. Using hierarchical methods this infinite dimensional MDP can be modeled as semi Markov decision process (SMDP), which involve solving the MDP at some "distinguished states" which we will call as *landmark states*.

A hierarchical-based generalized sampling-based motion planners will be developed (Figure 3.1) and details of the development will be discussed in section 3.3. These generalized sampling-based motion planners are generalization of *probabilistic roadmaps* (PRM) and *rapidly exploring random tree* (RRT) and are called *Generalized PRM* (GPRM) and *Generalized RRT* (GRRT) respectively.

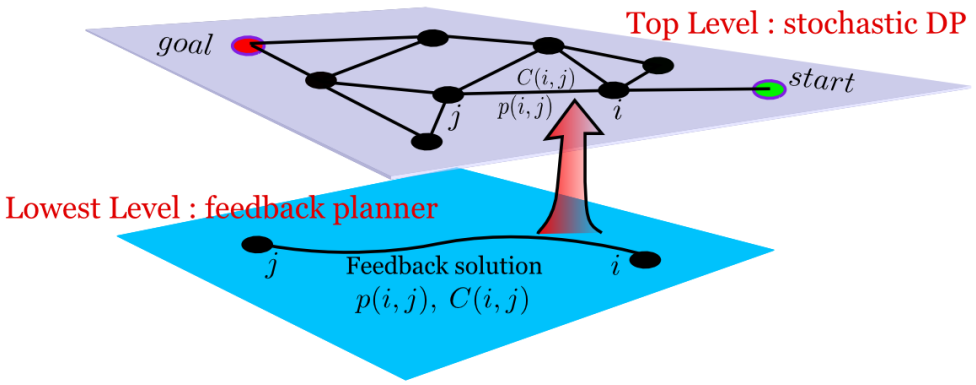


Fig. 3.1. Depicting Hierarchical Planning in Levels (for Single Agent)

The solution methodology proposed in these generalized sampling based motion planners involve :

- sample configurations (landmarks) from the free configuration space, the method of doing this differs in GPRM and GRRT ($\mathcal{C}_{free}$),
- find a feedback based control solution to transition of robot from one landmark state to another, (the lowest level planner)
- estimate the transition cost/ probabilities for each of these transitions by using the stochastic maps provided and Monte Carlo simulations.
- Generate a topological graph $\mathcal{G}(V, E)$ using the sampled landmarks (V) and the connections (E) made.
- Run stochastic dynamic programming (DP) (the top level planner), an optimization algorithm over the developed graph to get solution in GPRM
- Keep developing the graph till the start configuration is connected to the desired goal configuration in GRRT
- Ensure the probability of the final path achieved either in GPRM or GRRT, has probability of success p_s associated with it and it maintains $p_s > p_{min}$, where p_{min} is a priori specified for a given environment and is the minimum acceptable success probability for any path solution.

There is a collision detection module which ensures that the sampled configurations q are obstacle free i.e. $p(\mathcal{O}|q) = 0$, where $\mathcal{O}$ represents the obstacles and also the path (the edge on $\mathcal{G}$) generated by the lowest level planner (the feedback solution) in between two configurations or landmarks have a $p_{edge} > 0$. This collision detection module has the stochastic map provided as the input.

Furthermore, to develop efficient algorithm to address high dimensional dynamical systems, a novel adaptive sampling methodology is developed and discussed in

Section 4. The novel methodology developed has been extended to GPRM case and can be extended to GRRT but has not been developed and discussed. The adaptive sampling methodology is specific to the generalized sampling based motion planners developed because of the *transition probabilities information* encoded in the solution of GPRM and hence uses this information to improve the subsequent sampling to solve the motion planning problem efficiently.

3.3 Generalized Sampling-Based Motion Planners

In this section, we present the generalized sampling-based motion planners, GPRM and GRRT, which extend the traditional PRM and RRT algorithms to systems with uncertainty. In the following sections, we first formulate the uncertainty models, followed by a detailed description of the generalized algorithms.

3.3.1 Model

The generalized sampling-based algorithms require an uncertainty model for both the motion of the robot and a model for map uncertainty. In the following discussion, we outline the models that are used in this section of the dissertation.

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

$$\dot{x} = f(x) + g(x)u + h(x)w \tag{3.1}$$

where x represents the state of the robot, w represents the white noise perturbation, and u represents the control input to the robot. The aforementioned equation is a nonparametric model of uncertainty in the robot motion model and will be used throughout this dissertation for the lowest level control law designs.

We assume that the uncertainty in the map is specified through a binary occupancy value $p(O/y)$, i.e. the probability that there is a obstcle at the point y in the map. The occupancy values in the map can be considered the output from a mapping algorithm. However, in this dissertation, we shall not cover 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, e.g. $\mathcal{C}_{free}$. This case means that any state whose configuration is in $\mathcal{C}_{free}$ is safe. This condition, in turn, induces a free space in the state space of the robot, e.g. $\mathcal{X}_{free}$. 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 $\mathcal{X}_{free}$.

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

3.3.2 GPRM

In motion planning, the objective is to plan the path of a robot from a start state to an end state. PRM attempts to accomplish this condition by the following two approaches:

1. randomly sampling the state space of the robot, and

2. connecting every sampled point with its k-nearest neighbors using some local open-loop planner such as a straight line planner while checking for collisions with obstacles.

The result of 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; otherwise, more points are sampled in the state space of the robot until a graph that contains such a feasible path is found. In the case of systems with uncertainty, it may be impossible to find a path that succeeds with probability 1, and hence, we are interested in finding paths that have a success probability above a prescribed minimum threshold p_{min}. The algorithm is analyzed and later in this work shown to be probabilistic complete. Hence if there exist a path with probability of success greater than p_{min}, the algorithm will find it with probability 1. In situations when the algorithm is applied to maps with no possible solution, the algorithm will stop after a large number of iterations and return failure.

The pseudo-code for the GPRM algorithm is shown as follows. As shown in the pseudo-code, steps 2-3 and 5 are different from the traditional PRM algorithm. In the following sections, we discuss these steps of the algorithm in detail.

Algorithm 3.1: GPRM Algorithm

Data: the start state x_0, the goat state x_g, the minimum probability of success p_{min}

1 Initialize the GPRM with nodes x_0 and x_g;

2 **for** $p_s > p_{min}$ **do**

3 Sample equilibrium states in $\mathcal{X}_{free}$ probabilistically using a uniform distribution ;

4 Grow the GPRM by connecting every sampled state in the domain with it k-nearest neighbors using suitable obstacle-free feedback controllers;

5 Evaluate the cost of every connection in the resulting graph using Monte Carlo simulations;

6 Plan on the resulting graph using the evaluated edge cost from step 5;

7 Evaluate the probability of success p_s of the resulting path from step 6, and set $p_s = 0$ if there is no path.;

Step 4: 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 neighborhood 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 neighborhood of the point x_g. Then, the aforementioned case implies that the probability of the state of the robot $p(x(t))$ is mostly concentrated in the region Ω_{x_g} as $t \to \infty$.

Step 5: 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 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 condition can precisely be stated 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 in either landmark x_i or x_j due to 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 prespecified neighborhood of x_i, e.g. Ω_i, and turned off when the state of the robot enters some neighborhood of the node x_j, e.g. Ω_j, at which time the feedback controller, to get it to one of the neighboring nodes of x_j, is switched on (this situation is shown in Figure 3.2). Let one particular instance of a trajectory, e.g. the Nth instance, which goes from $\Omega_i \to \Omega_j$ under the feedback controller $u(:, x_j)$, be $x_0^{(N)}, \ldots, x_{t(N)}^{(N)}$, where $t(N)$ denotes the time that the controller terminates. This time $t(N)$ is stopping time and is a random variable, because it depends on the particular realization. The probability of success of the Nth realization is given by :

$$p_s^{ij,(N)} = (1 - p(O/x_0^{(N)})) \cdots (1 - p(O/x_{t(N)}^{(N)})) \tag{3.2}$$

where, as aforementioned, $p(O/x)$ 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 and time. Then, if we do repeated simulations, the probability of success and cost of the controller $u(:, x_g)$ in controlling the robot from x_i to x_j can be approximated as :

$$p_s^{ij} \approx \frac{1}{M} \sum_{N=1}^{M} p_s^{ij,(N)} \tag{3.3}$$

$$c_s^{ij} \approx \frac{1}{M} \sum_{N=1}^{M} c_s^{ij,(N)} \tag{3.4}$$

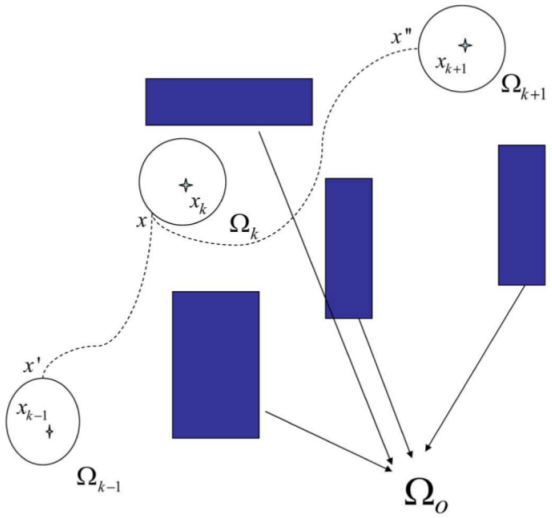


Fig. 3.2. Illustration of Robot Motion Under the GPRM Methodology

Due to law of large numbers, it follows that, as $M \to \infty$ the aforementioned estimates converge too 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 give by :

$$c^{ij} = p_s^{ij} c_s^{ij} + (1 - p_s^{ij}) c_F \tag{3.5}$$

where c_F is some heurestically defined, suitable high cost of failure. The aforementioned equation allows to evaluate the edge costs in the graph that is formed by connecting any node to it k-nearest neighbors.

Step 7 : In tradition 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 ensure 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 case has to be tested. Thus, once a minimum cost path is found on the graph according to the edge costs as previously defined, the probability of success of the individual segments of the path, which in turn, is from step 5. Thus, if the success probability is higher than the threshold, the planning problem is solved; otherwise, more points have to be sampled in the state space.

Remark 1. The feedback controller is designed for a workspace without any obstacles, because otherwise, the controller design is very complicated due to the constraints imposed on the robotic system by its workspace. The requirement of feedback controllers that stabilize a system to a given equilibrium point x_g is mild. In fact, for any fully actuated system, linearizing the system about the given equilibrium and designing a linear quadratic (LQ) controller for the linearized system results in such

a stabilizing controller, at least locally. For nonholonomic and underactuated systems, in general, such linearized techniques may not provide a stabilizing controller due to Brockett's theorem [39]. In that case, suitable nonlinear controllers may be designed to stabilize the system about a nominal trajectory, which can be obtained using optimal control techniques. In the example section, we use a dynamic feedback linearization-based controller to design stabilizing controllers about equilibrium configurations of a unicyle robot.

Remark 2. The analytical evaluation of the probability of success of an option is difficult, because it is equivalent to the "first passage time" problem for a stochastic nonlinear system. In general, Monte Carlo techniques, including sequential Monte Carlo techniques, are the most efficient method for evaluating such probabilities [40]. In this dissertation, we use simple Monte Carlo to evaluate these probabilities The time of execution of the options $t(\omega)$, where ω is a particular realization of the trajectory under the option, can, in general, be infinite, i.e. the option may take an infinite time to terminate. However, the expected time to terminate can be shown to be finite, $E[t] < \infty$ (refer Appendix 1).

3.3.3 GRRT

The traditional RRT algorithm attempts to connect a start point and an end point in the workspace of a robot by growing a tree using a random sampling as follows.

1. Randomly pick a point in the state space of the robot
2. Find the nearest node on the tree according to some pre-specified metric
3. Connect the nearest node on the tree to the sampled node using some local planner while checking for collision
4. Add the new node to the tree if the robot does not collide with obstacles

The tree is grown in this manner until a feasible path is found from the start point to the goal point. Due to uncertainty, it might not be possible to find a path that succeeds in connecting two points with probability 1. Hence, in the current scenario, we require that the path has a minimum pre-specified probability of success p_{min}.

We now present the generalized version of the RRT algorithm, i.e. GRRT. The pseudo-code for the algorithm is presented as follows (steps 2-4 of the algorithm are different from the traditional RRT algorithm, and in the following sections, we discuss the details about these differences, starting with step 3).

Algorithm 3.2: GRRT Algorithm

Data: Start state x_0, goal node x_g, minimum probability of success p_{min}

Input: Initialize tree with x_0, set $p(x_0) = 1$, set $N = 1$

1 **for** *tree reached goal node* x_g **do**

2 Generate node x_N at random (x_N is an equilibrium state in $\mathcal{X}_{free}$);

3 Connect x_N to the node x^* on the tree, using local feedback control, that satisfies

$$x^* = \arg\max_i p(x_i)p(x_i, x_N) \tag{3.6}$$

where $p(x_i, x_N)$ is the probability of successfully transitioning from $x_i \to x_N$ under the local feedback law;

4 Set $p(x_N) := p(x^*)p(x^*, x_N)$;

5 **if** $p(x_n) > p_{min}$ **then**

6 add node x_N to the tree with label $p(x_N)$ and set $N = N + 1$;

7 **else**

8 Goto step 2;

Step 3: The nodes (or, more precisely, the neighborhoods of the nodes) are connected by local feedback controllers that have been designed using control design

techniques and the probability of success of the controller evaluated as in the GPRM algorithm. The reason that we chose the node as in (refer Equation 3.6) has to do with the proof of completeness of the resulting algorithm (refer Appendix 1). 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 (Equation 3.6) is the best node in terms of the probability of success of transitioning from $x_0 \rightarrow x^* \rightarrow x_N$, where, as aforementioned, x_0 is the root node.

Step 4 and 5: Step 4 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 that the threshold of p_{min}. Hence, we include the tree pruning in step 5. 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 $x_0, x_1, \ldots, x_K$ is given by $p(x_0, x_1)p(x_1, x_2) \cdots p(x_{m-1}, x_m) = p(x_{K-1})p(x_{K-1}, x_K)$ according to the labeling convention that we have used, where, as aforementioned, $p(x)$ represents the probability of successfully transitioning from root node to node x.

GRRT algorithm is also analyzed and later in this work shown to be probabilistic complete. Hence if there exist a path with probability of success greater than p_{min}, the algorithm will find it with probability 1. In situations when the algorithm is applied to maps with no possible solution, the algorithm will stop after a large number of iterations and return failure.

3.4 Numerical Experiments

In this section, we will detail the application of the generalized sampling-based motion to a fully actuated holonomic point robot and an underactuated nonholonomic unicycle model.

3.4.1 Fully Actuated Point Robot

In this section, we apply the generalized planners developed in the previous section to an idealized holonomic point robot. We will deal with a planner robotic system, but the extension to a 3-D system is quite straightforward. Note that the higher level planning algorithms do not change with the robot model and only the lowest level controllers change with different or complicated robot models. The design of local point-to-point feedback controllers for holonomic systems such the fully actuated point robot considered here is quite trivial and can be done using the standard LQ control theory [1].

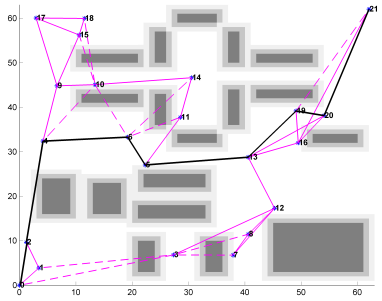
The robot motion model was assumed to be :

$$\ddot{q} = u + w \tag{3.7}$$

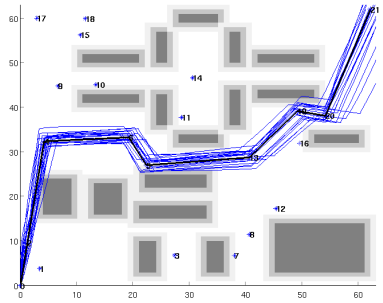
where q is the position vector, u are the input forces, and w is a white noise term that quantifies the uncertainty in the motion model of the robot. We assumed that the state of the robot could perfectly be sensed. We used several different maps and assumed that the map uncertainty in each case was specified to us using a discrete occupancy grid (OG) representation, i.e. we are given a distribution $p(O/x_{ij})$ that denotes the probability that there is an obstacle in the (i, j)th grid in the map.

The local feedback controllers that connect the sampled points were designed using linear quadratic regulation (LQR) techniques [1]. The cost function used penalized both the control effort and the state deviation from the goal equilibrium point. The safe recurrent classes Ω_k around some sampled point x_k were defined to be some ball of radius ϵ, where ϵ was chosen in a heurestic manner. We found the probability of success and the cost of operation, where the cost of operation was quadratic cost,of the feedback controllers that join two nodes on the graph using repeated Monte Carlo simulations accordingly to (Equation 3.3) and (Equation 3.4). We have tested the controllers using a naive Monte Carlo method, but for higher

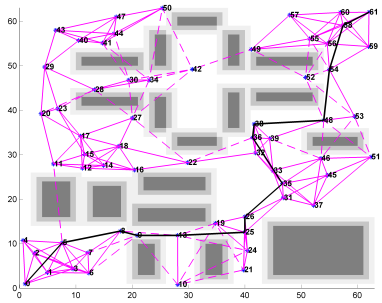
dimensional problems, very efficient subset simulations technique exist [40], which can be leveraged to find the success probabilities in an efficient manner.



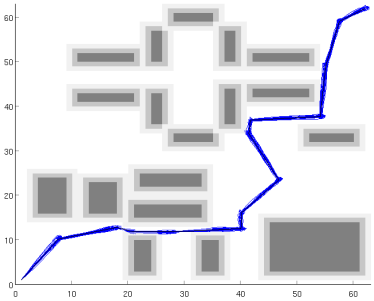
(a) Standard PRM



(b) $p_s = 2\%$, Trajectories Ensemble



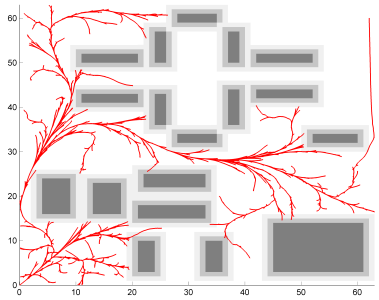
(c) Modified PRM



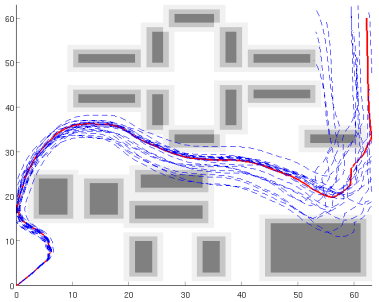
(d) $p_s = 89.68\%$, Trajectories Ensemble

Fig. 3.3. Comparison of GPRM with Traditional PRM : Map 1 ($p_s \equiv$ Probability of Success)

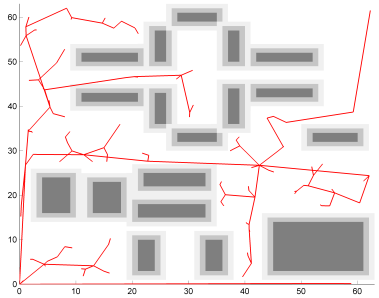
The result of our simulation experiments are shown in Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6. Each of these figures presents the performance of the GPRM and GRRT algorithms, along with their traditional counterparts on two different maps. We have performed our simulations on several other maps, but the paucity of space does not allow us to present all these results. Figure 3.4(a) and Figure 3.6(a) represent the tree built RRT in the map. In Figure 3.4(b) and Figure 3.6(b), we represent the final nominal (noise-free) path from the start node to the goal node found by RRT in red and the ensemble of trajectories that result



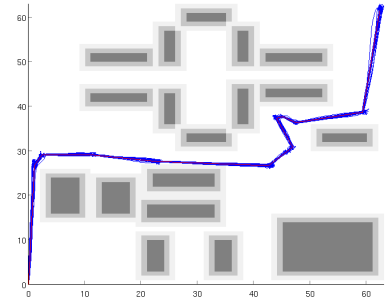
(a) Standard RRT tree



(b) $p_s = 12\%$, Trajectories Ensemble

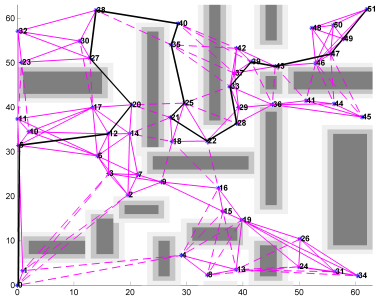


(c) Feedback based RRT tree

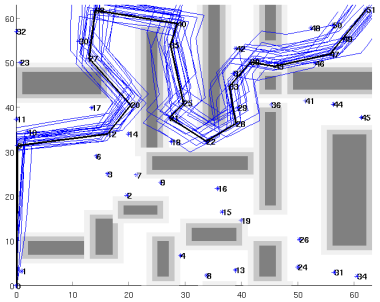


(d) $p_s = 100\%$, Trajectories Ensemble

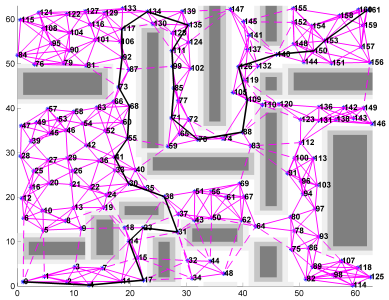
Fig. 3.4. Comparison of GRRT with Traditional RRT: Map 1 ($p_s \equiv$ Probability of Success)



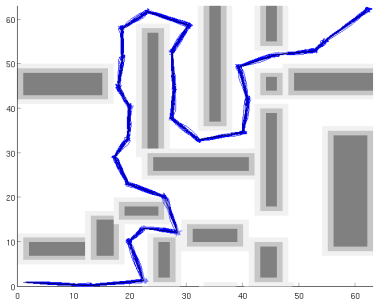
(a) Standard PRM



(b) $p_s = 6\%$, Trajectories Ensemble

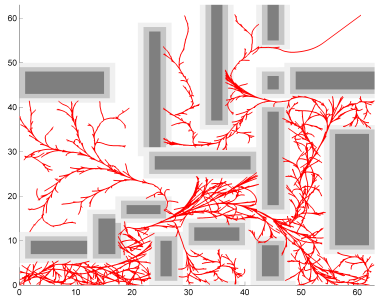


(c) Modified PRM

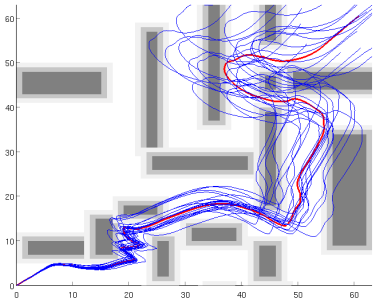


(d) $p_s = 96.83\%$, Trajectories Ensemble

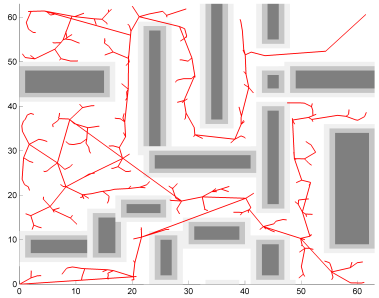
Fig. 3.5. Comparison of GPRM with Traditional PRM: Map 6 ($p_s \equiv$ Probability of Success)



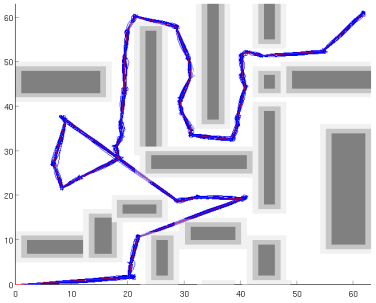
(a) Standard RRT tree



(b) $p_s = 0\%$, Trajectories Ensemble



(c) Feedback based RRT tree



(d) $p_s = 87.9\%$, Trajectories Ensemble

Fig. 3.6. Comparison of GRRT with Traditional RRT: Map 6 ($p_s \equiv$ Probability of Success)

due to the process uncertainty in the motion model in blue, along with probability of success of the path. In Figure 3.4(c) and Figure 3.6(c), we represent the tree built by GRRT, showing the noise-free paths between nodes. In Figure 3.4(d) and Figure 3.6(d), we represent the nominal noise-free path, along with the ensemble of trajectories due to process uncertainty, as well the probability of success of the path. Similarly, in Figure 3.3(a) and Figure 3.5(a), we show the graph built by PRM and the nominal path and the trajectory ensemble, along with the probability of success of the path. The same data are presented for GPRM in Figure 3.3(c) ,(d) , Figure 3.5(c) and (d). The graphs and trees constructed by GPRM and GRRT are virtual, because there is not one single path that connects the nodes but an entire bundle of them between any two nodes because of the uncertainty in the robot motion model.hence, the connection encodes the feedback controller that joins the two nodes, rather than an actual path that joins them as in the case in traditional PRM or RRT. However, for visual comprehension, we only present the noise-free paths in the GRRT and GPRM tree and graph figures, respectively [see Figure 3.3(c), Figure 3.4(c), Figure 3.5 (c) and Figure 3.6(c)]. As shown in the figures, the performance of GPRM and GRRT is significantly better that the performance of the traditional PRM and RRT algorithms. For instance, in map 1, the probabilities of success of the GRRT and GPRM algorithms are 100% and 89.68%, respectively, whereas the probabilities of success of the traditional RRT and PRM are 12% and 2%, respectively. A similar observation holds for map 6. Although the nominal noise-free path in PRM and RRT does not collide with any obstacles, the presence of the noise in the robot motion model leads to collisions in most cases due to the open-loop nature of the plan. This case is clearly shown in Figure 3.3(b), Figure 3.4(b), Figure 3.5(b) and Figure 3.6(b), where the trajectory ensemble grows in size over time and diverges from the nominal noise-free path and thereby leads to collisions, which were not present in the nominal path. Moreover, it is also shown in the same figures that there is no guarantee that the robot will reach the goal under these plans. In contrast, the robustness

that is attained due to feedback can be gauged from Figure 3.3(d), Figure 3.4(d) , Figure 3.5(d) and Figure 3.6(d), where the ensemble of trajectories is tightly bundled around the nominal noise-free path due to the presence of feedback. We note here that the very low success probabilities of the traditional methods are due to the complicated nature of the two maps. In similar maps, there have better performance but the GRRT and GPRM performance is always significantly better. This result is not surprising, because the original PRM and RRT algorithms are open-loop planners and were not developed for uncertain robot models and state spaces. The control in the open-loop planners is a function of time alone and is based on the nominal dynamics (unperturbed dynamics). Thus, in the presence of perturbations or noise, the path if the robot can substantially deviate, and the nominal performance of the robot cannot be maintained. However, the feedback plans in GRRT and GPRM assures robustness to such perturbations, because the control is a function of the state of the robot (not time), and hence, even when the robot path deviates from the nominal, the control law can still guide it towards the goal. In fact, all other techniques that modify the PRM and RRT to handle uncertainty, such as [17,16,18] and [15], suffer from the exact same problem, because they open-loop planners that are designed to handle only stationary map uncertainty and thus, similar to PRM and RR as aforementioned, cannot be robust to the dynamic process uncertainty in the robot motion model.

3.4.2 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

$$\dot{x} = vcos\theta + w_x \tag{3.8}$$

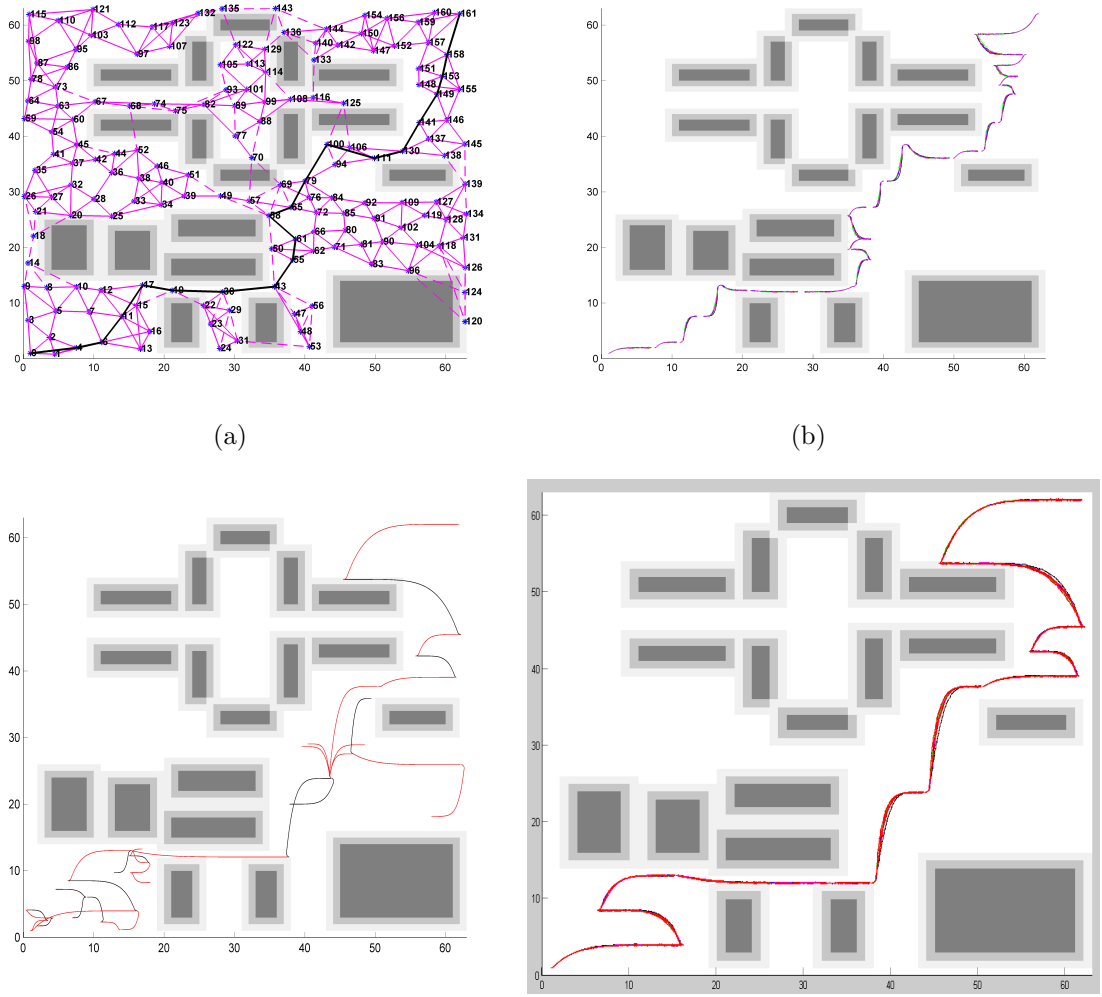
$$\dot{y} = vsin\theta + w_y \tag{3.9}$$

$$\dot{\theta} = \omega + w_\theta \tag{3.10}$$

where (x, y, θ) represents the pose of the robot, the velocity v and the angular velocity ω represent the control inputs to the problem, and w_x, w_y and w_θ are uncorrelated white noise terms. We assume that the robot can approximated by a point in this dissertation. 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. Because the unicycle model is a nonholonomic system, the design of feedback controllers that stabilizes the robot about a particular equilibrium configuration is much more involved than in the case of the fully actuated robot considered in the previous section. In fact, the standard linear control theory cannot be used in the design of feedback controllers for such systems, even locally [41], [42], because it is known from Brockett's theorem that a static feedback controller that can stabilize such systems about any equilibrium does not exist. Thus, suitable nonlinear control techniques have to be resorted to design feedback laws [41], [42]. We chose a dynamic feedback linearization-based controller design that has been treated in detail in [42]. This controller can stabilize the robot about any given configuration in a smooth manner and with exponential convergence.

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

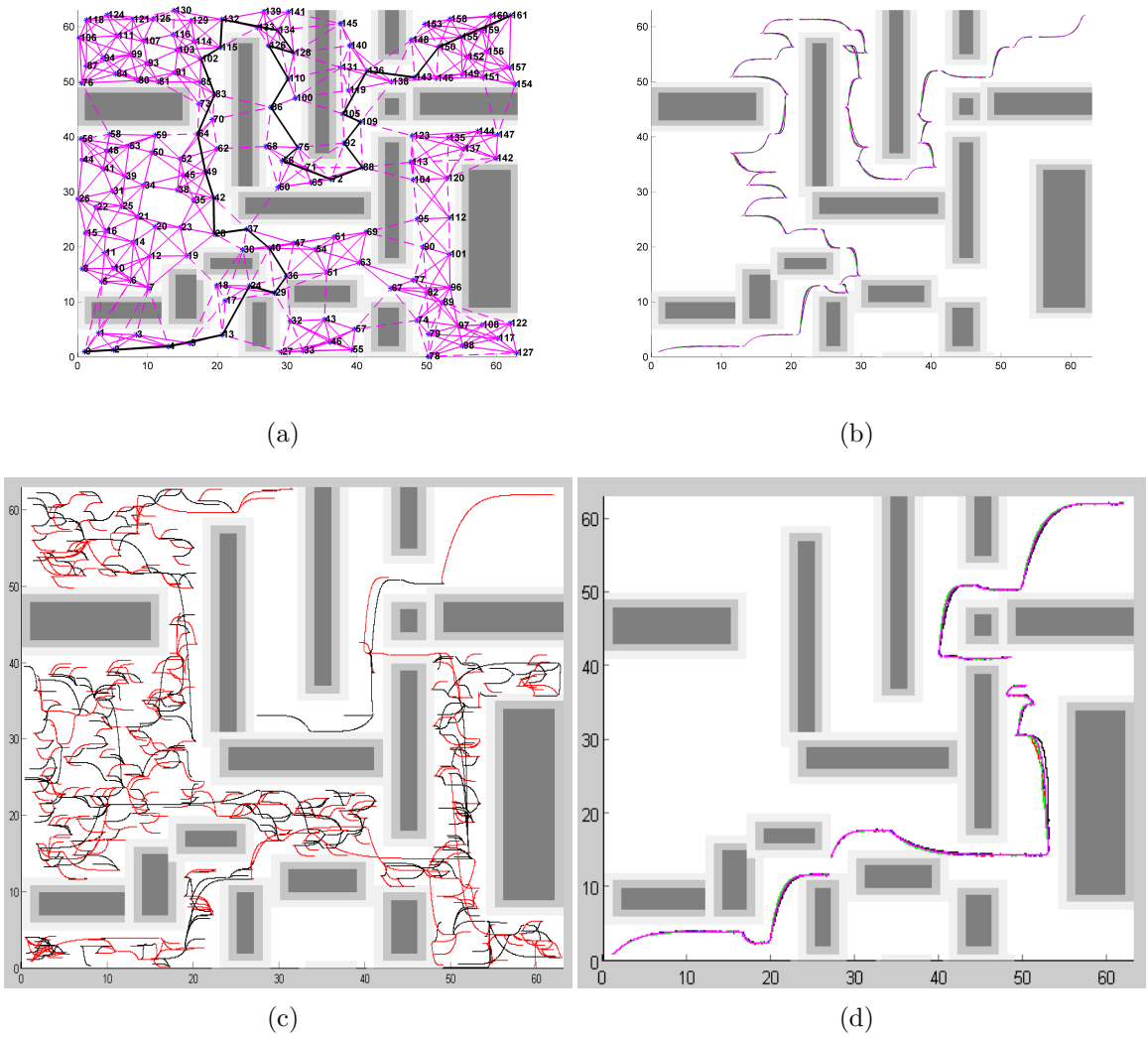
allowable linear speed ($\sigma_{x,y} = 0.3\, v_{max}$), whereas the noise in the θ equation has intensity equal to 10% of the maximum allowable angular speed ($\sigma_\theta = 0.1\, \omega_{max}$).



(a) (b) (c) (d)

Fig. 3.7. Performance of GRRT and GPRM on the Unicycle Robot: Map 1 ($p_s \equiv$ probability of success). (a) GPRM, (b) Bundle of final trajectories, with $p_s = 100\%$. (c) GRRT with nonholonomic constraints. (d) Bundle of final trajectories, with $p_s = 93.33\%$

The results of our numerical simulations are shown in Figure 3.7 and Figure 3.8. Figure 3.7(a) and Figure 3.8(a) represent the tree of feasible trajectories shown with-



(a) (b)

(c) (d)

Fig. 3.8. Performance of GRRT and GPRM on the Unicycle Robot: Map 6 ($p_s \equiv$ probability of success). (a) GPRM, (b) Bundle of final trajectories, with $p_s = 100\%$. (c) GRRT with nonholonomic constraints. (d) Bundle of final trajectories, with $p_s = 100\%$

out any noise in the system. Figure 3.7 (b) and Figure 3.8(b) show the nominal path, along with the final path ensemble around it. Similarly Figure 3.7(c) and Figure 3.8(c) show 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. Figure 3.7(d) and Figure 3.8(d) represent the path ensemble around the nominal trajectory. We performed the experiments on the set of six maps that we has used in the case of the fully actuated robot, except in this case, because due to the non-holonomic constraints on the motion of the robot, the trajectories of the robot are smoother that in the case of the fully actuated robot. This result is not so surprising, because the algorithms are exactly the same for the two cases, except in the design of the local feedback controllers.

GPRM and GRRT are both successful at handling motion uncertainty however the GRRT algorithm is easier to search because of the tree structure. One avenue is to use the GRRT algorithm as the option in the GPRM algorithm for large degree of freedom (DOF) systems and complicated maps, because this approach would control the number of nodes in the GPRM, thereby reducing the search complexity in GPRM while, at the same time, making the options more powerful that if we were to use only primitive local feedback controllers. This case will be one of our future avenues for research. Thus, in this section, we have shown the application of the generalized sampling-based feedback motion planners to both fully actuated and under-actuated robotic systems. As shown in the results, the planner have excellent performance in wither case in quite complicated maps, in the presence of motion uncertainty and uncertainty in the map.

3.5 Conclusion

This section has presented generalized versions of sampling-based motion planners PRM and RRT, i.e. GPRM and GRRT. These algorithms generalize the PRM and RRT methodologies to the case where there is uncertainty in both the robot

motion model and the map provided to the robot. We have analyzed the algorithms and shown their probabilistic completeness. The algorithms were tested on an idealized planer fully actuated holonomic robot and an under-actuated nonholonomic unicycle, and the results clearly show that the performance of the generalized planners is significantly better than the performance of their traditional counterparts, mainly because the traditional PRM and RRT algorithms were not designed to take uncertainty into account. Although we have obtained very promising initial results, much remains to be done. As aforementioned, we do not foresee any significant difficulty in applying the algorithms to robots that can be described by rigid body equations of motion or any other fully actuated robot system, because the systems are essentially feedback equivalent to the planer robot described in this section [43]. Furthermore, it was noted that the GPRM algorithm has to sample far more points before it attains a satisfactory probability of success compared to GRRT. We will explore the use of more sophisticated sampling strategies to increase the efficiency of the planner and increase its probability of success. Moreover, using GRRT as options within GPRMs might help us reduce the number of points that need to be sampled by GPRM to obtain an adequate probability of success. Furthermore, this approach might allow our techniques to scale to larger maps that the ones considered here. Finally, and perhaps most importantly, we would like to relax the assumption that the state of the robot is perfectly known, instead assuming that we only have noisy measurements of the state relative to the map. This approach implies that we need to solve the planning problem in conjunction with the simultaneous localization and mapping (SLAM) problems, thereby leading to a solution to the simultaneous planning, localization, and mapping (SPLAM) problem, which is partially observable Markov decision process (POMDP) and is orders of magnitude more complex compared to the MDPs considered in this section.

4. ADAPTIVE SAMPLING FOR GENERALIZED SAMPLING-BASED MOTION PLANNERS*

4.1 Introduction

In this section* a novel adaptive sampling methodology for the generalized sampling-based motion planners (introduced in the previous section) is presented.

The general motion planning problem in robotics is to find a collision free path for a robot from one configuration to another, in a given obstacle space.

Exact planners are intractable for most practical problems because the complexity grows exponentially with the dimensionality of the problem [44]. Randomized Sampling based methods were introduced to provide approximate solutions, while avoiding the prohibitive cost of computing the exact representation of the free space. Probabilistic Roadmaps (PRM) are one of the most successful sampling based methods for multi-query planning, which sample the domain in a random fashion and build a roadmap over these samples to represent the free space [3]. For single-query planning Expansive Space Tree planners (ESTs) [45, 46] and Rapidly-exploring random trees (RRTs) [4] were developed. Then there are Sampling based Roadmap for Trees (SRTs) planner [47, 48] which construct a PRM style roadmap of single query planner trees. Some of the recent efforts to get optimal solution for motion planning using these sampling based motion planners were presented in [49] where PRM* and RRT* were developed.

To address highly constrained motions and domains, a key idea is to bias the sampling towards good regions of the configuration space [3], and various different sampling strategies to do the same have been proposed. These planners make local hypothesis that identify poor visibility regions [32] in the free space, and [32] categorizes the research efforts related to this based of the different methodologies adopted.

*Reprinted with permission from "Adaptive sampling for generalized sampling-based motion planners", by S. Kumar and S. Chakravorty, 2010, *IEEE Conference on Decision and Control*, 7688-7693 ©2010 IEEE,

Some use the information of workspace geometry, broadly categorized as *Workspace based sampling strategies*. Techniques in this category are watershed labeling algorithm [50] and workspace importance sampling [51]. Some use geometric patterns and reject unpromising samples, categorized as *Filtering based sampling strategies*. Techniques under this category are Gaussian strategy [52], bridge test [53], and Vis-PRM [54] and medial axis sampling [55, 56, 19]. Some use information gained during the roadmap construction, categorized as *Adaptive Sampling Strategies* and techniques include two-phase connectivity expansion strategies [3], and multiphase sampling [57]. There also exists a *Deformation Strategy for Sampling*, which tries to deform the domain into a more expansive domain [58]. Furthermore, in [59], the different research efforts related to importance sampling and different connections strategies other than k-near neighbors were further discussed in detail. [60] presents an Obstacle-based PRM (OBPRM), one of the first and very successful representatives of obstacle-based sampling methods. [61, 62] were other manipulability based importance sampling approaches. Connection sampling methods [63, 3] generates samples that facilitate the connection of the roadmap. Alternate PRM connection strategies other than k-near neighbors were also developed. Some of the earlier efforts related to this discussed creating sparse roadmaps as in [64, 3], where computation of edges which were part of the same connected components were avoided. Then in-order to capture connectivity of the $\mathcal{C}_{free}$ other strategies were aimed at connecting different components of the roadmap [63, 65]. Lazy PRM were introduced in [66, 67, 68], which presented the idea of evaluating the collision checks only when it is absolutely necessary, i.e., lazy evaluation, in-order to speed up performance. Furthermore, a combination of lazy PRM and ESTs were presented in [69] as Single-query, Bi-directional, Lazy collision checking (SBL) planners. All the above strategies spend more time generating a node when compared to a naive uniform sampling, and, adopted strategies of evaluating edges at a later stage of planning, with the expectation that a much smaller roadmap is required to answer

queries, resulting in faster computation time. These strategies were studied and analyzed in refs. [70, 32, 71, 72, 59] where various measures/ metrics such as *connectivity*, *coverage* and *completeness* were proposed to evaluate their effectiveness. In ref. [73], an attempt is made to provide metrics for the sampling process during the roadmap construction. Thus, sampling intelligently can achieve a significant speedup when compared to naive uniform sampling.

Unfortunately PRM and its variants work in the deterministic framework, and with the introduction of map and robot model uncertainty, the technique is no longer robust. Furthermore, PRM does not account for the dynamics of the robot. Rapidly-exploring random trees (RRTs) incorporate randomized sampling of the domain, as in PRM, while also incorporating the dynamics of the robot while planning [4]. However, like PRMs, RRTs are open loop planners, and thus are not robust to map and model uncertainty. The generalized sampling based motion planners, *Generalized-PRM* (GPRM) and *Generalized-RRT* (GRRT), were introduced to incorporate stochastic models of map and model uncertainty along with the dynamical constraints of the robot, and provide a feedback solution to the motion planning problem [29], [28]. We would like to mention other attempts to generalize PRMs and RRTs to handle map uncertainty [17, 16, 18, 22]. However, none of these techniques provide a feedback solution to the planning problem and therefore are not robust to model uncertainty.

In this section, we introduce a novel strategy for adaptive sampling in GPRM. The strategy proposed here incorporates the information of the probabilities encoded in the connections of the GPRM. With this extra information, which is unique to planners incorporating uncertainty, the sampling strategy biases the samples such that the efficiency and the overall success probability for the planning increases in GPRM. We show that motion planning problem on complex maps can be efficiently solved using GPRM, in conjunction with the adaptive sampling strategy, while simultaneously increasing the success probability of the solution. A preliminary version of

this work has been published in the 49^{th} *IEEE Conference on Decision and Control*, 2010 (IEEE CDC '10) [30] and a journal version has been accepted and will appear in *Journal of Control Theory and Application, Special Issue on Approximate Dynamic Programming*, 2010. In this section, along with the detailed algorithm of the proposed methodology, we extend the domain of application of the proposed algorithm to n-link manipulators.

The rest of the section is organized as follows. section 4.2 discusses hierarchical planning methods and the GPRM algorithm in brief, section 4.3 introduces conceptualization, development and the algorithm of the *Adaptive Sampling* Strategy for GPRM. section 4.4 discusses the application of GPRM along with Adaptive Sampling on two different dynamical systems along with results.

4.2 Generalized Sampling Based Methods

The basic motion planning problem is to find a collision free path for a robot in a given obstacle space. With the introduction of map and model uncertainty, one can no longer have the same formulation of the motion planning problem. In the presence of stochastic model uncertainty, there is a need for feedback control, which is then associated with a probability that the robot reaches the goal without hitting the obstacles. Generalized Sampling Based Algorithms [29, 28] were introduced to address the problem of feedback motion planning in such constrained work spaces. Before going into the details of the methodology of [29, 28], we note that the complexity of the motion planning problem has increased due to:

- Introduction of model uncertainty in the dynamics of the robot, which implies that we have to obtain satisfactory performance over an ensemble of paths instead of a single path.
- Introduction of map uncertainty, implies the planner has to succeed for an ensemble of maps.

The notion of collision avoidance and collision-free paths as the solution to the motion planning problem, can no longer be satisfied, and therefore the above criteria need to be replaced by a solution/ path with a high probability of success. The motion planning problem can be re-framed as : *To solve the motion planning problem in the presence of map uncertainty and model uncertainty, generate a feedback solution with a probability of success above an a-priori specified probability,* p_{min}.

4.2.1 Hierarchical Methods and Generalized Probabilistic Roadmaps (GPRM)

If the uncertainties in the robot model and environment can be modeled probabilistically, the robot motion planning problem can be formulated as Markov Decision Problem (MDP) [2]. These MDPs are computationally intractable for anything but small state/ control spaces and especially hard to solve in continuous state and control spaces. Hierarchical Methods can be used to break down the complexity of the problem. The *Generalized Probabilistic Roadmaps*(GPRM) [29, 28], is a sampling based hierarchical method which extends the *Probabilistic Roadmaps* (PRM) [3] technique for deterministic path planning, to systems with stochastic model and map uncertainty.

In the following paragraph we briefly introduce GPRM, more details can be found in [29]. The state of the robot is given by $x = (q, \dot{q})$, where q represents configuration of the robot and $\dot{q}$ the generalized velocities. The free region in the map corresponds to a free region in the configuration space, C_{free}, which induces a free region in the state-space of the robot, say χ_{free}. GPRM samples equilibrium states (i.e. state wherein the velocities are zero) in χ_{free}, which are called *landmarks*.

The planning problem of guiding the robot from the *start* landmark to the *goal* landmark is divided into two hierarchical levels. The lowest level planner guides the robot from one landmark to another using feedback control and accounts for the model uncertainty in the robot dynamics, specified by the following equation:

$$\dot{x} = f(x) + g(x)u + h(x)w \tag{4.1}$$

where x is the state of the robot, w is a white noise perturbation, and u is the control. However, the control does not account for constraints, i.e. obstacles in the map, which are specified by $p(O/y)$, the probability that a point y in map is *occupied*.

The interaction between feedback planner and the obstacles in the map result in a *transition probability* and *transition cost* for the robot from one landmark to next. Figure 4.1 depicts a sample path between the landmarks s and r given a feedback controller u that guides it towards r. The control u is taken at state s, the agent will reach one of the k-nearest neighbors of s. The transition probability, of an individual path, $p_{s,r}$ is given by :

$$p_{s,r} = \prod_{y} (1 - p(O/y)) \tag{4.2}$$

where y represents the grids along the path. A *failure state*, say x_{fail}, is introduced and $1 - p_{s,r}$ is the probability of landing in the failure state. The transition cost, $c_{s,r}$ is directly proportional to the probability of transitioning to the failure state, x_{fail}. Due to the presence of model uncertainty, the average cost $c(s, u)$ and the average transition probability $p(r/s, u)$, i.e. average probability of reaching state r given current state s and control action u, have to be formed by averaging over all such sample paths. This is achieved using Monte Carlo simulations.

The top level planner, works on global map in the landmark space. It uses the information of the metrics of the lowest level planner, minimizes the cost-to-go from each landmarks over all possible policies, and gives the optimal control policy over the landmark map.

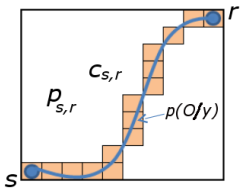


Fig. 4.1. *Transition Cost* and *Transition Probability*

The optimal control action $u^*(\cdot)$ for each state/ landmark of the map is the outcome of the top level planner. The optimal cost-to-go $J^*(\cdot)$, required in calculation of $u^*(\cdot)$, is found as the solution of the Bellman fixed point equation/ Dynamic Programming equation :

$$J^*(s) = \min_u \{c(s,u) + \sum_r (p(r/s,u)J^*(r)\} \tag{4.3}$$

$$u^*(s) = \operatorname*{argmin}_u \{c(s,u) + \sum_r p(r/s,u)J^*(r)\} \tag{4.4}$$

where $J^*(s)$ is the optimal cost-to-go from state s, $u^*(s)$ is the optimal control action to be taken at state s. Here, control u at state s is the next landmark among the k-nearest neighbors of s that the robot is guided to, $p(r/s,u)$ is the probability of transition from $s \to r$ given the robot is guided towards the landmark specified by u, and $c(s,u)$ is the cost of transition. Note that $p(r/s,u)$ and $c(s,u)$ are got by evaluating the lowest level feedback planner. The details of the algorithm and the calculation of metrics are in [29, 28].

4.2.2 Algorithm GPRM

The pseudo-code for the generalized probabilistic roadmaps (GPRM) algorithm was given in 3.1 in section subsection 3.3.2.

A few points have to be made regarding the feedback controllers specified in step 2 above :

- Due to model uncertainty present in the dynamical system, it is impossible to control the robot exactly to the point x_g even in the absence of obstacles
- In the case of stochastic systems, a feedback controller is necessary due to uncertainty. The feedback controller ensures that even in presence of uncertainty

in the model, the robot reaches a neighborhood of the target equilibrium state with a high probability, in the absence of obstacles

- The feedback controller is designed for a workspace without any obstacles as otherwise the controller design is complicated

The feedback controller can be designed in many ways. For linear systems LQR based controllers can be used. For non-linear system, the system can be linearized about an equilibrium point and a feedback controller can be designed for the linearized system. Other non-linear feedback controllers may also be used, such as the dynamic feedback controller used for the non-holonomic system in [28]. The feedback controller operates between one landmark and another and in presence of model uncertainty ensures the robot reaches a neighborhood of the target equilibrium state in the absence of obstacles. In the presence of obstacles, Monte Carlo simulations are used to compute the *transition probability* and *transition cost* in using the feedback planner to guide the robot from one landmark to another. The feedback controller used in the work presented here is state-feedback based LQR controller.

The GPRM is capable of handling model and map uncertainty as discussed above, but as the complexity of the map, i.e the size of the map and the clutter of the obstacles increase, the number of landmarks required to find a solution becomes large, thereby greatly increasing the computational resources required. A logical extension for complicated maps is to sample in areas where samples are required, i.e. use an adaptive sampling strategy. The next section describes such an adaptive sampling algorithm.

4.3 Adaptive Sampling

In sampling based motion planning algorithms, the number of samples determine the complexity of computation required to solve the problem. For a complex domain, a naive uniform sampling will require a large number of samples and hence, more

computational resources. Introduction of adaptive sampling adds intelligence to the planning algorithm, by efficiently adding new samples.

4.3.1 Adaptive Sampling Details

In a sampling based motion planners framework, co-ordinates of configuration space are sampled in random fashion which is mapped into the obstacle space as shown in Figure 4.2(a) (they are referred to as equilibrium states, x_g or *landmarks* in GPRM framework). A connectivity graph is constructed over the landmarks as shown in Figure 4.2(b), it is based on the cost and transition probabilities computed from the lowest level feedback planner used in GPRM.

Using the information encoded in the connectivity graph, we introduce the major ingredients of the adaptive sampling strategy in the following.

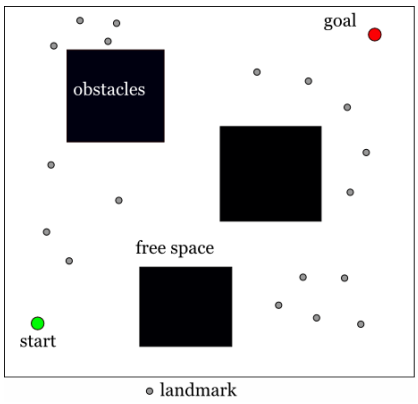
Identification of Start and Goal Clouds

A cloud of samples is referred to as a collection of landmarks which are inter-connected with transition probabilities higher than p_{min}, in the connectivity graph of the map/ obstacle space. The start and the goal clouds are the cloud of samples containing the start and the goal (or end) landmarks respectively.

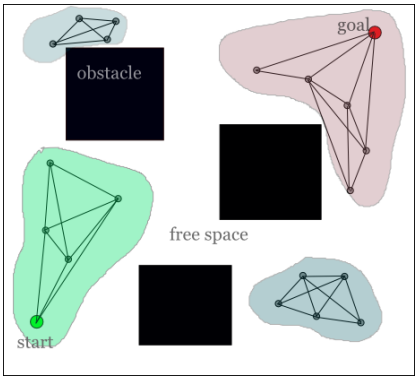
The motion planning problem, in the generalized sampling based motion planning framework, is to find a path[1] with success probability higher than p_{min} between the start and the goal landmark. The idea is to identify the cloud of samples as shown in Figure 4.2(c) containing the *start landmark state* and the *goal landmark state* and try to connect them during the re-sampling phase and hence solve the motion planning problem.

To identify these clouds the information carried by the connectivity graph is used.

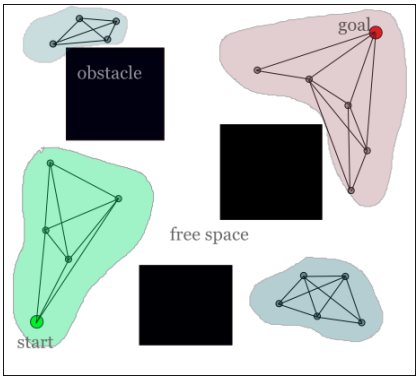
[1]A path here implies a local feedback controller guides the robot from one landmark to another, while the higher level planner guides the robot regarding the landmark, to go to next.



(a) Landmarks



(b) Connections



(c) Cloud of Sampled

Fig. 4.2. Problem Domain with Free Space, Obstacles, Start, Goal Positions

The connectivity graphs in the generalized motion planner framework encode both the *transition cost* and the *transition probability* information.

We assign *goal proximity probability*, $\bar{p}_g(x)$, and *start proximity probability*, $\bar{p}_s(x)$, to each of the landmarks, x. Proximity probability is a metric defined between two landmark states (say x_a and x_b), and it carries the information of the probability of transition from state x_a to x_b, given by $\bar{p}_b(x_a)$, and vice-versa, $\bar{p}_a(x_b)$. The goal proximity probability, $\bar{p}_g(x)$ is defined as the proximity probability between a landmark, x, and the goal landmark state, x_g, along with a constraint that $\bar{p}_g(x) > p_{min}$, where p_{min} is given. It is calculated by traversing from the goal landmark, x_g to the concerned landmark, x, keeping track of all the transition probability in the path. Similarly, the start proximity probability $\bar{p}_s(x)$ is calculated by traversing from the start landmark, x_s, to the concerned landmark, x, and keeping track of the transition probabilities along the way. These metrics, once calculated, will suggest landmarks which are connected to the goal and the start landmarks, with a overall transition probability greater than the threshold probability (p_{min}) of the domain. In this way the cloud of samples connected to the start and the goal with a high transition probability can be computed.

Identification of other clouds

We also want to identify clouds other than the start and the goal clouds that are present in the workspace. We compute this information to identify and differentiate between the *good* and *bad* samples. These *good* and *bad* samples will be discussed in item 3 below.

The process of computing the information about clouds can be stated as:

- pick a landmark, x, and assign a group identification, $g_{id}(x)$, representing cloud information

- all the directly and indirectly connected landmarks are assigned the same group identification, $g_{id}(\cdot)$
- restart the process with a new landmark which has no assigned group yet
- continue till all the landmarks are covered, i.e each landmark has a assigned group identification, $g_{id}(\cdot)$

Once the process is complete, all samples having same $g_{id}(\cdot)$ lie in that group.

Sampling **good** landmarks

Sampling of landmarks in the configuration space is usually done using uniform sampling over the configuration space, when no knowledge is available to bias the sampling. Sampling ***good*** landmarks x_{good} involves sampling landmarks which have the potential to solve the motion planning problem, or about rejecting the bad landmarks x_{bad} from a set of sampled landmarks. We re-sample the space, i.e. generate a set of new landmarks, X_{new}, and find the k-nearest neighbors of each landmark in X_{new}. Based on the potential connections[2], every new landmark can be categorized completely, refer Figure 4.3, as a landmark whose neighbors:

i. $\in X_{new}$ only, the set of new landmarks generated, (refer Figure 4.3(a))

ii. $\in$ different clouds, (refer Figure 4.3(b))

iii. $\in$ different clouds and X_{new}, (refer Figure 4.3(c))

iv. $\in X_{new}$ and a specific cloud, (refer Figure 4.3(d))

v. $\in$ a specific cloud only, (refer Figure 4.3(e))

Samples in *Category* (*item iv*) *and* (*item v*) are categorized as "bad" since

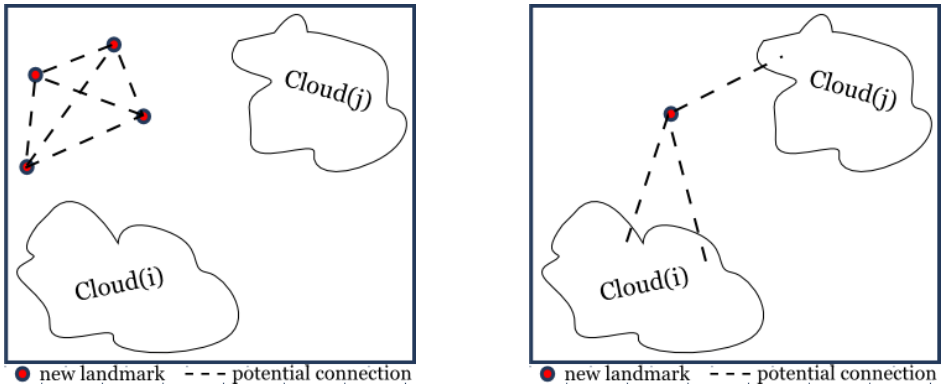
[2]The connections with k-nearest neighbors, prior to computing the transition probabilities, which either establishes a connection or discards it.

obviously they have minimal potential to solve the problem. Hence, using the cloud information we reject the identified bad samples.

Identifying Weak Link / Links in a Low Probability Connected Path

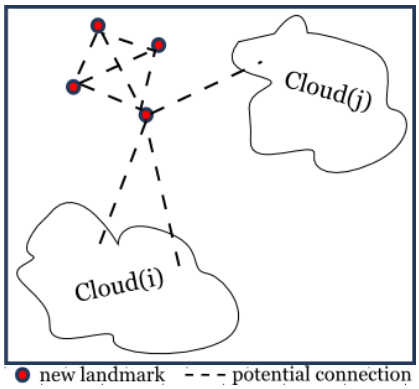
The connectivity graph of a map has the *transition cost* and *transition probability* information. In contrast, in the deterministic framework of sampling based motion planners, such as PRM, these graphs only carry the *transition cost* information. In GPRM the top level planner searches for a high probability path over the domain, and returns a path connecting the start landmark x_s and the goal landmark x_g, and a success probability associated with it, say p_{path}. There could be cases where in spite of the connectivity, the p_{path} is less than the desired threshold success probability p_{min}. Such an outcome can be used as a starting point for finding a neighboring path, $path'$ with a path probability $p_{path'}$, which has a success probability $p_{path'} > p_{min}$. We identify the *weak link / links*[3] of the low probability path and then sample around these in search of $path'$. Finding a neighboring path with a higher probability of success in the vicinity of a low-probability path may not always be feasible, as has been experienced during numerical simulations, but results show that the technique works fine most of the time.

[3]Connections in the connectivity graph, which are responsible for low success probability of the path.

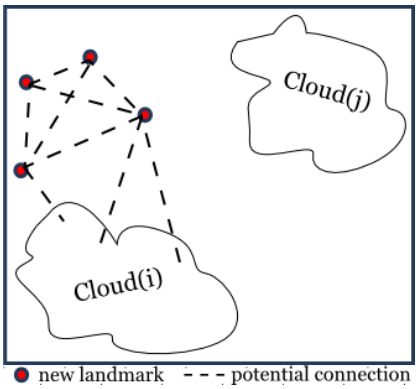


(a) C_1

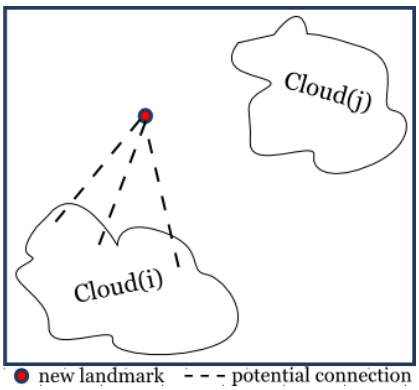
(b) C_2



(c) C_3



(d) C_4



(e) C_5

Fig. 4.3. Categories of New Landmarks Sampled

Based on the ingredients of the Adaptive Sampling Strategy as described above, the algorithm can be summarized as follows :

Algorithm 4.1: AGPRM - Adaptive Sampling GPRM

```
1  Invoke GPRM over the given map initially with a small number of randomly
   selected landmarks and p_min;
2  while a path with high success probability NOT found do
3      Assign p̄_s(·) and p̄_g(·) to all landmarks;
4      Identify landmarks with high p̄_s(·) and p̄_g(·) and their g_id(·);
5      Pick a pair of landmarks, one with high p̄_g(·) and another with high p̄_s(·);
6      for each pair found do
7          Sample[a] between these landmarks;
8          Identify bad samples x_bad and reject them;
9          flag← EvaluateImportantConnections();
10         if flag then
11             Perform GPRM;
12             if path with high success probability found then
13                 STOP;
14             else if a low probability path found then
15                 while 0 < p_path < p_min do
16                     Find weak link / links in the low probability path;
17                     Sample[b] between the pair of landmarks along the weak link;
18                     Discard bad samples;
19                     Using EvaluateImportantConnections() perform GPRM;
20                     if a path with p_path > p_min found then
21                         STOP;
```

[a] In simulations the samples were drawn from a Gaussian distribution, with mean placed at the arithmetic mean of generalized positions of start and goal landmark and the standard deviation σ being 2-norm of the distance between start and mean.

[b] In simulations, a biased distribution was assumed, i.e. an elliptical distribution with major axis aligned along start and goal configurations.

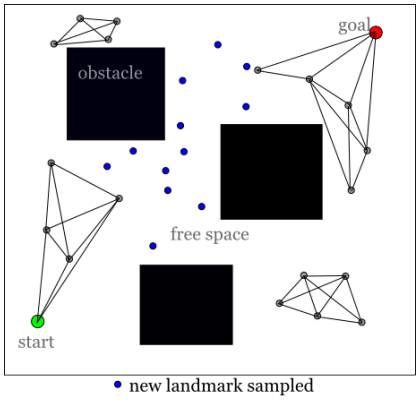
Algorithm 4.2: AGPRM - EvaluateImportantConnections()

Data: Start landmark, Goal landmark, set of new samples

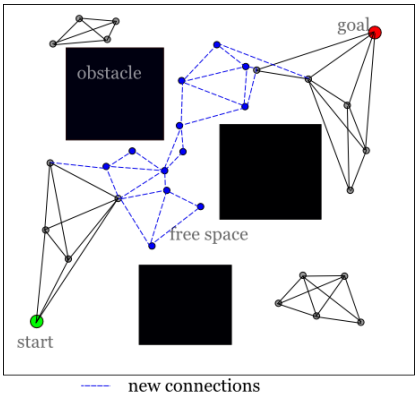
```
1  Generate connections using k − near landmarks;
2  Find the connections involving start landmark and goal landmark;
3  Evaluate these connections now;
4  if Any connection having start landmark succeeded then
5      flag_start ← true;
6  else
7      flag_start ← false;
8  if Any connection having goal landmark succeeded then
9      flag_goal ← true;
10 else
11     flag_goal ← false;
12 flag = flag_start ∧ flag_goal;
13 if flag then
14     Evaluate all other connections;
15 else
16     Discard samples involved in other connections;
17 return flag;
```

The Figure 4.4 depicts in brief the stages in the adaptive sampling methodology.

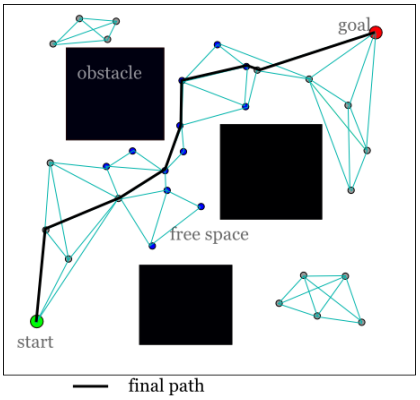
AGPRM uses the probabilistic complete algorithm GPRM in its core and the proposed adaptive sampling technique attempts to improve the solution in terms of sampling efficiency and performance. In situations when the algorithm is applied to maps with no possible solution, the algorithm will stop after a large number of iterations and return failure.



(a) Assign **good** landmarks between *start* and *goal* clouds, in Figure 4.2(c)



(b) Assign new connections



(c) The Final Path

Fig. 4.4. Adaptive Sampling in Steps, (Build-Up on Figure 4.2)

4.4 Results and Discussion

The Adaptive Sampling methodology developed is applied along with GPRM to point robot dynamics and 3-link manipulator case. Results presented here study the improvement with respect to the number of samples required to solve the problem. The Adaptive sampling algorithm is not time optimized as of yet, but will be a topic of future research.

4.4.1 Point Dynamics Robot

First a fully actuated point robot is studied. The dynamics of the robot is given by:

$$\ddot{q} = u + w, \tag{4.5}$$

where q is the generalized position vector of the robot, u are the input forces and w is a white noise term that quantifies the uncertainty in the motion model of the robot. This case was solved using basic GPRM, refer [29]. Numerical simulation results are presented for a set of maps with varying degrees of complexity. In general, the results indicate that:

- The quality of sampling improved, i.e the landmarks were generated in required regions.
- The number of landmarks required to solve any complex map is approximately reduced to half the number required for solving the same map with basic GPRM, with progressively higher rewards in larger/ complex maps (refer Table Table 4.1).

.

Each of the maps (Figure 4.5 - Figure 4.8) discussed in the results section has two sub-figures : sub-figure (a) represents the initial landmarks the adaptive sampling

Table 4.1
Result Comparison : Point-Dynamics **GPRM** with and without **Adaptive Sampling**

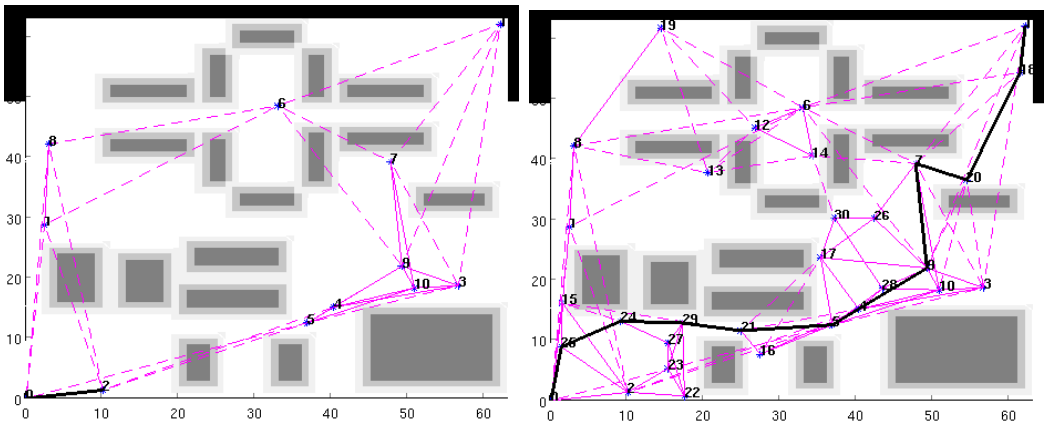
Map#	Number of Samples Required ($p_{min} = 0.8$)		
	basic GPRM	AGPRM	AGPRM η
1	62 ($p_s = 0.896$)	30 ($p_s = 1.0$)	2.07
3	72 ($p_s = 0.889$)	32 ($p_s = 0.889$)	2.25
5	72 ($p_s = 1.0$)	61 ($p_s = 1.0$)	1.18
6	162 ($p_s = 0.889$)	64 ($p_s = 0.889$)	2.53
10	182 ($p_s = 1.0$)	52 ($p_s = 1.0$)	3.50

starts with, and sub-figure (b) represents the final solution for the map with the additional landmarks sampled, and a path shown between the start and goal query. In the results shown in Figure 4.5, the final connectivity graph on the map shows the adaptive nature of the sampling done to solve the map. There are areas in the map where more sampling was done and areas where no sampling has been done. This is something to be expected from an adaptive sampling algorithm. Maps with more complexity were also solved and Figure 4.6 - Figure 4.8 represent the solutions. Some maps have always challenged the sampling and motion planning algorithms, one of them is the single passage map, the solution to which is given in Figure 4.9. The algorithm was able to solve the map with minimal increase in landmarks for the map.

Efficiency (η) in Table (Table 4.1) is defined as efficiency of the Adaptive Sampling based GPRM (AGPRM) and is given by the ratio of *GPRM Samples* required to the *AGPRM Samples* required for solving the given query in the given obstacle space.

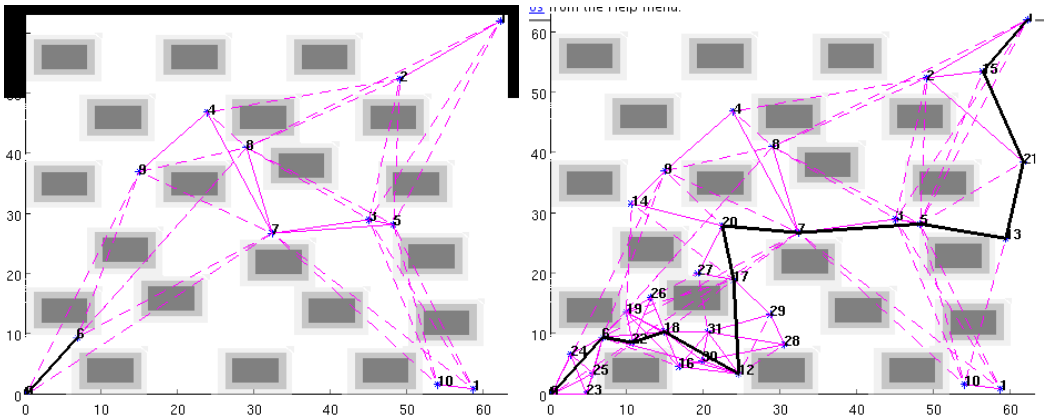
The maps discussed till now have dimensions 60 x 60 units. Figure 4.10 represents one of the largest map the algorithm was tried on, it is 150 x 150 units in area.

The maps discussed here were also solved using the basic GPRM algorithm and the results when compared with the adaptive sampling case, suggest that the number



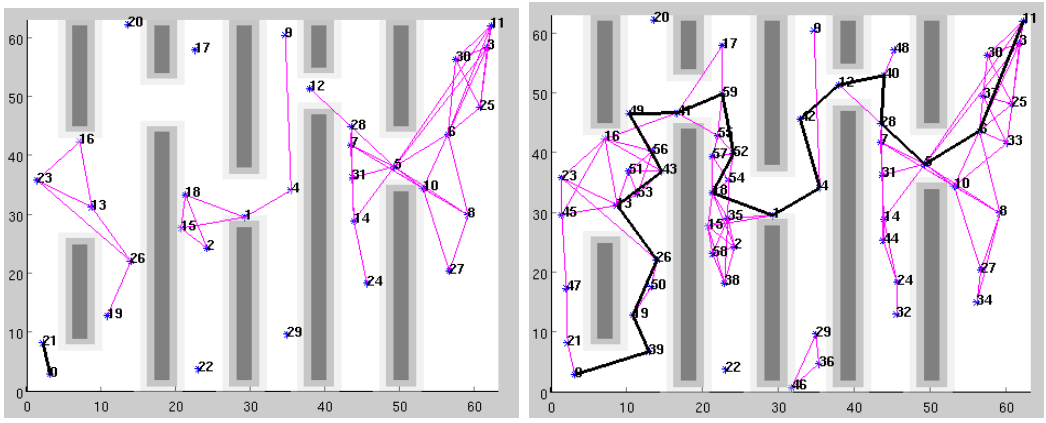
(a) Initial Landmarks (b) Query Solved

Fig. 4.5. AGPRM with Point Robot : Map 1



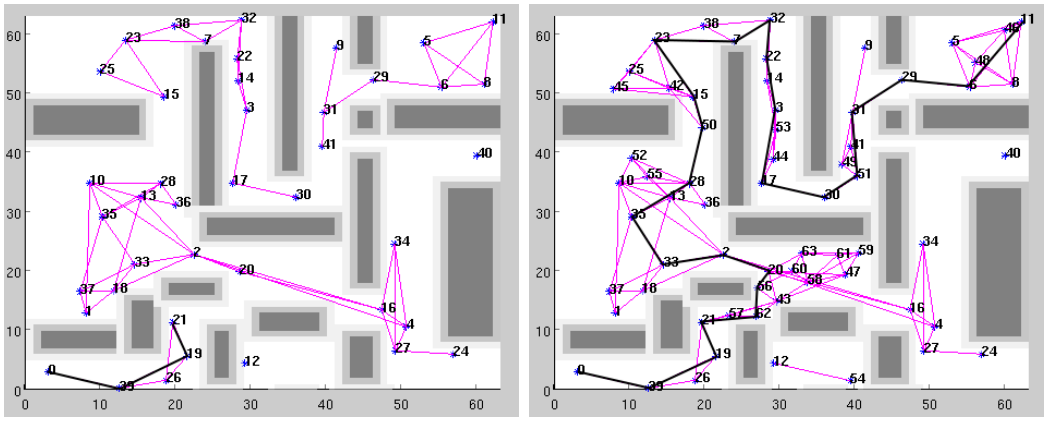
(a) Initial Landmarks (b) Query Solved

Fig. 4.6. AGPRM with Point Robot : Map 3



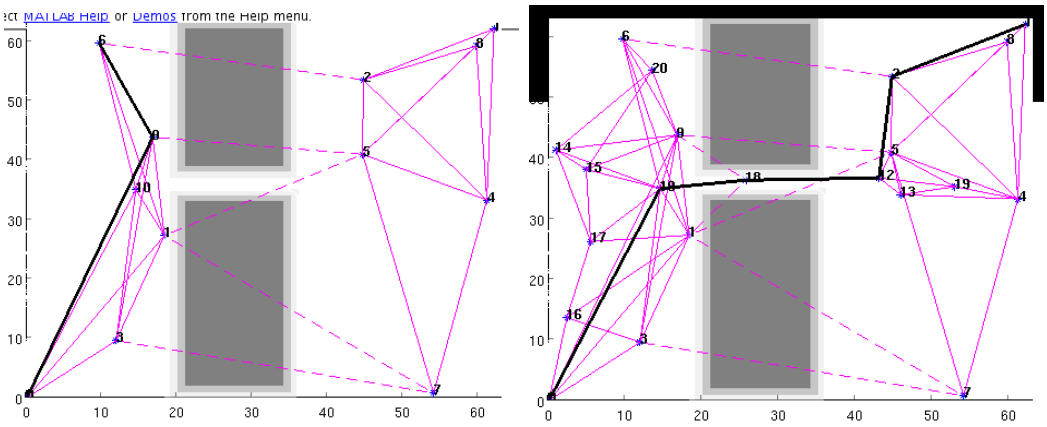
(a) Initial Landmarks (b) Query Solved

Fig. 4.7. AGPRM with Point Robot : Map 5



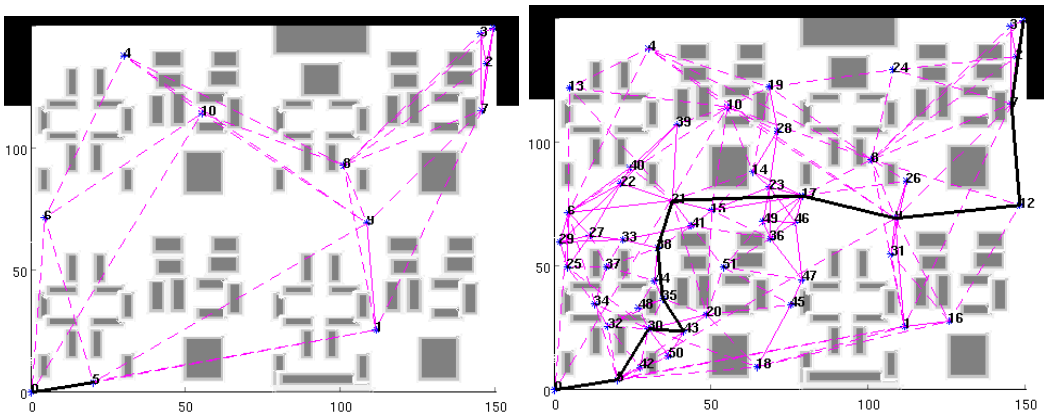
(a) Initial Landmarks (b) Query Solved

Fig. 4.8. AGPRM with Point Robot : Map 6



(a) Initial Landmarks (b) Query Solved

Fig. 4.9. AGPRM with Point Robot : Map 9



(a) Initial Landmarks (b) Query Solved

Fig. 4.10. AGPRM with Point Robot : Map 10

of samples required to solve the maps have approximately been reduced by half or more (refer Table Table 4.1).

4.4.2 n - Link Manipulator

The GPRM algorithm along with the Adaptive Sampling developed is applied to an n-link manipulator. An n-link manipulator with fixed arm length is a dynamical system with n degrees of freedom (DOF). Here we have taken a fixed-base n-link manipulator operating in a plane perpendicular to gravity with $n = 3$ (Figure 4.11), 5 and 8 links. The dynamics of the system is developed and a feedback controller is designed to stabilize the manipulator about any given configuration by linearizing the dynamical system about the configuration and using linear quadratic control techniques. This feedback controller is then applied on the non-linear system. The developed dynamics, along with the designed controller, is used in the simulations involved in the GPRM algorithm. The maps studied are of varying complexity.

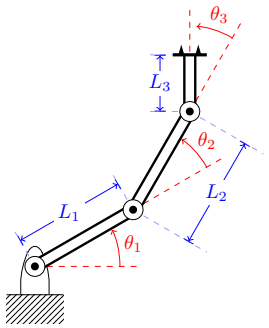
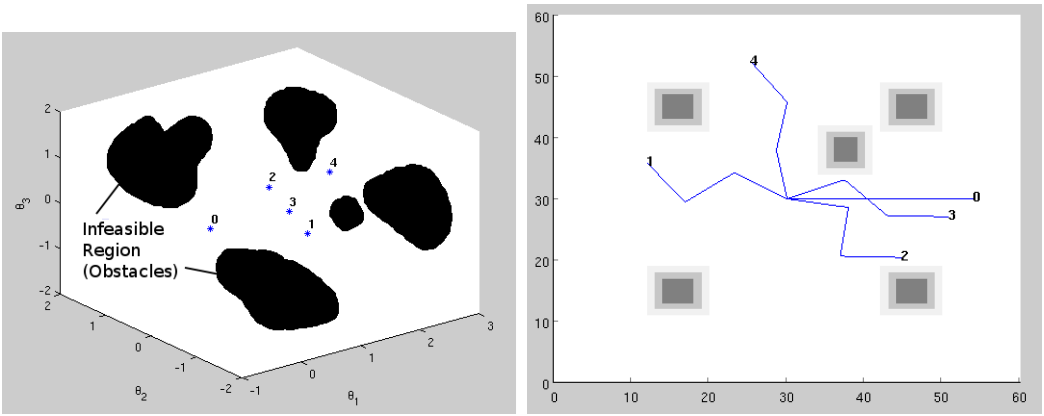


Fig. 4.11. Three-Link Manipulator

The configuration space of 3-link system is given by the link angles $[\theta_1, \theta_2, \theta_3]$. There exist an obstacle/ physical space, (Figure 4.12 (a) and (b)), in which the manipulator has to operate. Each configuration space entity is a configuration of the link manipulator in the obstacle space. And the obstacles in the obstacle/ physical



(a) Configuration Space*

(b) Obstacle Space

Fig. 4.12. Configuration and Obstacle Space (* The infeasible region representing obstacles in the configuration space is hypothetical and for understanding)

Table 4.2
Result Comparison : 3-Link-Manipulator **GPRM** with and without **Adaptive Sampling**

Map#	Number of Samples Required ($p_{min} = 0.75$)		
	basic GPRM	AGPRM	AGPRM η
1	32 ($p_s = 1.0$)	21 ($p_s = 1.0$)	1.52
2 a	102 ($p_s = 1.0$)	22 ($p_s = 1.0$)	4.64
2 b	102 ($p_s = 0.797$)	26 ($p_s = 0.797$)	3.92

space generate an infeasible region in the configuration space of the link manipulator (Figure 4.12 (a)). The landmarks generated from the configuration space/ state space have to avoid this infeasible region and hence, avoid the link configurations which clash with the obstacles in the physical space. A similar approarch during motion planning helps achieve obstacle avoidance.

The results of the GPRM algorithm along with the Adaptive Sampling methodology, on the link manipulator case has been depicted in Figure 4.13-Figure 4.15. Table (Table 4.2) compares the number of samples required by the Adaptive Sampling based algorithm for the 3-link manipulator case with basic GPRM with uniform sampling. Efficiency (η) in Table (Table 4.2) has been defined prior to Table (Table 4.1).

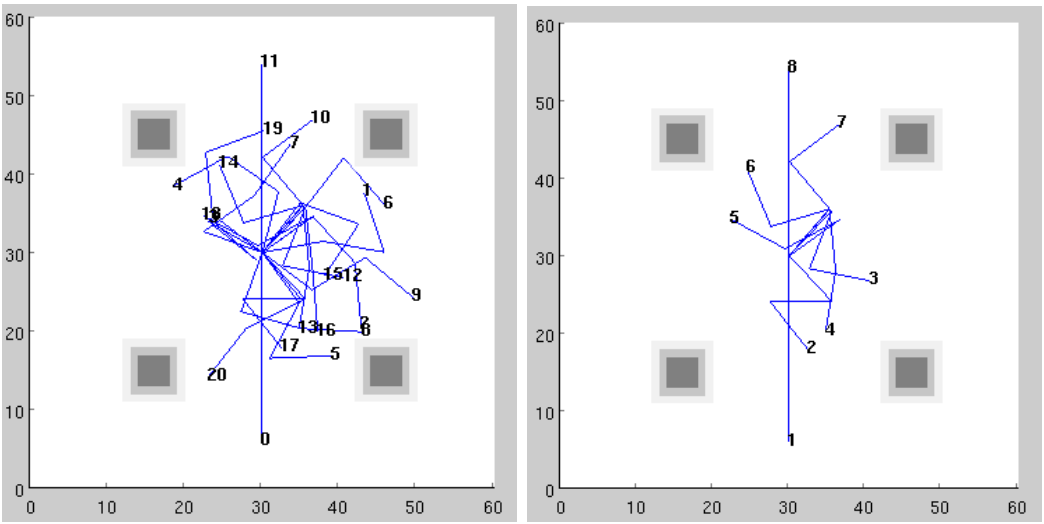
For each map, two figures are shown. Fig (a) shows all the landmarks generated while searching to reach the final configuration. Fig (b) shows the sequence of landmarks in the optimal solution. The numbering of the intermediate landmarks in Fig (b) is for understanding the sequence of travel of the link manipulator from the start to end configuration.

We studied performance on two different maps. Results for Map 1 are shown in Figure 4.13. Map 2 (Figure 4.14, Figure 4.15) was studied with two different initial configuration (i.e. Map 2 a, b):

- Map 2 a with vertically down initial configuration and,

- Map 2 b with initial configuration flat right.

The problem in Map 2b, is a hard problem because the end configuration is closer to the initial configuration, but the shortest path is blocked due to obstacles. The results indicate that the number of samples required are similar but the probability of success (p_s) gets reduced for the basic GPRM and AGPRM algorithms.

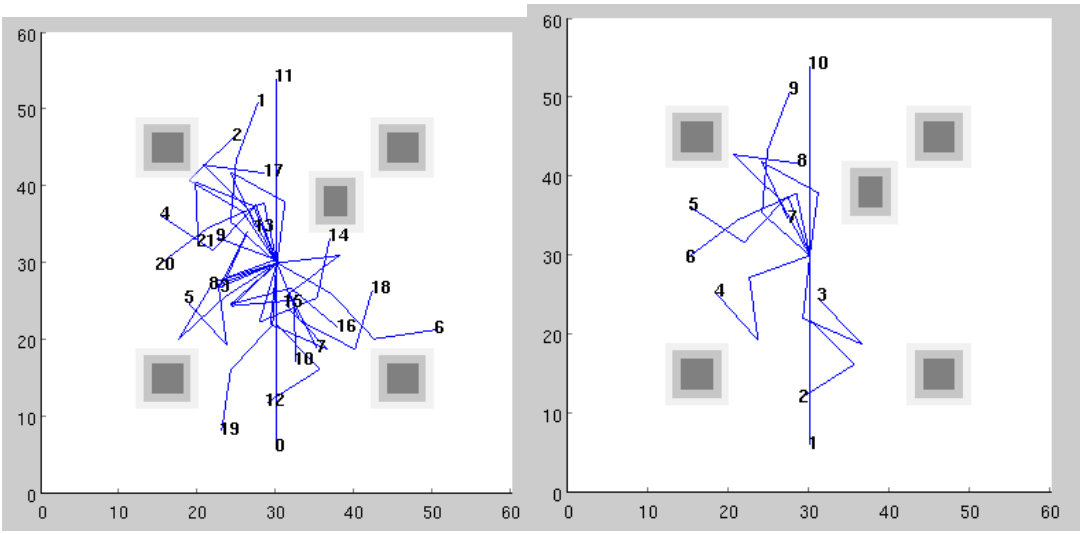


(a) Landmarks (b) Query Solved

Fig. 4.13. AGPRM with 3-Link Manipulator : Map 1

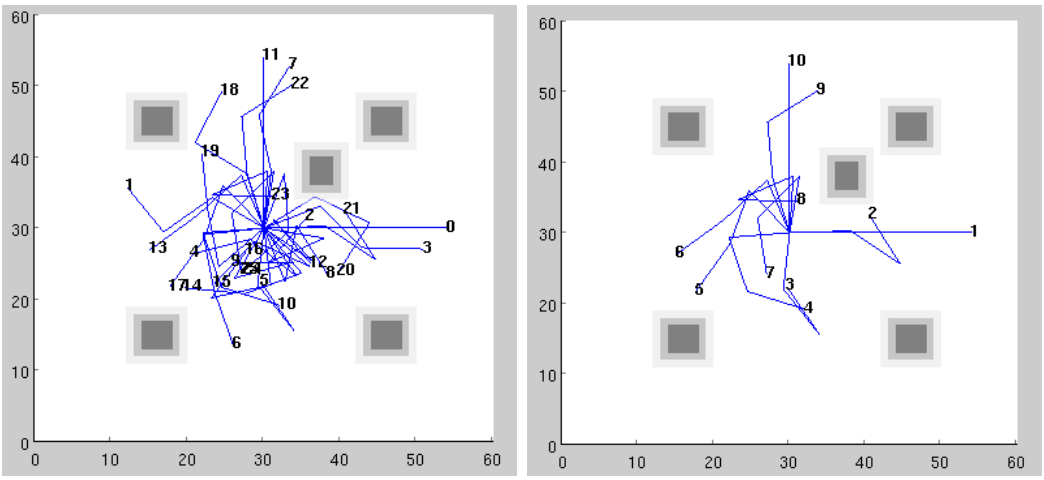
5-Link Manipulator

After improved results using AGPRM for 3-link manipulator, the algorithm was tested on higher dimensional state-space systems with highly non-linear dynamics. Figure 4.16 shows one of the results obtained from applying AGPRM to solve the motion planning problem for 5-link case with process uncertainty and the aforementioned stochastic maps (i.e. Map 1, 2a, 2b). The figures depict that the algorithm was successful in solving the 5-link case under different maps with different initial conditions. The dimension of the map has to be increased as the length of each link



(a) Landmarks (b) Query Solved

Fig. 4.14. AGPRM with 3-Link Manipulator : Map 2 a



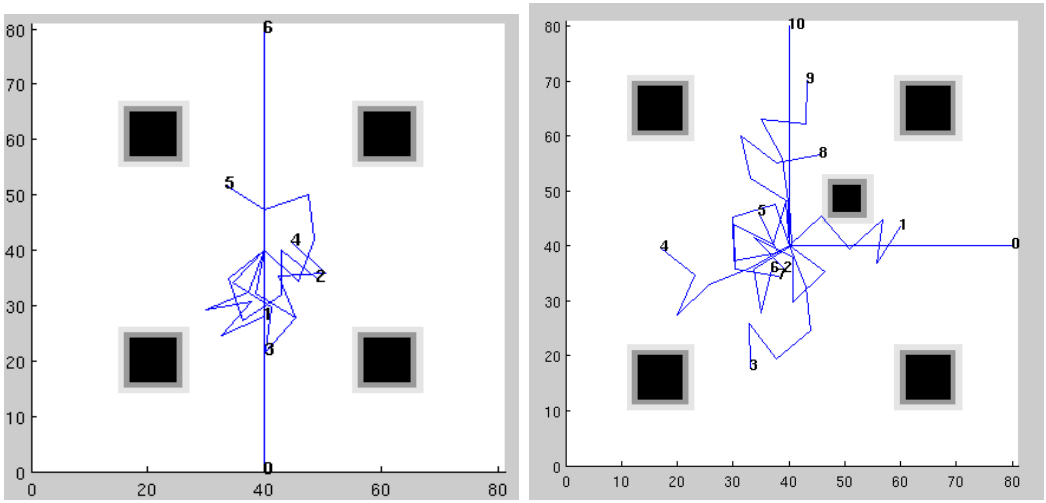
(a) Landmarks (b) Query Solved

Fig. 4.15. AGPRM with 3-Link Manipulator : Map 2 b (different initial configuration)

Table 4.3
5-Link Manipulator - AGPRM Results

Map (80×80)	No. of samples
1	180
2 b	390

was kept same, keeping the obstacle configurations similar. The Table 4.3 shows the number of samples required to solve the problem.



(a) Map 1 solution (b) Map 2 b solution

Fig. 4.16. AGPRM with 5-Link Manipulator

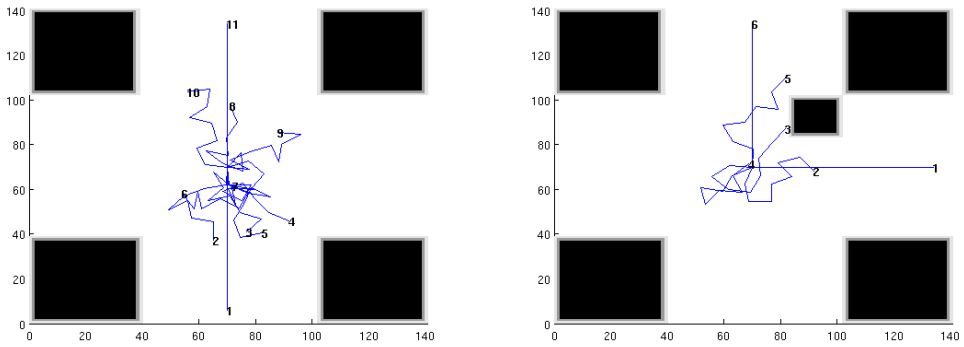
8-Link Manipulator

Figure 4.17 shows the results obtained by applying AGPRM algorithm on a 8-link manipulator, i.e. a 16 dimensional state-space system. These results essentially depicts that the algorithm is capable of handling highly non-linear high-dimension state space systems while solving the motion planning problem in presence of uncer-

Table 4.4
8-Link Manipulator - AGPRM Results

Map (140×140)	No. of samples
1	255
2 b	250

tainty. The dimension of the map has to be increased to 140×140 units of length to accommodate 8-links, with each link having the same length as for 3-link and 5-link case. Table 4.4 shows the number of samples required to solve the motion planning problem on corresponding maps for the 8-link manipulator. The number of samples required for solving the Map 2b for 8-link required less number of samples compared to solving a similar map for 5-link case, this could be because of certain on-going improvements done in the algorithm.



(a) Map 1 solution (b) Map 2 b solution

Fig. 4.17. AGPRM with 8-Link Manipulator

4.5 Conclusions

This section of the dissertation presents an adaptive sampling strategy for the generalized probabilistic roadmaps framework. The strategy was tested on an idealized point robot with fully actuated dynamics and a 3-link manipulator having 3-DOF and 6-dimensional state space with stochastic map and model uncertainty. The numerical simulations were done on several complicated maps for the point dynamics robot and for the 3-link manipulator. The results are promising when compared to basic GPRM, and suggests that a solution to complicated maps, where a basic GPRM might fail or would require a high number of landmarks, is possible with significantly less number of landmarks, using the adaptive sampling strategy. Furthermore once the efficiency of the algorithm was tested, the algorithm was applied to other highly non-linear systems with very high-dimensional state-spaces. The algorithm was successfully applied to 5-link and 8-link manipulator case (i.e. upto 16 dimensional state space systems) to solve the motion planning problem in presence of uncertainty and hence depicted the capability of solving very high-dimensional state space systems. The results here indicate that we have increased the efficiency of sampling, and the probability of success associated with the solution of GPRM algorithms along with showcasing the capability to handle highly non-linear systems with very high dimensional state-spaces. The results presented here have not been time optimized and this is a subject of ongoing research. We will also be working towards the proof of convergence of the adaptive sampling techniques and the proof and simulation results of approaching *resolution complete maps* with multiple queries.

5. MULTI-AGENT PROBLEM

5.1 Introduction

In a single-agent system only one agent interacts with the environment, where a *multi-agent system* (MAS) consists of multiple agents which execute actions and influence their surroundings. Each agent receives observations and selects actions individually, but it is the resulting *joint action* which influences the environment and generates the reward[1] for the agents. This has extremely important consequences on the characteristics and the complexity of the problem.

5.1.1 Characteristics of Multi-Agent System (MAS)

A multi-agent system results in increased complexity in both the action and state space whenever a new agent is added to the system. Since the total number of joint actions is defined as the cross-product of the individual action sets, the action space scales exponentially with the number of agents. The same is true for the state space of the multi-agent system also. Some fundamental characteristics of a multi-agent system [74, 75] are discussed below.

Environment : In single-agent system the environment is assumed to be static and hence transition and reward function do not depend on the time step t. However, when other agents are part of the environment then new state and received reward also depend on the actions selected by the other agents, i.e. environment is dynamic. Collision check and avoidance algorithms will change in such a scenario.

Homogeneous and heterogeneous agents : Agents in a MAS can be either *homogeneous* or *heterogeneous*. Homogeneous agents have identical capabilities,

[1] cost of transition

and heterogeneous agents have different designs and different capabilities, for instance different equations of motion.

Control : The control is decentralized, each agent selects action individually, but the system is affected by the joint action, that is, the combination of all selected actions.

Knowledge : This is the information an agent has about the world and the task it has to solve. An agent has specific internal knowledge : the actions it can perform, the transition and reward functions.

Observability : This is the degree to which agents either individually or as a team, identify the current world state. Ref. [76] gives the following four models for observability:

- *Individual observability* : Every agent observes the complete unique world state.
- *Collective observability* : The combined observations of all agents uniquely identify the world state. Each agent only observes a part of the state.
- *Collective partial observability* : Each agent observes part of the full state information but there are no assumptions about the combined observations of the agents.
- *Non-observability* The agents receive no feedback from the world.

Communication : An ideal situation is that the agents are able to communicate instantaneously to all agents for free and there are no limitations in the number of messages. Because of communication constraints and delay, perfect communication may not be feasible.

5.1.2 Formal Description

The parameters of the multi-agent system can be summarized as [74, 75]:

- A discrete time step t
- A ground of n agents $A = \{A_1, A_2, \ldots, A_n\}$
- A set of environments states $\boldsymbol{X}$, $\boldsymbol{x}^t \in \boldsymbol{X}$
- A set of actions $\mathcal{A}_i$ for every agent i, and $a_i^t \in \mathcal{A}_i$. The joint action $\boldsymbol{a} \in \mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the vector of all individual actions.
- A set of observations Ω_i for every agent i
- A state transition function $T : \boldsymbol{X} \times \mathcal{A} \times \boldsymbol{X} \to [0, 1]$ which gives the transition probability $p(\boldsymbol{x}^t | \boldsymbol{a}^{t-1}, \boldsymbol{x}^{t-1})$ that the system moves to state $\boldsymbol{x}^t$ when the joint action $\boldsymbol{a}^{t-1}$ is performed in state $\boldsymbol{x}^{t-1}$.
- An observation function $O : \boldsymbol{X} \times \mathcal{A} \times \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n \to [0, 1]$ which defines the probability $p(o_1^t, \ldots, o_n^t | \boldsymbol{x}^t, \boldsymbol{a}^{t-1})$ that the observations are observed by the agents $1, \ldots, n$ in state $\boldsymbol{x}^t$ after joint action $\boldsymbol{a}^{t-1}$ is performed. It can be concisely written as $p(\boldsymbol{o}^t | \boldsymbol{x}^t, \boldsymbol{a}^{t-1}, \boldsymbol{x}^{t-1})$
- A reward function $R_i : \boldsymbol{X} \times \mathcal{A} \to \mathbb{R}$ which provides agent i with a reward $r_i^{t+1} \in R_i(\boldsymbol{x}^t, \boldsymbol{a}^t)$ based on the joint actions $\boldsymbol{a}^t$ taken in state $\boldsymbol{x}^t$. The global reward function $R(\boldsymbol{x}^t, \boldsymbol{a}^t) = \sum_{i=1}^{n} R_i(\boldsymbol{x}^t, \boldsymbol{a}^t)$ is the sum of all individual rewards received by the n agents.

5.1.3 Models to tackle MAS

The different approaches of dealing with multi-agent systems are [74, 75] :

Multiagent MDP (MMDP) : This model follows the general model from MDP with two additional assumptions. First, each agent has knowledge of the global

reward. Secondly, the system has full observability, i.e. each agent observes the complete state $\boldsymbol{x}^t$.

Collaborative multiagent MDP : Each agent in this model acts individually, has full observability, and has knowledge of the individual reward and not the shared global reward. It uses *known dependencies between the agents* to create a factorized representation of transition and reward function, and hence, needs to observe the state variables of agents on which it depends.

Stochastic Games : All involved agents have to select an action a_i, and the resulting joint action $\boldsymbol{a}$ provides each player i an individual payoff $R_i(\boldsymbol{a})$. This assumes full observability and complete knowledge of reward function but action information of other agents is not available. The agents try to maximize individual reward.

Decentralized POMDP : It assumes that the observations of the agents are uncertain and is an extension of the single-agent POMDP model with collective partial observability.

The general multi-agent problem has been formulated and the different approaches of solving the motion planning problem in presence of process and sensing uncertainty was discussed above. We intend to solve the multi-agent motion planning problem in presence of process uncertainty and map uncertainty, however we assume perfect state sensors. The multi-agent problem which we would address in this dissertation is given in the following section 5.2.

5.1.4 Coordination Problem

The multi-agent motion planning problem in presence of process uncertainty and stochastic maps can be posed as an *multi-agent Markov decision process* (MMDP). In traditional methods of solving an MMDP, it is treated as a single large MDP and

standard solution techniques available for MDPs are applied, this was done by [77]. The goal of solving an MMDP should be that of finding the best/optimal policy for the system of agents. However actions are taken at an individual level of the agents and its should be ensured that without using communication (i.e. real-time data transfer not allowed), the combined actions of all the agents should result in an optimal policy for the system of agents. The problem of identifying individual policy for each agent which results in the optimal joint policy is called the **coordination problem**.

Researchers working on solving this problem have come up with possible solutions such as *coordination graphs* (CGs) as in [74] and *max-plus* algorithm in conjunction with CGs as in [78]. In [74] the solution to cooperative action selection for a system of agents (or coordination problem) is proposed as constructing a *coordination graph* and optimizing over it using variable elimination algorithm. In [78], the researchers proposed a improved optimization technique, *max-plus* algorithm, which replaces the variable elimination procedure while optimizing the *coordination graphs*.

5.1.5 Multiple Traveling Salesman Problem (MTSP)

Traveling salesman problem (TSP) is a well known combinatorial optimization problem in operational research and theoretical computer science. The problem has been solved efficiently in [79]. A generalization of *traveling salesman problem* is the *multiple traveling salesman problem* (MTSP) which consists of determining the routes for multiple salesmen. Ref. [80] discusses an overview of formations and solution procedures for MTSP. A generalization of MTSP is one in which the agents can start at multiple depots (i.e. start locations) and the type of agents can be *heterogeneous*. The generalized MTSP problem statement is as follows : Given a set of heterogeneous agents that start from distinct depots (or start locations), a set of targets (or goal locations), find an assignment of targets to be visited by each agent along with the sequence in which it should be visited so that each target is visited at

least once by an agent, all the agents return to their respective depots after visiting the targets, and the total cost incurred by the collection of agents is minimized.

Problem Statement

The different parameters in the MTSP problem [31] are:

- n targets and m vehicles located at distinct depots. (for simplification agents will be referred as vehicles here)
- $V = \{V_1, \ldots, V_m\}$, with m vertices, representing the vehicles (i.e. the vertex V_i corresponds to the ith vehicle)
- $T = \{T_1, \ldots, T_n\}$, represents n targets
- $V^i = V_i \cup T$: set of all the vertices corresponding to the ith vehicle.
- $E^i = V^i \times V^i$ denote the set of all edges (pair of vertices) corresponding to ith vehicle
- $C^i : E \to \mathbb{R}_+$ denote the cost function with $C^i(a, b)$, representing the cost of traveling from vertex a to vertex b for vehicle i.
- Cost functions are asymmetric, i.e. $C^i(a, b)$ need not be equal to $C^i(b, a)$, $\forall\, a, b \in V^i, i = 1, \ldots, m$.

A vehicle either does not visit any target or visits a subset of targets in T. If the ith vehicle does not visit any target, then its tour, $TOUR_i = \phi$ and it corresponding cost, $C(TOUR_i) = 0$. If the ith vehicle visits at least one target, then its tour may be represented by an ordered set $\{V_i, T_{i_1}, \ldots, T_{i_{r_i}}, V_i\}$ where $T_{i_l}, l = 1, \ldots, r_i$

corresponds to r_i distinct targets being visited in that sequence by the ith vehicle. And the associated cost with the tour for the ith vehicle will be given by:

$$C(TOUR_i) = C^i(V_i, T_{i_1}) + \sum_{k=1}^{r_i - 1} C(T_{i_k}, T_{i_{k+1}}) + C^i(T_{i_{r_i}}, V_i). \tag{5.1}$$

Hence we need to find tours for the vehicles such that each target is visited at least once and the overall cost defined by $\sum_{i \in V} C(TOUR_i)$ is minimized.

Ref. [80] summarizes different formulations and solution procedures for multiple traveling salesman problem. The paper discusses exact algorithms, heuristic solution procedures and transformations to single TSP, to solve MTSP. The transformation based solution approaches essentially transforms the MTSP to a single TSP on an expanded graph and uses already available solution techniques for TSPs (such as [79]). One of the major concerns with respect to transformation's solutions are their scalability with increasing number of agents. A recent work [31] presents an effective transformation method to solve the generalized[2] MTSP. In this work the generalized MTSP is transformed first to a *one in a set* asymmetric TSP (ATSP) and then this *one in a set* ATSP is transformed to a single ATSP by Noon-Bean transformation [81]. The single ATSP is solved using the Lin-Kernighan-Helsgaun (LKH) solver [79] and it is shown that solving the single ATSP on a transformed graph is equivalent to solving the generalized MTSP.

5.2 A Class of The Multi-Agent Problems

The motion planning problem for single agent in presence of uncertainty in the form of process uncertainty, and stochastic maps, has been solved using the proposed generalized sampling based motion planners, i.e. GPRM and GRRT. Once the single agent problem has been solved some of the important extensions are to : dynamic maps, multi-agent systems and problems with sensing uncertainty. We shall not solve

[2]Heterogeneous agents starting at multiple depots

problem with sensing uncertainty in this dissertation. We would like to generalize our solution methodology to multi-agent scenarios in the presence of process uncertainty and stochastic maps. The particular multi-agent problem that we would like to solve in this dissertation will be discussed in subsection 5.2.1.

Researchers solve the *coordination problem* to solve the MMDP posed by the multi-agent motion planning problem (refer subsection 5.1.4). All the successful approaches [74, 78] were top-down in solution methodology, i.e. the *coordination problem* was solved in the MMDP framework.

In this work, we intend to present an approximate approach of solving a particular MMDP for the start locations and a particular cost structure as explained in subsection 5.2.1. In Section 3 and 4 we successfully solved the single agent motion planning problem in presence of uncertainty posed as MDP which was converted to SMDP and solved thereafter. Our approach to solving the MMDP posed by multi-agents is bottom-up where we intend solve multiple MDPs for single agents and use the cost of transition and transition probabilities generated by the solution to solve the *coordination problem* and hence solve the overall MMDP posed by the systems of agents. This *coordination problem* is equivalent to the routing problem solved by the TSP research community as discussed in subsection 5.1.5. Hence we intend to use the already existing literature related to solution of MTSPs to solve the routing problem (or the coordination problem in MMDPs). Details of the approach to solve the MMDP will be discussed in further sections.

5.2.1 A Class of Multi-Agent Problems in Presence of Uncertainty

We want to solve the motion planning problem, under given scenario :

- m agents, with m initial configurations, i.e. $\boldsymbol{q}_I = \{q_{I_1}, q_{I_2}, \ldots, q_{I_m}\}$,
- n goal locations, i.e. $\boldsymbol{q}_G = \{q_{G_1}, q_{G_2}, \ldots, q_{G_n}\}$,
- Process uncertainty present in robot motion model,

- Environment given by stochastic maps, i.e. static obstacle probabilities.

Given the above multi-agent scenario, additional considerations need to be made:

1. Number of agents and number of goal locations might not be same, i.e. $m \neq n$ (general case). If, the number of agents and number of goal locations are same, then solving m single agents problems can be straightforward. But as the goal locations are different in number, some agents will have to go to more than one goal and some might not have to go to any goal. Hence with the given scenario one has to solve a *routing problem* (or the *coordination problem* as discusses in subsection 5.1.4) for the multi-agent system.

2. Due to presence of other agents in the given static stochastic map, in addition to collision with obstacles, we need to address collision with other agents. Furthermore, this has to be a real-time solution.

3. Typically in a multi-agent system, one can assume presence of heterogeneous agents, i.e. agents having different capabilities. Thus, there is a need to discuss homogeneous as well as heterogeneous agents scenario.

The *routing problem*, in item 1 above, has been solved extensively in the *traveling salesman problem* (TSP) research community, primarily in deterministic framework [79]. The generalized multi-agent routing problem has been posed as a *multiple traveling salesman problem* (MTSP) [80,31] and there have been multiple approaches to solve it. We aim to use the already existing routing problem solution techniques for multi-agent systems developed in [31], and in synergistic manner apply it along with GPRM to the multi-agent systems in presence of process uncertainty and stochastic maps. We expect this generalized technique - *multi-agent adaptive sampling generalized probabilistic roadmaps*(MAGPRM), will help us solve the feedback motion planning problems in high dimensional state spaces under uncertainty in a multiple agent scenario, with GPRM as the underlying framework, which successfully solved the single agent scenario.

The section 5.3 will discuss the solution approach for the multi-agent motion planning problem under uncertainty.

5.3 Solution Approach : **Problem 2**

In order to solve the motion planning problem under uncertainty involving multi-agent systems, stated in **Problem 2** (refer subsection 1.2.2), a solution methodology is proposed using the *multiple traveling salesman problem* (MTSPs) [31].

We have solved the motion planning problem under uncertainty for a single agent case (**Problem 1**). We propose to address the multi-agent motion planning problem under uncertainty by solving the following sub-problems using proposed methodologies along with GPRM as follows:

Routing Problem : This is the problem of identifying which agents will go to which goal locations. Hence given m agents and their initial configurations, $\boldsymbol{q}_I = \{q_I(i)\}$, $i = 1, \ldots, m$, and n target final configurations, $\boldsymbol{q}_G = \{q_G(i)\}$, $i = 1, \ldots, n$, and given $m \neq n$ (general case), how to determine which set of goals any given agent will go to. Due to the condition of $m \neq n$, some agents may go to multiple goal configurations and/ or other agents may not go to any goal configuration.

Solving the "routing problem" amounts to solving a *passive/offline co-ordination* problem, i.e., co-ordination between agents before starting the execution of the planning. MTSP is the tool through which we plan to solve this routing problem. We will solve the original problem of multi-agent motion planning under uncertainty using GPRM in conjunction with MTSP, in a hierarchical fashion. The solution of GPRM will generate transition costs and probabilities between any pair of goal locations[3] for any given agent. Using these cost and transition probabilities, the MTSP algorithm will be used to solve the "routing problem"

[3]Goal locations here includes the initial configurations and the desired configurations

and hence, solve the multi-agent motion planning problem under uncertainty.

Inter-Agent Collision Avoidance : Inclusion of multi-agents in the domain creates a need to address the *inter-agent collision avoidance* through an updated collision detection module which, in general, cannot be assured by the routing problem. To address this problem, we shall use *collective partial observability* (ref. subsection 5.1.1), and develop schemes with guaranteed collision avoidance.

Homogeneous and Heterogeneous Agents : In case of all agents being homogeneous in dynamics and capabilities, a single graph based solution can be provided using GPRM along with MTSP. For heterogeneous agents, solving the motion planning problem will involve constructing topological graphs in GPRM for every type of agent present in the system, i.e. given m agents consisting of t type of agents will need t number of graphs and a number of GPRMs solutions.

5.4 Routing Problem as MTSP

This section is focused on solution of the *routing problem* explained in the section 5.3. We intend to solve this sub-problem using *multiple traveling salesman problem*(MTSP) solution methodology. The details of MTSP were presented in subsection 5.1.5. MTSP solution methodology is primarily developed for deterministic framework, but we intend to use it in conjunction with GPRM/AGPRM, developed for single agent motion planning problem under uncertainty in Section 3 and 4. In this section we develop the synergistic coupling of MTSP solutions with GPRM to solve the *routing problem*, a sub-problem of the actual problem of *multi-agent motion planning motion planning under uncertainty*.

We will first re-iterate some existing definitions and develop some new ones to mathematically present the problem and then solve it.

5.4.1 Definitions

General

$\mathcal{C}$: The configuration space of the agent. A configuration is given by q, i.e a generalized position.

$\mathcal{X}$: The state-space of the agent. A state is given by, $x = (q, \dot{q})$, i.e. comprised of generalized position and generalized velocity.

l^i : i^{th} landmark, i.e. a sample in state-space ($l^i \in \mathcal{X}$).

$\mathcal{L}$: Set of landmarks, i.e. $\mathcal{L} = \{l^i\}, \forall i$, on a given stochastic map.

$\mathcal{G}$: Set of start and goal locations[4] in a multiple agents scenario on a given stochastic map. These start and goal locations are a part of the set of landmarks, i.e. $\mathcal{G} \subset \mathcal{L}$

$\mathcal{A}$: Set of all agents, $\{a_i\}, \forall i$. (a_i is the i^{th} agent)

q_i : Configuration of a_i, where $q_i \in \mathcal{C}$

Q_k : Configuration of all agents at time step k, i.e. $Q_k = \{q_1^k, q_2^k, \ldots, q_{|\mathcal{A}|}^k\}$ and $Q_k \in \mathcal{C}^{|\mathcal{A}|}$

$\mathcal{U}$: Set of controls. ($u \in \mathcal{U}$)

$\mathcal{M}$: Set of lower level controllers. ($\mu \in \mathcal{M}$)

[4] landmarks

Controls

$\mu(\cdot)$: The lower level (say $Level_1$ in MAGPRM or *lowest level* of GPRM as in Figure 3.1 or Figure 5.1) controller for the agents. In MAGPRM it is a feedback controller, parametrized using landmark. Also $\mu \in \mathcal{M}$ and :

$$\mu(\cdot) : \mathcal{X} \mapsto \mathcal{U}$$

$\pi(\cdot)$: Policy operator at $Level_2$ of MAGPRM (or *top level* of GPRM as in Figure 3.1 or Figure 5.1), i.e. solution of GPRM for a single agent.

$$\pi(\cdot) : \mathcal{L} \mapsto \mathcal{M}$$

Given a goal landmark, l^{goal}, π is a solution provided by GPRM developed in section subsection 3.3.2. This solution is dependent on l^{goal} and hence the operator π can be rigorously written as follows:

$$\pi(\cdot \ ; \ l^{goal}) : \mathcal{L} \mapsto \mathcal{M}$$

$\gamma(\cdot)$: An operator at $Level_3$ of MAGPRM (need to introduce a new level above $Level_1$ and $Level_2$).

$$\gamma(\cdot) : \mathcal{A} \times \mathcal{G} \mapsto \mathcal{G}$$

Let this be the solution of MAGPRM, i.e. given a particular $a_i \in \mathcal{A}$ and the location($\in \mathcal{G}$) of a_i, say $g \in \mathcal{G}$, the operator outputs the next goal location for a_i, i.e. $g' \in \mathcal{G}$. This g' parametrizes the $Level_2$ $\pi(\cdot)$ operator, i.e.:

$$\pi(\cdot \ ; \ g') : \mathcal{L} \mapsto \mathcal{M}$$

Furthermore in terms of goal landmark, $l_i^{goal} \in \mathcal{L}$ for the i^{th} agent, the location $g' \in \mathcal{G}$ where $\mathcal{G} \subset \mathcal{L}$, is given by :

$$g' = l_i^{goal}, \text{ and hence}$$
$$\pi(\cdot\ ;\ l_i^{goal}) : \mathcal{L} \mapsto \mathcal{M}, \text{ for } i^{th}\ agent$$

Solution from GPRM is a feedback policy, i.e. given any landmark the policy will suggest which should be the next landmark in the domain. Ideally we would like the operator of $Level_3$ of MAGPRM to also be a feedback policy. We will be discussing this in detail in subsection 5.4.3.

5.4.2 Solution of MTSP

In [31], a solution methodology is proposed for the generalized MTSP problem formulation stated in subsection 5.1.5. The solution methodology involves two transformation steps:

- Converting a generalized MTSP to a *one-in-a-set ATSP* (where ATSP : Asymmetric Traveling Salesman Problem).
- Converting a one-in-a-set ATSP to a single ATSP. This is done using the *Noon-Bean Transformation* [81].

The generalized MTSP is posed as a single ATSP by the proposed transformations which involve cost modifications. The single ATSP can be solved using the well-known TSP solver, LKH [79]. Solving the single ATSP and working backwards gives the solution to the generalized MTSP. Details of the algorithm developed can be seen in [31].

5.4.3 Solving Multi-Agent Systems in Presence of Uncertainty

In this section, the single agent generalized sampling based motion planners (GPRM) will be generalized to MAGPRM (multi-agent AGPRM), to solve the multi-agent system problem using the MTSP solution methodology stated in the previous section along with the GPRM technique.

As GPRM was a hierarchical approach , as shown in Figure 3.1, to solve the MDP posed by the motion planning problem in presence of process uncertainty and stochastic maps, the proposed algorithm, MAGPRM, for solving the multi-agent motion planning involves introduction of a new level in the existing hierarchy. GPRM solved the single agent case, as shown in Figure 3.1 or Figure 5.1, in a hierarchical fashion. The lowest level (say $Level_1$) in GPRM solves the motion planning problem between one landmark to another and inherently was generating the cost of transition and transition probabilities between two landmarks. These transition probabilities and costs induced an abstract MDP, on the discrete set of landmark states, i.e. the higher level ($Level_2$ in Figure 5.1). We used Dynamic Programming to solve the $Level_2$ abstract MDP with the transition cost and probabilities generated by the $Level_1$. Hence using these two levels the single agent motion planning problem, between any two start and goal locations, under uncertainty is solved.

In a multi-agent motion planning scenario, having m agents and n goal locations, there are two additional sub-problems that need to be solved, as discussed in section 5.3, namely *routing problem* and *inter-agent collision avoidance*. In order to solve the *routing problem* we introduce $Level_3$ which comprises of a graph whose vertices are landmarks ($g \in \mathcal{G}$), i.e. the m agents' initial locations and the n desired goal locations. The edges of the graph in $Level_3$ are abstract connections from "agent locations to goal locations" and "goal to goal locations". "Agent locations to agent locations" connections are avoided as a part of assumption that an agent should not go to another agent's location.

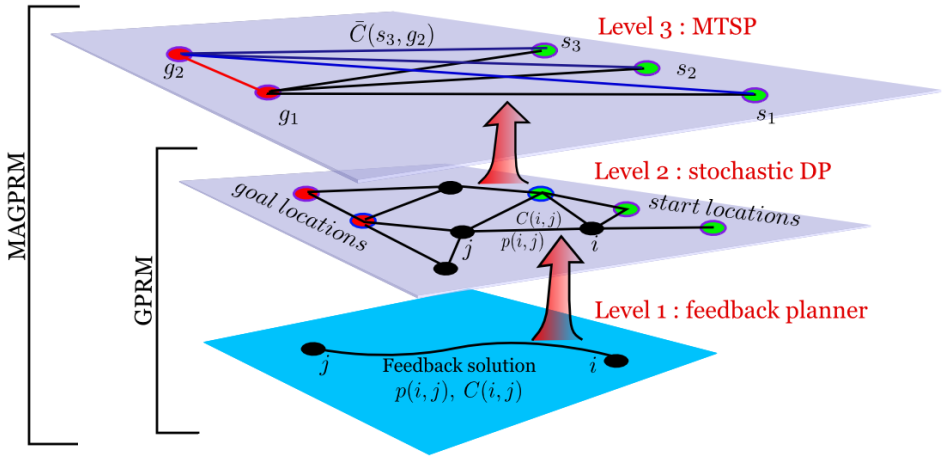


Fig. 5.1. Depicting Hierarchical Planning in Levels (for Multi Agents)

Using GPRMs the cost of transition and the path probability associated with these edges in $Level_3$ is computed (Figure 5.1). These costs and transition probabilities associated with every edge is specific to agents, but gets simplified in case of presence of multiple agents of same type. The number of GPRMs that needs to be solved are $2n(m+n)$[5]. Using the computed costs[6] and after two cost transformations, as discussed in subsection 5.4.2 the prospective MTSP algorithm [31] via the LKH solver solves the *routing problem*, i.e. allotment of goal locations to different agents.

In this MTSP level, i.e. $Level_3$ of MAGPRM, the solution is an operator operator $\gamma(\cdot)$. Currently the solution of MTSP, for each agent, is a sequence of goal locations to be visited. This solution ($\gamma(\cdot)$) is definitely not a feedback policy, i.e. if the sequence is broken by the agents due to any plausible reason, the policy does not remain optimal. Hence in order to make $\gamma(\cdot)$ a feedback policy additional considerations need to be made.

In $Level_3$ the abstract graph consists of nodes which are the agent/start locations ($s \in \mathcal{G}$) and the goal locations ($g \in \mathcal{G}$) and $\mathcal{G} = \{s\} \cup \{g\}$. The cost of the edges of this graph has been calculated, by running a number of GPRMs, and stored. In the event that an agent deviates and visits a non-assigned goal location, then the operator $\gamma(\cdot)$ should be able to provide the next best move for the agents with the remaining set of unvisited goal locations. In order to make the operator $\gamma(\cdot)$ a feedback policy, every time a deviation from the assigned sequence of goal locations is witnessed, the MTSP algorithm needs to recompute in *real-time* the operator $\gamma(\cdot)$ with the current agent locations and the remaining unvisited goal locations.

[5]These are the number of edges that needs to be evaluated. m agents to n goals and vice versa gives $2mn$ edges, n goals to n goals gives $2n^2$ edges and hence the total is $2n(m+n)$

[6]The MTSP algorithm only takes costs as input, the costs computed by GPRM do take into account the transition probabilities.

The set of agent and goal locations are provided a priori and these locations are treated as landmarks in the GPRM level. In an event of change of these location ($\in \mathcal{G}$), i.e. addition/deletion of new goal locations, addition/deletion of new agent locations, after a solution of MAGPRM has been found, the new locations will have to be added as additional landmarks and it has to be connected to the existing graph via appropriate feedback controllers for different agents. This will result in modified graphs at $Level_2$ of MAGPRM and the cost for the abstract graph edges at $Level_3$ of MAGPRM needs to be recomputed. Once the updated costs are available, the proposed MTSP solution procedure needs to be re-run, to get the new policy $\gamma(\cdot)$. In order to have a well connected graph at GPRM level after the additional locations are added, this step needs to be done *offline* to ensure connectivity. The notable part here is that even if there is a change of set of locations after a MAGPRM solution is found, the addition computational burden is minimal if the GPRM graphs are well connected as we need to only solve the MTSP.

5.4.4 Multi-Agent GPRM (MAGPRM) Algorithm

The proposed methodology of solving the multi-agent motion planning problem is summarized in 5.1 below.

Algorithm 5.1: Multi-Agent GPRM (MAGPRM)

Data: Set of agents ($\mathcal{A}$), start locations $\boldsymbol{x}_0$, goal locations $\boldsymbol{x}_g$, p_{min} for the environment

1 **for** i^{th} *agent at start location* $x_{0_i} \in \boldsymbol{x}_0$ **do**
2 **for** j^{th} *goal location,* $x_{g_j} \in \boldsymbol{x}_g$ **do**
3 **while** $p_s(x_{0_i} \rightarrow x_{g_j}) < p_{min}$ **do**
4 **if** i^{th} *agent's type already evaluated* **then**
5 Use already existing roadmap to build and connect further;
6 Construct AGPRM, parametrized with goal location x_{g_j} and agent-type of i^{th} agent;
7 Construct a cost of transitions matrix for each agent-type (i.e. cost of transitions between *agents* $\leftrightarrow$ *goals* and *goals* $\leftrightarrow$ *goals*);
8 Solve the *routing problem* for each agent, using the above generated costs in prospective MTSP algorithm;

The proposed algorithm solves the general[7] multiple agent motion planning problem in presence of process uncertainty and stochastic maps. This algorithm will provide an *offline* routing solution for individual agents. The real-time implementation of this solution requires an *inter-agent collision avoidance* module which will be discussed in the following section 5.5.

The algorithm constructs a roadmap using AGPRM between each pair of start and goal locations. The loop at *line 3* depicts this; with each pair a new AGPRM, parametrized at the current goal location, is solved.

[7] Involving heterogeneous agents

In the case of, multiple agents of same agent-type, *lines 4-5* ensures that the already existing roadmap and the already existing landmarks in the domain are used to build further or to find a solution to the motion planning between ith agent and the jth goal location.

In the case of heterogeneous agents, different roadmaps for different types of agents have to be constructed and hence, is more computational intensive. The landmarks might still be shared but transition costs and transition probabilities calculations will involve running the simulations and constructing a different roadmap each time a new type of agent comes into the system. *Line 7* emphasizes this feature of the algorithm.

Once the cost of transitions between each start and goal locations are computed, the routing problem is solved using the prospective MTSP algorithm. The solution outcome of MAGPRM is the sequence of goal locations to be visited by individual agents, which takes into account the process uncertainty in the dynamics propagation of the agents and traversal along a stochastic map.

5.4.5 Probability of Success for MAGPRM

An important point to be discussed here is the probability of success (p_{path}) associated with the final paths suggested by the MAGPRM algorithm. Inherently there is a threshold probability (p_{min}) provided for the map when performing a GPRM, which ensures the solution has $p_{path} \geq p_{min}$. A GPRM run between each pair of start and goal location gets a probability of success in accordance to it. The cost of these transitions from start to goal locations generated from GPRM incorporate the probability of success in it. These costs are used by the MTSP algorithm at $Level_3$ of MAGPRM. The chosen MTSP algorithm does not take constraints such as:

- limiting the number of goal locations to be visited by an agent.

- the probability of success of the final paths to have a minimum threshold.

Hence the hard constraint of finding a solution having probability of success greater than threshold provided cannot be ensured at MTSP level, and hence sometimes the solution by MAGPRM may have probability of success for agents less than the threshold.

Let path probability for MAGPRM be denoted by p^M_{path} and that of GPRM as p^G_{path}. MAGPRM's solution is a combination of multiple GPRM solution at $Level_2$ and as probabilities are multiplicative, this enforces the upper bound for p^M_{path}, i.e. $p^M_{path} \leq \min p^G_{path}$ always. Hence the p_{min} for GPRM directly is not applicable to MAGPRM and it has the relationship

$$p^M_{min} < p^G_{min}. \tag{5.2}$$

As mentioned above the p^M_{min} is not enforced as a hard constraint when MTSP solution is found, due to lack of capabilities of prospective MTSP algorithm to handle constraints. An adaptive solution to this problem can be formulated as follows : in an event of, $p^M_{path} < p^M_{min}$, the algorithm increases the p^G_{min} for GPRM solutions in $Level_2$, which can be achieved by *adaptive sampling*, if possible. But there is a possibility of no improvement in p^M_{path}, under which condition the solution outcome of MAGPRM is reported as such. This proposed adaptive solution has not been implemented in this dissertation.

5.5 Inter-Agent Collision Avoidance

The multi-agent scenario will introduce moving agents in the system. In addition to collision avoidance module responsible for detecting the collisions of agents with the obstacles, another *online* collision avoidance and detection module is required to handle inter-agent collision. This section is focused on developing the *online* collision detection and avoidance module.

5.5.1 Definitions

General

$\mathcal{P}_i$: Priority of a_i. The priority of agents are assigned (heuristically) based on the type of agent. More information when **Alert zone** is discussed ahead.

$d(\cdot)$: Function defining distance between two agents,

$$d(a_i, a_j) = ||q_i - q_j||$$

Collision Zone

$v_{max}^{t_i}$: maximum velocity of agent-type t_i [8], for i^{th} agent.

ϵ^{t_i} : radius of collision, for a particular agent-type.

$collision(\cdot)$: Function defining collision between two agents as:

$$\epsilon = \max\{\epsilon^{t_i}, \epsilon^{t_j}\}$$

$$collision(a_i, a_j) = \begin{cases} 1 & \text{if } d(a_i, a_j) \leq \epsilon, i \neq j \\ 0 & \text{else} \end{cases}$$

$COLLISION(\cdot)$: Collision function for checking collision in between all agents.

$$COLLISION(Q_k) = \begin{cases} 1 & \text{if } \exists\ i, j : collision(a_i, a_j) = 1 \\ 0 & \text{else} \end{cases}$$

Alert Zone

Ω_i : **Alert-zone** for the i^{th} agent, $\Omega_i \subset \mathcal{C}$

[8] This is required for heterogeneous (i.e. multiple agent-type) multi-agent case

r^{t_i} : radius of the **alert-zone** of i^{th} agent of agent-type t_i, also assume $r^{t_i} = kv^{t_i}_{max}$, where k is some constant. r^{t_i} and ϵ^{t_i} need to follow:

$$v^{max} = \max\{v^{t_i}_{max}, \forall i\}$$
$$\epsilon^{t_i} < r^{t_i} - v^{max} * dt$$

which ensure that after entering an alert zone, even traveling with maximum velocity, an agent cannot enter the collision zone of another agent after one time step.

Priority assignment : Agents are assigned priority by arranging them with r^{t_i} in descending order, i.e. agents with largest r^{t_i} will have highest priority. And with the same agent-type, t_i, the priority is assigned randomly.

$alert(\cdot)$: Function defining alertness between two agents as:

$$alert(a_i, a_j) = \begin{cases} 1 & \text{if } d(a_i, a_j) \leq r^{t_i} \;||\; d(a_i, a_j) \leq r^{t_j}, i \neq j \\ 0 & \text{else} \end{cases}$$

$ALERT(\cdot)$: Alert function for checking alertness in between all agents.

$$ALERT(Q_k) = \begin{cases} 1 & \text{if } \exists\ i, j : alert(a_i, a_j) = 1 \\ 0 & \text{else} \end{cases}$$

$q_j \in \Omega_i$: a_j is in **alert-zone** of a_i, i.e. $alert(a_i, a_j) = 1$ & $d(a_i, a_j) \leq r^{t_i}$. ($\implies q_i \in \Omega_i$)

$\mathcal{S}^1_i$: Set of agents in **alert-zone** of a_i (in the 1^{st} level), i.e. $\mathcal{S}^1_i = \{a_j \mid q_j \in \Omega_i\}$

n^1_i : Number of agents in the set $\mathcal{S}^1_i$, i.e. $n^1_i = |\mathcal{S}^1_i|$

$\mathcal{P}_{max}(\cdot)$: The highest priority in a set of agents, for example in $\mathcal{S}_i^1$,

$$\mathcal{P}_{max}(\mathcal{S}_i^1) = \max_j \{\mathcal{P}_j \mid a_j \in \mathcal{S}_j^1\}$$

$a_{max}(\cdot)$: The highest priority agent in a set of agents, for example in $\mathcal{S}_i^1$,

$$a_{max}(\mathcal{S}_i^1) = \arg\max_{a_j \in \mathcal{S}_i^1} \{\mathcal{P}_j\}$$

Controls

$\beta(\cdot)$: The decision operator to avoid possible collision.

$$\beta(\cdot) : \mathcal{X}^{|A|} \times \mathcal{A} \times \mathcal{M} \mapsto \mathcal{M} \cup \{\mu_0\}$$

where μ_0 is a local controller that stops the agent at its current position. Given all agents, their states and their current controllers, this operator takes the decision for every $a_i \in \mathcal{A}$, i.e. which controller $\mu_k^i(\cdot)$ the i^{th} agent should take at current time step, k. The control options as output of $\beta(\cdot)$ are restricted to:

continue : Use the already planned $\mu_k^i(\cdot)$ for a_i, i.e. no change required.

stop : Replace $\mu_k^i(\cdot)$ with $\mu_0(\cdot)$ for a_i, i.e. the agent needs to stop at the current position.

evasive action : Pick a new controller $\tilde{\mu}_k^i(\cdot) \in \mathcal{M}$ for a_i.

5.5.2 The Algorithm

Algorithm 5.2: Collision Detection and Avoidance

Data: $\mathcal{A}, \mathcal{M}, \mathcal{P}, Q_0, ALERT(Q_0) = 0,$

Result: $COLLISION(Q_k) = 0, \forall k$

for *every* t_k **do**
  Initialize $\forall i, \mu_k^i \leftarrow$ **continue**;
  if $ALERT(Q_k) = 1$ **then**
    Construct the set of agents, $\mathcal{B} = \{a_i \mid \exists j, alert(i,j) = 1\}$;
    if $COLLISION(Q_k) = 1$ **then**
      return **fail**;
    else
      $\mathcal{B}'$: Sort $\mathcal{B}$ based on priority $\mathcal{P}$;
      for *every* a_i, *with* $\mu_k^i \leftarrow$ **continue** | **take evasive action**; *starting with* $a_i = a_{max}(\mathcal{B}')$ **do**
        for *every* $a_j \in \mathcal{S}_i^1 \backslash a_i$ **do**
          $\mu_k^j \leftarrow$ **stop**, i.e. $\mu_k^j \leftarrow \mu_0$;
        for *every* $a_j \in \mathcal{S}_i^1 \backslash a_i$ **do**
          Check for future collision, i.e. calculate $collision^{k+1}(a_i, a_j)$;
          if $collision^{k+1}(a_i, a_j) = true$ **then**
            $\mu_k^i \leftarrow$ **evasive action**, i.e. $\mu_k^i \leftarrow \tilde{\mu}_k^i$;
  else
    continue;
  Propagate a_i using μ_k^i, $\forall i$;

This algorithm for the collision detection and avoidance is for every agent at every time-step of propagation. The functions used in the algorithm have been explained in the subsection 5.5.1. With Q_k depicting the set of configuration of all the agents,

the function $ALERT(Q_k)$ is checked for every time step k and if its triggered then the *collision detection and avoidance* module ensures that collision in-between agents at every time-step is avoided.

In the event of $ALERT(Q_k)$ being triggered, a set of all those agents for which $alert(\cdot)$ is triggered is created. All these agents are sorted such the highest priority agent is arranged first[9]. Starting with the highest priority agent, as listed in the algorithm from lines 10-15, the agents can either **continue, stop** or take **evasive action**. An analysis of the algorithm is discussed in Appendix 2. A formal proof showing that this online *collision detection and avoidance* module will ensure that the agents, while following the solution from MAGPRM, will not collide with each other at any time-step and hence, the complete trajectory of the agents will be collision free is given in Appendix 2.

Hence, an online implementation of this *collision detection and avoidance* module along with the solution outcome of MAGPRM solves the motion planning problem for multiple homogeneous/ heterogeneous agents in presence of process uncertainty and stochastic maps.

[9]Any possible conflict, in probable next action of different agents, will not arise by handling of agents with respect to their priority. Details covered in Appendix 2.

5.6 Results and Discussion

In this section, we will detail the application of the multi-agent GPRM (i.e. MAGPRM) algorithm to scenarios with homogeneous and heterogeneous agents in the problem. The agents involved in these numerical experiments are dubins car and a simplified three dimensional vehicle, mimicking a helicopter.

5.6.1 Vehicle Models Used

In order to apply the MAGPRM algorithm to heterogeneous agents scenario atleast two different types of agents were required. The numerical experiments done using MAGPRM involves the following two types of robot models used along with their specific feedback controllers.

Nonholonomic Unicyle robot

This is the same as explained in subsection 3.4.2. The equations of motion are given by :

$$\dot{x} = v\cos\theta + w_x \tag{5.3}$$

$$\dot{y} = v\sin\theta + w_y \tag{5.4}$$

$$\dot{\theta} = \omega + w_\theta \tag{5.5}$$

where (x, y, θ) represents the pose of the robot, the velocity v and the angular velocity ω represents the control inputs to the problem and w_x, w_y and w_θ are the uncorrelated noise terms for the different states of the robot. A sampled pose is in the (x, y, θ) spaces and the local feedback controller used to stabilize the robot about these sampled equilibrium configurations is given by [42] which is a dynamic feedback linearization-based controller.

Simplified 3D helicopter robot

A simplified three-dimensional helicopter robot is constructed using a Dubins car for two-dimensional traveling (as in subsection 3.4.2) and a double integrator for the altitude traversal ($z-$direction). Hence the dynamics of this simplified robot can be given by:

$$\dot{x} = v\cos\theta + w_x \tag{5.6}$$

$$\dot{y} = v\sin\theta + w_y \tag{5.7}$$

$$\dot{\theta} = \omega + w_\theta \tag{5.8}$$

$$\ddot{z} = u_z + w_z \tag{5.9}$$

where (x, y, θ, z) represents the pose of the robot, the ground velocity (two-dimensional) v, the angular velocity ω and u_z, the input forces in the $z-$direction, represents the control inputs to the problem. And w_x, w_y, w_θ and w_z are the uncorrelated noise terms for each of the states of the robot. Our sampled poses are in the $(x, y, \theta, z, \dot{z})$ spaces. The local feedback controllers can stabilize the robot about any of the equilibrium configurations sampled.

A dynamic feedback linearization-based controller design is chosen as in subsection 3.4.2 for the Dubins car model. A LQR based feedback controller is designed for the double integrator in $z-$direction as in subsection 3.4.1.

Uncertainty was added to the robot motion model by adding white noise to the robot dynamics equations are aforementioned, with the intensity of the white noise being approximately a fraction of the maximum allowable vehicle linear and angular speed, i.e. the noise in the x, y equations had intensity equal to 30% of the maximum allowable linear speed ($\sigma_{x,y} = 0.3\, v_{max}$), whereas the noise in the θ equation has intensity equal to 10% of the maximum allowable angular speed ($\sigma_\theta = 0.1\,\omega_{max}$). For the $z-$direction double integrator in Equation 5.9, the noise was considered to have $\sigma_z = 0.3\, u_z^{max}$, where u_z^{max} is the maximum input force possible.

In simulations, for GPRMs the threshold success probability, $p^G_{min} = 0.85$ and that for MTSP level was a priori assigned to $p^M_{min} = 0.7$ and this is as discussed in subsection 5.4.5.

5.6.2 Homogeneous Agents

In these numerical experiments, multiple homogeneous agents[10] starting at different locations on a stochastic map were supposed to cover a given number of goal locations. As the agents are homogeneous, the *cost of transition* and the *probability of transition* from one landmark to another is the same given all agents are working with the same map and the same sampled landmarks.

The result of our simulation experiments are shown in Figure 5.2 and Figure 5.3. Figure 5.2(a), Figure 5.3(a) and Figure 5.3(b) show three different cases, i.e. different number of agents starting at different start locations and have to visit a different number of goal locations.

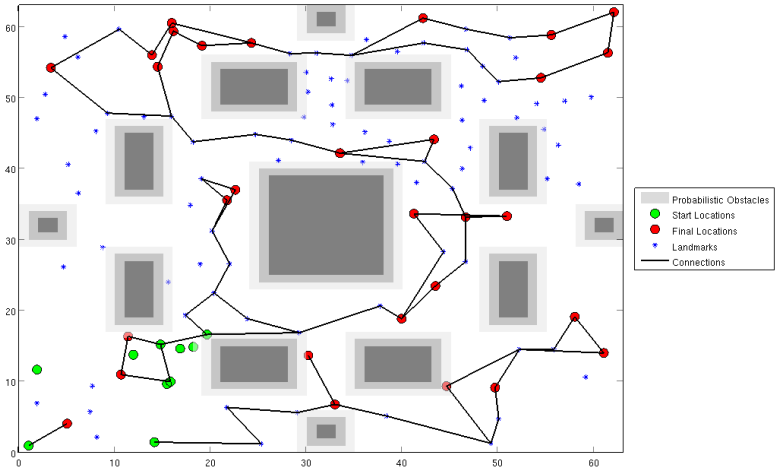
The Figure 5.2 depicts a case in which all the start locations for the robots were constricted to a smaller region compared to the spread of the goal locations, i.e. throughout the map. Figure 5.2(a) shows the solution of MAGPRM (i.e. at $Level_2$ of MAGPRM) in terms of the goal locations to be visited by the active[11] agents and the various landmarks used to navigate through those assigned goal locations. Figure 5.2(b) shows the actual trajectories (i.e. at $Level_1$ of MAGPRM) of the active agents based on the dynamics (i.e. Nonholonomic unicycle robot) and the corresponding feedback controller as explained in subsection 5.6.1.

[10] All agents are Dubins car

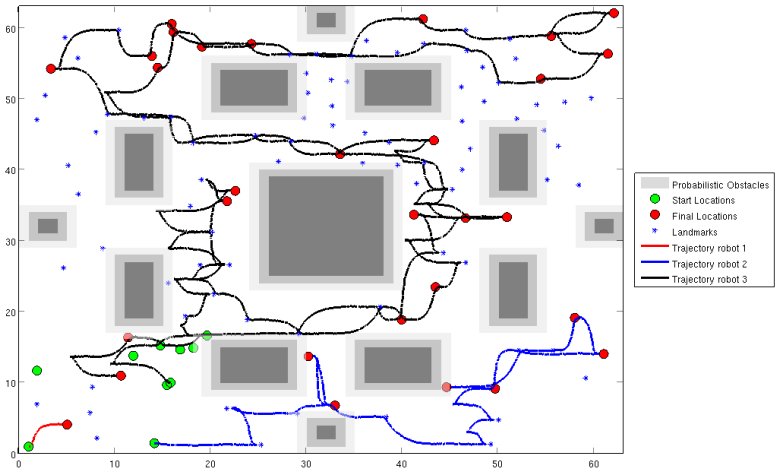
[11] In MAGPRM solution not all agents are were assigned a goal location and hence active agents are the ones which has been assigned atleast one goal location.

An important observation to be made in this result is that even if multiple agents were present, the solution provided by MAGPRM suggests only a handful of agents to cover the set of goal locations. Inherently the solution of MTSP (i.e. at $Level_3$ of MAGPRM) is driven by space partitioning and that phenomena is seen in this solution even if the costs generated by GPRM, which is fed to MTSP to get a solution, have the travel time penalized. The probability of success associated with the final paths of the individual agents are ~ 0.7.

Figure 5.3(a) and Figure 5.3(b) shows results obtained by MAGPRM, depicting coverage of goal location by the agents starting from their start locations. Both the solutions have 40 goal locations and the number of agents are 5 and 10 respectively. In Figure 5.3(a) 3 out of 5 available agents are active. The probability associated with the final solution have probability of success $p_{path} = 0.7$ (i.e. the threshold probability). In Figure 5.3(b) 4 out of the 10 available agents were active. The solution of MAGPRM in these solutions also show the space partitioning behavior. The probability of success in this solution is also $p_{path} = 0.7$.

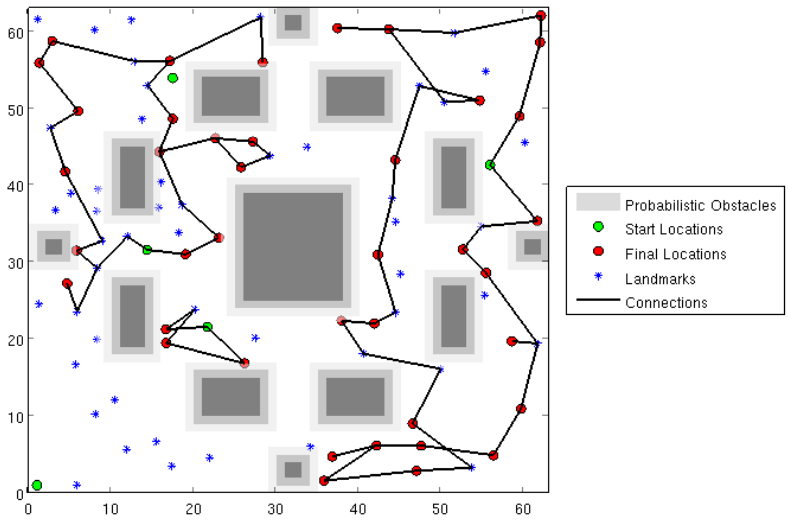


(a) MAGPRM - Showing solutions for individual vehicles

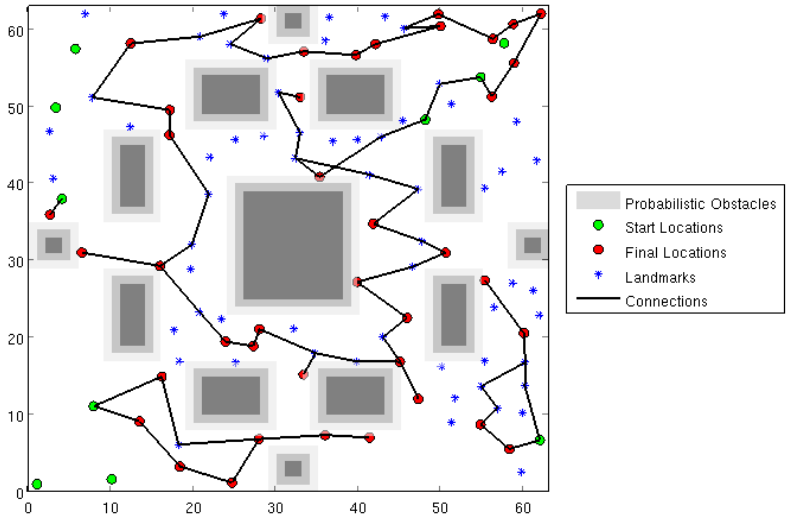


(b) MAGPRM - Showing trajectories for individual vehicles

Fig. 5.2. MAGPRM Solutions and Trajectories



(a) MAGPRM - 5 vehicles and 40 final locations



(b) MAGPRM - 10 vehicles and 40 final locations

Fig. 5.3. MAGPRM Solutions

5.6.3 Heterogeneous Agents

In these set of experiments, heterogeneous[12] agents are present in the map. The equations of motion and the feedback controller associated with each type of agent has been explained in subsection 5.6.1. Three dimensional static stochastic maps are used for these simulations. In each of these simulations there are several 3-dimensional goal locations along with 2-dimensional goal locations. The Dubins car can only cover the 2-dimensional goal locations and the simplified 3D helicopter robot can traverse to both 2-dimensional and 3-dimensional goal locations.

The initial set of landmarks sampled were more towards 2-dimensional goals compared to 3-dimensional goals. The connections of 3-dimensional goal locations was further facilitated using AGPRM, hence the solutions shown below has less number of 3-dimensional sampled landmarks.

Figure 5.4 shows the solution outcome of MAGPRM. The Dubins car will go to only one of the 2-dimensional goal locations and the simplified 3D helicopter robot will be covering all other 2-dimensional and 3-dimensional goal locations. A partitioning of space is seen in the solution.

Figure 5.5 and Figure 5.6 shows the MAGPRM solution for another case with multiple obstacles present in the map. The Dubins car is covering all the 2-dimensional goal locations and the simplified 3D helicopter robot is covering only the 3-dimensional goal locations. Figure 5.5(a) shows the solution of MAGPRM and Figure 5.5(b) shows the actual trajectories of the robots. A partitioning of space is again visible.

The collision avoidance and detection module explained in section 5.5 is implemented and comes into play if required while the robots start to navigate to the assigned goal locations. The solutions shown in Figure 5.4 and Figure 5.6 do not require the usage of this collision related module, since the solution has the partitioning of space which implies that the robots do not come close enough to trigger the collision module.

[12] A Dubins car and a simplified 3D robot

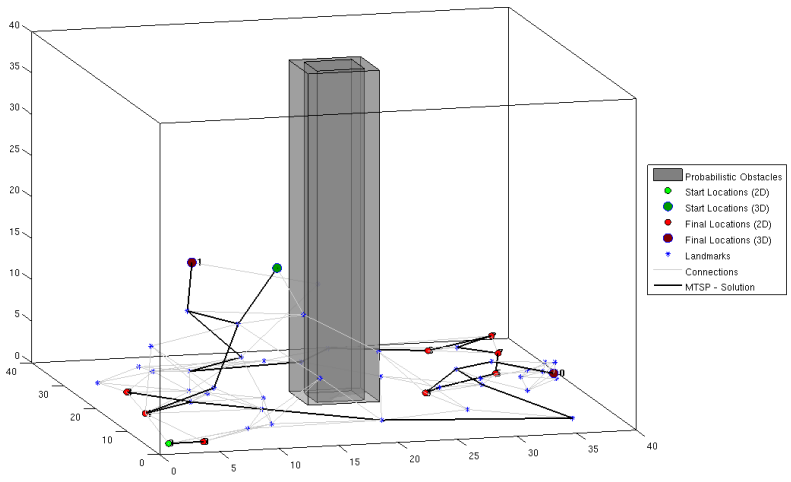
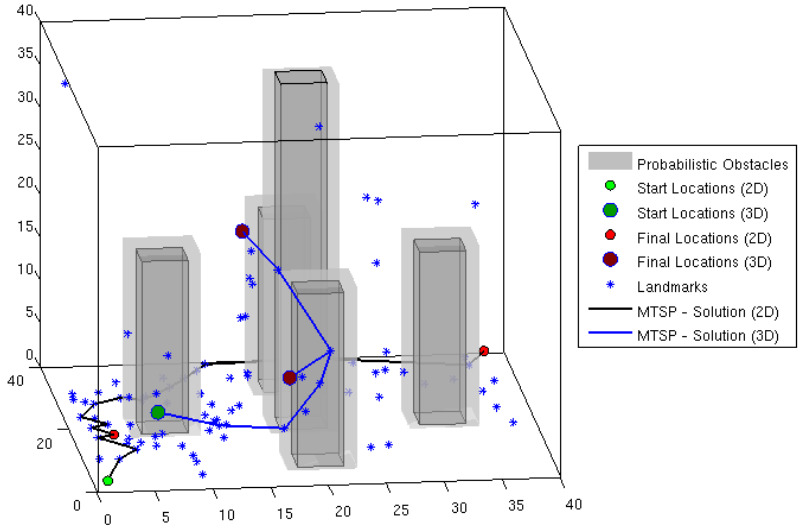
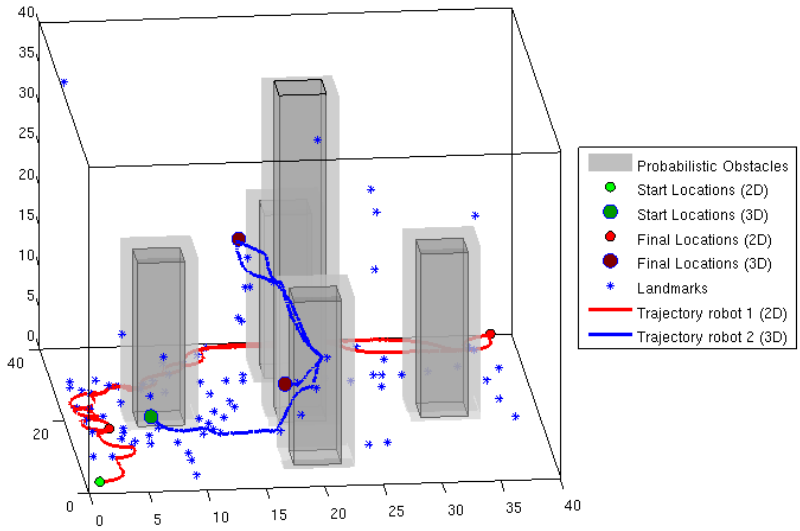


Fig. 5.4. MAGPRM with Dubins' Car and 3D Vehicle with 1-Obstacle



(a) MTSP solution for the robots



(b) Trajectories of the robots

Fig. 5.5. MAGPRM with Dubins' Car and 3D Vehicle with 5-Obstacles : Case 1

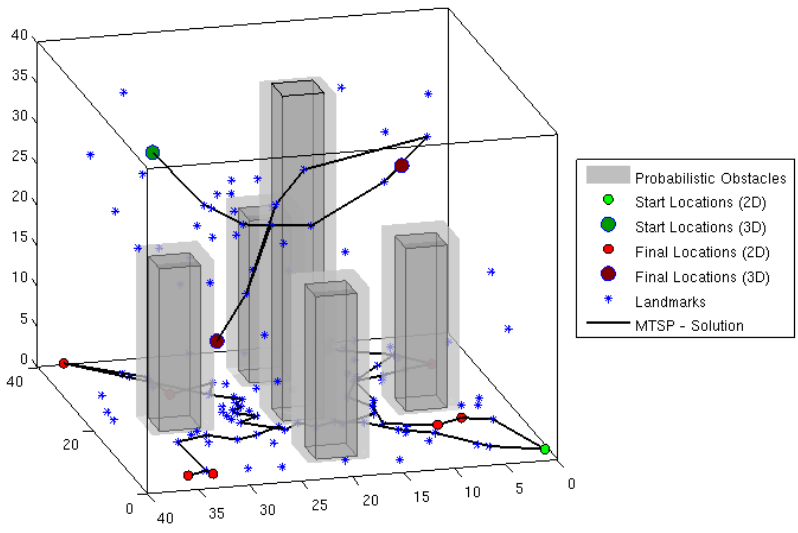


Fig. 5.6. MAGPRM with Dubins' Car and 3D Vehicle with 5-Obstacles : Case 2

The heterogeneous agents case discussed in this section depicts the power of MAGPRM. The stochastic decision making problem in presence of heterogeneous agents for which the computation of the cost of transitions are different (which was not the case in homogeneous agents) is solved. In order to solve this heterogeneous agents problem using MAGPRM, multiple GPRMs are solved in the underlying framework and a solution graph is generated for each type of agent.

5.7 Conclusion

In this section of the dissertation, we have presented the motion planning problem under uncertainty in presence of multiple agents. In order to solve the overall problem in conjunction with our existing solution methodology for single agent (i.e. GPRM), there are two sub-problems namely *routing problem* and *inter-agent collision avoidance* which needs to be additionally solved. To solve the *routing problem* an existing solution approach to the *multiple traveling salesman problem* is used. The MTSP solution methodology in conjunction with GPRM results in the MAGPRM algorithm whose solution will be an offline solution to motion planning problem for multiple agent in presence of process uncertainty and stochastic maps. To solve the *inter-agent collision avoidance* problem a heuristic algorithm is developed which guarantees collision free trajectories for every agent in real-time. Numerical experiments using these algorithms were performed on *homogeneous agents* and *heterogeneous agents* scenario for maps of different difficulty levels and different number of start, goal locations and number of agents. Results show that the algorithm does indeed solve the motion planning problem for multiple agents in presence of process uncertainty and stochastic maps.

6. CONCLUSION AND FUTURE DIRECTIONS

The motion planning problem is a sequential decision making problem and optimal control is the most general framework for solving such a problem. These problems for real-world systems cannot be solved in a deterministic framework since in the real-world, the evolution of dynamics involves uncertainty. An approach to solve motion planning problems in presence of uncertainty is to incorporate the uncertainty in the motion model, which increases the complexity of the problem. The work presented in this dissertation focuses on such problems where the uncertainty is from process noise, and uncertain environments, while perfect state sensing is assumed.

The motion planning problem can be formulated as a Markov decision process (MDP), if the uncertainties in the robot motion and environments can be modeled probabilistically. The complexity of solving these MDPs grow exponentially as the dimension of the problem increases and hence, it is nearly impossible to solve the problem even without constraints. Using hierarchical methods, these MDPs can be transformed into a semi-Markov decision process (SMDP) which only needs to be solved at certain landmark states. Sampling based algorithms like probabilistic roadmaps (PRM) and rapidly exploring random trees (RRTs) have been successful in solving very high dimensional deterministic robotic motion planning problem. However they are not robust to system with uncertainties and hence, one of the goals of this work is to generalize PRM/RRT to solve motion planning with uncertainty. It is shown that the SMDPs are the right framework to extend PRM/RRT to systems with uncertainties.

6.1 Contributions

In this dissertation, we give a systematic way of handling process uncertainty, stochastic maps and solve, in continuous state and control spaces, the motion planning problem that has an a priori specified minimum required success probability.

6.1.1 Contribution 1 : Generalized PRM (GPRM) & Generalized RRT (GRRT)

We present a generalization of randomized sampling based algorithms PRM and RRT, to incorporate the process uncertainty, and obstacle location uncertainty, termed as *generalized PRM* (GPRM) and *generalized RRT* (GRRT) algorithms. The controllers used at the lower level of these planners are feedback controllers which ensure convergence of trajectories to goal landmarks while mitigating the effects of process uncertainty. GPRM incorporates these feedback controllers into the topological graph construction phase, and in GRRT, the dynamically feasible trajectories incorporate these feedback controllers while expanding the tree. These algorithms are analyzed and a formal proof is presented proving these algorithms to be probabilistic complete, i.e. given a solution exists, as the number of sampled configurations increases the algorithms find the solution with probability 1. The algorithms have been discussed in detail and have been implemented on different robotic systems such as point robot dynamics and Dubins car demonstrating the capability of the algorithm to handle non-linear dynamics in presence of uncertainty. The results indicate that the algorithms solve the motion planning problem for a single agent in continuous control spaces in the presence of process uncertainty, with constraints such as obstacles and velocity/acceleration constraints.

A preliminary version of this work has been published in *IEEE International Conference on Systems, Man and Cybernatics*, 2009 (IEEE SMC '09) [28] and a journal version has appeared in *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 2011 [29].

6.1.2 Contribution 2 : Adaptive GPRM (AGPRM)

A novel adaptive sampling technique is proposed for these generalized planners in Section 4 to increase the efficiency and overall success probability of these planners and in order for them to tackle high dimensional problems. The proposed *adaptive*

GPRM (AGPRM) algorithm has been developed on the basis of information of transition probabilities encoded in the GPRM, which are unique to these generalized algorithms. The algorithm is implemented on a point robot and n-link manipulators, with n up to 8 links, i.e. a 16-dimensional state-space. The results demonstrate the ability of the proposed algorithm to handle highly non-linear systems with very high-dimensional state space.

A preliminary version of this work has been published in the 49^{th} *IEEE Conference on Decision and Control*, 2010 (IEEE CDC '10) [30] and a journal version has been accepted and will appear in *Journal of Control Theory and Application, Special Issue on Approximate Dynamic Programming*, 2010.

6.1.3 Contribution 3 : Multi-agent AGPRM (MAGPRM)

The multiple agent motion planning problem in presence of process uncertainty and stochastic maps maybe posed as a multi-agent Markov decision process (MMDP). In order to approximately solve this MMDP, a *coordination* sub-problem needs to be solved. In the single agent case, GPRM solves the MDP posed by the motion planning problem in presence of uncertainty. In presence of heterogeneous multi-agents with given start and desired goal configurations, an abstract graph, specific to an agent type, can be constructed with all start and goal configurations as nodes and its edges represents the cost of transitions given by a GPRM between that pair of start and goal configuration. Solution of the agent routing problem over this graph using a *multi-agent traveling salesman problem* (MTSP) solution technique solves the *coordination problem*. The solution methodology, called the *multi-agent AGPRM* (MAGPRM) is the result to solve the multi-agent motion planning problem under uncertainty, using a MTSP solution technique in conjunction with GPRM.

The solution of MAGPRM does not take into account the presence of other moving agents in the domain, hence, for real-time implementation an *inter-agent collision detection and avoidance* module was designed which ensures that no two

agents collide at any time-step. A formal analysis and proof of performance of this module is also presented.

Numerical experiments were performed on a set of *homogeneous* and *heterogeneous* agents, comprising of Dubins car and three-dimensional simplified helicopter robot. Results demonstrate that an optimal joint policy, for the systems of agents, is achieved and the individual agents routing has a probability of success above the minimum required success probability.

6.2 Future Directions

In this section several interesting directions for future research involving alternative approaches and extension of the current work is presented.

Immediate extensions of the current work can be done in a couple of areas. To provide solution convergence guarantees for any adaptive sampling algorithm has always been a challenge: researchers usually demonstrate the converge using numerical simulations. In the proposed adaptive sampling based GPRM (AGPRM), numerical results have shown improvement in efficiency, and the algorithm converges to a solution. A formal proof of this convergence will be an immediate extension.

Another immediate extension can be done by implementing the adaptive solution to achieve some pre-specified p^M_{min} for a MAGPRM solution, as discussed in subsection 5.4.5. Adaptive sampling implementation for GPRM improves the efficiency for achieving the requisite p^G_{min}. Due to lack of capabilities of prospective MTSP solution technique to handle constraints, a solution of MAGPRM is not guaranteed to achieve p^M_{min}, thus introducing an adaptive solution technique to handle this can achieve similar performance as in AGPRM.

On a similar note, an alternative MTSP solution technique which is capable of handling constraints can eliminate the problem of not-achieving p^M_{min} by a MAGPRM solution. Using such a solution technique will ensure that the specified lower bound, p^M_{min} is followed, i.e. $p^M_{path} \geq p^M_{min}$. Furthermore, such a MTSP solution technique

can ensure that the number of goal locations visited by agents are bounded, if it is a requirement.

In our approach to solving motion planning problems under uncertainty, the problem was posed as a MDP. Using randomized sampling of landmarks and Monte-Carlo simulation in between landmarks, transition costs and transition probabilities were computed, which were used to solve the MDP using Dynamic Programming (DP). Many reinforcement learning algorithms, in machine learning, are closely related to DP techniques. But compared to DP, the reinforcement learning algorithms do not need the knowledge of the underlying MDP, these techniques learn the transition costs and transition probabilities through offline/online simulation. Hence, generalizing GPRM using reinforcement learning will be an interesting direction for investigation.

The proposed MAGPRM algorithm in this work, is an approximate method of solving a particular MMDP for a set of start locations. To solve the general MMDP, i.e. solving the stochastic optimization problem for the joint states and control spaces for multi-agents, by using the solution of GPRM, for underlying MDPs for individual agents, will be another interesting investigation (i.e. posing the MMDP in terms of GPRM formulation). In a general MMDP, the joint state and control spaces of all the agents increase the complexity of the decision process.

The MAGPRM algorithm developed in this dissertation to solve the heterogeneous multi-agent motion planning problem under uncertainty, can also be perceived as a tool to measure performance for different controller options for the same agent. It can be a powerful tool to compare multiple controllers, or in suggesting which controller to be used in different domains of the problem.

Incorporating sensing uncertainty in the motion planning algorithm is a ongoing research effort and is a very important addition to the problem. Another important extension is to solve the motion planning problem with increasing stochastic map's dimension/knowledge, i.e. with new observations made in the domain, the mapping

algorithm updates the map and then the motion planning algorithm extends its existing graph to cover the updated map.

REFERENCES

[1] A. Bryson and Y. Ho, *Applied optimal control*. New York, NY: American Institute of Aeronautics and Astronautics, 1979.

[2] S. LaValle, *Planning algorithms*. Cambridge, U.K.: Cambridge Univ Pr, 2006.

[3] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep. 98-11, Computer Science Dept., Iowa State University, 1998.

[5] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, p. 233, 2002.

[6] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc., 1994.

[7] D. Bertsekas, *Dynamic Programming and Optimal Control: 3rd Edition*. Nashua, NH: Athena Scientific, 2007.

[8] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic programming*. Nashua, NH: Athena Scientific, 1996.

[9] R. Sutton and A. Barto, *Reinforcement Learning*, vol. 18. Cambridge, MA: MIT Press, 1998.

[10] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, p. 378, 2001.

[11] R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[12] R. Parr, "Hierarchical control and learning for Markov decision processes". PhD thesis, Dept. Comput. Sci., Univ. of California, Berkeley, CA, 1998.

[13] S. LaValle, "Robot motion planning: A game-theoretic foundation," *Algorithmica*, vol. 26, no. 3, pp. 430–465, 2000.

[14] N. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, pp. 113–120, IEEE, 1996.

[15] L. Guibas, D. Hsu, H. Kurniawati, and E. Rehman, "Bounded uncertainty roadmaps for path planning," *Algorithmic Foundation of Robotics VIII*, pp. 199–215, 2009.

[16] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *IEEE International Conference on Robotics and Automation*, pp. 3313–3318, IEEE, 2007.

[17] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 1261–1267, IEEE, 2006.

[18] N. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *IEEE International Conference on Robotics and Automation*, pp. 1617–1624, IEEE, 2007.

[19] S. Wilmarth, N. Amato, and P. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1024–1031, IEEE, 1999.

[20] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *The International Journal of Robotics Research*, vol. 28, no. 11-12, p. 1448, 2009.

[21] R. Alterovitz, M. Branicky, and K. Goldberg, "Motion planning under uncertainty for image-guided medical needle steering," *The International Journal of Robotics Research*, vol. 27, no. 11-12, p. 1361, 2008.

[22] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, Citeseer, 2007.

[23] D. Mellinger and V. Kumar, "Control and planning for vehicles with uncertainty in dynamics," in *IEEE International Conference on Robotics and Automation*, pp. 960–965, IEEE.

[24] R. Pepy, M. Kieffer, and E. Walter, "Reliable robust path planner," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1655–1660, IEEE, 2008.

[25] T. Schouwenaars, B. Mettler, E. Feron, and J. How, "Robust motion planning using a maneuver automation with built-in uncertainties," in *Proc. American Control Conference*, vol. 3, pp. 2211–2216, IEEE, 2003.

[26] Y. Huang and K. Gupta, "RRT-SLAM for motion planning with motion and map uncertainty for robot exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1077–1082, IEEE, 2008.

[27] A. Lazanas and J. Latombe, "Motion planning with uncertainty: A landmark approach," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 287–317, 1995.

[28] S. Chakravorty and S. Kumar, "Generalized sampling based motion planners with application to nonholonomic systems," in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 4077–4082, IEEE, 2009.

[29] S. Chakravorty and S. Kumar, "Generalized sampling-based motion planners," *IEEE Transactions on Systems, Man, and Cybernatics—Part B: Cybernetics*, vol. 41, no. 3, p. 855, 2011.

[30] S. Kumar and S. Chakravorty, "Adaptive sampling for generalized sampling based motion planners," in *49th IEEE Conference on Decision and Control*, pp. 7688–7693, IEEE, 2010.

[31] P. Oberlin, S. Rathinam, and S. Darbha, "Today's Traveling Salesman Problem," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 4, pp. 70–77, 2010.

[32] D. Hsu, J. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Robotics Research*, pp. 83–97.

[33] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, p. 534, 1999.

[34] D. Conner, "Integrating planning and control for constrained dynamical systems". PhD thesis, Robotics Inst. Comput. Sci., Carnegie Mellon Univ., Pittsburg, PA, 2007.

[35] D. Conner, H. Choset, and A. Rizzi, "Flow-through policies for hybrid controller synthesis applied to fully actuated systems," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 136–146, 2009.

[36] A. Quaid and A. Rizzi, "Robust and efficient motion planning for a planar robot using hybrid control," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, pp. 4021–4026, IEEE, 2000.

[37] L. Yang and S. Lavalle, "The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 419–432, 2004.

[38] S. Patel, S. Jung, J. Ostrowski, R. Rao, and C. Taylor, "Sensor based door navigation for a nonholonomic vehicle," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3081–3086, IEEE, 2002.

[39] R. Brockett, "Asymptotic stability and feedback stabilization," *Differential Geometric Control Theory*, pp. 181–208, 1983.

[40] S. Au and J. Beck, "Subset simulation and its application to seismic risk based on dynamic analysis," *Journal of Engineering Mechanics*, vol. 129, p. 901, 2003.

[41] I. Kolmanovsky and N. McClamroch, "Developments in nonholonomic control problems," *Control Systems Magazine*, vol. 15, no. 6, pp. 20–36, 1995.

[42] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[43] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996.

[44] J. Reif, "Complexity of the mover's problem and generalizations extended abstract," in *Proc. 20th Annual IEEE Conference on Foundations of Computer Science*, pp. 421–427, 1979.

[45] D. Hsu, J. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726, IEEE, 1999.

[46] D. Hsu, "Randomized single-query motion planning in expansive spaces". PhD thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 2000.

[47] K. Bekris, B. Chen, A. Ladd, E. Plaku, and L. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 656–661, IEEE, 2003.

[48] M. Akinc, K. Bekris, B. Chen, A. Ladd, E. Plaku, and L. Kavraki, "Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps," *Robotics Research*, pp. 80–89, 2005.

[49] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[50] J. van den Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *The International Journal of Robotics Research*, vol. 24, no. 12, p. 1055, 2005.

[51] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2004.

[52] V. Boor, M. Overmars, and A. Van Der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2, 1999.

[53] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 4420–4426, Citeseer, 2003.

[54] T. Simeon, J. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[55] L. Guibas, C. Holleman, and L. Kavraki, "A probabilistic roadmap planner for flexible objects with aworkspace medial-axis-based sampling approach," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1999.

[56] J. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 4439–4444, Citeseer, 2003.

[57] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. Amato, "A machine learning approach for feature-sensitive motion planning," *Algorithmic Foundations of Robotics VI*, pp. 361–376, 2004.

[58] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pp. 141–154, 1998.

[59] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. Cambridge, MA: The MIT Press, 2005.

[60] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," *Robotics: The Algorithmic Perspective*, pp. 630–637, 1998.

[61] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, p. 999, 2002.

[62] P. Leven and S. Hutchinson, "Using manipulability to bias sampling during the construction of probabilistic roadmaps," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 6, pp. 1020–1026, 2003.

[63] L. Kavraki, "Random networks in configuration space for fast path planning," 1995.

[64] P. Svestka, "A probabilistic approach to motion planning for car-like robots," *RUU-CS*, no. 93-18, 1993.

[65] M. Morales, S. Rodriguez, and N. Amato, "Improving the connectivity of prm roadmaps," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 4427–4432, IEEE, 2003.

[66] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, pp. 521–528, IEEE, 2000.

[67] R. Bohlin, "Path planning in practice; lazy evaluation on a multi-resolution grid," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 49–54, IEEE, 2001.

[68] R. Bohlin and L. Kavraki, "A randomized algorithm for robot path planning based on lazy evaluation," *Handbook on Randomized Computing*, pp. 221–249, 2001.

[69] G. Sanchez and J. Latombe, "On delaying collision checking in prm planning: Application to multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 1, p. 5, 2002.

[70] R. Geraerts *et al.*, "Sampling-based motion planning: Analysis and path quality," 2006.

[71] L. Kavraki, M. Kolountzakis, and J. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[72] S. Lindemann and S. LaValle, "Current issues in sampling-based motion planning," *Robotics Research*, pp. 36–54, 2005.

[73] R. Pearce, M. Morales, and N. Amato, "Structural Improvement Filtering Strategy for PRM," *Robotics: Science and Systems IV*, p. 167, 2009.

[74] C. Guestrin, "Planning under uncertainty in complex structured environments". PhD thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 2003.

[75] J. Kok, "Coordination and learning in cooperative multiagent systems". PhD thesis, Dept. Comput. Sci., Univ. of Amsterdam, Amsterdam, The Netherlands, 2006.

[76] D. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 389–423, 2002.

[77] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proc. 6th conference on Theoretical aspects of rationality and knowledge*, pp. 195–210, Morgan Kaufmann Publishers Inc., 1996.

[78] J. Kok and N. Vlassis, "Using the max-plus algorithm for multiagent decision making in coordination graphs," *RoboCup 2005: Robot Soccer World Cup IX*, pp. 1–12, 2006.

[79] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[80] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[81] C. Noon and J. Bean, "An efficient transformation of the generalized traveling salesman problem," *Ann Arbor*, vol. 1001, pp. 48109–2117, 1989.

[82] A. Ladd and L. Kavraki, "Measure theoretic analysis of probabilistic path planning," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 229–242, 2004.

[83] D. Isaacson and R. Madsen, *Markov chains, theory and applications*, vol. 4. New York: Wiley, 1976.

[84] C. Hsu, *Cell-to-cell mapping: a method of global analysis for nonlinear systems*. New York, NY: Springer, 1987.

[85] M. Dellnitz and O. Junge, "On the approximation of complicated dynamical behavior," *SIAM Journal on Numerical Analysis*, vol. 36, no. 2, pp. 491–515, 1999.

APPENDIX 1

ANALYSIS OF GENERALIZED SAMPLING-BASED MOTION PLANNERS

We will show that both the GPRM and the GRRT algorithms are probabilistically complete, i.e. the algorithms will find a feasible path, given that such a path exists as the number of samples goes to infinity. We use the path isolation techniques commonly used in showing the probabilistic completeness of PRMs [82]. The basic idea of the proof is very simple : we show that, is there is a feasible "safe" path parametrized by a sequence of points, in the sense that will be formalized as follows, from a start state of a goal state, then there is a finite neighborhoods of each of these points such that, if samples are chosen from these neighborhoods, the path formed by sequencing the sampled points is also feasible. Hence, because these neighborhoods are finite, samples will be chosen in each of these neighborhoods with probability 1 as the number of sampled goes to infinity, and thus, the algorithms are probabilistically complete. The analysis of the generalized sampling based motion planners requires the use of the theory of Markov chains. The reader is referred to [83] for relevant details with regard to Markov chains.

Let $x_0, x_1, \ldots, x_N (= x_g)$ denote a particular path that attains the goal configuration x_g with a success probability that is higher than the minimum threshold probability p_{min}. The former statement has to be made mathematically precise and is done as follows. Recall that, due to the stochasticity of the system, it is never possible for the system to exactly attain some state x_k in the sequence; instead, we stop the controller, drawing the robot to x_k, and switch to the controller, drawing it to x_{k+1}, when the robot enters some pre-specified neighborhood Ω_k of the point x_k. The situation is illustrated in Figure 3.2. The robot moved from $x' \in \Omega_{k-1}$ to some $x \in \Omega_k$ under the action of control $u(;, x_k)$. Once it reaches x, the controller $u(:, x_{k+1})$ switches on, and the robot moves from $x \in \Omega_k$ to $x'' \in \Omega_{k+1}$, at which point the controller $u(:, x_{k+2})$ switches on, and so on, until the robot reaches some

point y in the neighborhood Ω_N of x_N. In addition, note that all the obstacles in the Figure 3.2 are lumped into the single sink cell Ω_O. Thus, the aforementioned statement, which is carefully stated, implies that the system transitions through the neighborhoods $\Omega_0 \rightarrow \Omega_1 \cdots \rightarrow \Omega_N$ with a probability higher than the minimum threshold p_{min}, successfully transitioning from $x \rightarrow y$, denoted by $p(y/x)$, is greater than the minimum threshold probability p_{min}.

First, let us focus on the transition from the set Ω_{k-1} to the set Ω_k. Let the feedback controller based on drawing the robot to the point x_k be denoted by $u(:, x_k)$. This feedback control induces a Markov process on the state space of the robot, according to which the robot moves in a probabilistic manner. Assuming that we discretize the time by some small ΔT, let the transition density function of the resulting Markov chain be denoted by $p(y/x; x_k)$ (on the obstacle free state space), where the dependence on x_k shows the explicit dependence of the transition probabilities of Monte Carlo on the point x_k. We need to find out the probability that the robot makes it form some $x \in \Omega_{k-1}$ to some $x' \in \Omega_k$ without colliding with any of the obstacles in the state space oof the robot. Note that te aforementioned transition probabilities do not account for the obstacles in the state space. Let us further discretize the continuous state Markov chain into a finite-state Markon chain through some small spatial discretization Δx into a finite set of cells $\mathcal{C}_i$ with measure $\mu(\mathcal{C}_i)$. The evolution of the continuous state Markon chain is approximated by a discrete Monte Carlo that evolves on the cell space. The transition probabilities between individual cells in the cell space are given by

$$P(\mathcal{C}_j/\mathcal{C}_i) = \int_{\mathcal{C}_i} (\int_{\mathcal{C}_i} p(y/x) \frac{1}{\mu(\mathcal{C}_i)}) dy. \tag{1.1}$$

This discrete MC is an arbitrary good approximation of the continuous MC as the size of the spatial discretization gets small under mild regularities conditions on the system dynamics, in particular, the existence of a smooth transition density function $p(y/x)$, and ergodicity, i.e. the property that any initial distribution decays

to a unique stationary distribution. The approximation is good in the sense that the difference between the piecewise constant approximation in cell space and the true probability density function (pdf) evolution in continuous space is arbitrarily small as the size of the spatial discretization becomes small. Such cell -based techniques are a well-established method for the global analysis of nonlinear systems known as generalized cell-to-cell mapping [84], [85]. For simplicity, all our subsequent analysis is done on the finite state MC formed by the spatial discretization of the state space. The implicit assumption is that the discretization is fine enough such that the finite MC. The original transition probabilities of the MC in the obstacle-free state space have to be modified to obtain the probabilities of success of transitioning from cell x to cell x' (we slightly abuse the notation here to avoid the notational inconvenience of using the symbol $\mathcal{C}_i$ for cells).

Let all the cell that correspond to the obstacles in the state space be lumped into a single sink cell $\{x_o\} = \Omega_O$. Once the robot hits an obstacle in its state space, it is deemed to have failed, and hence, $p(x/x_o; x_k) = 0$ for any x in the state space of the robot. In addition, for simplicity, let the number of cells in the set Ω_k be n_R for all the sets $\Omega_1, \Omega_2, \ldots, \Omega_N$. Once the robot enters the set Ω_k, it is captured, and the next feedback law to get it to Ω_{k+1} starts. Thus, $p(x'/x; x_k) = 0$ for all $x \in \Omega_k$ and any $x' \in \Omega$. Thus, Ω_k and $Omega_O$ are the recurrent classes of the modified MC, i.e. the states that the MC infinitely often visits as the time increases to infinity. The rest of the states are transient, i.e. the MC stops visiting these states as time increases to infinity. Because the number of states in the safe recurrent class Ω_k is fixed and the number of sink states that correspond to the obstacle recurrent class is one, the number of transient states is also fixed, e.g. denoted by the number n_T. We further assume that the MC on the obstacle-free map is ergodic, i.e. any

initial distribution asymptotically converges to a unique stationary distribution. The transition probability of the modified MC can be written as the following matrix [83]:

$$\tilde{P}(x_k) = \begin{bmatrix} \tilde{P}_1(x_k) & 0 & 0 \\ 0 & \tilde{P}_2(x_k) & 0 \\ R_1(x_k) & R_2(x_k) & Q(x_k) \end{bmatrix} \tag{1.2}$$

where $\tilde{P}_1(x_k)$ is an $n_R \times n_R$ matrix, $\tilde{P}_2(x_k)$ is a scalar, $R_1(x_k)$ is an $n_T \times n_R$ matrix, $R_2(x_k)$ is an $n_T \times 1$ matrix, and $Q(x_k)$ is an $n_T \times n_T$ matrix. $\tilde{P}_1$ represents the transition matrix (in our case, I_{n_R}, i.e. the $n_R \times n_R$ identity matrix) that governs the transitions within the "safe" recurrent class Ω_k, $\tilde{P}_2$ is simply oe and denotes that once the robot is in the sink state it stays there, R_1 denotes the transition probabilities that one of the transient states transitions to one of the states in the "safe" recurrent class, R_2 represent the transition probabilities that one the transient states transitions into the sink state Ω_O and Q represents the probabilities that the transient states stay within the transient class of states. The original transition probability matrix in the obstacle-free case is modified only in the first $n_R + 1$ rows. In particular, note that R_1, R_2, and Q are the same for the original and the modified MCs. Due to the ergodicity of the unconstrained MC, it is ensured that trajectories that start from any robot state is bound to get captured in either the safe set or the failure sink state. This condition implies that there are two recurrent classes in the modified MC.

The transition probability matrix that contains the probabilities that some transient state gets absorbed into some satte within the safe recurrent class Ω_k is given by [83]

$$P_a(x_k) = (I - Q(x_k))^{-1} R_1(x_k) \tag{1.3}$$

where the explicit mention of the point x_k shows the dependence of the absorption probability matrix on the feedback control $u(:, x_k)$. Note that the absorption probability matrix P_a is an $n_T \times n_R$ matrix and that the (i, j)th element in the matrix denotes the probability that the ith element in the transient class is absorbed into the safe recurrent class Ω_k through the jth element. To understand the aforementioned formula, note that the probabilities that the transient states remain within the transient class after n steps is given by Q^n. Thus, the probability that the transient states get absorbed into the safe recurrent class after n steps is $Q^n R_1$, and the probability that the transient states get absorbed into the sink class is $Q^n R_2$. It may be shown that the infinite sum $I+Q+Q^2+Q^3+\cdots$ exists, and hence, $(I-Q)^{-1}$ exists, because Q is sub-stochastic, i.e. its row sum is less than or equal to one. Hence, $||Q||_\infty \leq 1$, where $||.||_\infty$ is the matrix norm induced by the max norm [83]. Thus, the probability that some state in the transient class eventually gets absorbed into some state of the safe recurrent class Ω_k is given by $(I - Q(x_k))^{-1} R_1(x_k)$, and the probability that some state is absorbed into the sink class is given by $(I - Q(x_k))^{-1} R_2(x_k)$.

Next we consider the sequence of transitions $\Omega_0 \to \Omega_1 \cdots \to \Omega_N$. Again, recall the assumption that there are exactly n_R states in each of the aforementioned N safe recurrent classes. In (equation), the matrix $P_a(x_k)$ contains the probabilities that any one of the transient states of the MC induced by $u(:; x_k)$ (let us denote the MC by $\mathcal{M}(x_k)$) is absorbed into any one of the safe recurrent states in the safe recurrent class Ω_k. However, in view of the aforementioned sequence of transformations from one safe recurrent class to the next, we are only interested in the transient state of the MC $\mathcal{M}(x_k)$ that correspond to the safe recurrent class Ω_{k-1} of the MC induced by $u(:; x_{k-1})$, $\mathcal{M}(x_{k-1})$. The matrix that contains the probabilities that some state in Ω_{k-1} is absorbed into some state in the safe recurrent class Ω_k is give by

$$P_a(x_{k-1}, x_k) = \Gamma_k (I - Q(x_k))^{-1} R_1(x_k) \tag{1.4}$$

where Γ_k is a constant $n_R \times n_T$ matrix that maps the safe recurrent states in Ω_{k-1} into the transient states of $\mathcal{M}(x_k)$. The matrix Γ_k is independent of the choice of x_{k-1} and x_k, given that Ω_{k-1} and Ω_k are fixed. Then, it follows that the transition probability matrix that denotes the transitions from $\Omega_0 \to \Omega_1 \to \cdots \to \Omega_N$ is give by

$$P_a(x_0, x_1, \ldots, x_N) = \prod_{k=1}^{N} \Gamma_k (I - Q(x_k))^{-1} R_1(x_k) \tag{1.5}$$

The product matrix $P_a(x_0, \ldots, x_N)$ is $n_R \times n_R$, because each of the component matrices in the product is $n_R \times n_R$. The (i, j)th element of the matrix denotes the probability that the ith state of the safe recurrent class Ω_0 is absorbed into the jth state of the safe recurrent class Ω_N, under the sequence of controllers $u(:, x_1), u(:, x_2), \ldots, u(:; x_N)$.

Suppose now that the sequence of controllers parameterized by the points $(x_1, \ldots, x_N)$ is slightly perturbed to the parameter $(x'_1, \ldots, x'_N)$. It is reasonable to assume now that the transition probability matrix of the MC due to $u(:, x_k)$ is close to the transition probability matrix of the MC induced by $u(:, x'_k)$ if x_k and x'_k are close to each other, given that the safe recurrent class Ω_k remains the same. In other words, we assume that $P(x_k)$, the transition probability matrix on the obstacle-free space for the robot under control $u(:; x_k)$, parametrized by the point x_k, is a continuous function of x_k. Because the matrices $Q(x_k)$ and $R_1(x_k)$ are unchanged once the MC is modified to account for the obstacles in the state space, it follows that these matrices are also continuous functions of x_k. This case, in turn, implies that $(I - Q(x_k))^{-1}$ is also continuous in parameter x_k, and hence the product matrix $P_a(x_0, \ldots, x_N)$ is continuous with respect to the set of parameters $(x_0, \ldots, x_N)$, which is the case, because the kth component of the product is continuous with respect to x_k and thus, trivially, also continuous with respect to the parameter $(x_0, \ldots, x_N)$. Hence the product is also continuous with respect to parameter $(x_0, \ldots, x_N)$. Therefore, it

follows that, given any $\epsilon > 0$, there exists a $\delta > 0$ such that, if $||x_k - x'_k|| < \delta$ for all k, $|P_a(x_0, \ldots, x_N)(i,j) - P_a(x'_0, \ldots, x'_N)(i,j)| < \epsilon$ for all possible elements (i,j) of the matrices.

Take $\epsilon = min_{(i,j)} P_a(x_0, \ldots, x_N)(i,j) - p_{min}$. Due to the aforementioned argument, it follows that there exists δ^* such that if $||x_k - x'_k|| < \delta^*$,

$$|P_a(x_0, \ldots, x_N)(i,j) - P_a(x'_0, \ldots, x'_N), (i,j)| < min_{(i,j)} P_a(x_0, \ldots, x_N)(i,j) - p_{min}$$

which in turn, implies that $P_a(x_0, \ldots, x_N)(i,j) > P_{min}$ for all (i,j). Noting that $\delta^* > 0$, with probability 1, it is true that a point x'_k will eventually be chosen in δ^* balls around each of the x_k as the number of sampled points go to infinity. Thus, this case shows that the GPRM algorithm is probabilistically complete.

The aforementioned development may be summed up in the following result.

Proposition 1.1. Let $p(x)$ denote the (discretized) transition probability matric that corresponds to the feedback controller $u(:, x)$ on the obstacle-free state space. Assume that the transition probability matric $P(x)$ is continuous in x and is ergodic for all x. Let $(x_0, \ldots, x_N)$ parametrize the sequence of controllers $u(:, x_0), \ldots, u(:, x_N)$ with associated safe recurrent classes $\Omega_0, \Omega_1, \ldots, \Omega_N$ such that a feasible path exists from x_0 to $x_N (= x_g)$, i.e. $P_a(x_0, \ldots, x_N)(i,j) > p_{min}$ for all elements (i,j), where the (i,j)th element of the absorption probability matrix P_a denotes the probability that the ith state in Ω_0 successfully transitions of the jth state in Ω_N. Give then the safe recurrent classes $\Omega_0, \ldots, \Omega_N$ are unchanged, there exists a $\delta^* > 0$ such that, if $||x_k - x'_k|| < \delta^*$, i.e. $x'_k \in \mathcal{B}_{\delta^*(x_k)}$, $P_a(x'_0, \ldots, x'_N)(i,j) > p_{min}$ for all (i,j). Consequently, the GPRM algorithm finds a feasible path, if one exists as aforementioned, with probability 1. Hence, the GPRM algorithm is probabilistically complete.

The follow remarks are due to the assumptions made in Proposition 1.

Remark 3. The transition probabilities of the MC underlying the closed-loop system under $u(:, x)$ is given by $p(y'/y, u(y, x))$. Thus, if the transition probability function

is continuous in u, which is true under mild regularity conditions on the system dynamics, and if the feedback control $u(:, x)$ is continuous in x, it follows that the controlled MC transition probabilities are continuous in x. The condition that the feedback control law continuously varies with the equilibrium about which it stabilizes is a mild assumption. For instance, in the LQ case, this case amounts to the assumption that the feedback gain matrix $K(x)$ smoothly varies with the equilibrium x. This condition, in turn, reduces to the smoothness to the solutions of the associated algebraic Riccati equation with respect to the open-loop system matrix $A(x)$ about equilibrium x.

Remark 4. We may also find the expected time for states in Ω_{k-1} to get absorbed into Ω_k or Ω_O. The expected times of absorption from the states in Ω_{k-1} is given by $\Gamma_k(I - Q(x_k))^{-1}\bar{1}$, where $\bar{1}$ is a vector of ones. Thus, the expected time of execution of the feedback controller $u(:, x_k)$ is finite. Hence, it follows that the expected total time of operation of the entire sequence of controllers $u(:, x_1)$ through $u(:, x_N)$ is also finite.

We may also give explicit bounds on the expected number of samples needed to get a safe feasible path and the probability of failure of the GPRM, given that a fixed number of samples have been drawn. Without loss of generality, assume that the feasible space of planning is $[0, 1]^d$. Let the volume of the regions $\mathcal{B}_{\delta^*}(x_k)$ previously defined be p^*. Then, the following results directly follow from the traditional PRM analysis ([82], Th. IV.2, Corollary IV.3)

Proposition 1.2. We have the same conditions as in Proposition 1, Given that there exists a safe path $x_1, \ldots, x_N$, the expected number of iterations required by GPRM to find a feasible path is previously bounded by $E[M] \leq (H(N)/p^*)$, where $H(N)$ is the Nth harmonic number and is $\mathcal{O}(logN)$. Moreover, the probability of not finding a feasible path after M iterations is bounded as $P_f \leq N(1 - p)^M$.

Next we show the probabilistic completeness of the GRRT algorithm. We shall retain all the machinery that was developed for the GPRM algorithm and use Proposition 1. The proof of completeness is through induction.

Let $(x_0, \ldots, x_N)$ be a sequence of nodes such that the path through them is feasible based on Proposition 1. Then due to Proposition 1, it follows that there exists balls $\mathcal{B}_\delta$ around each x_k such that, if $x'_k \in \mathcal{B}_\delta(x_k)$, the path through $x'_0, \ldots, x'_N$ is also feasible. For notational ease, let us simple denote the balls $\mathcal{B}_\delta(x_k)$ as $\mathcal{B}_k$. The statement for the induction is given as follows.

Suppose that some $x'_0 \in \mathcal{B}_0, \ldots, x'_M \in \mathcal{B}_M$ are chosen in that order, i.e. first x'_1, then x'_2, and so on. Then there exists a feasible path from x_0 to x_M based on Proposition 1.

First we prove the statement for $M = 1$. Suppose that some $x'_1 \in \mathcal{B}_1$ is chosen and not connected to $x'_0 \in \mathcal{B}_0$ (if it is connected, the result is trivial). Then, through the construction of GRRT, in particular because of (eq. ref), this case means that there is some other node in the tree, e.g. x, such that $p(x'_1) = p(x, x'_1)p(x) > p(x, x'_0) > p_{min}$. Hence it follows that there is a safe path from x_0 to x_1.

Next we assume that the statement is true for $M = k - 1$ and show that it is true for $M = k$. Supposed that, for the first time, $x'_k \in \mathcal{B}_k$ is chosen after some $x'_0 \in \mathcal{B}_0, \ldots, x'_{k-1} \in \mathcal{B}_{k-1}$ have been chosen. If the node is not connected to x'_{k-1} ans is instead connected to some x, then using arguments exactly the same as aforementioned, it follows that $p(x'_k) = p(x)p(x, x'_k) > p(x'_{k-1})p(x'_k, x'_{k-1}) > p_{min}$. The reason for the last inequality is given as follows. Because we have assumed the statement for $k - 1$, it means that there is safe path till x'_{k-1} that is, at least, as safe as going through $x'_0, x'_1, \ldots, x'_{k-1}$. Because we know that, due to Proposition 1, $p(x'_0, x'_1, \ldots, x'_{k-1})p(x'_k, x'_{k-1}) > o_{min}$, it follows that $p(x'_{k-1})p(x'_k, x'_{k-1}) > pmin$. Hence the result is true for k if it is true for $k-1$. Thus, it follows that the statement is true for N.

It is true with probability 1 that some $x'_0 \in \mathcal{B}_0, \ldots, x'_N \in \mathcal{B}_N$ will be chosen in that order, and hence, it follows that the GRRT algorithm is probabilistically complete, which is summed up as the following proposition.

Proposition 1.3. We have the same conditions as in Proposition 1. Given that there is a safe feasible path, GRRT finds a feasible path with probability 1, and hence, it is probabilistically complete.

The aforementioned results have been proven for a deterministic map. The extension to the case of map uncertainty is reasonably straightforward but is left out here due to spatial constraints.

APPENDIX 2
ANALYSIS OF THE COLLISION AVOIDANCE AND DETECTION ALGORITHM

Definition 2.1. Let Q_k denote the configuration $\{q_1, q_2, \ldots, q_{|\mathcal{A}|}\}$ at time step k, where q_i is the configuration of the i^{th} agent and $\mathcal{A}$, denotes the set of all agents. Let $COLLISION(\cdot)$ be a collision detector binary function for the whole configuration of agents, i.e. Q_k. Also let there be a binary function $ALERT(\cdot)$ over the Q_k, for alerting the agents of invasion of their corresponding **alert zone**.

Detailed definitions were covered in the previous section.

Assumption 2.1. Let $\beta(\cdot)$ be a decision operator for the agents, dictating the actions in an event of possible collision. The action **evasive action** available to the decision operator $\beta(\cdot)$, for resolving possible collision, do exist.

This assumption essentially states that if required a given agent can deviate to a new landmark (i.e. other than its planned next landmark, l^k), given some conditions. Also there exist a nearby unoccupied[1] landmark, $l^{k'}$ and there exists a controller $\mu(\cdot; l^{k'})$ which will take the agent from its current configuration to this new landmark.

Assumption 2.2. Let $collision^k(a_i, a_j)$ be a binary function which check for collision between a_i and a_j at time step k. Given $v(a_j) = 0$ (velocity of a_j), the q_k^i (configuration of a_i at time step k) and μ_k^i (the controller for a_i at time step k), then $collision^{k+1}(a_i, a_j)$ can be calculated.

The configuration of a_i after one time step can be computed. And based on the predicted q_{k+1}^i for a_i and given $q_{k+1}^j = q_k^j$ for a_j, the $collision^{k+1}(a_j, a_j)$ can be computed based on predicted values.

Lemma 2.1. Given $ALERT(Q_{k-1})$ was **false**, and $ALERT(Q_k)$ is **true** then $COLLISION(Q_k)$ is never **true**.

[1] by other agents

Proof. This essentially means that the **collision zone** and **alert zone** definitions are such that 2.1 holds. The radius of **collision zone** is given by ϵ^{t_i} and that of **alert zone** is denoted by r^{t_i}, where t_i is the agent-type of the i^{th} agent. The relation between ϵ^{t_i} and r^{t_i} is given by:

$$v^{max} = \max\{v_{max}^{t_i}\}, \tag{2.1}$$

$$r^{t_i} = \kappa v_{max}^{t_i}, \tag{2.2}$$

$$\epsilon^{t_i} < r^{t_i} - v^{max} dt \tag{2.3}$$

where κ is some constant. The v^{max} denotes the maximum of $v_{max}^{t_i}$, i.e. the maximum velocity of any agent. The maximum distance any agent can travel in one time step is $v^{max}dt$, hence if an agent is outside another agent's **alert zone**, in one time step it cannot penetrate both **alert zone** and **collision zone** in the same time step. So if the ϵ^{t_i} is given then using Equation 2.3, r^{t_i} can be designed using an appropriate κ. Also κ allows design of r^{t_i} such that no invasion of **collision zone** happens in the next n time steps ($n \geq 1$) also. □

Lemma 2.2. Let μ_k^i denotes the control for i^{th} agent at time step k. An agent with current configuration q_k^i for which $\mu_k^i = \mu_0$, then q_{k+1}^i is collision free.

Proof. It states that an agent with **stop** action at time step k is collision free at time step $k+1$. As this agent is **stop**, the only possibility of collision is from any of the moving agents, but all moving agents outside **alert zone** of this agent will not have possible collision due to 2.1. And agents within the **alert zone** will not have collision with this stopped agent due to existence of **evasive action** as per 2.1. □

Proposition 2.1. Given $ALERT(Q_k)$ is $true$, the decision operator $\beta(\cdot)$ (2.1) ensures that $COLLISION(Q_{k+1})$ is $false$.

Proof. The proof is constructed using the mathematical induction methodology. Firstly it is proved that in a set of agents $\mathcal{B}'$ (algorithm 5.2) for which alert has

been triggered, this statement holds for the highest priority agent $\in \mathcal{B}'$. Then assuming this statement holds for the first $m-1$ agents, it will be shown that it holds for the m^{th} agent also.

To prove that 2.1 is true for the highest priority agent in $\mathcal{B}'$ in algorithm 5.2. Let there be total of n agents and m agents for which alert is triggered ($m \leq n$). Let $a_1 = a_{max}(\mathcal{B}')$ denote the highest priority agent (with q_k^1 as its configuration at time step k) in $\mathcal{B}'$ with agents in its **alert zone** denoted by $\mathcal{S}_1^1$ and let the next agent priority wise as a_2 (with $\mathcal{S}_2^1$) and so on. The algorithms signals other agents in a_1's **alert zone** to **stop**. And in case of probable collision with any of these agents in the **alert zone** in the next time step, a_1 should take an **evasive action**, which exists as per 2.1. For all other agents who are moving are $\notin \mathcal{S}_1^1$, as 2.1 is true the $collision^{k+1}(a_1, a_j) = false, \forall a_j \notin \mathcal{S}_1^1$ as per 2.1. Hence q_{k+1}^1 is collision free, given that q_k^1 is collision free.

Let this be true for $m-1$ agents, i.e. $\{q_{k+1}^1, q_{k+1}^2, \ldots, q_{k+1}^{m-1}\}$ is collision free, now to prove that q_{k+1}^m is also collision free. Either a_m is **stopped** or moving. If a_m is **stopped** it will be collision free in the next time step as per 2.2. If a_m is moving, means it was not in the **alert zone** of any of the moving higher priority agents (i.e. $\mathcal{P}_m < \mathcal{P}_i, \forall a_i \in \mathcal{B}'$) as all the higher priority agents are collision free. And for all other agents which are outside a_m's **alert zone** will ensure collision free time step $k+1$ for a_m as per 2.1 and for agents in the **alert zone** who are stopped, as per 2.1 the q_{k+1}^m of m^{th} agent will be collision free.

Hence the full configuration Q_{k+1} for all the agents are collision free in the next time step. □

VITA

Name: Sandip Kumar

Address: Department of Aerospace Engineering,
H.R. Bright Building, Rm. 701, Ross St. - TAMU 3141,
Texas A&M University,
College Station, TX - 77843 - 3141

Email: to.sandip@gmail.com

Education: B. Tech., Mechanical Engineering,
Indian Institute of Technology,
Kharagpur, India, 2000 - 2004

Ph.D., Aerospace Engineering,
Texas A&M University,
College Station, TX, USA, 2007 - 2011