COMPACT REPRESENTATIONS FOR EFFICIENT ROBOT MOTION PLANNING WITH FORMAL GUARANTEES

BY

ANDREW DOBSON

A dissertation submitted to the

School of Graduate Studies—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Kostas E. Bekris

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE DISSERTATION

Compact Representations for Efficient Robot Motion Planning with Formal Guarantees

by Andrew Dobson

Dissertation Director: Kostas E. Bekris

This work provides compact representations for single- and multi-robot motion planning in the context of prehensile robot manipulation. First, the asymptotic near-optimality and probabilistic near-optimality properties of sampling-based motion planners are defined and discussed. Probabilistic near-optimality is leveraged to provide practical and grounded stopping criteria for these methods which probabilistically guarantee the methods return high-quality paths. It is also shown how these methods can be leveraged to produce a compact planning representation, which is a lightweight structure that is quick to query and easy to store. The work also outlines a compact representation for solving multi-arm manipulation tasks, and integrates a scalable, asymptotically optimal multi-robot motion planning method to provide scalable, globally asymptotically optimal task and motion planning in object transfer prehensile manipulation domains.

Acknowledgements

First and foremost, I would like to thank my advisor, Kostas E. Bekris, who took a chance on me as an undergrad and helped mold my work into the rigorous scientific study it is today. Though he is at times strict, he is ever forward-thinking, and I am grateful for his vigilance and dedication, even through late nights. I would also like to thank each and every PRACSYS lab member I have worked with. Alexis Oyama, Ilias Apostolopoulos, Athanasios Krontiris, Yanbo Li, James Marble, Ryan Luna, Andrew Kimmel, Zakary Littlefield, Rahul Shome, Justin Cardoza, Colin Rennie, Shaojun Zhu, Zacharias Psarakis, Hristian Courtev, I would not have had the mettle to survive the long deadlines nor the powerful coding skills and robust code base to support my work if it were not for each and every one of you.

I would also like to thank my external collaborators, Professor Alberto DeSouza for reminding me that life is about discovery and finding your passion, Professor George Moustakides for his patient guidance and imparting some of his formidable mathematical knowledge, and Professor Dan Halperin for treating me as one of his own students and working diligently to integrate our ideas into collaborative work. I extend my thanks to Kiril Solovey and Oren Salzman for being stalwart collaborators and freely sharing research ideas and theoretical knowledge. I would like to thank the DHS CCICADA Center of Excellence at Rutgers University for their generous financial and intellectual support. I would especially like to thank the CCICADA director Fred Roberts for his sage advice and tireless efforts on our joint projects, as well as thanking all members of CCICADA who made weekly project meetings productive and thought-provoking, and especially to Brian Ricks for his dedication to the project. I also extend thanks to the promising undergraduate researchers I had the opportunity to work with over these years, Robert Kolchmeyer, Meera Murti, Poorva Sampat, and Andrew Wells.

I also thank my parents for giving me a stable platform to start my college career as an undergrad, and to my entire family for their support and belief that I could become the first Ph.D. recipient of the family. Thank you to the various administrative staffs of UNR and Rutgers for your unrelenting service though piles of paperwork and travel reimbursement forms. And finally, I would like to thank each my committee members, William Steiger, Jingjin Yu, and Devin Balkcom, for dedicating their time to the review and consideration of this work.

Dedication

This work is dedicated to my grandfather, Toney Brown, who remains a stalwart champion of education and pushing the limits of human knowledge.

Table of Contents

Chapter 1

Introduction

An important step in creating efficient, automated robotic processing centers is performing long-horizon motion and task planning in an efficient and scalable manner. Typically, a robotic system has a series of actuators that allow it to interact with the world, a set of sensors to observe the state of the world, and an internal model of its environment, allowing it to reason about the effects of motion in the environment. There is a wide range of fascinating and computationally difficult problems to solve in order to achieve the wide deployment of robots in everyday workspaces. Robot vision and perception, hardware design, controller modulation, model learning, feedback control, and long-horizon motion and task planning are just some of the problems that have to be solved in order to make a functioning robotic system. This has led robotics to be highly multidisciplinary with an interesting mix of ideas constantly innovating these subareas.

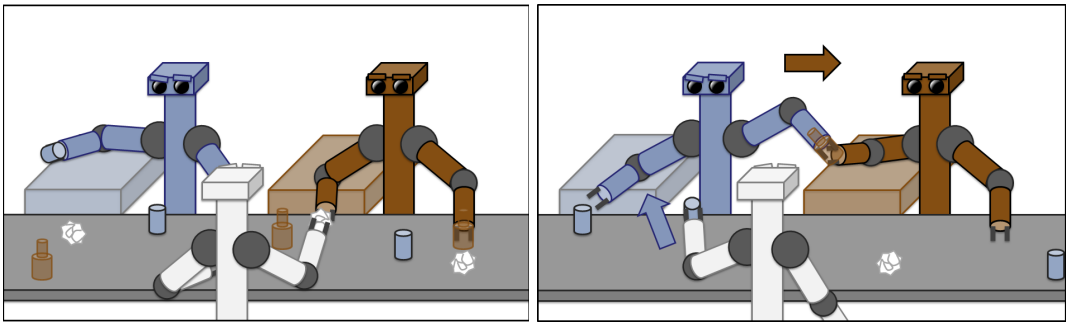

Figure 1.1: The motivating problem of this work. (Left) objects begin coming down the conveyor belt and need to be sorted into their appropriate bins, i.e., the bottles should go to the brown bin. (Right) As the conveyor belt moves, some arms might have to pick up objects for which they cannot access the bin. In this case, the robots should perform hand-off of the objects to other robots who do have access to the appropriate bin.

This work is specifically motivated by the problem of multi-robot task and motion

planning to solve object transfer manipulation tasks. For instance, consider a related motivating challenge shown in Figure 1, which corresponds to an automated recycling problem. This is an object transfer task where the objective is to move every object to a desired target region based on its class, i.e., bottles should be moved to the brown bin, aluminum cans should be moved to the silver bin, and so on. This problem is of particular interest as it represents a less-structured environment relative to those handled by current automation systems. That is, most manufacturing plants assume a highly structured environment where motion plans for moving a particular part can be computed a single time in an offline fashion, since the starting state of the object to transfer is always the same. In this problem, this will generally not be the case, and often the object to be manipulated will not have a known, predefined shape during preprocessing, making these traditional approaches to the problem poorly suited to the task.

The task and motion planning aspects of the problem are addressed in this work, motivated by solving these problems quickly given no prior knowledge of where objects will be in the scene. Solutions are presented that are able to provide formal guarantees in terms of overall path quality for the given task and motion planning problem. The key contribution and overall theme of this work is developing compact planning representations that are efficient to query while also providing formal guarantees on the length of solutions returned by the method. This work will generally assume solutions to other subproblems, such as robot vision and perception, grasp planning, and feedback control, and will therefore focus more on the motion planning and task reasoning aspects of the problem.

1.1 Objective

This work seeks to develop methods for efficiently solving the object transfer manipulation task. Specifically, it focuses on providing **formal guarantees** on path quality using **compact planning representations**. *The objective is to provide methods that can be deployed in multi-arm object transfer manipulation problems like the automated recycling center example and provide high throughput by employing provably near-optimal*

solutions. This section continues by discussing these two objectives, which are at times in opposition, in greater detail.

1.1.1 Formal Guarantees on Path Quality

Throughout this work, sampling-based motion planning is heavily used and forms the backbone of the final search method proposed in chapter 6 [73, 90]. These methods were originally introduced as a means to overcome the difficulty of performing motion planning for high-dimensional systems, but gave up on traditional guarantees, such as completeness and optimality. Analysis showed that these methods could provide asymptotic properties instead, i.e., formal guarantees, generally in terms of returning paths and relating the quality of those paths, which hold as the algorithm is run to an infinite iteration limit [85, 68]. Prior work showed that these methods can provide *probabilistic completeness* [71], *asymptotic optimality* [68], and *asymptotic near-optimality* [103], which are concepts expanded upon in chapters 2-4. In theory, it would be desirable to be able to create an oracle function that would take as input any start and goal pair for the motion planning problem and returns the optimal sequence of controls for the robot to achieve this. This oracle could for example correspond to an infinitely dense planning structure, over which an optimal discrete search could be performed.

1.1.2 Compact Representations

In general, sampling-based planners construct a planning structure in the form of a tree or a graph that is used to solve motion planning queries, i.e., for providing long-term control sequences to bring the robotic system from some start configuration to a goal configuration. Ideally, such a planning structure is lightweight and can quickly be queried for a solution. The proposed concept of an oracle presented in the previous subsection runs counter to this idea, as it was posed as a solution to providing optimal solutions, which in the general case may require that the underlying data structure is dense. Instead, this work will show that compact representations can provide near-optimality guarantees while maintaining a relatively sparse data structure, having orders of magnitude fewer nodes and edges than competing methods which provide

asymptotic optimality guarantees. Furthermore, this work presents compact representations for performing the multi-robot object transfer manipulation task, providing minimal representations to speed up online search.

1.2 Dissertation Overview and Contributions

The outline and contributions of this work are now described in further detail. The manuscript is divided into two high-level sections. The first section is comprised of chapters 3 and 4 and focuses on the problem of single-robot motion planning, both extending the types of formal guarantees that can be provided with sampling-based planners, as well as introducing a compact representation for efficient motion planning. The second section is comprised of chapters 5 and 6 and focuses on the multi-robot and task planning aspects of the problem, first by extending formal guarantees to efficient multi-robot sampling-based motion planning, and then by presenting a compact representation for solving object transfer manipulation tasks. Note that most of the work presented here is taken directly from related publications. Large portions of chapters 3, 4, 6, and 5 are taken directly from the referenced publications by the author.

Chapter 2 introduces the basic concepts and foundational work in the areas of algorithmic motion planning, robot task planning, and manipulation planning. It begins by reviewing algorithmic motion planning, giving an overview of the types of the generally leveraged in this work. These are known as sampling-based planners, and gain practical efficiency by having lightweight modules for quickly validating or invalidating individual configurations for a robotic system. The chapter spends time illustrating the state-of-the-art and giving the basic framework upon which much of this dissertation's work is founded: the Probabilistic Roadmap Method (`PRM`) [73]. The chapter also introduces the basic concepts and terminology of multi-robot task, motion, and manipulation planning. It discusses the current state-of-the-art in these directions and gives a general overview of the types of approaches taken to solve task and motion planning problems. It also outlines assumptions about the problem to be solved and presents the object transfer task studied in this work.

Chapter 3 discusses the properties of sampling-based motion planners and proposes that the asymptotically optimal variants of these approaches have finite-time near optimality properties which formally guarantee that the methods return paths of bounded length relative to clearance-robust optimal paths. This joint work with George V. Moustakides formalizes the notion of **probabilistic near-optimality** [27], proving that a variant of the PRM* method [68] exhibits these properties. It draws bounds and leverages them to provide practical tools for generating principled stopping criteria for these methods, as well as for estimating solution non-existence probability. It probabilistically guarantees that the length of the solution returned by PRM* is within a finite bound of a clearance-robust optimal path after a finite amount of computation, which is a novel kind of property for these kinds of methods. Furthermore, as part of the proof presented in chapter 3, a novel approximation to an unsolved variant of the ball-line picking problem [129] is drawn, which may have relevance to other fields beyond this work. Experimental validation shows that this bound accurately reflects the actual execution of sampling-based planners, but also re-captures the *curse of dimensionality*, showing that it requires exponentially many samples to provide path quality guarantees for higher dimensional systems.

Chapter 4 attempts to reconcile the need for exponentially many nodes in the planning structure by relaxing optimality constraints. Instead, it develops planners that exhibit **asymptotic near-optimality** properties [103], and does so by formalizing a method to create a sparse roadmap spanner (SPARS) [31]. A sparse roadmap spanner is a **compact planning representation** containing orders of magnitude fewer nodes and edges than asymptotically optimal roadmap methods while still providing high-quality paths in practice. The methods provide formal near-optimality guarantees, asymptotically guaranteeing that the method converges to returning paths within a bound of the optimal solution. In addition to these guarantees, the method is shown to be asymptotically sparse, which guarantees that the probability of growing the structure as the method runs to infinity tends to 0. Experimental validation shows these approaches quickly provide paths, which are of much higher quality than theoretical bounds suggest.

Chapter 5 contains the collaborative contributions in the direction of multi-robot motion planning. This work which was coauthored by Kiril Solovey, Rahul Shome, and Dan Halperin extends the efficient `dRRT` framework [139] to be **asymptotically optimal** [33], and shows that with practical speed-ups, the method quickly provides high-quality solutions to multi-robot motion planning problems even for a large number of robotic systems. Furthermore, this is all accomplished without explicitly constructing a roadmap in the composite configuration space of the robots. Instead, it defines a tensor product roadmap implicitly, which makes this a different kind of compact representation. The method then extends an explicit planning tree over this structure to perform online search. Experimental results show that the method is scalable to planning for several high-dimensional robotic systems while quickly returning initial solutions of low cost.

Chapter 6 addresses the object transfer manipulation task, incorporating the results from the previous chapters. It outlines both appropriate preprocessing and online search methods for performing this task. The precomputation begins by constructing a **compact representation** of the many arm manipulation task topology that was generalized from prior work for dual-arm manipulation problems [52, 30]. It then constructs roadmaps for each robot arm and employs the multi-arm motion planning framework `dRRT`* detailed in the previous chapter, and by integrating this method, it provides formal proof showing the method is **globally asymptotically optimal**. That is, not only does the approach asymptotically converge to returning optimal motion plans for the arms involved in the task, but it also converges to returning the optimal sequence of actions, object grasps, and placements for solving the task. Experimental results show that the approach is scalable to several robot manipulators and quickly generates initial solutions.

Finally, **chapter 7** restates the key contributions of the work and summarizes open questions and interesting directions for future investigation. Furthermore, it cites the potential for this work to inform future investigation of these methods.

Chapter 2

Foundations of Robot Task, Motion, and Manipulation Planning

This chapter begins by introducing the concept of single-system **Robot Motion Planning**, which remains the general theme of this work through the first half of the work. These concepts are primarily detailed by chapters 3 and 4. Informally, motion planning is the process by which a robot system uses an internal representation of its environment to compute a sequence of controls to actuate the system, so as to move the robot system from an initial position to a target position while remaining safe, i.e., avoiding collisions and maintaining constraints, such as keeping balanced. The canonical version of the problem is known as the **Piano Mover's Problem**, which is posed for a collection of kinematic, polyhedral rigid bodies in a polyhedral workspace. Even this simple version of the problem is $\texttt{PSPACE} - \texttt{HARD}$ [122]. This initial work showed that the difficulty of the problem scales exponentially with the number of degrees of freedom of the robotic system.

A complete algorithm for solving the Piano Mover's Problem was presented in the thesis of Canny that directly shows this exponential complexity [17]. This algorithm operates in the robotic system's configuration space ($\mathcal{C}$), where a point in this space is called a configuration. A configuration is the parameterization of a robotic system's degrees of freedom, where this set of parameters fully specifies the location of every rigid body geometry of the robotic system. Many of the approaches examined in this chapter partition this space into two sets: the invalid or obstacle set $\mathcal{C}^{\text{inv}}$ where the robotic system is violating constraints of the problem (most commonly collisions with static geometry in the scene), and $\mathcal{C}^{\text{free}}$ where the robotic system's configuration is valid and safe. These partitions cover the whole space, i.e., $\mathcal{C} = \mathcal{C}^{\text{free}} \cup \mathcal{C}^{\text{inv}}$.

The algorithm posed by Canny introduced the concept of a roadmap, a one dimensional subspace of $\mathcal{C}^{\text{free}}$ that captures its connectivity, which is generally represented by a graph data structure. The algorithm however is impractical, similar to other early methods that attempted to leverage $\mathcal{C}^{\text{free}}$ approximations [14, 101, 67].

Other approaches not explored in this work approach the problem from different directions. Some approaches apply a grid discretization over the configuration space and perform discrete search directly [144]. Another paradigm designed to solve these problems is based on artificial potential fields [75, 61, 42], though complete versions of this methodological approach are difficult to apply in general configuration spaces [77, 123]. A similar technique that solved difficult problems took a stochastic approach to avoid local minima [6, 88]. The difficulties faced by these methods and the advent of efficient collision checking primitives would motivate the development of sampling-based motion-planning approaches, which will *largely* be the focus of this dissertation. The principle ideas behind sampling-based planners are explored in Section 2.1. Another approach to the problem is to directly produce minimum-cost trajectories via optimization methods [9, 164].

Ideally, efficient motion planners would be able to quickly identify compact graphical representations that nearly approximate the free configuration space, similar to the original roadmap concept. This is the primary objective behind sampling-based roadmap planners, which will be explored in more detail in Section 2.1.2 [73, 72]. Ideally, such a roadmap is quick to return results to shortest-path queries [89, 1]. Furthermore, it would be beneficial to provide guarantees on the length of paths returned by such a roadmap, bounded by the length of the true optimal path through $\mathcal{C}^{\text{free}}$.

In the second half of this dissertation, the focus shifts to more challenging practical problems and the motivating application of automated recycling, outlined in chapters 5 and 6. To this end, it begins exploring robotic applications that require multiple robot manipulators with fixed bases operating within the same workspace. These robots will be centrally coordinated in order perform manipulation tasks, and planned for in a coupled fashion in order to provide formal guarantees.

A naïve approach to solving the motion planning problem for these systems would

be to construct a sampling-based roadmap or tree for all of the robots simultaneously. That is, the method would operate directly in the composite configuration space of R robot arms simultaneously. While this theoretically provides asymptotic optimality (or asymptotic near-optimality) for the multi-robot problem, it does not scale due to the exponential memory dependence on the problem's dimensionality. In such a setup, the dimensionality of the problem increases very quickly with the number of robots, R. In the motivating example, each robot manipulator typically has around seven degrees of freedom, quickly making this approach infeasible for more than two arms.

Beyond the problem of achieving simultaneous motion for these robot systems, this work also proposes methods for performing manipulation task planning in order to address the target recycling application. This requires finding a sequence of high-level plans that allow the robot arms to manipulate a target object, placing it in an appropriate target location. This requires both finding appropriate subsets of arms $R_{\text{task}} \subset R$ to manipulate the object with and finding a sequence of such sets of arms that accomplishes the given task. Even given an appropriate sequence of arm sets, the method must also leverage appropriate multi-robot motion planning to achieve these actions. Naïvely enumerating these sequences is obviously infeasible. The motion planning aspect of this problem is further complicated by the need to perform manipulation tasks. Due to these manipulation tasks, these methods will generally require specialized sampling methods in order to correctly plan for manipulation, as the set of configurations that correspond to object grasping configurations are a lower-dimensional manifold of the entire configuration space. This means that a sampling scheme which naïvely samples in the whole configuration space will find such a transition point with probability zero.

2.1 Sampling-based Motion Planning

Sampling-based motion planning techniques provide efficient solutions in practice, even for high-dimensional, geometrically complex problems [89, 91, 22]. Two primary families of planners have emerged. Probabilistic Roadmap Methods (`PRM`s) preprocess a robot configuration space ($\mathcal{C}$ -space) to create a multi-query structure [73, 72]. Tree-based

planners, such as the Rapidly-exploring Random Tree (RRT) are suited to rapid single-query planning, especially for dynamic systems [90, 92]. There are also methods that lie somewhere between, in order to reap the advantages of both algorithmic approaches [117, 2]. In general, these methods rely on several primitives to construct a planning structure in the configuration space, which are detailed next.

2.1.1 Sampling-based Planning Primitives

As implied by the name, these methods employ some sampling method to generate free configurations in the configuration space. There has been success in the field with several approaches to sampling, though most commonly these methods are leveraged using uniform random samplers. Another approach that has good properties in terms of low sample dispersion is quasi-random deterministic sampling sequences [13, 124, 157, 64, 63, 65], and there exist comparative studies on the subject [43]. Another approach is to attempt to leverage $\mathcal{C}$-space projections to produce samples on the medial axis, so as to create paths with maximum clearance from obstacles [160, 51]. As approximating invalid regions in the configuration space is difficult, other approaches adapt sampling strategies and leverage information from invalid samples to appropriately generate samples within narrow free passages within the $\mathcal{C}$-space [57]. Others yet have shown the merits of sampling using non-uniform distributions [12].

Traditionally, these approaches are applied to find shortest paths through the configuration space according to a distance function.

Definition 1 (Distance Function)**.** *The distance function $d(\cdot,\cdot)$ takes two configurations in $\mathcal{C}$ and returns a real value, i.e., $d(q_i, q_j) \to \mathbb{R}$, which expresses the distance of the two configurations in the absence of obstacles and generally satisfies metric or pseudometric properties.*

In order to construct the planning structure in $\mathcal{C}^{\text{free}}$, sampling-based methods rely on being able to accept or reject samples based on whether a configuration is safe. That is, the approach leverages a method which determines whether configuration q lies within $\mathcal{C}^{\text{free}}$ or $\mathcal{C}^{\text{inv}}$. Prototypically, these methods rely on the availability of a collision

checker to determine the validity of samples, but the general term for such a module is a validity checker.

Definition 2 (Validity Checker)**.** *Given an individual configuration q, a validity checker returns whether q lies within $\mathcal{C}^{\text{free}}$ or $\mathcal{C}^{\text{inv}}$.*

The following section will outline Probabilistic Roadmap Methods, which produce a planning structure that is a graph. Samples drawn within $\mathcal{C}^{\text{free}}$ will be added as nodes, but in order to build the planning structure, the sampled configurations must be connected by edges. This is typically accomplished with the aid of a local planner.

Definition 3 (Local Planner)**.** *Given configurations q_{begin}, q_{end}, a local planner $\mathbb{L}(\cdot,\cdot)$ returns the optimal path between the configurations* **in the absence of obstacles***, i.e., $\mathbb{L}(q_{begin}, q_{end}) \rightarrow \pi_{\mathbb{L}}$, where $\pi_{\mathbb{L}}(t) \rightarrow \mathcal{C}$ and $t \in [0,1]$, satisfying $\pi_{\mathbb{L}}(0) = q_{begin}$ and $\pi_{\mathbb{L}}(1) = q_{end}$.*

Typically, a straight line between q_{begin} and q_{end} in $\mathcal{C}$ is used for local planning, especially in the canonical Piano Mover's Problem. In order to add local paths as edges in the planning structure, it must be that for all $q \in \mathbb{L}(q_{\text{begin}}, q_{\text{end}})$, $q \in \mathcal{C}^{\text{free}}$. While there exist efficient and complete methods for checking if an entire path lies entirely within $\mathcal{C}^{\text{free}}$ [135], a sampling-based process is commonly used as an approximation.

2.1.2 Probabilistic Roadmap Methods

One of the first popular sampling-based motion planning methods in order to construct a roadmap in $\mathcal{C}^{\text{free}}$ was the **Probabilistic Roadmap Method** (`PRM`) [73]. The high-level operations of `PRM` are outlined in Algorithm 1. For a set number of iterations n (Line 2), the algorithm samples a configuration in $\mathcal{C}^{\text{free}}$ (Line 3) and adds it as a node to the roadmap (Line 4). It then tries to connect it with a local path to a set of k-closest neighbors among the existing nodes (k-`PRM`) or those within a δ-ball (δ-`PRM`) (Lines 5,6). If any local path lies entirely within $\mathcal{C}^{\text{free}}$ (Line 8), an edge is added to the roadmap (Line 9). The advantages of the method lie in its simplicity, and generality while scaling to higher-dimensional problem instances than other competing methods

Algorithm 1: PRM(n)

```
V ← ∅; E ← ∅;
for i = 1 ... n do
    v ← SampleFree;
    V ← V ∪ v;
    r_i ← CONNECT_RADIUS(i);
    U ← NEAR(V, v, r_i);
    for u ∈ U do
        if 𝕃(v, u) ∈ C^free then
            E ← E ∪ {𝕃(v, u)};
        end
    end
end
return G = (V, E);
```

were capable of. While strict completeness and path non-existence cannot typically be proven for the PRM, probabilistic completeness can be provided instead.

Definition 4 (Probabilistic Completeness)**.** *Let* $(\mathcal{C}^{\text{free}}, q_{start}, q_{goal})$ *be a motion planning problem that admits a continuous trajectory* $\pi : [0,1] \to \mathcal{C}^{\text{free}}$ *subject to* $\pi(0) = q_{start}$ *and* $\pi(1) = q_{goal}$*. Then, an algorithm* ALG *that is run for* n *iterations is probabilistically complete if the probability that* ALG *returns a solution to the motion planning problem converges to* 1 *as* n *tends to infinity.*

This was originally proven for d-dimensional manifolds [71, 58], for non-holonomic robots [148], and then later for a broad class of problems [85].

Many early variants of PRM focused on providing solutions as quickly as possible while addressing issues related to narrow passages [3]. Some approaches were tailored to improve different criteria, such as returning high-clearance paths [160], and others to return high quality solutions experimentally [21, 121]. While the types of plans produced by these methods are open-loop and generally susceptible to noise or model errors, they have been extended to address such problems; for instance, some methods perform belief space planning for dealing with uncertainty [120, 15].

The efficiency of probabilistic roadmaps is dominated by the presence of "narrow passages", which require sampling from a low-measure subset of the configuration space in order to discover them. In difficult problem instances, the optimal path that answers

a given query may be required to traverse one or more of these narrow passages, greatly decreasing the probability that a sampling-based approach returns such a path. This motivated the development of variations that appropriately sample configurations to speed up the construction of sufficiently connected roadmaps [3, 58, 12, 160, 51, 11, 39, 57, 95, 117]. Furthermore, the PRM framework has been adapted so as to solve a variety of different challenges beyond the basic Piano Mover's Problem, involving multiple robots [127], manipulation planning [109], assembly planning [147], planning for flexible objects [87] and bioinformatics applications [4].

A variation of PRM closely related to the work presented in chapter 4 is the Visibility-based PRM [137]. This prior work rejects specific samples while adding only those required for coverage and connectivity purposes. The combination of these two properties is sufficient for probabilistic completeness. The resulting roadmaps are lightweight tree structures, because two nodes are not connected if they belong to the same connected component. Furthermore, the technique provides an automatic stopping criterion: when M consecutive samples fail to be added to the roadmap, then a probabilistic estimation of the percentage of free space not covered by the nodes of the data structure is $\frac{1}{M}$. Unfortunately, the path quality of paths returned by this method can be quite poor.

This led to work that aimed to identify "useful cycles", which adds edges between roadmap nodes if the existing path connecting the nodes is sufficiently lengthy [110]. The resulting structure is no longer a tree, but remains sparse and improves path quality. The "useful cycles" criterion has been combined with the Reachability Roadmap variant of PRM to return high clearance paths in 2D and 3D $\mathcal{C}$-spaces [44]. One way to improve path quality is through a post-processing, smoothing phase. Methods that reason about path homotopy provide solutions that can be smoothed to optimal ones [62, 133]. Hybridization graphs can be seen as a smoothing process that combines multiple solutions into a single, better quality one [121]. While such smoothing-based approaches are valid alternatives in certain cases, they construct relatively dense roadmaps and increase the online query resolution time. The proposed SPARS framework addresses these issues, creating a compact planning representation that is very quick to query, while providing path quality guarantees and practically returning paths of low cost.

2.1.3 Random Tree Methods

An extremely popular alternative to roadmap-based methods explore the $\mathcal{C}$-space by incrementally propagating from existing nodes in the planning structure to grow a tree. One such common and popular method is the Rapidly-exploring Random Tree (`RRT`) approach [93]. This method is commonly employed due to the inherent "Voronoi Bias" it exhibits. That is, the tree is automatically biased toward quickly growing toward unexplored regions of the space. Around the same time, a similar tree-based approach was proposed that probabilistically biases the expansion of these trees toward unexplored regions of the space [59, 60].

These methods tend to be more efficient in quickly answering individual motion queries and they can be easily applied to problems involving robot dynamics because they do not depend on the existence of a steering method that exactly connects two configurations of the system. Such a steering function is typically required in the construction of a roadmap. Furthermore, by definition they already return sparse data structures, as a tree is minimally sparse; however, it does not ensure that the number of nodes in this tree remain low.

Nevertheless, tree-based approaches do not provide the same properties in terms of preprocessing the entire free configuration space in order to be able to answer multiple, unknown queries. Moreover, the basic `RRT` approach has been shown to almost certainly converge to suboptimal solutions [108]. It can be extended, however, to an asymptotically optimal variant, known as `RRT`* [68], which however does require a steering method. Further work in this direction has proven that by employing random controls, asymptotic optimality can be ensured for these types of methods without the need for a steering function [98]. These tree-based methods can also benefit from sparse roadmaps that can quickly return C-space distances among obstacles [97]. Another approach focusing on practical applications incorporates feedback control policies directly into a rapidly-exploring tree structure [151].

2.2 Properties of Sampling-based Planners

Formal analysis showed that sampling-based planners provide probabilistic completeness, which guarantees that if the provided problem instance has a solution, the methods solve the problem with probability asymptotically approaching 1 [58, 71, 85, 18]. While many approaches adopt a quasi-random sampling scheme [13], much of the body of analysis has focused on uniform random sampling. This is aided by a body of literature that supports such uniform processes, such as studies into concentration of measure [36]. Furthermore, in the absence of obstacles in the configuration space, roadmap-based methods return random geometric graphs [141], which have been extensively studied in other contexts [116]. Further work showed that solution non-existence can be detected under certain conditions [107] or that a solution is guaranteed to exist under others [99].

Important recent work has provided the conditions under which sampling-based methods are **asymptotically optimal** [68, 69].

Definition 5 (Asymptotic Optimality)**.** *Let* $(\mathcal{C}^{\text{free}}, q_{start}, q_{goal})$ *be a robustly feasible motion planning problem that admits a continuous trajectory* $\pi : [0,1] \to \mathcal{C}^{\text{free}}$ *subject to* $\pi(0) = q_{start}$ *and* $\pi(1) = q_{goal}$*. Then, an algorithm* `ALG` *that is run for* n *iterations is asymptotically optimal if the probability that* `ALG` *returns a solution of minimum cost to the motion planning problem converges to* 1 *as* n *tends to infinity.*

The analysis indicates that for the `PRM` methods, the critical algorithmic requirement for asymptotic optimality is the number of neighbors that each new sample should be connected to. Table 2.1 briefly gives an overview of different `PRM`-based approaches and their asymptotic properties. A simple `PRM` that connects samples to neighbors within a δ-ball is asymptotically optimal, but results in a very dense roadmap. The roadmap's density can be reduced by considering the k-nearest neighbors of the sample, but this version is not asymptotically optimal. $\texttt{PRM}^*$ and $\texttt{k} - \texttt{PRM}^*$ rectify this by selecting the minimum number of neighbors required for asymptotic optimality, which is a logarithmic function of the number of nodes already in the planning structure [68, 69]. Nevertheless, all samples are added as nodes, resulting in a large graph and

the resulting structure can still be relatively slow to query. Roadmaps with asymptotic optimality properties have large memory requirements and take longer to query than other approaches focusing on sparsity.

algorithm	edges	optimal?
δ-PRM	$O(n^2)$	asymptotically optimal
k-PRM	$O(kn)$	no
PRM*	$O(n \log n)$	asymptotically optimal
k $-$ PRM*	$O(n \log n)$	asymptotically optimal
SRS	$O(an^{1+\frac{1}{a}})$	asymptotically near-optimal
IRS	$O(n \log n)$	asymptotically near-optimal

Table 2.1: PRM variations and asymptotic optimality properties. The parameter n corresponds to the number of nodes in the roadmap.

Note that these are asymptotic results, and can not be used to argue much about the solution quality returned after a finite amount of computation, unlike the properties guaranteed in chapter 3. The results from that chapter show that asymptotically optimal planners can however provide good quality paths when halted after finite computation, even if strict asymptotic optimality constraints are relaxed [103, 126, 32].

An approach to returning sparser, high-quality roadmaps is to relax the optimality guarantees by utilizing graph spanners [114]. Spanners are subgraphs, where the shortest path between two nodes on the subgraph is no longer than t times the shortest path on the original graph, where t is called the stretch factor of the spanner. Applying an efficient spanner [8] on the output of k $-$ PRM* resulted in a Sequential Roadmap Spanner (SRS) [105], which reduces the expected number of edges and provides asymptotic near-optimality, i.e., as more time is spent on constructing the roadmap, the quality of solutions converges to a value at most t times the optimal. An incremental integration of spanners with k $-$ PRM* (IRS) has been experimentally shown to provide even better results [104]. The path quality degradation with these methods is quite smaller in practice than the theoretical guarantees.

These spanners, however, only remove edges, and the resulting roadmaps, similar to asymptotically optimal solutions, still need to include every $\mathcal{C}$ -space sample drawn as a node to achieve this property. An initial attempt towards not including all nodes was a simple extension of the IRS approach but did not provide any theoretical guarantees [106]. Chapter 4 introduces the SPARS approach, which was the first method to

provide formal asymptotic near-optimality guarantees where the probability of adding new nodes to the roadmap tends to zero [31]. An updated version of the approach, SPARS2, achieves the same objectives while reducing the memory requirements upon construction [28].

2.2.1 Contributions Relative to the State-of-the-art

Chapter 3 formalizes *Probabilistic Near-Optimality* (PNO) for efficient sampling-based roadmap methods using limited assumptions. A previous contribution of the author highlighted PNO properties for a PRM variant that asymptotically converges to a dense planning structure, relying on Monte Carlo experiments to draw bounds on path length [27]. That chapter formally shows the following contributions:

- The probabilistic near-optimality of PRM* is shown, which allows for intelligent stopping criteria and probabilistic bounds on solution non-existence for general setups.
- To the best of the author's knowledge, this chapter provides a novel approximation to a previously unsolved problem in geometric probability to draw closed-form path length guarantees.
- The analysis works for a variant of PRM* constructing $O(n \log n)$ edges (equivalent to the asymptotically optimal), as opposed to the previously considered PNO-PRM*, which creates $O(n^2)$ edges [27].

This guarantee is similar to Probably Approximately Correct (PAC) solutions in the machine learning literature [153]. Such properties can significantly impact the practical application of these methods. For example, in robot task planning, a higher-level planner often queries a motion planner to determine if a solution exists to perform some action [55]. PNO guarantees can inform a high-level task planner of expected path degradation relative to the optimal path along a homotopic class, as well as provide a probabilistic bound on solution non-existence, which can be used to prune away certain actions with a high confidence that it is infeasible. This further results in automated stopping criteria that are informed on solution quality and the progress of the sampling process rather than the common practice of using an arbitrary, ad hoc stopping criterion.

PNO properties imply that solutions will very often be within a known bound of the optimal. Such a bound can be used to provide many practical benefits. This work formally provides PNO properties for PRM*, reasoning over a theoretical construction of hyper-balls in the $\mathcal{C}$ -space tiled along a clearance-robust optimal path. It is also shown how to apply these properties in practice to ensure high-quality paths, and experimental validation of the bounds is provided. Due to the difficulty of the motion planning problem, PNO properties often require many samples to provide near-optimal paths with high confidence; however, this bound also provides error and confidence bounds for a budget of algorithm iterations.

Based on this paper, it is now possible to estimate in an informed manner the number of iterations needed for PRM* to return with high probability a path within a desired bound of the optimum for different planning challenges. In practice, it will be seen that for sufficiently high-dimensional problems, the exponential dependency on problem dimensionality is recovered. To this end, the following chapter proposes methods for providing formal path quality guarantees while retaining asymptotically sparse planning structures.

Chapter 4 describes how to generate compact planning structures, which return high-quality paths. Specifically, it proposes the sparse roadmap spanner (SPARS) framework. This framework constructs sparse roadmap spanner over the free configuration space $\mathcal{C}^{\text{free}}$, and it is shown that these structures asymptotically converge to returning near-optimal paths while guaranteeing the probability of node addition tends to 0. These structures are practical and can be constructed in a time and memory efficient manner. Examples of these sparse roadmaps constructed with this framework can be seen in Figure 2.1

In particular, the planners for constructing sparse roadmap spanners presented in this chapter have the following properties:

- probabilistic completeness
- asymptotic near-optimality with additive cost
- asymptotic sparsity: the probability of growing the roadmap asymptotically tends

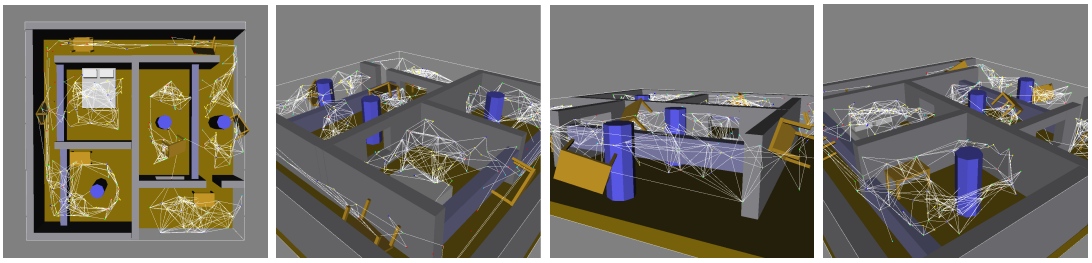

Figure 2.1: A roadmap spanner in the $SE(3)$ "Beam Site" environment, loaded in the `OMPL` software package [24]. Configurations for a table moving along a solution path are highlighted.

to 0.

The resulting roadmap can probabilistically guarantee that it can answer any path planning query in the $\mathcal{C}$ with paths of length bounded by:

$$t \cdot I^*_{\delta_n} + 4 \cdot \Delta, \tag{2.1}$$

where t and Δ are input parameters to the algorithm, and $I^*_{\delta_n}$ is the cost of the optimum path in the free configuration space, if one exists. This framework indicates that finite-sized data structures with this property should be possible to algorithmically construct, though it is unclear what the properties of the space should be in order to guarantee this. The framework is grounded in a concrete implementation, originally called the `SPARS` method, which constructs two graphs in parallel. In addition to the spanner, the method also constructs an asymptotically optimal dense graph using `PRM`* that includes all $\mathcal{C}$-space samples. The use of the dense graph is a significant limitation of this original approach. For this reason, a second implementation of the framework, `SPARS2`, is also presented that provides the same theoretical guarantees without explicitly maintaining the dense graph.

2.3 Multi-Robot Motion Planning

This section reviews related work in the field of multi-robot motion planning problems. Specifically, the focus is on simultaneous motion of R robots $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_R\}$, each operating in their own configuration space $\mathcal{C}_{\mathcal{A}_i}$ It is straightforward to treat all of

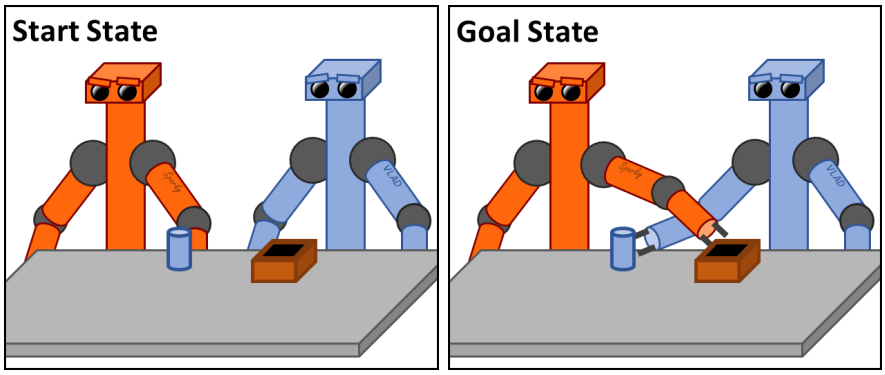

Figure 2.2: Simultaneous planning for multiple high-dimensional systems is a difficult, motivating challenge for this work.

these systems as a single robot by operating in the composite configuration space $\mathcal{C} = \mathcal{C}_{\mathcal{A}_1} \times \cdots \times \mathcal{C}_{\mathcal{A}_R}$, which is the Cartesian product of these configuration spaces. Early attempts at multi-robot motion planning quickly discovered the difficulty of naïvely planning in the composite robot configuration space.

Directly solving the problem in this fashion is referred to as coupled multi-robot planning. There is a great deal of literature on multi-robot planning for pebble motion style problems where simple, low dimensional systems are coordinated on a shared motion graph [162, 163, 84]. There have been a variety of planners developed to address the challenge for higher dimensional systems [128]; however, coupled methods often do not scale well, though they have completeness guarantees. The motivating challenge in Figure 2.2 remains especially challenging due to the relatively high number of degrees of freedom for each robot. The alternative to coupled planning is decoupled planing, where paths for robots are computed individually and then coordinated online to avoid robot-to-robot collision [94], which scales much better. Decoupled methods typically lack completeness and optimality guarantees; however, hybrid approaches can achieve optimal decoupling to retain guarantees [155], even for complex, high-dimensional systems [159].

Other early attempts focused on finding proper methods for composing several roadmaps constructed for each of the robots in an intelligent manner. These approaches created a super-graph over these roadmaps [48, 47], but a more modern approach known

as the discrete RRT (dRRT) method [139] provides scalability by searching over the implicit tensor product [149] of the single-robot roadmaps. This inspired a recent extension, which is detailed further in chapter 5 known as dRRT*, which achieves asymptotic optimality as well as improved performance through an informed search process [33].

Other work focuses on many of the complicating factors of practical planning, such as dealing with complex kinodynamic constraints [115], while others focus on coordinated manipulation [134]. There exist control-based methods that have the advantage that they scale to hundreds or thousands of robots, but generally lack global guarantees and only perform local, reactive collision avoidance rather than long-horizon planning [154, 150]. Many efforts in this domain focus on dealing with a large number of objects, such as in the context of rearrangement planning [112, 83] and navigation [156, 19, 96, 145] or manipulation among movable objects [146]. This work assumes objects are directly reachable so that rearrangement is not needed but it can potentially be integrated with such solutions when this is not the case.

2.3.1 Contributions Relative to the State-of-the-art

Chapter 5 outlines the dRRT* method, which is an extension of the prior dRRT method. It achieves asymptotic optimality for the multi-robot motion planning problem without requiring explicitly representing the entire planning structure in memory, making it a practically efficient approach that quickly generates initial solutions. Formally, chapter 5 sets up the multi-robot motion planning problem, and provides the following contributions:

- Outlines modifications to the dRRT method to speed up its practical performance.
- Proves the updated dRRT* method is asymptotically optimal for the multi-robot motion planning problem.
- Outlines simulated verification of the method that shows it scales well to a large number of robots.

This represents significant progress in terms of extending asymptotic optimality to practical multi-robot methods. It also suggests that explicitly representing large, dense planning structures is not necessary to obtain optimal path quality guarantees for high-dimensional systems, which was a practical limitation of the finite time properties of motion planners outlined in chapter 3.

2.4 Task Planning and Robot Manipulation

While motion planning focuses on computing controls to move robotic systems from a known start configuration to a known goal configuration, task planning focuses on determining a sequence of high-level, often abstract actions to change the state of the world. While both motion and task planning have many roots in traditional artificial intelligence work [53, 125, 89, 91], task planning often focused on traditional logic-based approaches. The high-level reasoning employed simple predicate logic [37], while eventually moving on to more expressive languages, such as linear temporal logic (LTL) [119], and to one of many flavors of planning domain definition languages (PDDL) [46, 80].

Traditionally, to solve integrated task and motion planning problems, a hierarchical scheme was employed that first found a sequence of actions that would logically solve the problem specification, and then attempted to use a motion planner to determine how to physically perform each action [111]. If the motion planner fails to produce a solution, then a new sequence of actions must be determined, and this ends up being an inefficient solution to the problem. Instead, more advanced techniques attempt to perform integrated task and motion planning, where both the task planner and motion planner are informed about constraints in each other to allow direct computation of motion plans that perform high-level actions [16, 54, 66, 41, 118].

There are also incremental task and motion planning methods, which solve the problem in an iteratively deepening fashion [26]. This related approach, like several others, casts the task planning problem as a constraint satisfaction problem. Related approaches determine a high-level plan skeleton and then directly use incremental CSP

solvers over discretely sampled object grasps and placements [102, 41]. This prior work attempts to factor the problem into smaller, independent sets of state and control, much like the proposed work reasons over multiple robotic systems. The prior work, while efficiently solving complex problem instances, does not provide asymptotic optimality guarantees.

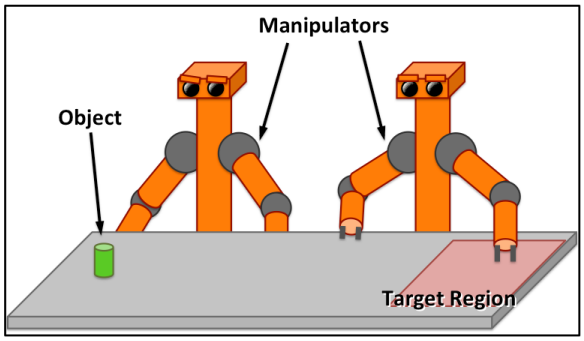

Figure 2.3: An example with $n = 4$ arms. In this setup, no arm can reach both the object's starting pose and the target region.

A particularly interesting task planning problem is that of robot manipulation: using a robotic system with a manipulator in order to interact with the physical geometry around it. Many such manipulation tasks have general interest in a wide variety of applications, and even relatively simple tasks, such as the object transfer problem illustrated Figure 2.3 prove to be challenging problems.

There are many efforts to perform integrated task and motion planning for manipulation [16], several of which are hierarchical in nature [161, 7], with extensions that interleave planning and execution, but require reversible plans [66]. Others attempt to leave the task and motion planning as black boxes, instead developing an intermediate layer to coordinate off-the-shelf planners [143]. An effective technique for improving the practical performance of these methods is to defer resolution of geometric constraints arising from motion planning for as long as possible [35]. The current work shares this objective through a similar lazy evaluation procedure. To increase the chances of motion plans being found for specified high-level actions, some sampling-based motion planners for manipulation planning directly plan in task-constrained manifolds, such as keeping an end-effector level with a table [10] or for folding clothes [86]. There is work that focuses on leveraging information about an arm's reachability to more efficiently

search for planning solutions [45, 152].

The current work performs a forward search, and is compared against a similar forward search approach [55]. other methods similarly perform forward search and a related work [40] incorporates the `FF` heuristic [56] into a task-level specification, allowing geometric constraints to be encoded into the heuristic. This prior work preprocesses object poses and grasp information during the construction of roadmaps for the robots. This limits the applicability of the approach to a small set of predefined objects, which the current work aims to avoid.

There is also work that explicitly reasons about the multi-modal nature of multi-arm manipulation [79, 78]. Early efforts provided the "manipulation graph" abstraction, which formally reasons about robot-object contacts [25], and a formalization of constraint manifolds, which correspond to object grasps and stable pose states, was developed for single-arm manipulation [136]. While there has been extensive work on dual-arm manipulation problems [138], formally identifying the topology of dual-arm manipulation was not done until recently [52]. This was then extended to the general multi-arm case [30]. For multi-arm manipulation, a method proposed the computation of a trajectory for the movable object as a heuristic guidance for computing the manipulation action of multiple arms along the object path [23].

A closely related recent work has shown that asymptotically optimal manipulation task planning can be performed with sampling-based approaches [158]. Unfortunately, the corresponding approach suffered from lack of scalability. While sharing the theoretical asymptotic optimality properties of the prior approach, the current work introduces scalable preprocessing primitives, which generalize to many types of objects, and performs an informed online search that is scalable for several robot manipulators. Further work in this direction [132] showed that a previously proposed practical approach [54] can achieve asymptotic optimality by preprocessing a set of roadmaps, one for each of a finite set of object contacts, which are then merged. Such an approach is limited in that the preprocessing assumes a single, known object, and does not scale to novel objects or novel object poses.

2.4.1 Contributions Relative to the State-of-the-art

In chapter 6 a compact representation of multi-arm manipulation is formalized that is sufficient to solve complex object transfer tasks using fixed-base manipulators performing hand-offs. The chapter also formalizes how to perform adequate offline preprocessing over the representation, which provides helpful search heuristics [40] as well as efficient online search to quickly produce solutions. The search integrates the dRRT* method from chapter 5 for scalable multi-arm motion planning within the framework in such a way that the desirable asymptotic optimality guarantees [158] are achieved but in a computationally efficient manner. Formally, chapter 6 sets up the R-arm object transfer manipulation problem, providing the following contributions:

- A minimal topology for prehensile multi-arm manipulation problems is codified, generalizing the "manipulation graph" of prior work for single-arm manipulation [136, 79, 78, 25].
- A framework for appropriately preprocessing over this topology and performing efficient online search to quickly produce solutions to the object transfer problem.
- A proof for the asymptotic optimality of the method for the integrated task and motion planning for the object transfer problem.
- Experimental validation of the method that shows scalability and practical efficiency of the approach.

This means this work automatically identifies which arms are required and in what order to solve the object transfer problem, including the sequence of hand-offs and required grasps to achieve them. The provided topology is a "multi-arm manipulation graph" ($G_{\mathtt{MAM}}$), which has high-level states corresponding to stable object poses and hand-offs, restricting these modes based on the spatial interaction of the robots.

Chapter 3

Probabilistic Near-Optimality after Finite Computation

This chapter outlines a fundamental property of sampling-based planners that has been investigated only recently. These planners exhibit **Probabilistic Near-Optimality** in finite time; that is, by examining the rate of convergence of these methods to returning near optimal solutions, it is possible to have tight probabilistic bounds on the quality of the path returned by such a method. Previously the literature focused on asymptotic properties, which can only indicate how the algorithms behave in a hypothetical infinite-horizon planning setup. While asymptotic properties have proven useful for guaranteeing some amount of robustness for these methods, they are often lacking in practical guarantees for actually running these methods with, necessarily, a finite time budget.

3.1 Problem Setup

This chapter will reason over the configuration space $\mathcal{C}$ as a metric space, using the Euclidean L_2-norm as a distance metric. For robustly feasible motion planning problems, there exists a set of δ-robust paths that answer a query (q_{start}, q_{goal}), of which, the path of minimum length from that set is denoted as $\pi^*_{\delta_n}$ with length $I^*_{\delta_n}$. Such a path is known as a robust-feasible path:

Definition 6 (Robust Feasible Path)**.** *A path, π_δ, is δ-robust if π_δ has strong δ-clearance from $\mathcal{C}^{\text{inv}}$ of at least δ.*

Formally, this chapter focuses on methods solving the following problem:

Definition 7 (Robustly Feasible Motion Planning)**.** *For a Robustly Feasible Motion*

Planning (RFMP) Problem, $(\mathcal{C}, q_{start}, q_{goal}, \delta_0)$*, and a clearance value* δ_0 *so that a* δ_0*-robust path* π_{δ_0} *exists,* $\pi_{\delta_0}(0) = q_{start}$ *and* $\pi_{\delta_0}(0) = q_{goal}$*, find a solution path* $\pi_n(\tau) \rightarrow \mathcal{C}^{\text{free}}$ *so that* $\pi(0) = q_{start}$ *and* $\pi(1) = q_{goal}$*.*

Traditionally, sampling-based motion planners that can solve this problem offer no insight into the probability that an answer is returned, nor if such an answer will be close to optimal. The analysis takes a volumetric reasoning over the δ_n-robust optimal path, $\pi^*_{\delta_n}$. Note that this reasoning does not necessarily consider a specific optimal path, but rather any such optimal path of length $I^*_{\delta_n}$ and clearance δ_n.

This chapter examines a variant of PRM*, which is an asymptotically sparser approach than the PNO-PRM* method proposed in some of the initial work in this direction [27]. The high-level operations of PRM* are outlined in Algorithm 1, in Section 2.1.2. The difference between PRM, PNO-PRM*, and PRM* is the connection radius used (Line 5). While the original PRM method and PNO-PRM* were proposed using a fixed constant for the connection radius, the new PRM* variant use a radius that decreases over time, while still maintaining the necesssary connectivity for asymptotic optimality. Both methods use as input a desired clearance, δ_0. This clearance corresponds to the clearance for the δ-robust optimal path the methods reason over. PNO-PRM* uses a fixed-radius for CONNECT_RADIUS of $\frac{3\delta_0}{2}$, producing an asymptotically dense data structure. The proposed PRM* variant instead uses $\gamma_{\text{PNO}}(\frac{\log n}{n})^{\frac{1}{d}}$, which produces an asymptotically sparser structure. The analysis will show that the proposed variant of PRM* uses double the connection radius of the original PRM*, leading to the following property:

Property 1 (Probabilistic Near-Optimality for RFMP)**.** *An algorithm ALG is probabilistically near-optimal for an RFMP problem* $(\mathcal{C}, q_{start}, q_{goal}, \delta_0)$*, if for a finite iteration* $n > n_0$ *of ALG and an error threshold* ϵ*, there is a probability* $\mathbb{P}_{success}$ *such that a path* π_n *of length* I_n *exists in the planning structure computed by ALG and answers the query such that:*

$$\mathbb{P}(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n}) < 1 - \mathbb{P}_{success}$$

where $I^*_{\delta_n}$ *is the length of the optimum* δ_n*-robust path* $\pi^*_{\delta_n}$ *for a value* $\delta_n < \delta_0$*, and* n_0 *is some minimum iteration threshold.*

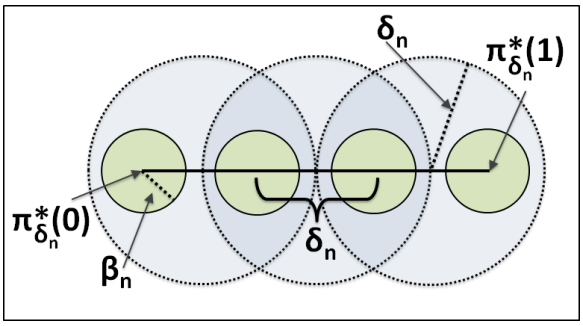

Figure 3.1: Hyperballs over an optimal path with radius β_n and separation δ_n. Consecutive balls lie entirely within some clearance ball $\mathcal{B}_{\delta_n}(\pi^*_{\delta_n}(\tau_t))$.

A bound on n_0 arises from the analysis, with an exact characterization provided by Equation 3.6.

The analysis considers a theoretical construction of hyperballs as illustrated in Figure 3.1. Consider $M_n + 1 = \lceil \frac{I^*_{\delta_n}}{\delta_n} \rceil + 1$ balls, B_n centered along $\pi^*_{\delta_n}$, i.e., $B_n = \{\mathcal{B}_{\beta_n}(\pi^*_{\delta_n}(\tau_0)), \ldots, \mathcal{B}_{\beta_n}(\pi^*_{\delta_n}(\tau_{M_n}))\}$, having radius $\beta_n \leq \frac{1}{2}\delta_n$, where $\delta_n = \frac{1}{2}r_n$, and where r_n is the connection radius used by this PRM* variant. The hyperball centers are δ_n distance apart from each other and are disjoint. Any pair of points that lie in consecutive hyperballs will necessarily be tested for connection by PRM*. This construction is similar to that provided in the original analysis of PRM*, when $\theta_1 = 2$ [69].

The early attempt at recovering these properties using PNO-PRM* used a fixed number of hyperballs of decreasing radius [27], whereas the results derived here consider the number of hyperballs, and the separation and radius of the hyperballs as functions over the number of drawn samples, n. By considering such a set of hyperballs, it is also the case that asymptotic optimality results can also be recovered by this approach. Consider for each hyperball, the set of samples S generated by PRM*, which lie in this hyperball. If $|S| \geq 1$ for all of the hyperballs, then there must exist at least one path through these hyperballs in the PRM* graph. The analysis considers the existence and length I_n of this path in order to provide PNO guarantees.

To ensure this path is constructed, the connection radius r_n must be derived. Furthermore, an equivalent connection number is derived for the equivalent $\mathtt{k}-\mathtt{PRM}^*$ algorithm. New algorithmic parameters, $\gamma_{\mathtt{PNO}}$ and $k_{\mathtt{PNO}}$, must be derived to ensure connections within the new hyperball construction are attempted. These Lemmas and derivations are stated throughout Section 3.2.

To aid the reader, the following summary of notation employed in this work is provided for reference.

- $I^*_{\delta_n}$: The length of the clearance robust optimal path answering some query at iteration n.
- δ_0: Desired clearance value for the clearance robust optimal path to cover.
- ϵ: A multiplicative bound with respect to optimal path length, $\epsilon \geq 0$.
- $\mathbb{P}_{success}$: The probability of the algorithm returning a path within ϵ of the optimal: $\mathbb{P}_{success} \in (0, 1)$.
- $M_n = \lceil \frac{I^*_{\delta_n}}{\delta_n} \rceil$: The number of segments which comprise the path returned by the algorithm.

3.2 Probabilistic Near-Optimality for Sampling-Based Planners

This section represents the bulk of the contribution for this chapter. At a high-level, it is broken down into three steps. First, appropriate connectivity constants are derived for the particular $\mathtt{PRM}^*$ variants studied in this chapter. Then, a precise bound on the probability of returning a covering path is provided in the following section. Lastly, by analyzing a problem in geometric probability, a bound on the length of a covering path is derived, and the results from the covering path probability are synthesized to create the final bound on path length returned by the method.

3.2.1 Deriving $\mathtt{PRM}^*$ Connectivity Constants

This section begins by examining the required connectivity constant of the given $\mathtt{PRM}^*$ variant, employing the same type of reasoning as the derivation for $\gamma_{\mathtt{PRM}^*}$ in the literature

[69]. The objective is to leverage a bound on the probability that PRM* will fail to produce a sample in each of the hyperballs over $\pi^*_{\delta_n}$ to derive an appropriate constant for the connection radius. The section continues to prove the following:

Lemma 1 (Connectivity constant $\gamma_{\texttt{PNO}}$)**.** *To ensure* PNO *properties of* PRM*, *it suffices to use* γ *parameter:*

$$\gamma_{\texttt{PNO}} > 4\left(\left(1+\frac{1}{d}\right)\left(\frac{|\mathcal{C}^{\text{free}}|}{V_d}\right)\right)^{\frac{1}{d}} = 2\cdot\gamma_{\texttt{PRM}^*}$$

Proof. Let the connection radius employed by the PRM* variant be $r_n = \gamma_{\texttt{PNO}}\left(\frac{\ln n}{n}\right)^{\frac{1}{d}}$. Then, by construction, this connection radius is at least four times larger than the radius of a hyperball, i.e., $\beta_n < \frac{1}{4}\gamma_{\texttt{PNO}}\left(\frac{\ln n}{n}\right)^{\frac{1}{d}}$. Then,

$$|\mathcal{B}_{\beta_n}| = V_d\beta_n^d < V_d\big(\frac{r_n}{4}\big)^d = V_d\cdot\frac{\ln n}{n}\big(\frac{\gamma_{\texttt{PNO}}}{4}\big)^d,$$

where $V_d = |\mathcal{B}_1|$ is the d-dimensional constant for the volume of a hyperball. Also by construction, $\delta_n \geq \frac{1}{2}\gamma_{\texttt{PNO}}\left(\frac{\ln n}{n}\right)^{\frac{1}{d}}$. Then, the number of path segments constructed over $\pi^*_{\delta_n}$ can be bounded by $M_n \leq \frac{I^*_{\delta_n}}{\delta_n} = \frac{2I^*_{\delta_n}}{\gamma_{\texttt{PNO}}}\left(\frac{n}{\ln n}\right)^{\frac{1}{d}}$.

Then, in line with previous work in the literature [71, 69], the probability of failure can be bounded using the probability that a single hyperball contains no sample. The event that a single hyperball does not contain a sample is denoted as $\mathcal{F}$, and has probability:

$$\mathbb{P}(\mathcal{F}) = \left(1-\frac{|\mathcal{B}_{\beta_n}|}{|\mathcal{C}^{\text{free}}|}\right)^n < \left(1-\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\frac{\ln n}{n}\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d\right)^n$$

Then, since $(1-x)^t \leq e^{-tx}$,

$$\mathbb{P}(\mathcal{F}) \leq e^{-\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\ln n\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d} = n^{-\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d} \tag{3.1}$$

Now, compute bounds on the event Ω^C that at least one ball does not contain a sample as:

$$\mathbb{P}(\Omega^C) = \mathbb{P}\big(\bigcup_{M_n}\mathcal{F}\big) \leq \sum_{i=1}^{M_n}\mathbb{P}(\mathcal{F}) = (M_n)\mathbb{P}(\mathcal{F})$$

Substituting the computed value for M_n, and $\mathbb{P}(\mathcal{F})$ from Eq. 3.1:

$$\mathbb{P}(\Omega^C) \;\leq\; \left(\frac{2I^*_{\delta_n}}{\gamma_{\texttt{PNO}}}\Big(\frac{n}{\ln n}\Big)^{\frac{1}{d}}\right)n^{-\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d} \;=\; \left(\frac{2I^*_{\delta_n}}{\gamma_{\texttt{PNO}}}\frac{1}{(\ln n)^{\frac{1}{d}}}\right)n^{-\left(\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d-\frac{1}{d}\right)}$$

Now, if $\sum_{i=1}^{\infty}\mathbb{P}(\Omega^C)$ is less than infinity, this implies by the Borel-Cantelli theorem that $\mathbb{P}(\limsup_{n\to\infty}\Omega^C)=0$ [49]. Furthermore, by the Zero-one Law, $\mathbb{P}(\limsup_{n\to\infty}\Omega^C)=0\Rightarrow\mathbb{P}(\limsup_{n\to\infty}\Omega^{\beta}_n)=1$, meaning the probability of coverage converges to 1 in the limit.

In order for the sum to be less than infinity, it is sufficient to show that the exponent, $\frac{V_d}{|\mathcal{C}^{\text{free}}|}\cdot\left(\frac{\gamma_{\texttt{PNO}}}{4}\right)^d-\frac{1}{d}<1$. The algorithm can ensure this by using an appropriate value of $\gamma_{\texttt{PNO}}$. Solving the inequality for $\gamma_{\texttt{PNO}}$ shows that it suffices that:

$$\gamma_{\texttt{PNO}} > 4\left(\Big(1+\frac{1}{d}\Big)\Big(\frac{|\mathcal{C}^{\text{free}}|}{V_d}\Big)\right)^{\frac{1}{d}}$$

□

Next, the connectivity constant for the k-nearest neighbor variant of the algorithm is provided. The high-level idea is that it will be shown that two events happen infinitely often with the given $k(n)$; the set of hyperballs each contain at least one sample, and that each ball of radius δ_n has no more than $k(n)$ samples inside it. From this, it is clear that if $\texttt{k}-\texttt{PRM}^*$ attempts to connect each sample with $k(n)$ neighbors, it will attempt connections between samples in neighboring hyperballs. Then, this section continues on to show the following Lemma:

Lemma 2 (Connectivity constant $k_{\texttt{PNO}}$)**.** *To ensure* PNO *properties of* $\texttt{k}-\texttt{PRM}^*$*, it suffices to use:*

$$k_{\texttt{PNO}} = 2^d$$

Proof. By using the computed value of $\gamma_{\texttt{PNO}}$ from above,

$$M_n \leq \frac{1}{2}I^*_{\delta_n}\left(\Big(\frac{V_d}{|\mathcal{C}^{\text{free}}|}\Big)\Big(\frac{n}{\ln n}\Big)\Big(\frac{1}{1+\frac{1}{d}}\Big)\right)^{\frac{1}{d}},\text{ and}$$

$$|\mathcal{B}(\delta_n)| \leq \Big(V_d\cdot 4^d\Big)\Big(\frac{1}{2}\Big)^d\Big(1+\frac{1}{d}\Big)\Big(\frac{|\mathcal{C}^{\text{free}}|\ln n}{nV_d}\Big) = 2^d\Big(1+\frac{1}{d}\Big)\Big(\frac{|\mathcal{C}^{\text{free}}|\ln n}{n}\Big)$$

Let A be an indicator random variable that takes value 1 when there is a sample in some arbitrary hyperball of radius δ_n. Then, $\mathbb{E}[A] = \frac{|\mathcal{B}(\delta_n)|}{|\mathcal{C}^{\text{free}}|} = 2^d(1+\frac{1}{d})(\frac{\ln n}{n})$. Since each sample is drawn independently of the others, the number of samples in a ball can be expressed as a random variable N, such that $\mathbb{E}[N] = n\mathbb{E}[A] = 2^d(1+\frac{1}{d})\ln n$. Due to A being a Bernoulli random variable, the Chernoff Bound can be employed to bound the probability of N taking large values, namely:

$$\mathbb{P}(N > (1+t)\mathbb{E}[N]) \leq \left(\frac{e^t}{(1+t)^{(1+t)}}\right)^{\mathbb{E}[N]}, t > 0$$

Then, let $t = e - 1$. Substituting this above yields:

$$\mathbb{P}(N > e\mathbb{E}[N]) \leq e^{-\mathbb{E}[N]} = e^{-2^d(1+\frac{1}{d})\ln n} = n^{-2^d(1+\frac{1}{d})}$$

Now, in order for the $k(n)$ connections to attempt connections outside of a δ_n-ball, it must be that:

$$k(n) = k_{\texttt{PNO}}e\Big(1+\frac{1}{d}\Big) \geq 2^d e\Big(1+\frac{1}{d}\Big) = e\mathbb{E}[N],$$

which clearly holds if $k_{\texttt{PNO}} = 2^d$. This implies that $\mathbb{P}(N > k(n)) \leq n^{-2^d(1+\frac{1}{d})}$.

Finally, consider the event ζ that even one of the balls has more than $k(n)$ samples:

$$\mathbb{P}(\zeta) = \mathbb{P}\Big(\bigcup_{M_n}\mathbb{P}(N > k(n))\Big) \leq \sum_{n}^{M}\mathbb{P}(N > k(n)) = M_n\mathbb{P}(N > k(n))$$

$$\mathbb{P}(\zeta) \leq \frac{I^*_{\delta_n}}{2}\left(\frac{V_d}{\ln n|\mathcal{C}^{\text{free}}|(1+\frac{1}{d})}\right)^{\frac{1}{d}} n^{-2^d(1+\frac{1}{d})+\frac{1}{d}}$$

Then, it is clear that $\sum_{i=1}^{\infty}\mathbb{P}(\zeta) < \infty$, which by the Borel-Cantelli Theorem implies that $\mathbb{P}(\limsup_{n\to\infty}\zeta) = 0$, and furthermore, $\mathbb{P}(\limsup_{n\to\infty}\zeta^C) = 1$ via the Zero-one Law, i.e., the number of samples in the δ_n-ball is almost certainly less than $k(n)$.

Finally, using the result showing the convergence of $\mathbb{P}(\Omega_n^\beta)$ to 1, and the above result for $\mathbb{P}(\zeta^C)$, it can be concluded that $\mathbb{P}(\limsup_{n\to\infty}(\Omega_n^\beta \cap \zeta^C)) = 1$, implying that for the choice of $k(n)$, $\texttt{k}-\texttt{PRM}^*$ attempts the appropriate connections. □

3.2.2 Probability of Path Coverage

Foundational work on roadmap-based approaches sought to characterize how quickly these methods return valid solutions [71]. An exponential bound on this probability was derived as:

$$\mathbb{P}(\Omega_n^\beta) \geq 1 - \frac{2I^*_{\delta_n}}{\delta_n} e^{-\frac{\pi \cdot \delta_n^2 \cdot n}{4|\mathcal{C}^{\text{free}}|}} \tag{3.2}$$

Here, $\mathbb{P}(\Omega_n^\beta)$ represents the probability that a path has been generated in a set of hyperballs centered around a clearance-robust optimal path. This bound is sufficient to show that these methods converge exponentially quickly to returning such a solution. This bound is somewhat loose, so a tighter bound was derived for the PNO-PRM* approach as a first attempt [27], while this work extends such an analysis to PRM*. As an example, for a two-dimensional configuration space with $|\mathcal{C}^{\text{free}}| = 100$, $I^*_{\delta_n} = 10$, $\delta_n = 1$, and for $\mathbb{P}(\Omega_n^\beta) = 0.99$, the derived bound accurately reports 97 samples are required as opposed to nearly 980 with Equation 3.2. This bound follows from prior work in the literature [71, 69, 27]. The derivation of γ_{PNO} also gives the value for the hyperballs' radii, $\beta_n = \left(\left(1 + \frac{1}{d}\right)\left(\frac{|\mathcal{C}^{\text{free}}|}{V_d}\right)\left(\frac{\ln n}{n}\right)\right)^{\frac{1}{d}}$, as well as the number of covering hyperballs, $M_n + 1 = \lceil \frac{I^*_{\delta_n}}{\delta_n} \rceil + 1$. Substituting these values into the existing coverage probability results from related work yields:

$$\mathbb{P}(\Omega_n^\beta) \approx \left(1 - \left(1 - a\right)^n\right)^{\frac{1}{2} I^*_{\delta_n} \left(b\right)^{-\frac{1}{d}} + 1}$$

where $a = \frac{V_d \left(\sqrt[d]{\left(1+\frac{1}{d}\right)\left(\frac{|\mathcal{C}^{\text{free}}| \ln n}{nV_d}\right)} \right)^d}{|\mathcal{C}^{\text{free}}|}$, $b = \left(1 + \frac{1}{d}\right)\left(\frac{|\mathcal{C}^{\text{free}}| \ln n}{nV_d}\right)$, and V_d is the d-dimensional constant for the volume of a hyperball, i.e., $|\mathcal{B}(r)| = V_d r^d$. The value of a arises from the relative size of the hyperballs in the free space, while b is related to M_n. Simplifying this expression yields the following Lemma:

Lemma 3 (Probability of Path Coverage)**.** *Let Ω_n^β be the event that for one execution of* PRM* *there exists at least one sample in each of the $M_n + 1$ hyperballs of radius β_n*

over the clearance robust optimal path, $\pi^*_{\delta_n}$, for a specific value of $n > n_0$ and β_n. Then,

$$\mathbb{P}(\Omega_n^\beta) \approx \left(1 - \left(1 - a\right)^n\right)^{\frac{1}{2} I^*_{\delta_n} \left(b\right)^{-\frac{1}{d}} + 1} \tag{3.3}$$

Where $a = \left(1 + \frac{1}{d}\right)\left(\frac{\log n}{n}\right)$ *and* $b = \frac{|\mathcal{C}^{\text{free}}|}{V_d} a$.

3.2.3 Bounding Path Quality

This section employs the result from Section 3.2.2 to determine path quality bounds for PRM*. The high-level approach to determining this bound is done in four steps. First, Chebyshev's Inequality is leveraged to express this bound in terms of the mean and variance of I_n, the length of PRM*'s returned path. Then, the expected value and variance of I_n are derived. Finally, the results are combined to give the final bound.

A bound in terms of mean and variance

This section outlines the high-level idea behind applying the Chebyshev inequality to create the probabilistic bound on I_n. Let Ω^C be the event that there does not exist a sample in each of the hyperballs covering a path, i.e., $\mathbb{P}(\Omega^C) = 1 - \mathbb{P}(\Omega_n^\beta)$. Then, the value for $\mathbb{P}(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n})$ can be expressed as:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big)\mathbb{P}(\Omega_n^\beta) + \mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega^C\big)\mathbb{P}(\Omega^C)$$

It is assumed that the probability of a path being larger than δ is quite high if Ω_n^β has not happened, i.e., $\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega^C\big)$ is close to 1; therefore, this probability can be upper bounded by 1.

Let y be a random variable identically distributed with I_n, but having 0 mean, i.e., $y = I_n - \mathbb{E}[I_n]$. Then,

$$\mathbb{P}(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n}) = \mathbb{P}\big(\mathbb{E}[I_n] + y - I^*_{\delta_n} > \epsilon I^*_{\delta_n} \mid \Omega_n^\beta\big) + \mathbb{P}\big(\mathbb{E}[I_n] + y - I^*_{\delta_n} < -\epsilon I^*_{\delta_n} \mid \Omega_n^\beta\big),$$

Let Ω^C be the event that there does not exist a sample in each of the hyperballs covering a path, i.e., $\mathbb{P}(\Omega^C) = 1 - \mathbb{P}(\Omega_n^\beta)$. Then, the value for $\mathbb{P}(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n})$ can be expressed as:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big)\mathbb{P}(\Omega_n^\beta) + \mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega^C\big)\mathbb{P}(\Omega^C)$$

This is because the probability of returning a low quality path is expressed as a sum of probabilities, when event Ω_n^β has occurred, and when Ω_n^β has not occurred. Since $\mathbb{P}(\Omega^C) = 1 - \mathbb{P}(\Omega_n^\beta)$, then via Lemma 3, both $\mathbb{P}(\Omega^C)$ and $\mathbb{P}(\Omega_n^\beta)$ are known for known n and β. It is assumed that the probability of a path being larger than ϵ is quite high if Ω_n^β has not happened, i.e., $\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega^C\big)$ is close to 1; therefore, this probability can be upper bounded by 1. All that remains is to compute $\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big)$. Let y be a random variable identically distributed with I_n, but having 0 mean, i.e., $y = I_n - \mathbb{E}[I_n]$. Then, let

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big) = \mathbb{P}\big(|\mathbb{E}[I_n] + y - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big)$$

Then, the absolute value can be removed, as $|X| > a \Rightarrow X > a$ or $X < -a$. Then, the probability is equal to the sum:

$$\mathbb{P}\big(\mathbb{E}[I_n] + y - I^*_{\delta_n} > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big) + \mathbb{P}\big(\mathbb{E}[I_n] + y - I^*_{\delta_n} < -\epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big),$$

where due to symmetry,

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big) = 2\mathbb{P}\big(\mathbb{E}[I_n] + y - I^*_{\delta_n} > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big)$$

Rearranging the terms inside the probability yields:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big) = 2\mathbb{P}\big(y > (\epsilon + 1)I^*_{\delta_n} - \mathbb{E}[I_n] \mid \Omega_n^\beta\big)$$

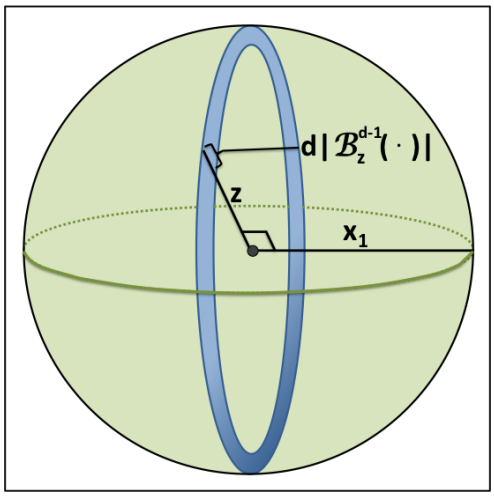

Figure 3.2: The differential over a lower-dimensional hyperball, illustrated for $d = 3$.

This probability will be bounded with Chebyshev's Inequality, which states:

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq a) \leq \frac{Var(X)}{a^2}$$

In order to employ this inequality, both $\mathbb{E}[I_n]$ and $Var(I_n)$ for the length of a path in the PRM* planning structure, I_n are needed.

Approximation of $\mathbb{E}[I_n]$ in $\mathbb{R}^d$

As required for the bound, this section derives a tight approximation of $\mathbb{E}[I_n]$. Let, $\mathbb{E}[I_n] = \sum_{m=1}^{M} \mathbb{E}[I_m]$, where I_m is the length of a single segment between hyperballs. Then, because all I_m are I.I.D., $\mathbb{E}[I_n] = M\mathbb{E}[I_1]$. Then, $\mathbb{E}[I_1]$ is approximated via two integrations over the endpoints of the segment, as outlined in Figure 3.3.

Let, $\mathbb{E}[I_n] = \sum_{m=1}^{M} \mathbb{E}[I_m]$, where I_m is the length of a single segment between two random samples in consecutive disjoint balls. Then, assuming all I_m are I.I.D., $\mathbb{E}[I_n] = M\mathbb{E}[I_1]$. Then, to compute $\mathbb{E}[I_n]$, $\mathbb{E}[I_1]$ is computed. This requires the solution to a unique ball-line picking problem variant. Computing this value requires integration over the possible locations of the endpoints of the segment, as illustrated in Figure 3.3.

The integration is broken into two steps, and the first integral will be for the situation depicted in Figure 3.3 (left). The objective is to get an expected value for the distance between points x and x'. Here, x represents a random point within the first hyperball,

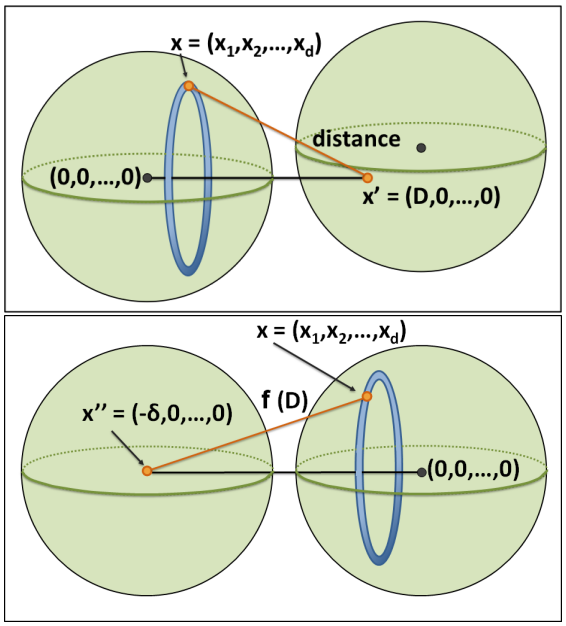

Figure 3.3: Illustrations in 3D of the mean calculation. (top) The first set of integrals is performed over the left hyperball, averaging the distance between points $(x_1, x_2, \ldots, x_d)$ and $(D, 0, \ldots, 0)$. (bottom) Using the result from the first set of integrals, a second set of integrals is performed over the second hyperball, yielding the expected value.

while x' is some fixed point within the second hyperball, which has distance D from the center of the first hyperball. Without loss of generality, x' can be displaced along only the first coordinate, x_1. To get an expected value, this distance is integrated over all points within the first hyperball, and then divided by the volume of the d-dimensional hyperball. In this work, the volume of a d-dimensional hypersphere of radius β_n is denoted $|\mathcal{B}_{\beta_n}| = V_d \beta_n^d$, where V_d is a constant dependent on the dimension of the space. Taking the distance between $x = (x_1, x_2, \ldots, x_d)$ and $x' = (D, 0, \ldots, 0)$ to be $\sqrt{(x_1 - D)^2 + x_2^2 + \ldots + x_d^2}$ produces the following integral:

$$A = \frac{1}{V_d \beta_n^d} \int \cdots \int_{x_1^2 + \ldots + x_d^2 \leq \beta_n^2} \sqrt{(x_1 - D)^2 + x_2^2 + \ldots + x_d^2} \quad dx_1 \ldots dx_d,$$

This integral will be converted from a d-dimensional integral into a double integral using substitution. First, let $z^2 = x_2^2 + \ldots + x_d^2$. This allows performing the integral over only two variables, x_1 and z; however, the form of the integral changes, as the differential is adapted as illustrated in Figure 3.2. This differential, $d|\mathcal{B}_{d-1}(z)|$, is taken over a lower dimensional hypersphere, of dimension $d - 1$, as z is taking the place of

$d-1$ coordinates. Then:

$$A = \frac{1}{V_d \beta_n^d} \iint_{x_1^2+z^2 \leq \beta_n^2} \sqrt{(x_1 - D)^2 + z^2} \quad d|\mathcal{B}_z^{d-1}(\cdot)| dx_1 dz,$$

where $d|\mathcal{B}_{d-1}(z)| = \frac{d}{dz} V_{d-1} z^{d-1}$. Taking this derivative, $\frac{d}{dz} V_{d-1} z^{d-1} = (d-1) V_{d-1} z^{d-2}$, and substituting into A yields:

$$A = \frac{(d-1)V_{d-1}}{V_d \beta_n^d} \iint_{x_1^2+z^2 \leq \beta_n^2} z^{d-2} \sqrt{x_1^2 + D^2 - 2Dx_1 + z^2} \quad dx_1 dz$$

The integral can be represented in terms of polar coordinates, where $x_1 = r\cos\theta$, $z = r\sin\theta$, and $dx_1\ dz = r\ d\theta\ dr$. This gives

$$A = \frac{(d-1)V_{d-1}}{V_d \beta_n^d} \int_0^{\beta_n} r \int_0^{\pi} (r\sin\theta)^{d-2} \sqrt{r^2 + D^2 - 2Dr\cos\theta} \quad d\theta dr$$

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta_n^d} \int_0^{\beta_n} r \int_0^{\pi} (r\sin\theta)^{d-2} \sqrt{1 + \big(\frac{r}{D}\big)^2 - 2\big(\frac{r}{D}\big)\cos\theta} \quad d\theta dr$$

A second-order Taylor Approximation for the square root is taken. Let $f(u) = \sqrt{1+u}$, where $u = \big(\frac{r}{D}\big)^2 - 2\big(\frac{r}{D}\big)\cos\theta$. The approximation will be taken about the point $u = 0$. This is reasonable given that overall, β_n is considered to be smaller than the separation between consecutive hyperballs, δ_n. Take the second-order Taylor Approximation as:

$$f(u) \approx f(0) + f'(0) \cdot u + \frac{1}{2!} f''(0) \cdot u^2.$$

Taking a derivative of f yields $f'(u) = \frac{1}{2}(1+u)^{-\frac{1}{2}}$ and $f''(u) = -\frac{1}{4}(1+u)^{-\frac{3}{2}}$. Then,

$$f(u) \approx \sqrt{1} + \frac{1}{2}(1)^{-\frac{1}{2}} \cdot u - \frac{1}{2} \cdot \frac{1}{4}(1)^{-\frac{3}{2}} \cdot u^2 = 1 + \frac{1}{2}u - \frac{1}{8}u^2$$

substituting $u = \big(\frac{r}{D}\big)^2 - 2\big(\frac{r}{D}\big)\cos\theta$,

$$f(u) \approx 1 + \frac{1}{2}\big(\big(\frac{r}{D}\big)^2 - 2\big(\frac{r}{D}\big)\cos\theta\big) - \frac{1}{8}\big(\big(\frac{r}{D}\big)^2 - 2\big(\frac{r}{D}\big)\cos\theta\big)^2$$

$$f(u) \approx 1 + \frac{1}{2}\left(\left(\frac{r}{D}\right)^2 - 2\left(\frac{r}{D}\right)\cos\theta\right) - \frac{1}{8}\left(\left(\frac{r}{D}\right)^4 - 4\left(\frac{r}{D}\right)^3\cos\theta + 4\left(\frac{r}{D}\right)^2\cos^2\theta\right)$$

Then, as this is a second-order approximation, the third- and fourth-order terms are considered negligible, and thus, the approximation results in:

$$f(u) \approx 1 + \frac{1}{2}\left(\left(\frac{r}{D}\right)^2 - 2\left(\frac{r}{D}\right)\cos\theta\right) - \frac{1}{8}\left(4\left(\frac{r}{D}\right)^2\cos^2\theta\right)$$

Substituting the result:

$$A = \frac{D(d-1)V_{d-1}}{V_d\beta_n^d}\int_0^{\beta_n} r\int_0^{\pi}(r\sin\theta)^{d-2} \left(1 + \frac{1}{2}\left(\left(\frac{r}{D}\right)^2 + 2\left(\frac{r}{D}\right)\cos\theta\right) - \frac{1}{8}\left(4\left(\frac{r}{D}\right)^2\cos^2\theta\right)\right)\, d\theta dr$$

Simplifying this integral requires the following Lemmas:

Lemma 4 (Value of $\int_0^{\pi}(\sin\theta)^d d\theta$)**.** *In terms of the hyperball volume constant, V_d,*

$$\int_0^{\pi}(\sin\theta)^{d-2}d\theta = S_{d-2} = \frac{dV_d}{(d-1)V_{d-1}}$$

Proof. For simplicity, let $\int_0^{\pi} sin^d(\theta)\ d\theta$ be denoted as S_d. Then:

$$|\mathcal{B}_\beta| = V_d\beta^d,$$

where $V_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}$ is a constant dependent on the dimension, d. Then, the volume can be computed as an integral of the following form:

$$V_d\beta^d = \iint_{x_1^2+\rho^2\leq\beta^2} dx_1\ d(V_{d-1}\rho^{d-1}),$$

where the second differential is over a sphere of radius ρ of dimension $d-1$. Then,

$$V_d\beta^d = \iint_{x_1^2+\rho^2\leq\beta^2}(d-1)V_{d-1}\rho^{d-2}\ dx_1\ d\rho$$

Now, to simplify this integral, it will be converted to polar coordinates, using $x_1 = r\ \cos\theta$, $\rho = r\ \sin\theta$, and $dx_1\ d\rho = r\ d\theta\ dr$. Substituting these values yields:

$$V_d\beta^d = (d-1)V_{d-1}\int_0^\beta \int_0^\pi r^{d-2}(\sin\theta)^{d-2} r \; d\theta \; dr$$

$$V_d\beta^d = (d-1)V_{d-1}\int_0^\beta r^{d-1} S_{d-2} \; d\theta \; dr$$

$$V_d\beta^d = (d-1)V_{d-1}S_{d-2}\frac{\beta^d}{d} \; d\theta \; dr$$

$$S_{d-2} = \frac{dV_d}{(d-1)V_{d-1}}$$

□

Lemma 5 (Recurrence relation of $\int_0^\pi (\sin\theta)^d d\theta$)**.** *For $S_d = \int_0^\pi (\sin\theta)^d d\theta$, the following recurrence relation holds:*

$$\int_0^\pi (\sin\theta)^d d\theta = S_d = \frac{d-1}{d} S_{d-2}$$

Proof.

To determine this recurrence, the following expression will be solved for x:

$$S_d = xS_{d-2}$$

Substitute the result from Lemma 4, getting:

$$x = \frac{(d+2)V_{d+2}}{(d+1)V_{d+1}} \cdot \frac{(d-1)V_{d-1}}{dV_d}.$$

Then, substituting the value of V_d yields:

$$x = \frac{(d-1)(d+2)}{d(d+1)}\Big(\frac{\Gamma(\frac{d+2}{2})}{\Gamma(\frac{d+4}{2})}\Big)\Big(\frac{\Gamma(\frac{d+3}{2})}{\Gamma(\frac{d+1}{2})}\Big)$$

$$x = \frac{(d-1)(d+2)}{d(d+1)}\Big(\frac{2}{d+2}\Big)\Big(\frac{d+1}{2}\Big)$$

$$x = \frac{d-1}{d}$$

□

Applying these Lemmas:

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta^d} \int_0^{\beta_n} r \left(\int_0^{\pi} (r \sin\theta)^{d-2} \Big(1 + \frac{1}{2}\big(\frac{r}{D}\big)^2\Big) d\theta \right.$$
$$\left. + \int_0^{\pi} (r \sin\theta)^{d-2} (\cos\theta) d\theta - \frac{1}{2}\big(\frac{r}{D}\big)^2 \int_0^{\pi} (sin\theta)^{d-2} cos^2\theta \; d\theta \right) dr$$

The second integral over θ will integrate to 0, due to the presence of cosine, while the other terms leverage Lemmas 4 and 5:

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta_n^d} \int_0^{\beta_n} r^{d-1} \left(S_{d-2}\Big(1 + \frac{1}{2}\big(\frac{r}{D}\big)^2\Big) - \frac{1}{2}\big(\frac{r}{D}\big)^2 \Big(S_{d-2} - S_d\Big) \right) dr$$

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta_n^d} S_{d-2} \cdot \int_0^{\beta_n} r^{d-1} \left(\Big(1 + \frac{d-1}{2d}\big(\frac{r}{D}\big)^2\Big) \right) dr$$

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta_n^d} S_{d-2} \cdot \int_0^{\beta_n} \left(r^{d-1} + \frac{d-1}{2d}\Big(\frac{r^{d+1}}{D^2}\Big) \right) dr$$

$$A = \frac{D(d-1)V_{d-1}}{V_d \beta_n^d} \frac{dV_d}{(d-1)V_{d-1}} \cdot \left(\frac{\beta_n^d}{d} + \frac{d-1}{2d}\Big(\frac{\beta_n^{d+2}}{(d+2)D^2}\Big) \right) = D + \frac{(d-1)\beta_n^2}{2(d+2)D}$$

This is only an intermediate result, however, and it must be integrated over once again to consider all possible placements of the point x' in the second hyperball, as illustrated in Figure 3.3(right). In order to do so, write D in terms of ϵ by taking the distance between $x = (x_1, x_2, \ldots, x_d)$ and $x'' = (-\delta_n, 0, \ldots, 0)$. Then, $D = \sqrt{(x_1+\epsilon)^2 + x_2^2 + \ldots + x_d^2}$, and $\mathbb{E}[I_1]$ is computed as:

$$\mathbb{E}[I_1] \quad = \quad \frac{1}{V_d \beta_n^d} \int \cdots \int_{x_1^2+\ldots+x_d^2 \leq \beta_n^2} D \quad + \quad \frac{(d-1)\beta_n^2}{2(d+2)D} \quad dx_1 \ldots dx_d,$$

Steps similar to what was just taken to derive the intermediate result are used to compute this integral. As a matter of simplicity, note that the second term inside the integral is already a second-order term, which means taking the integral will result in higher-order terms. Since $D = \sqrt{(x_1+\delta_n)^2 + x_2^2 + \ldots + x_d^2}$, the second term will take only the constant term of the Taylor Approximation for D. Then, taking $z^2 = x_2^2 + \ldots + x_d^2$:

$$\mathbb{E}[I_m] = \frac{1}{V_d \beta_n^d} \int \cdots \int_{x_1^2+\ldots+x_d^2 \leq \beta_n^2} \sqrt{(x_1+\delta_n)^2 + x_2^2 + \ldots + x_d^2} + \frac{(d-1)\beta_n^2}{2(d+2)\sqrt{(x_1+\delta_n)^2 + x_2^2 + \ldots + x_d^2}} \, dx_1 \ldots dx_d$$

$$\mathbb{E}[I_m] \; = \; \frac{1}{V_d \beta_n^d} \iint_{x_1^2+z^2 \leq \beta_n^2} \sqrt{(x_1+\delta_n)^2 + z^2} \; + \; \frac{(d-1)\beta_n^2}{2(d+2)\sqrt{(x_1+\delta_n)^2 + z^2}} \; dx_1 \; dz$$

Again, perform a polar coordinate transformation so as to take the integral:

$$\mathbb{E}[I_m] = \frac{1}{V_d \beta_n^d} \int_0^{\beta_n} r \int_0^{\pi} \Big(\sqrt{r^2 + \delta_n^2 - 2\delta_n r \cos\theta} + \frac{(d-1)\beta_n^2}{2(d+2)\sqrt{r^2 + \delta_n^2 - 2\delta_n r \cos\theta}} \Big) d|\mathcal{B}_{r\sin\theta}^{d-1}(\cdot)| \; d\theta \; dr$$

$$\mathbb{E}[I_m] = \frac{(d-1)V_{d-1}}{V_d \beta_n^d} \int_0^{\beta_n} r \int_0^{\pi} (r \; \sin\theta)^{d-2} \Big(\sqrt{r^2 + \delta_n^2 - 2\delta_n r \cos\theta} + \frac{(d-1)\beta_n^2}{2(d+2)\sqrt{r^2 + \delta_n^2 - 2\delta_n r \cos\theta}} \Big) \; d\theta \; dr$$

Now, to compute this integral, a Taylor Approximation will be taken. Again, recall that because the second term in the integral is second order, a 0^{th}-order approximation is taken for that term:

$$\mathbb{E}[I_m] \approx \frac{(d-1)V_{d-1}}{V_d\beta_n^d}\int_0^{\beta_n} r\int_0^{\pi}(r\ \sin\theta)^{d-2}$$
$$\Big(\delta_n\Big(1+\frac{1}{2}\big(\big(\frac{r}{\delta_n}\big)^2+2\big(\frac{r}{\delta_n}\big)\cos\theta\big)-\frac{1}{8}\big(4\big(\frac{r}{\delta_n}\big)^2\cos^2\theta\big)\Big)+\frac{(d-1)\beta_n^2}{2(d+2)\delta_n}\Big)\ d\theta\ dr$$

Then, performing steps similar to above, rewrite in terms of S_{d-2}, as well as splitting the last term into a separate integral:

$$\mathbb{E}[I_m] \approx \frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}\int_0^{\beta_n} r^{d-1}\int_0^{\pi}\Big((\sin\theta)^{d-2}$$
$$+\frac{r^2}{2\delta_n^2}(\sin\theta)^d\Big)\ d\theta\ dr+\frac{(d-1)V_{d-1}}{V_d\beta_n^d}\cdot\int_0^{\beta_n} r^{d-1}\int_0^{\pi}(\sin\theta)^{d-2}\frac{(d-1)\beta_n^2}{2(d+2)\delta_n}\ d\theta\ dr$$

$$\mathbb{E}[I_m] \approx \frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}S_{d-2}\int_0^{\beta_n} r^{d-1}\Big(1$$
$$+\frac{(d-1)r^2}{2d\delta_n^2}\Big)\ dr+\frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}S_{d-2}\int_0^{\beta_n} r^{d-1}\frac{(d-1)\beta_n^2}{2(d+2)\delta_n^2}\ dr$$

$$\mathbb{E}[I_m] \approx \frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}S_{d-2}\Big(\frac{\beta_n^d}{d}+\frac{(d-1)\beta_n^{d+2}}{2d(d+2)\delta_n^2}\Big)$$
$$+\frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}S_{d-2}\Big(\frac{(d-1)\beta_n^{d+2}}{2d(d+2)\delta_n^2}\Big)$$

$$\mathbb{E}[I_m] \approx \frac{\delta_n(d-1)V_{d-1}}{V_d\beta_n^d}\frac{dV_{d-1}}{(d-1)V_{d-1}}\cdot\Big(\frac{\beta_n^d}{d}+\frac{(d-1)\beta_n^{d+2}}{2d(d+2)\delta_n^2}+\frac{(d-1)\beta_n^{d+2}}{2d(d+2)\delta_n^2}\Big)$$

$$\mathbb{E}[I_m] \approx \delta_n+\frac{(d-1)\beta_n^2}{(d+2)\delta_n}$$

Now, using this result for the expected value of a single segment, I_m, the expected value of the entire path consisting of M such segments is:

$$\mathbb{E}[I_n] \approx M\Big(\delta_n+\frac{(d-1)\beta_n^2}{(d+2)\delta_n}\Big)$$

Euclidean	$\lambda_n = 0.5$	$\lambda_n = 0.125$
dimension	% error	% error
2	0.1730%	0.0050%
3	0.0473%	0.0205%
10	0.9413%	0.0128%
100	1.9147%	0.0129%

Figure 3.4: Simulation comparison for $\mathbb{E}[I_n]$, using 120,000 data points for each entry, for differing $\lambda_n = \frac{\beta_n}{\delta_n}$, where the error is $100 \cdot \frac{|\mathbb{E}[I_n] - I_n|}{I_n}$.

This result must be integrated over the second hypersphere to attain the value of $\mathbb{E}[I_n]$. The steps taken will be very similar to the steps taken to reach the intermediate result, as the form of the integral is not significantly different. After performing simplification, the following Lemma arises:

Lemma 6 (Expected value of I_n)**.** *The path built over the set of $M_n + 1$ hyperballs having radius β_n has expected length:*

$$\mathbb{E}[I_n] \approx M_n\Big(\delta_n + \frac{(d-1)\beta_n^2}{(d+2)\delta_n}\Big)$$

To verify the approximation, Monte Carlo experiments were employed. The relative error of the approximation to the simulated values are shown in Figure 3.4.

Computation of the Variance of I_n in $\mathbb{R}^d$

At a high-level, the derivation goes as follows: to compute the $Var(I_n)$, leverage the definition of the variance of a random variable, i.e., $Var(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$:

$$Var\big(\sum_{m=1}^{M} I_m\big) \;=\; \mathbb{E}[\sum_{m=1}^{M} I_m^2] \;-\; \big(\mathbb{E}[\sum_{m=1}^{M} I_m]\big)^2 \;=\; \sum_{m=1}^{M}\sum_{k=1}^{M} \mathbb{E}[I_m I_k] \;-\; \big(\mathbb{E}[\sum_{m=1}^{M} I_m]\big)^2$$

The second term can be simplified due to the linearity of expectation:

$$Var\big(\mathbb{E}[I_m I_k] - M^2\big(\mathbb{E}[I_m]\big)^2$$

Many of the terms of the double sum are independent, and only the variance terms

of each segment and the covariance between adjacent segments contribute to the sum. This allows this variance to be simplified as:

$$Var\Big(\sum_{m=1}^{M} I_m\Big) \quad = \quad M\mathbb{E}[I_1^2] \;+\; (2M \;-\; 2)\mathbb{E}[I_1 I_2] \;+\; (2 \;-\; 3M)\big(\mathbb{E}[I_1]\big)^2$$

Both $\mathbb{E}[I_1^2]$ and $\mathbb{E}[I_1 I_2]$ are unknown. Therefore, this section contintues by deriving these expressions.

Lemma 7 (Expected value of I_1^2)**.** *For two consecutive hyperballs, the expected squared distance between random points in those balls is*

$$\mathbb{E}[I_1^2] = \delta_n^2 + \frac{2d}{d+2}\beta_n^2$$

Proof. The derivation of $\mathbb{E}[I_1^2]$ follows the same general steps as the computation of $\mathbb{E}[I_1]$; however, the form of the integral is simpler in this case. The integral over one of the two balls is of the form:

$$A \quad = \quad \frac{1}{V_d {\beta_n}^d} \int \cdots \int_{x_1^2+\ldots+x_d^2 \le {\beta_n}^2} \Big(\sqrt{(D-x_1)^2 + x_2^2 + \ldots + x_d^2}\Big)^2 dx_1 \ldots dx_d,$$

A represents an intermediate result. Then, take $z^2 = x_2^2 + \ldots + x_d^2$. Substituting these into the above integral yields:

$$\frac{1}{V_d {\beta_n}^d} \iint_{x_1^2+z^2 \le {\beta_n}^2} \big((D-x_1)^2 + z^2 \; d|\mathcal{B}_z^{d-1}(\cdot)|\big) \; dx_1 dz$$

$$= \quad \frac{1}{V_d {\beta_n}^d} \iint_{x_1^2+z^2 \le {\beta_n}^2} V_{d-1}(d \;-\; 1) z^{d-2}\big((D \;-\; x_1)^2 \;+\; z^2\big) \quad dx_1 dz$$

Then, to compute this integral, perform the integration over polar coordinates:

$$\frac{V_{d-1}(d-1)}{V_d {\beta_n}^d} \int_0^{\beta_n} r \int_0^{\pi} (r \quad \sin\theta)^{d-2}\big((D \;-\; (r \quad \cos\theta))^2 \;+\; (r \quad \sin\theta)^2\big) \quad d\theta \quad dr$$

$$= \quad \frac{V_{d-1}(d-1)}{V_d {\beta_n}^d} \int_0^{\beta_n} r^{d-1} \int_0^{\pi} (\sin\theta)^{d-2} \big(D^2 \ + \ r^2 \ - \ 2Dr\cos\theta\big) \quad d\theta \quad dr$$

Then, applying Lemmas 4 and 5:

$$\frac{V_{d-1}(d-1)}{V_d {\beta_n}^d} \int_0^{\beta_n} r^{d-1} \big(D^2 + r^2\big) S_{d-2} \ d\theta \ dr$$

$$= \frac{V_{d-1}(d-1)S_{d-2}}{V_d {\beta_n}^d} \big(D^2 \frac{{\beta_n}^d}{d} + \frac{d{\beta_n}^{d+2}}{d(d+2)}\big) \ d\theta \ dr$$

$$A = D^2 + \frac{d}{d+2}{\beta_n}^2$$

Now, this intermediate result is used to compute the final value of $\mathbb{E}[I_1^2]$. Begin again by integrating over this value:

$$\mathbb{E}[I_1^2] \qquad = \qquad \frac{1}{V_d {\beta_n}^d} \int \cdots \int_{x_1^2 + \ldots + x_d^2 \le {\beta_n}^2} D^2 \quad + \quad \frac{d}{d+2}{\beta_n}^2 dx_1 \ldots dx_d,$$

where now D is written in terms of δ_n as $D = \sqrt{(x_1+\delta_n)^2 + x_2^2 + \ldots + x_d^2}$. Again, take $z^2 = x_2^2 + \ldots + x_d^2$ and substitute in to get:

$$\frac{1}{V_d {\beta_n}^d} \iint_{x_1^2 + z^2 \le {\beta_n}^2} \big(((x_1 \ + \ \delta_n)^2 \ + \ z^2) \ + \ \frac{d}{d+2}{\beta_n}^2\big) d|\mathcal{B}_z^{d-1}(\cdot)| \quad dx_1 \quad dz$$

$$= \quad \frac{(d-1)V_{d-1}}{V_d {\beta_n}^d} \iint_{x_1^2 + z^2 \le {\beta_n}^2} \big((r^2 \ + \ \delta_n^2 \ + \ 2x_1\delta_n) \ + \ \frac{d}{d+2}{\beta_n}^2\big) z^{d-2} \quad dx_1 \quad dz$$

Rewrite in polar coordinates and splitting the integral:

$$\frac{(d-1)V_{d-1}}{V_d {\beta_n}^d} \Bigg(\int_0^{\beta_n} r \int_0^{\pi} r^d (\sin\theta)^{d-2} + (r\sin\theta)^{d-2}\delta_n^2 + 2\delta_n r\cos\theta d\theta \ dr$$
$$+ \frac{d}{d+2}{\beta_n}^2 \int_0^{\beta_n} r \int_0^{\pi} (r\sin\theta)^{d-2} \ d\theta \ dr \Bigg)$$

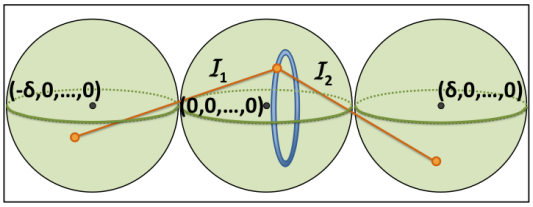

Figure 3.5: To compute $\mathbb{E}[I_1 I_2]$, integration is performed over the common point determining I_1 and I_2.

$$= \frac{(d-1)V_{d-1}}{V_d {\beta_n}^d}\left(\int_0^{\beta_n} r\big(S_{d-2}(r^d + \delta_n^2 r^{d-2})\big)dr + \frac{d}{d+2}{\beta_n}^2\int_0^{\beta_n} r^{d-1}S_{d-2}dr\right)$$

$$= \frac{(d-1)V_{d-1}}{V_d {\beta_n}^d}S_{d-2}\left(\frac{{\beta_n}^{d+2}}{d+2} + \frac{\delta_n^2{\beta_n}^d}{d} + \frac{d{\beta_n}^2}{d+2} + \frac{{\beta_n}^d}{d}\right)$$

$$\mathbb{E}[I_1^2] = \delta_n^2 + \frac{2d}{d+2}{\beta_n}^2$$

□

Lemma 8 (Expected value of $I_1 I_2$ in $\mathbb{R}^d$)**.** *For three consecutive hyperballs, the expected value of the product of the lengths of the segments connecting random samples inside those balls is*

$$\mathbb{E}[I_1 I_2] \approx \delta_n^2 + \Big(\frac{2d-3}{d+2}\Big)\beta_n^2$$

Proof. To compute $\mathbb{E}[I_1 I_2]$, a key observation is made. First, to retrieve this value, the reasoning must consider three consecutive hyperballs, where the distance between the samples of the first two balls, I_1, and the distance between the samples of the second and third balls, I_2, depend on each other through their common endpoint in the second ball. Consider, however, that if this second point is fixed, then the values of I_1 and I_2 become independent. Using this fact, and the intermediate result of the mean calculation, begin by simply multiplying these two means to get the intermediate result:

$$A = (D_1 + \frac{C_d\beta_n^2}{2D_1})(D_2 + \frac{C_d\beta_n^2}{2D_2})$$

$$A = D_1 D_2 + \frac{C_d \beta_n^2}{2}\Big(\frac{D_1}{D_2} + \frac{D_2}{D_1}\Big)$$

where the fourth-order term involving β_n^4 is negligible.

Now, to reintroduce the dependence of I_1 and I_2, integrate over the second ball. For the ease of exposition, it is assumed that the centers of the three hyperballs are collinear. Then, the integral to solve is the following:

$$\mathbb{E}[I_1 I_2] \;=\; \frac{1}{V_d {\beta_n}^d} \int \cdots \int_{x_1^2 + \ldots + x_d^2 \leq {\beta_n}^2} D_1 D_2 \;+\; \frac{(d-1){\beta_n}^2}{2(d+2)}\Big(\frac{D_2}{D_1} \;+\; \frac{D_1}{D_2}\Big)\; dx_1 \ldots dx_d$$

Where D_1 and D_2 are expressed in relation to the center of the middle hyperball as $D_1 = \sqrt{(x_1 - \epsilon)^2 + x_2^2 + \ldots + x_d^2}$ and $D_2 = \sqrt{(x_1 + \epsilon)^2 + x_2^2 + \ldots + x_d^2}$. Again, taking $z^2 = x_2^2 + \ldots x_d^2$ and then, because the second term is already second-order, take a 0^{th}-order approximation over only that term:

$$\mathbb{E}[I_1 I_2] \approx \frac{1}{V_d {\beta_n}^d} \iint_{x_1^2 + z^2 \leq {\beta_n}^2} \left(\sqrt{(x_1 - \delta_n)^2 + z^2}\sqrt{(x_1 + \delta_n)^2 + z^2} + \frac{(d-1){\beta_n}^2}{2(d+2)}\Big(\frac{\delta_n}{\delta_n} + \frac{\delta_n}{\delta_n}\Big)\right) d|\mathcal{B}_z^{d-1}(\cdot)|\; dx_1 dz$$

Again, convert this integral into polar coordinates:

$$\mathbb{E}[I_1 I_2] \approx \frac{(d-1)V_{d-1}}{V_d {\beta_n}^d} \int_0^{\beta_n} r \int_0^{\pi} (r\;\sin\theta)^{d-2} \left(\sqrt{r^2 + \delta_n^2 - 2\delta_n r\cos\theta}\sqrt{r^2 + \delta_n^2 + 2\delta_n r\cos\theta} + \frac{(d-1){\beta_n}^2}{(d+2)}\right) d\theta\; dr$$

$$\mathbb{E}[I_1 I_2] \approx \frac{\delta_n^2 (d-1)V_{d-1}}{V_d {\beta_n}^d} \int_0^{\beta_n} r^{d-1} \int_0^{\pi} (\sin\theta)^{d-2} \left(\sqrt{1 + 2\Big(\frac{r}{\delta_n}\Big)^2 - 4\Big(\frac{r}{\delta_n}\Big)^2 \cos^2\theta} + \frac{(d-1){\beta_n}^2}{(d+2)}\right) d\theta\; dr$$

Then, as the square root is prohibitive to integrate directly, a second-order Taylor approximation is employed:

$$\mathbb{E}[I_1 I_2] \approx \frac{\delta_n^2 (d-1) V_{d-1}}{V_d {\beta_n}^d} \int_0^{\beta_n} r^{d-1} \int_0^{\pi} (\sin\theta)^{d-2} \left(1 + \left(\frac{r}{\delta_n}\right)^2 - 2\left(\frac{r}{\delta_n}\right)^2 \cos^2\theta + \frac{(d-1){\beta_n}^2}{(d+2)}\right) d\theta \; dr$$

$$\mathbb{E}[I_1 I_2] \approx \frac{\delta_n^2 (d-1) V_{d-1}}{V_d {\beta_n}^d} \int_0^{\beta_n} r^{d-1} \int_0^{\pi} (\sin\theta)^{d-2} \left(1 - \left(\frac{r}{\delta_n}\right)^2 + 2\left(\frac{r}{\delta_n}\right)^2 \sin^2\theta + \frac{(d-1){\beta_n}^2}{(d+2)}\right) d\theta \; dr$$

$$\mathbb{E}[I_1 I_2] \approx \frac{\delta_n^2 (d-1) V_{d-1}}{V_d {\beta_n}^d} S_{d-2} \int_0^{\beta_n} r^{d-1} \left(1 - \left(\frac{r}{\delta_n}\right)^2 + \frac{2(d-1)}{d}\left(\frac{r}{\delta_n}\right)^2 + \frac{(d-1){\beta_n}^2}{(d+2)}\right) dr$$

$$\mathbb{E}[I_1 I_2] \quad \approx \quad \frac{d\delta_n^2}{{\beta_n}^d}\left(\frac{{\beta_n}^d}{d} \; - \; \frac{{\beta_n}^{d+2}}{\delta_n^2(d+2)} \; + \; \frac{(2d-2){\beta_n}^{d+2}}{d\delta_n^2(d+2)} \; + \; \frac{(d-1){\beta_n}^2}{(d+2)}\right)$$

$$\mathbb{E}[I_1 I_2] \approx \delta_n^2 + \left(\frac{2d-3}{d+2}\right){\beta_n}^2$$

□

Subsitute the values from Lemmas 6, 7, and 8 into the above form to get:

$$Var\left(\sum_{m=1}^{M} I_m\right) \approx M\left(\delta_n^2 + \frac{2d}{d+2}\beta^2\right) + (2M-2)\left(\delta_n^2 + \frac{2d-3}{d+2}\beta^2\right) + (2-3M)\left(\delta_n + \frac{(d-1)\beta^2}{(d+2)\delta_n}\right)^2$$

After algebraic simplification, the following Lemma is reached:

Lemma 9 (Variance of I_n)**.** *I_n has variance:*

$$Var(I_n) \approx \frac{2\beta_n^2}{d+2}$$

Euclidean	$\lambda_n = 0.5$	$\lambda_n = 0.125$
dimension	% error	% error
2	6.0245%	0.5739%
3	9.7691%	1.0655%
10	19.0989%	2.1429%
100	23.7279%	2.8191%

Figure 3.6: Simulation comparison for $Var(I_n)$, using 120,000 data points for each entry, where the error is $100 \cdot \frac{|Var(I_n) - Var^{MC}|}{Var^{MC}}$.

Monte Carlo simulations are used to verify that the drawn approximation of the variance characterizes the variance properly, as illustrated in Figure 3.6.

Finalizing the PNO guarantee of PRM*

Now that the mean and variance of I_n has been approximated, application of the Chebyshev inequality results in the following theorem:

Theorem 1 (Probabilistic Near-Optimality of PRM*). *For finite iterations n, PRM* is probabilistically near-optimal, returning a path of length I_n such that*

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) \leq 1 + \mathbb{P}\big(\Omega^\beta_n\big)(\chi - 1), \textit{where } \chi = \frac{\frac{4\lambda_n^2 \delta_n^2}{d+2}}{{I^*_{\delta_n}}^2 \big(\epsilon - \frac{(d-1)}{(d+2)}\lambda_n^2\big)^2} \quad (3.4)$$

Proof. Now that the mean and variance of I_n have been approximated, the derivation of the bound can continue. Recall that in Section 3.2.3, the bound was manipulated into the following form:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n} \mid \Omega^\beta_n\big) = 2\mathbb{P}\big(y > (\epsilon + 1)I^*_{\delta_n} - \mathbb{E}(I_n) \mid \Omega^\beta_n\big)$$

Now, substituting the computed values into this expression:

$$2\mathbb{P}\big(y > (\epsilon+1)M_n\delta_n - M_n\big(\delta_n + \frac{(d-1)\beta_n^2}{(d+2)\delta_n}\big) \mid \Omega^\beta_n\big)$$

$$= 2\mathbb{P}\big(y > M_n\delta_n\big(\epsilon - \frac{(d-1)\beta_n^2}{(d+2)\delta_n^2}\big) \mid \Omega^\beta_n\big)$$

Recall that the inequality leveraged is Chebyshev's Inequality:

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq a) \leq \frac{Var(X)}{a^2}$$

then application of this inequality yields:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| > \epsilon \cdot I^*_{\delta_n} \mid \Omega_n^\beta\big) \leq \chi, \text{where } \chi = \frac{\frac{4\lambda_n^2\delta_n^2}{d+2}}{{I^*_{\delta_n}}^2\big(\epsilon - \frac{(d-1)}{(d+2)}\lambda_n^2\big)^2},$$

where $\lambda_n = \frac{\beta_n}{\delta_n}$. Then, the unconditional probability can be bounded as:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) \leq \chi\mathbb{P}(\Omega_n^\beta) + (1 - \mathbb{P}(\Omega_n^\beta))$$

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) \leq 1 + \mathbb{P}\big(\Omega_n^\beta\big)(\chi - 1)$$

□

3.3 Using PNO properties in practice

The PNO bound on PRM* can be leveraged in several ways, for instance, the length of the clearance-robust optimal path in the same homotopic class as the currently returned solution can be estimated during runtime. Also, a probabilistic stopping criterion is proposed.

3.3.1 Online Prediction of $I^*_{\delta_n}$

Here it is shown how to predict the length of the optimal path in the same homotopic class as PRM*'s currently returned solution during execution within a confidence bound $\mathbb{P}_{success} = \mathbb{P}(|I_n - I^*_{\delta_n}| < \epsilon \cdot I^*_{\delta_n})$. Estimate $I^*_{\delta_n}$ by considering the number of hyperballs $M_n + 1$ to compute an estimate of the error, ϵ. Then, use the current returned path length from the algorithm, I_n, and set $I^*_{\delta_n} = \frac{I_n}{(\epsilon+1)}$.

In Section 3.2.3, it was shown that:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) \leq 1 + \mathbb{P}\big(\Omega_n^\beta\big)(\chi - 1)$$

Furthermore, it can be argued that:

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) = 2\mathbb{P}\big(I_n - I^*_{\delta_n} \geq \epsilon \cdot I^*_{\delta_n}\big),$$

$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon \cdot I^*_{\delta_n}\big) \geq \mathbb{P}\big(I_n - I^*_{\delta_n} \geq \epsilon \cdot I^*_{\delta_n}\big),$$

and it must also be that:

$$\mathbb{P}\big(I_n - I^*_{\delta_n} \geq \epsilon_n \cdot I^*_{\delta_n}\big) = \mathbb{P}\big(I^*_{\delta_n} \leq \frac{I_n}{\epsilon_n + 1}\big) \leq$$
$$\mathbb{P}\big(|I_n - I^*_{\delta_n}| \geq \epsilon_n \cdot I^*_{\delta_n}\big) \leq 1 + \mathbb{P}\big(\Omega^\beta_n\big)(\chi - 1)$$

Consider, however, that this result is only valid given that $\pi^*_{\delta_n}$ exists for the current value of δ_n. Therefore, it is critical that the algorithm executes at least until $\delta_n \leq \delta_0$, i.e., when $n \geq n_0$. Then all that remains is to solve the bound in terms of ϵ. It is known that

$$\mathbb{P}\big(I^*_{\delta_n} \leq \frac{I_n}{\epsilon_n + 1}\big) \leq 1 + \mathbb{P}\big(\Omega^\beta_n\big)(\chi - 1) \geq 1 - \mathbb{P}_{success}$$

Then, the goal is to solve for ϵ_n. Performing some algebraic manipulation:

$$\chi \geq 1 - \frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)}$$

Then, substituting the value for χ yields,

$$\frac{\frac{4\lambda_n^2\delta_n^2}{d+2}}{{I^*_{\delta_n}}^2\big(\epsilon_n - \frac{d-1}{d+2}\lambda_n^2\big)^2} \geq 1 - \frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)}$$

$$\frac{4\lambda_n^2\delta_n^2}{(1 - \frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)})(d+2){I^*_{\delta_n}}^2} \geq \big(\epsilon_n - \frac{(d-1)}{(d+2)}\lambda_n^2\big)^2$$

$$\sqrt{\frac{4\lambda_n^2\delta_n^2}{(1 - \frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)})(d+2){I^*_{\delta_n}}^2}} + \frac{(d-1)}{(d+2)}\lambda_n^2 \geq \epsilon_n$$

$$\epsilon_n \leq \frac{2\lambda_n \delta_n}{I^*_{\delta_n}} \sqrt{\frac{1}{(d+2)(1-\frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)})}} + \frac{(d-1)}{(d+2)}\lambda_n^2$$

Finally, estimate $I^*_{\delta_n}$ as $I^*_{\delta_n} \approx \frac{I_n}{(\epsilon_n+1)}$. This leads to the following Lemma:

Lemma 10 (Multiplicative bound ϵ_n)**.** *After $n > n_0$ iterations of* PRM**, with probability $\mathbb{P}_{success}$, if $\pi^*_{\delta_n}$ exists, then* PRM* *contains a path ϵ_n-bounded by $I^*_{\delta_n}$ where:*

$$\epsilon_n \leq \frac{2\lambda_n \delta_n}{I^*_{\delta_n}} \sqrt{\frac{1}{(d+2)(1-\frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)})}} + \frac{(d-1)}{(d+2)}\lambda_n^2 \tag{3.5}$$

3.3.2 Deriving probabilistic stopping criteria

By settting a desired confidence probability $\mathbb{P}_{DES}$ of returning a path within a desired quality bound ϵ_{DES}, an iteration limit n_0 can be derived. For input δ_0, there may exist some δ_0-robust optimal path π_0 of length $I^*_{\delta_0}$. It is possible to compute a required iteration n_0, such that a path π_0 covering π^*_0 has been computed, which has length I_0 bounded by ϵ_{DES} with probability $\mathbb{P}_{DES}$. Let $\lambda_n = \frac{1}{2}$ and then solve Equation 3.5 for $\mathbb{P}(\Omega^\beta_n)$ yields:

$$(d+2)\big(1-\frac{\mathbb{P}_{success}}{\mathbb{P}(\Omega^\beta_n)}\big) \geq \frac{1}{M_n^2\big(\epsilon - \frac{1}{4}\frac{(d-1)}{(d+2)}\big)^2}$$

Solving for $\mathbb{P}(\Omega^\beta_n)$, the right hand side will be denoted as ψ:

$$\mathbb{P}(\Omega^\beta_n) \geq \frac{1}{\frac{1}{\mathbb{P}_{DES}} \cdot \Big(1 - \frac{1}{M_0^2 \cdot (d+2)\big(\epsilon_{DES} - \frac{(d-1)}{4(d+2)}\big)^2}\Big)} = \psi$$

Then, substituting the form of Equation 3.3 using β_0, M_0, and n_0, and solving for n_0 yields:

$$n_0 \leq \left\lceil \frac{\log\left(1 - \sqrt[M_0]{\psi}\right)}{\log\left(1 - \frac{|\mathcal{B}_{\beta_0}|}{|\mathcal{C}^{\text{free}}|}\right)} \right\rceil$$

Here, n_0 represents a minimum number of samples PRM* must be run in order to guarantee PNO properties.

Small 3D: n	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.5$	$\epsilon = 1.0$
100	0.0028	0.1424	0.7601	0.9362
1000	0.0666	0.5815	0.9330	0.9835
100000	0.7675	0.9635	0.9957	0.9990
10000000	0.9785	0.9977	0.9998	0.9999
Large 3D: n	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.5$	$\epsilon = 1.0$
100	0.0000	0.0001	0.0016	0.0022
1000	0.0001	0.0009	0.0151	0.0206
100000	0.0064	0.1738	0.7675	0.9369
10000000	0.5567	0.8953	0.9847	0.9962
Small 6D: n	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.5$	$\epsilon = 1.0$
500	0.0000	0.0033	0.6335	0.8235
5000	0.0000	0.0845	0.8584	0.9712
500000	0.0017	0.5884	0.9669	0.9934
50000000	0.5914	0.8856	0.9924	0.9985
Large 6D: n	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.5$	$\epsilon = 1.0$
500	0.0000	0.0000	0.0533	0.0693
5000	0.0000	0.0058	0.6853	0.8908
500000	0.0004	0.3817	0.9383	0.9874
50000000	0.1828	0.8056	0.9856	0.9971

Figure 3.7: Probability of returning near-optimal paths for 3D and 6D collision-free problem instances. Here, $|\mathcal{C}^{\text{free}}| = 1000$ (Small 3D), 1000000 (Large 3D), 216000 (Small 6D), and 1728000 (Large 6D).

Lemma 11 (PNO iteration limit for PRM*)**.** *For given ϵ_{DES} and $\mathbb{P}_{DES}$, the graph of* PRM* *probabilistically contains a path π_0 of length I_0 with $\mathbb{P}\big(|I_0 - I^*_{\delta_0}| \geq \epsilon_{DES} \cdot I^*_{\delta_0}\big) \leq 1 - \mathbb{P}_{DES}$ after n_0 iterations, where*

$$n_0 \;\leq\; \left\lceil \frac{\log\left(1 - \sqrt[M_0]{\psi}\right)}{\log\left(1 - \frac{|\mathcal{B}_{\beta_0}|}{|\mathcal{C}^{\text{free}}|}\right)} \right\rceil, \text{where } \psi \;=\; \frac{1}{\frac{1}{\mathbb{P}_{DES}} \cdot \left(1 - \frac{1}{M_0^2 \cdot (d+2)\left(\epsilon_{DES} - \frac{(d-1)}{4(d+2)}\right)^2}\right)} \tag{3.6}$$

The existence of the optimal path is not guaranteed. When the stopping criterion is satisfied, if no path has been returned within the path length bound, with probability $\mathbb{P}(\Omega_n^\beta)$, no such optimal path exists. PRM* can still return paths in different homotopic classes, and if users have knowledge of these alternate paths, then multiple criteria can be checked for satisfaction. It is left to future work to determine the best approach for reasoning over multiple homotopic classes.

The tables in Figure 3.7 provide an indication of the required iterations for different

problems of varying dimensionality. These values are given for collision-free environments, and the presence of obstacles can reduce the size of $\mathcal{C}^{\text{free}}$, increasing these probabilities, but they can also reduce δ_n, which reduces these probabilities.

3.4 Indications from Simulation

To test the validity of the analysis, simulations were performed using the described PRM* in four environments on the PRACSYS simulation software [76]. The specific parameters of these environments can be found in Figure 3.10. Environments with obstacles are shown in Figure 3.8. The automated stopping criterion was tested to ensure it stops PRM* appropriately, such that the PNO properties are satisfied, i.e., the actual number of times PRM* succesfully returns a path within the bound is higher than $\mathbb{P}_{success}$.

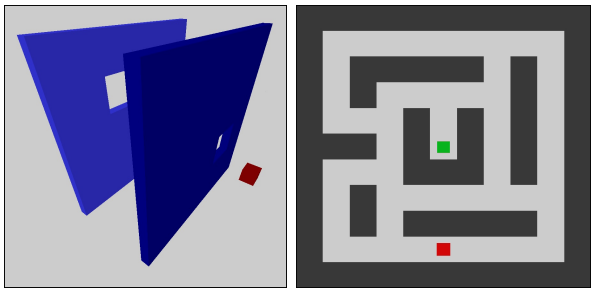

Figure 3.8: The environments with obstacles used as part of testing PNO properties: Barriers (Left), and Maze (Right)

For the desired path bound, the iteration limit n_0 was computed. Then, out of 500 experimental trials, the actual probability of successfully generating a path through the set of hyperballs over π_0 is computed. The stopping criterion properly selects n_0 so that $\mathbb{P}_{success}$ is greater than the input threshold $\mathbb{P}_{DES}$, which was set to 0.9 for this set of experiments. The probability of success for the algorithm over time is given in Figure 3.9 for the chosen environments. While the drawn bound on $\mathbb{P}_{success}$ becomes more loose for paths that are relatively short compared to their clearance, the bound generally performs very well, especially in environments with obstacles. Empty environments naturally have large clearance, causing some discrepancy to arise. The cause is likely due to simplifying assumptions of independence made during the approximation of the expected value and variance of I_n.

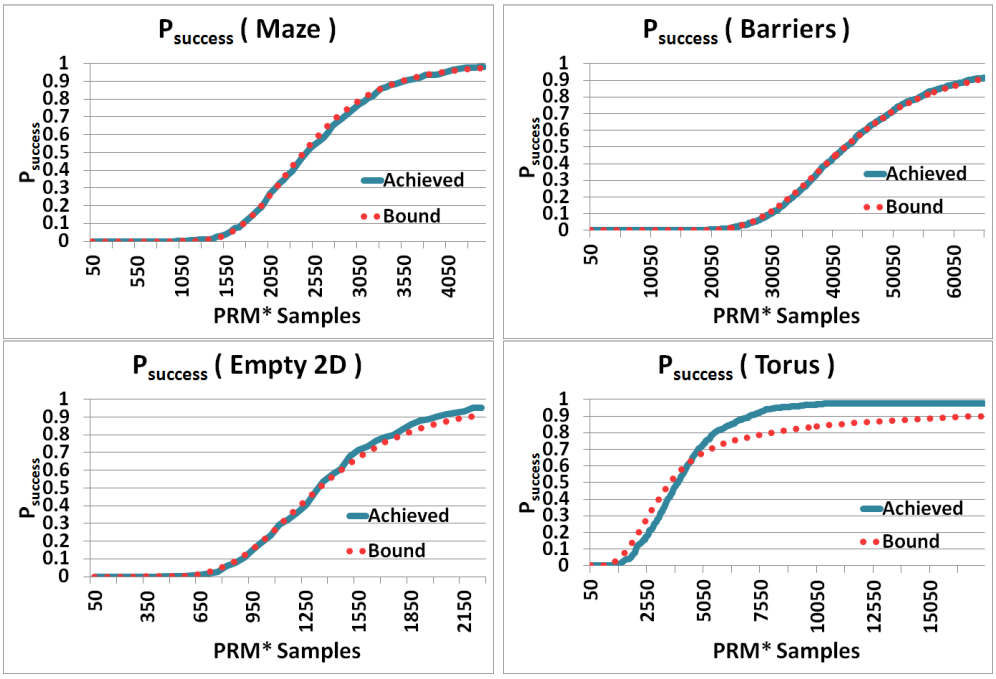

Figure 3.9: Probability of returning a path within the bound ϵ_{DES} for PRM* over time.

Environment	$\lvert\mathcal{C}^{\text{free}}\rvert$	ϵ_{DES}	δ_0	$I^*_{\delta_0}$
Maze	11150	0.2	5.0	240.0
Barriers	700	0.2	0.5	16.5
Empty ($2D$)	300	0.2	1.0	20.0
Empty (T^3)	248.0520	0.3	0.7	3.4641

Figure 3.10: Important parameters for the test environments used to validate the stopping criterion.

3.5 Discussion

This work extends PNO properties to PRM*. It shows PNO properties using an asymptotically sparse planning structure, and removes dependence on Monte Carlo simulations [27]. The analysis provides tight bounds for path quality, which are validated through simulation.

An important future work is to extend PNO properties to RRT*. Furthermore, these methods still generate many samples; thus, extending PNO to methods that add few nodes is important [29]. Path length bounds are drawn under the assumption of a Euclidean distance metric; however, for many robotic systems, this metric is not appropriate, so considering other cost metrics is fundamental.

Early versions of this work considered using lattice points instead of random samples. This proved problematic as the analysis would have to carefully consider the shape of the optimal path to attain reasonable estimates of path length. Also, as dimensionality increases, uniform grids become progressively worse at covering the space. This would require the identification of appropriate covering lattices in high-dimensions, and would likely require exponentially many points.

Bounds should become tighter if the framework is appropriately extended to consider multiple paths simultaneously. Further work will integrate `PNO` properties into task planning frameworks to create more efficient search methods. The bound can also be improved by finding analytical solutions where approximations were taken or by considering the effect of multiple samples in hyperballs.

Chapter 4

Asymptotic Near-Optimality with Compact Representations

In the prior chapter, it was shown that sampling-based motion planning approaches exhibit properties in terms of path quality guarantees after a finite amount of computation; however, the experimental results recapture the so-called curse of dimensionality inherent in the motion planning problem. That is, that to provide high-quality path guarantees with a great deal of certainty, the method requires exponentially many samples with increasing problem dimension. To combat this problem, this Chapter focuses on a different property called **Asymptotic Near-Optimality**, which guarantees a method asymptoticall converges to returning a near-optimal solution. By giving up on strict optimality requirements, much sparser planning structures can be created requiring many fewer samples to return paths of provably high quality.

4.1 Problem Setup and Objectives

This Chapter focuses on solving the Robustly Feasible Path Planning Problem, redefined below by providing a discrete, graphical representation of the free configuration space that can be stored efficiently and queried quickly while providing formal guarantees. The results presented here are for planning problems involving rigid bodies ($SE(2)$ and $SE(3)$). Nevertheless, the method is applicable in any space where an appropriate metric and sampling function exist.

Definition 8 (Robustly Feasible Motion Planning)**.** *For a Robustly Feasible Motion Planning (RFMP) Problem,* $(\mathcal{C}, q_{start}, q_{goal}, \delta_0)$*, and a clearance value* δ_0 *so that a* δ_0*-robust path* π_{δ_0} *exists,* $\pi_{\delta_0}(0) = q_{start}$ *and* $\pi_{\delta_0}(0) = q_{goal}$*, find a solution path* $\pi_n(\tau) \rightarrow \mathcal{C}^{\text{free}}$ *so that* $\pi(0) = q_{start}$ *and* $\pi(1) = q_{goal}$*.*

A naive way to define a graphical representation of $\mathcal{C}^{\text{free}}$ is by considering the implicit, exhaustive graph $G(V, E)$, where all the elements of $\mathcal{C}^{\text{free}}$ are nodes and all the collision free paths between them are edges. A compact data structure for answering shortest-path queries in continuous spaces must be able to identify which $\mathcal{C}$ points are not needed as roadmap nodes. Thus, a "roadmap spanner" is a subgraph $G_S(V_S \subset V, E_S \subset E)$ of this implicit, exhaustive graph of the continuous space and should satisfy the following properties:

1. All nodes in G are connected with a path in $\mathcal{C}^{\text{free}}$ to a node on G_S (coverage).
2. G_S has as many connected components as G (connectivity).
3. All shortest paths on G_S are no longer than t times the corresponding shortest paths in G (spanner property).

These properties allow for (a) arbitrary query points to connect to the roadmap, (b) paths to exist between any query points through G_S that can be connected in $\mathcal{C}^{\text{free}}$, and (c) asymptotic near-optimality for query points that lie on G_S. The objective is to also provide asymptotic near-optimality properties for query points that will not be lying on G_S. For such points, it is necessary to take into account the cost of connecting them to G_S, which gives rise to an additive term regarding the relative cost of the solution path computed by the spanner G_S and the optimum solution cost on G. Overall, the objective is to guarantee the following property.

Definition 9 (Asympt. Near-Optimality with Additive Cost)**.** *An algorithm is asymptotically near-optimal with additive cost if, for a path planning problem ($\mathcal{C}^{\text{free}}$, q_{init}, q_{goal}) and cost function $c : \Pi \to \mathbb{R}^{\geq 0}$ with a cl-robust optimal path of finite cost c^*, the probability it will find a path with cost $c \leq t \cdot c^* + \epsilon$, for a stretch factor $t \geq 1$ and additive error $\epsilon \geq 0$, converges to 1 as time approaches infinity.*

This paper presents algorithms that asymptotically converge to sparse data structures with the above property.

4.1.1 Sparse Roadmap Spanner Notation

To describe the proposed algorithms it will be helpful to introduce some new terminology. Every node in the sparse roadmap spanner G_S will be selected so that it represents its local neighborhood (i.e., spanner paths that initiate in the local neighborhood will go through this node). The size of the neighborhood is limited by a visibility radius Δ, i.e., a new sample will attempt to connect to nodes only within such a distance. This visibility radius is introduced both for computational reasons (avoiding the collision checking of long paths) and so as to provide near-optimality guarantees (i.e., bounding the additive ϵ term that arises from the cost of connecting query points to the sparse roadmap spanner). Then for any configuration $q \in \mathcal{C}^{\text{free}}$ it is possible to compute its representative among the existing nodes in G_S as follows:

Definition 10 (Representative)**.** *Given the nodes V_S of the sparse roadmap spanner G_S, a configuration q's representative $v \in V_G$ satisfies the following properties:*

- $d(q, v) \leq \Delta$ *for the visibility range* Δ,
- $\mathbb{L}(q, v) \subset \mathcal{C}^{\text{free}}$,
- $d(q, v) \leq d(q, v'), \forall\ v' \in V_G$ *so that* $\mathbb{L}(q, v') \subset \mathcal{C}^{\text{free}}$.

The representative of a configuration will be denoted as $\texttt{rep}(q)$. This notion gives rise to the dual term of a visibility region.

Definition 11 (Visibility Region)**.** *The visibility region of a node $v \in V_G$ is* $\texttt{vis}(v) = \{q \mid q \in \mathcal{C}^{\text{free}}, \texttt{rep}(q) = v\}$.

The boundaries of a visibility region will arise from three separate conditions. The first is that the region is bounded by Δ. Secondly, the region will be bounded by obstacle and visibility constraints, and lastly, it will be bounded by intersections with other visibility regions, called interfaces. An example of a visibility region can be seen in Fig. 4.1.1.

Definition 12 (Interface)**.** *Given the set of nodes V_G, an interface $i(v, v')$, between two nodes $v, v' \in V_G$ is the shared boundary of their visibility regions:*

$$i(v, v') = \texttt{vis}(v) \cap \texttt{vis}(v').$$

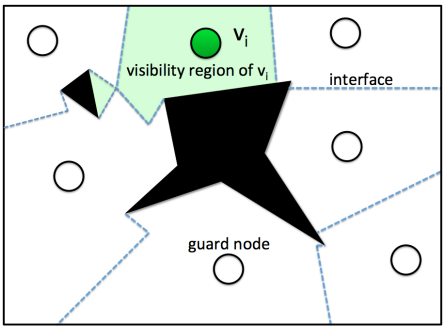

Figure 4.1: Visibility region of v_i, i.e., the configurations connected to v_i that have it as their closest node. The figure ignores the effects of the visibility range Δ.

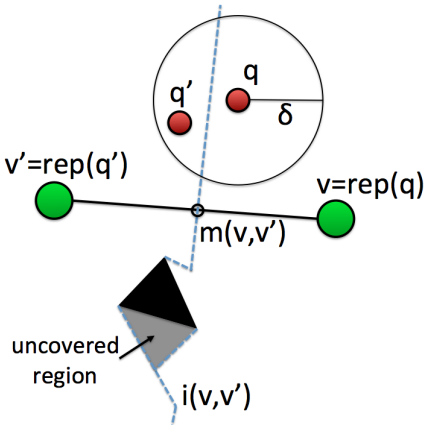

Figure 4.2: Two neighboring nodes v and v' define an interface $i(v, v')$: the shared boundary of their visibility regions.

Figure 4.2 illustrates how an interface arises between two nodes. Given the definition of an interface it is possible to also describe the notion of support, which is also highlighted in the same figure.

Definition 13 (Support)**.** *Given the set of nodes V_G, a configuration q supports the interface $i(v, v')$ if the following is true:*

- $\texttt{rep}(q) = v$,
- $\exists\ q' \in \mathcal{B}(q, \delta) : \texttt{rep}(q') = v' \wedge \mathbb{L}(q, q') \in \mathcal{C}^{\text{free}}$.

In the above definition, the term $\mathcal{B}(q, \delta)$ corresponds to the δ-radius hyper-ball centered at q. It becomes apparent from the above definition that the set of configurations

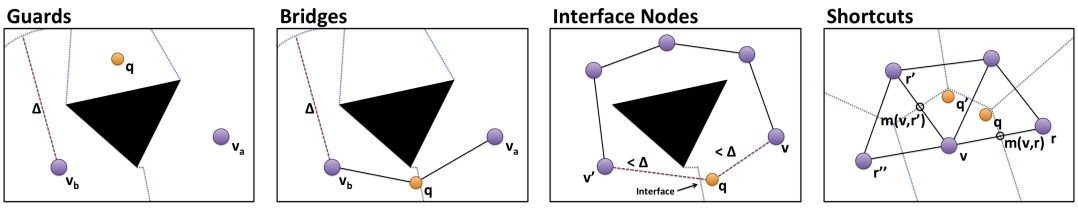

Figure 4.3: The four types of samples $\rho \in \mathcal{C}$ that sparse roadmap spanners are considering for addition. Nodes may be added for coverage (guards), connectivity (bridges), or to connect nodes that share an "interface" (interface nodes), or to satisfy near-optimality constraints when efficient $\mathcal{C}$ paths (ρ to ρ') are found (shortcuts).

that support an interface corresponds to all these $\mathcal{C}$ points that are within distance δ from the interface. The notion of support is introduced because it is easier to discover configurations through a sampling process that support an interface than the the configurations along an interface, as the first set has positive measure, while the second one has zero measure. Finally, the notion of midpoint will be also useful to describe the properties of the proposed algorithms.

Definition 14 (Midpoint)**.** *The midpoint between two configurations v and v' along the local path $\mathbb{L}(v, v')$ will be denoted as $m(v, v')$ and satisfies:*

$$m(v, v') \in \mathbb{L}(v, v') \wedge d(v,\ m(v, v')) = d(m(v, v'),\ v').$$

Note that if the local path $\mathbb{L}(v, v')$ is obstacle-free, then the midpoint $m(v, v')$ lies on the interface $i(v, v')$ between the two configurations, as shown in Figure 4.2.

4.2 Sparse Roadmap Spanner Methods

Algorithm 2 provides a high-level description of the proposed framework for the generation of sparse roadmap spanners. The approach constructs a graph $G_S(V_S, E_S)$ by generating random valid samples in $\mathcal{C}^{\text{free}}$, evaluating whether these configurations satisfy certain criteria to be added as nodes in V_S and then connecting them with local paths. There are four methods for promoting a configuration to the sparse roadmap G_S that are tested in order: "guards", "bridges", "interface nodes", and "shortcuts" as illustrated in Figure 4.3. A guard is added whenever a sample cannot be connected to any node already in V_S with a collision-free path of length Δ. A bridge is found

whenever a configuration can be connected to multiple nodes in V_S that are in disconnected components of G_S. Interface nodes are added when they reveal the existence of an interface between two spanner nodes, which do not share an edge in E_S. Shortcut nodes are added when a path is discovered in $\mathcal{C}^{\text{free}}$ that is significantly shorter than corresponding paths through G_S. Two variations of the basic framework will be described later on: `SPARS` (from `SPA`rse `R`oadmap `S`panner algorithm) and `SPARS2`, which differ on how they implement the identification of shortcut nodes.

Algorithm 2: `Sparse_Roadmap_Spanner`$(M, t, k, \delta, \Delta)$

```
1  failures ← 0;
2  {G_S, G_D} ← Initialize_Graphs();
3  while failures < M do
4  |  q ← Sample_Configuration(δ, G_D);
5  |  W ← Visible_Guards(q, Δ, G_S);
6  |  if W == ∅ then
7  |  |  Add_Guard(q, G_S);
8  |  end
9  |  else if any two w ∈ W not connected then
10 |  |  Add_Connector(q, W, G_S);
11 |  end
12 |  else
13 |  |  Add_Interface_Node(q, Δ, G_S);
14 |  end
15 |  if q ∉ V(G_S) then
16 |  |  Add_Shortcut(q, t, k, δ, Δ, G_S, G_D);
17 |  end
18 |  if no change in G_S then
19 |  |  failures++;
20 |  end
21 end
22 return G_S;
```

The framework receives five input parameters:

- M: Used for the termination criterion of the approach, similar to the Visibility-Based `PRM`[137].
- t: Corresponds to the stretch factor of the spanner.
- k: A parameter used by the `SPARS2` variation for sampling k configurations in the local neighborhood $\mathcal{B}(q, \delta)$.
- δ and Δ: The two distance thresholds in the $\mathcal{C}$ described in the previous section.

Δ corresponds to a "visibility" range for nodes in V_S, while δ is the radius of a local neighborhood for configurations in $\mathcal{C}^{\text{free}}$ that defines the support of an interface. Typically $\delta << \Delta$.

The method first samples a valid configuration q by employing the uniform random sampler and the validity checker (line 4). Then the algorithm computes the set W of existing nodes in G_S that are within distance Δ and with which q can be connected (line 5), i.e.,

$$\forall\ w \in W : \mathbb{L}(q, w) \in \mathcal{C}^{\text{free}}.$$

Then, there are four possible reasons for which a newly sampled configuration can be added to G_S:

1. *Coverage:* The sample q is in a part of $\mathcal{C}^{\text{free}}$ that is not covered by existing nodes in V_S, i.e., $W = \emptyset$ (lines 6-7). In this case, the sample q is added to the set of vertices: $V_S = V_S \cup q$. The nodes added for $\mathcal{C}$-space coverage will be called "guards". The purpose of these nodes is to ensure that whenever a query is given, the algorithm can connect the start and end query points to the sparse roadmap spanner with a collision-free local path.

2. *Connectivity:* The sample q is able to connect to at least two nodes that are otherwise disconnected: $\exists\ w_1, w_2 \in W$ so that there is no path in G_S that connects w_1 and w_2 (lines 5,8-9). In this case, the sample q is added to the set of vertices: $V_S = V_S \cup q$, and connected to the nodes:

$$\forall\ w \in W : E_S = E_S \cup \mathbb{L}(q, w).$$

3. *Connecting Interfaces:* The sample q reveals the existence of an interface between two nodes that do not share an edge (lines 10-11). The analysis section will show that it is necessary for all pairs of nodes that share an interface to also be connected with an edge. The reasoning is that the algorithm compares paths between midpoints on the roadmap spanner and their relationship with optimum paths in $\mathcal{C}^{\text{free}}$. If an interface exists between two nodes but they do not share an edge, then the midpoint is not on the roadmap spanner. Algorithm 3 details the steps that need to be taken in this case.

The algorithm `Add_Interface_Node` finds the two closest nodes v_1 and v_2 on the

Algorithm 3: `Add_Interface_Node`(q, Δ, G_S)

```
1  N ← Nearest_Guards(q, Δ, G_S);
2  v_1 ← arg min_{n∈N} d(q, n);
3  v_2 ← arg min_{n∈N, n≠v_1} d(q, n);
4  if L(v_1, q), L(q, v_2) ∈ C^free ∧ L(v_1, v_2) ∉ E_S then
5      if L(v_1, v_2) ∈ C^free then
6          E_S ← E_S ∪ L(v_1, v_2);
7      end
8      else
9          V_S ← V_S ∪ {q};
10         E_S ← E_S ∪ {L(v_1, q), L(q, v_2)};
11     end
12 end
```

graph G_S that are within Δ distance of the sample q ignoring obstacles(lines 1-3). Then if q can be connected to v_1 and v_2 but these two nodes are not directly connected (line 4), then the method has discovered the existence of an interface between two nodes, which do not share an edge. In this case, the method tries first to directly connect v_1 and v_2 (lines 5-6) and if this fails, then it adds q to V_S and the local paths $\mathbb{L}(v_1, q)$ and $\mathbb{L}(v_2, q)$ to E_S (lines 7-9).

Note that the method for discovering the existence of an interface does not directly correspond to the definition of the support of an interface as presented in the previous section and differs from the original presentation of the `SPARS` algorithm [31]. A method that would correspond to this definition would require to search the δ-sized hyper-ball centered at q in order to identify whether there are configurations in $\mathcal{B}(q, \delta)$ with different representatives than q. While this is a valid approach to detect an interface and will become necessary in the last part of the algorithm, it also more computationally expensive and takes longer time for the algorithm to discover interfaces. The method described here is a more efficient way to achieve the same objective that utilizes only information from a single $\mathcal{C}$ sample as well as the sample's connectivity properties with nodes of the sparse roadmap spanner. The analysis section will provide a proof that one sample is sufficient to reveal the existence of interfaces between nodes that do not share edges.

4. *Path Quality:* The fourth criterion is evaluated if none of the other ones has succeeded

in augmenting the graph (line 12 of algorithm `Sparse_Roadmap_Spanner`). Its purpose is to evaluate whether the new sample q reveals that there is a shortest path in $\mathcal{C}^{\text{free}}$ that is t times shorter or more than the path on the sparse roadmap spanner that will be used to answer a similar query (line 13). This work describes two alternative ways to implement this criterion, one that utilizes a dense graph G_D of configuration samples to estimate the shortest paths in $\mathcal{C}^{\text{free}}$ (`SPARS`) and another one that avoids the memory requirement of storing G_D and follows a conservative approximation for computing shortest paths in $\mathcal{C}^{\text{free}}$ (`SPARS2`). The following sections will provide the details of each variation.

If a sample fails all criteria, then it is not added to G_S and the parameter *failures* is incremented (lines 14-15). Should *failures* reach the threshold parameter M (line 3), the algorithm terminates and returns the graph computed up to that point (line 16).

4.2.1 Using PRM* to Find Shortest $\mathcal{C}^{\text{free}}$ Paths

The last criterion aims to guarantee that paths returned by the roadmap satisfy the spanner property relative to optimum paths in $\mathcal{C}^{\text{free}}$. The challenge for function `Add_Shortcut` is to provide an algorithmic way for checking whether this is true. The idea in the proposed framework is that to achieve this by reasoning locally within the visibility region of each node $v \in V_S$.

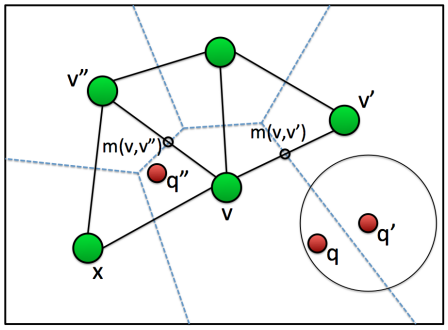

Figure 4.4: Configurations q and q' support $i(v, v')$, while q'' supports $i(v, v'')$. If the optimum path on the dense graph $\pi^*_D(q, q'')$ is t times shorter than the length of the spanner paths $\pi_S(m(v', v), m(v, v''))$ or $\pi_S(m(v', v), m(v, x))$, then configurations along $\pi^*_D(q, q'')$ are candidates for addition.

Consider the situation in Fig. 4.4 and a shortest path $\pi^*(q_{begin}, q_{end})$ in $\mathcal{C}^{\text{free}}$ that

goes through $\texttt{vis}(v)$. Path π^* can be partitioned into several segments, where each one is the intersection of π^* with the visibility region of a node. The roadmap will satisfy the spanner property if it can provide a path $\pi_S(q_{begin}, q_{end})$ that is less than t times the length of $\pi^*(q_{begin}, q_{end})$. Construct then the following spanner path to achieve this objective: replace each segment of the optimum path π^* that goes through a visibility region $\texttt{vis}(v)$, with the spanner path $\pi_S(m(v', v), m(v, v''))$ that connects the midpoints of the spanner edges connecting v with the nodes from whose visibility regions the optimum path enters and exits. For the first segment consider the path from q_{begin} to its representative $\texttt{rep}(q_{begin})$ and then to the midpoint of the edge connecting $\texttt{rep}(q_{begin})$ to the node that π^* is crossing into its visibility region. Similarly for the last segment and q_{end}. It is then sufficient that each segment of the optimum path is no more than t times shorter than the corresponding midpoint spanner path $\pi_S(m(v', v), m(v, v''))$. This is going to be certainly true, if the spanner path is shorter than t times the shortest $\mathcal{C}^{\text{free}}$ path between any two configurations on interfaces $i(v', v)$ and $i(v, v'')$.

The idea in SPARS is that it is possible to asymptotically compute all such shortest paths between interfaces by building in parallel with the roadmap a dense, asymptotically-optimal graph $G_D(V_D, E_D)$ using the PRM* algorithm. Every time that an interface path through the dense graph is revealed to be significantly shorter than the corresponding midpoint spanner path, the configurations along the interface path on the dense graph become candidates for addition to the roadmap. Thus, the algorithm operates as follows: Upon initialization of the roadmap G_S, it also initializes a dense graph G_D (line 2 of routine Sparse_Roadmap_Spanner). Furthermore, every time that a configuration is sampled, it is immediately added as a node in V_D and attempts are made to connect it to all nodes within a δ distance on the dense graph per the δ-PRM* algorithm [69] (line 4 of routine Sparse_Roadmap_Spanner). Consequently, the structure G_D will asymptotically converge to optimal $\mathcal{C}^{\text{free}}$ paths. The Add_Shortcut function in SPARS uses the dense graph G_D to compute interface paths for the representative of the current sampled configuration q.

Algorithm 4 first computes the representative of q (line 1) and then finds the set Q' of all neighbors of q in the dense graph within distance δ and their representatives V'

Algorithm 4: `Add_Shortcut`$(q, t, k, \delta, \Delta, G_S, G_D)$

```
1  v ← rep(q);
2  Q' ← Adjacent_Vertices(q, δ, G_D);
3  V' ← {v' | ∃ q' ∈ Q', so that v' = rep(q')};
4  for v' ∈ V' \ v do
5      for v'' ∈ V_S : ∃ i(v'', v) ∧
6      L(v'', v) ∈ E_G ∧ L(v'', v') ∉ E_G do
7          π_S ← Max_Spanner_Path(v, v', v'', G_S);
8          Q'' ← Interface_Support(v, v'');
9          π*_D ← argmin_{q''∈Q''} |π_D(q, q'')|;
10         if t · |π*_D| < |π_S| then
11             if L(v', v'') ∈ C^free then
12                 E_S ← E_S ∪ L(v', v'');
13             end
14             else
15                 Add_Path(G_S, {v' → π*_D → v''});
16             end
17         end
18     end
19 end
```

(lines 2-3). If there is a node $v' \in V'$ that is different than the representative v of q, then q supports an interface per the definition (line 4). Then the algorithm considers all nodes v'' in the roadmap spanner, which share an interface and an edge with v but do not share an edge with v' (lines 5-6). Note that it is possible for two nodes to share an edge but not an interface, as the algorithm ends up including edges that cross multiple visibility regions. The detection of an interface between two nodes v and v'' in this algorithm utilizes information from the underlying dense graph. In particular, the algorithm detects if two configurations q and q'', which share an edge in the dense graph G_D have different representatives, i.e., $q \in \texttt{vis}(v)$ and $q'' \in \texttt{vis}(v'')$, where $v \neq v''$. This means that there is an interface $i(v, v'')$. The situation is equivalent to what is displayed in Figure 4.4. If the two sparse nodes v and v'' are not already connected with an edge, the third criterion will augment the roadmap spanner in order to bridge this interface. In this process, it is not necessary to consider vertices v'' that share an edge with v'. This is because the edge $\mathbb{L}(v', v'')$ acts as a shortcut to the midpoint spanner path $\pi_S(m(v', v), m(v, v''))$. In this case there is no reason to compare against the interface path from $i(v', v)$ to $i(v, v'')$.

Algorithm 5: `Max_Spanner_Path`(v, v', v'', G_S)

```
1 Π_S ← { π_S( m(v', v), m(v, v'') ) };
2 for x : L(v, x), L(v'', x) ∈ E_S ∧ L(v', x) ∉ E_S do
3 |   if ∃i(x, v) then
4 |   |   Π_S ← Π_S ∪ { π_S( m(v', v), m(v, x) ) };
5 |   end
6 end
7 return argmax_{∀π∈Π_S} |π|;
```

Then for such a set of vertices v, v', v'' (i.e., v connected to v' and v'' but no edge between v and v'') the spanner path π_S is computed by calling function `Max_Spanner_Path` (line 6). Algorithm 5 provides the implementation for this procedure. Notice that the spanner path π_S ends up being the maximum length path among $\pi_S(m(v', v), m(v, v''))$ and all paths of the form $\pi_S(m(v', v), m(v, x))$, where x are nodes that share an interface and an edge with v, share an edge with v'' but do not share with v'. In order to keep the description brief, the reason for considering these additional vertices x will become apparent during the analysis of the method.

Once the spanner path π_S is found, function `Add_Shortcut` proceeds to find the corresponding shortest path between the interfaces $i(v', v)$ and $i(v, v'')$ given the addition of the new sample q that has been shown to support $i(v', v)$. The algorithm first finds the set of configurations that supports the interface $i(v, v'')$ (line 8) and the shortest path π_D^* on the dense graph between q and these configurations (line 9). Then the spanner property is checked between π_D^* and π_S (line 10). If it is violated, the algorithm first attempts to add a direct edge between nodes v' and v'' (lines 11-12). If this is not possible, then the entire path $\{v' \rightarrow \pi_D^* \rightarrow v''\}$ is considered for addition (lines 13-14). The implementation of `Add_Path` adds two configurations along the input path that support the interfaces $i(v', v)$ and $i(v, v'')$ and then tries to smooth the remaining path as much as possible so as to minimize the number of nodes and edges added to the sparse roadmap spanner. The addition of this path resolves the violation of the spanner property as argued in the analysis section.

4.2.2 Alternative to Storing a Dense Graph

Maintaining the dense graph G_D is very costly in terms of memory requirements for the algorithm during the construction process of the sparse roadmap spanner. In order to reduce the computational footprint of SPARS, a variant is presented that removes the reliance on the dense graph. SPARS2 accomplishes this by employing conservative approximations of the shortest paths between interfaces $i(v', v)$ and $i(v, v'')$ and making use of some alternative bookkeeping information stored on the nodes of the planning structure. Note that a dense graph is no longer initialized by function Sparse_Roadmap_Spanner and when a sample q is generated in SPARS2 it is no longer added to a dense graph (lines 2 and 4 of Sparse_Roadmap_Spanner).

Algorithm 6: Add_Shortcut2$(q, t, k, \delta, \Delta, G_S)$

```
1  v ← rep(q);
2  (Q', V') ← (∅, ∅);
3  for k iterations do
4  |  q' ← Sample_Near(q, δ);
5  |  if 𝕃(q, q') ∈ C^free then
6  |  |  v' ← rep(q');
7  |  |  if ∄ v' then
8  |  |  |  Add_Guard(q', G_S);
9  |  |  end
10 |  |  else if v' ≠ v then
11 |  |  |  Q' ← Q' ∪ q',  V' ← V' ∪ v';
12 |  |  end
13 |  end
14 end
15 if V' ≠ ∅ then
16 |  for each v' ∈ V' and q' ∈ Q' do
17 |  |  Update_Points(q, q', v, v', G_S);
18 |  |  Update_Points(q', q, v', v, G_S);
19 |  end
20 |  Test_Add_Paths(v, G_S);
21 |  for each v' ∈ V' do
22 |  |  Test_Add_Paths(v', G_S);
23 |  end
24 end
```

The first use of the dense graph in SPARS was to detect samples that support an interface (line 2 of Add_Shortcut). Since the dense graph is no longer maintained, an

alternative method is needed to detect samples that support interfaces. Lines 2 - 10 of Algorithm 6 provide a sampling-based method to achieve this objective. Each time a new configuration q is tested for the addition of a shortcut, k additional samples q' are generated in its δ-radius hyper-ball $\mathcal{B}(q, \delta)$ (lines 3-4). If the local path $\mathbb{L}(q, q')$ is free and the representative of q' is different than that of q (lines 5-6,9-10), then the requirements for identifying two configurations that support an interface have been met. Note that in the case that q' cannot be connected to any existing node, a new "guard" has been discovered and needs to be added as such to the roadmap (lines 7-8).

Once it is detected that sample q supports an interface of its representative v, it is necessary to check whether it reveals the existence of a new interface path that is not covered by the corresponding spanner midpoint path. If the dense graph were available, it would be possible to iterate over all configurations that support another interface of v and compute shortest paths between them and q. While computing shortest paths in $\mathcal{C}^{\text{free}}$ is not possible without the dense graph, it is still possible to detect samples that support interfaces as the previous paragraph indicated. Towards this objective, `SPARS2` maintains for each node v and for each pair of neighbors (v', v'') the following information (this information is stored on each node v of the corresponding graph data structure):

- A pair $P_v(v', v'')$ of configurations that support the corresponding interfaces $i(v', v)$ and $i(v, v'')$, which both belong in $\texttt{vis}(v)$, and define the shortest distance between any such pair of configurations.
- And a corresponding pair $\Xi_v(v', v'')$ of configurations that support the interfaces $i(v', v)$ and $i(v, v'')$, where the first belongs in $\texttt{vis}(v')$ and the second belongs in $\texttt{vis}(v'')$, and are the samples that reveal that the configurations in $P_v(v', v'')$ are supporting an interface.

Algorithm 7 is responsible for updating this bookkeeping information, and is illustrated by Figure 4.5. Given a configuration q that belongs in $\texttt{vis}(v)$ and a configuration $q' \in \mathcal{B}(q, \delta)$ that belongs in $\texttt{vis}(v')$ (both configurations support $i(v, v')$)) the algorithm considers all vertices v'', which share an interface and an edge with v but not an edge v' (lines 1-3). Then the algorithm retrieves the previously stored pair of configurations

Algorithm 7: `Update_Points`(q, q', v, v', G_S)

```
1  for v'' ∈ V_S \ v, v' do
2      if ∃ i(v'', v) then
3          if L(v, v'') ∈ E_S and L(v', v'') ∉ E_S then
4              (ρ', ρ'') ← P_v(v', v'');
5              (ξ', ξ'') ← Ξ_v(v', v'');
6              if d(q, ρ'') < d(ρ', ρ'') then
7                  P_v(v', v'') = (q, ρ'');
8                  Ξ_v(v', v'') = (q', ξ'');
9              end
10         end
11     end
12 end
```

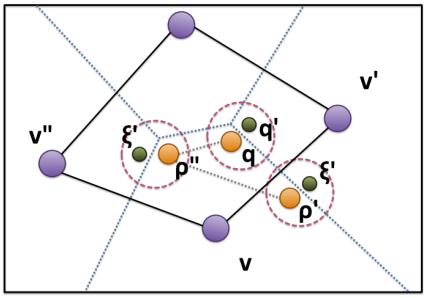

Figure 4.5: An example of the relative locations of the configurations used in the Algorithm 7. In this example, the new sample q is closer to ρ'' than ρ' was, so the pair of configurations for the connection between v' and v'' via v will be updated.

(ρ', ρ'') that support $i(v', v)$ and $i(v, v'')$ from the side of v, which correspond to the shortest distance among all such configurations (line 4). The corresponding $\Xi_v(v', v'')$ pair is also retrieved (line 5). If the new sample q is closer to the best representative of the interface $i(v, v'')$ (line 6), then the bookkeeping information is being updated with q and q'. Note that function `Update_Points` is called to update not only the information of v (line 13 of `Add_Shortcut2`) but also for the neighboring node v' of whose interface configuration q supports (line 14). A notable concern is how large this extra bookkeeping information becomes. It is expected that as the dimensionality of a problem increases, the space requirements in order to maintain this information will also increase significantly. It seems, however, that for problems up to 6 dimensions, the

memory requirements are quite manageable, as detailed in Section 4.4.

Once the bookkeeping information is updated, algorithm `Add_Shortcut2` proceeds to check if the new sample q revealed two interfaces that are closer than the corresponding spanner midpoint path by calling function `Test_Add_Paths` (lines 15-17). Note that this function is called only if `Update_Points` had to update the bookkeeping information.

Algorithm 8: `Test_Add_Paths`(v, G_S)

```
1  for v' ∈ V_S : 𝕃(v, v'') ∈ E_S do
2      for v'' ∈ V_S : ∃ i(v'', v) ∧
3      𝕃(v, v'') ∈ E_S ∧ 𝕃(v', v'') ∉ E_S do
4          (ρ', ρ'') ← P_v(v', v'');
5          (ξ', ξ'') ← Ξ_v(v', v'');
6          |π̂*_D| ← d(ρ', ρ'');
7          π_S ← Max_Spanner_Path(v, v', v'', G_S);
8          if t * |π̂*_D| < |π_S| then
9              if 𝕃(v', v'') ∈ C^free then
10                 E_S ← E_S ∪ 𝕃(v', v'');
11             end
12             else
13                 Add_Path(G_S, {v' → ξ' → ρ' → v → ρ'' → ξ'' → v''});
14             end
15         end
16     end
17 end
```

Algorithm 8 provides the implementation of `Test_Add_Paths` and in many ways it is similar to the second part of `Add_Shortcut` from `SPARS`. It reasons again for all vertices v'', which share an interface and an edge with v but not an edge v' (lines 1-3). For each such vertex v'' it retrieves from the bookkeeping information the shortest distance between two configurations in `vis`(v) that support $i(v', v)$ and $i(v, v'')$ (lines 4-6). This distance is a conservative approximation of the shortest path in $\mathcal{C}^{\text{free}}$ between these two interfaces, i.e., it will always converge over time to something shorter than the true shortest path. The corresponding spanner midpoint path π_S is computed as in `SPARS` (line 7) and then the spanner property is evaluated (line 8). If violated, an attempt to directly connect v' and v'' is made (lines 9-10). If this is not successful, then the entire path $\{v' \to \xi' \to \rho' \to v \to \rho'' \to \xi'' \to v''\}$ is considered for addition (lines 11-12). The implementation of `Add_Path` adds ξ' and ξ'' and then tries to smooth the remaining

path as much as possible so as to minimize the number of nodes and edges added to the sparse roadmap spanner.

4.3 Sparse Roadmap Spanner Analysis

The discussion on the properties initially relates to the entire framework for the generation of sparse roadmap spanners and then focuses on the `SPARS` (Section 4.1) and `SPARS2` (Section 4.2) variations. A series of lemmas argues the probabilistic completeness of the method, that paths returned by the proposed approaches converge to near-optimal planning structures and that the probability of adding nodes to the structure goes to zero as time progresses.

4.3.1 Probabilistic Completeness

The approach is equivalent to Visibility-based `PRM` [137] with regards to coverage and connectivity. From this equivalence, it is possible to argue that the framework achieves probabilistic completeness.

Theorem 2 (Coverage)**.** *For all* $q \in \mathcal{C}^{\text{free}} : \exists\ v \in V_S$ *so that* $\mathbb{L}(q, v) \in \mathcal{C}^{\text{free}}$ *with probability approaching* 1 *as* M *goes to infinity in the* `Sparse_Roadmap_Spanner` *algorithm.*

The argument to support the above statement can be found in the presentation of the Visibility-based `PRM` [137] as the framework adds all the nodes that would be added by this method and follows a similar termination condition. At a high level, each new guard inserted in G_S increases the coverage of $\mathcal{C}^{\text{free}}$ and the probability of generating configurations in non-covered regions decreases over time. The algorithm is then guaranteed to terminate for any finite input value M. When it stops, a probabilistic estimation of the percentage of free space not covered by spanner nodes is $\frac{1}{M}$, given uniform sampling. This means that future attempts to add spanner nodes will succeed with probability $(1 - \frac{1}{M})$. Consequently, as M goes to infinity, the resulting graph covers the entire space.

Note that relative to the Visibility-based `PRM`, the probability $(1 - \frac{1}{M})$ is more conservative, as the methods for generating sparse roadmap spanners assume a visibility

range limit Δ for graph nodes. The work on Visibility-based PRM has shown that even for fairly complicated problems in $SE(3)$, a relatively small number of guards is needed to probabilistically cover the space. Connectivity properties of the resulting sparse roadmap spanner can be argued in a similar manner.

Theorem 3 (Connectivity)**.** *For all $v, v' \in V_S$ that are connected with a collision-free path in $\mathcal{C}^{\text{free}}$, $\exists\ \pi_S(v, v')$ that connects them on G_S with probability 1 as M goes to infinity.*

The framework explicitly handles checking for connectivity in lines 8-9 of Algorithm 2. The algorithm adds an edge or a node every time it detects there is a way to connect two disconnected components. The probability of sampling a configuration that will connect such disconnected components of the graph depends on the environment. Narrow passages will make this connection more challenging. The probability of connecting any two disconnected components is 1, as the value of M goes to infinity, if the following assumption is true:

Assumption 1. *For any $v, v' \in \mathcal{C}^{\text{free}}$, if the set Q of configurations $q \in Q$ for which the following properties hold is non-empty:*

- $\mathbb{L}(q, v) \in \mathcal{C}^{\text{free}}$, $d(q, v) < \Delta$,
- $\mathbb{L}(q, v') \in \mathcal{C}^{\text{free}}$, $d(q, v') < \Delta$,

then Q has non-zero measure.

It can be argued that the above assumption relates to the definition of an expansive space [60]. The combination of the last two theorems provides probabilistic completeness, at least when the algorithm samples configurations q in a uniform way.

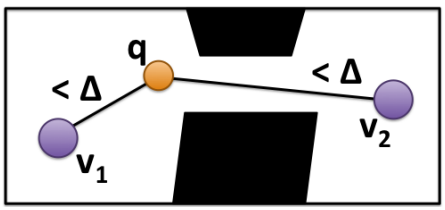

Figure 4.6: One sample can reveal interfaces.

In order to prove certain properties of the algorithm, it must be that all vertices in the planning structure that share an interface must also share an edge on the roadmap. The following theorem argues that this is indeed the case.

Theorem 4 (Connected Interfaces)**.** *For all $v_1, v_2 \in V_S$ that share an interface, then $\mathbb{L}(v_1, v_2) \in E_S$ with probability approaching 1 as M goes to infinity.*

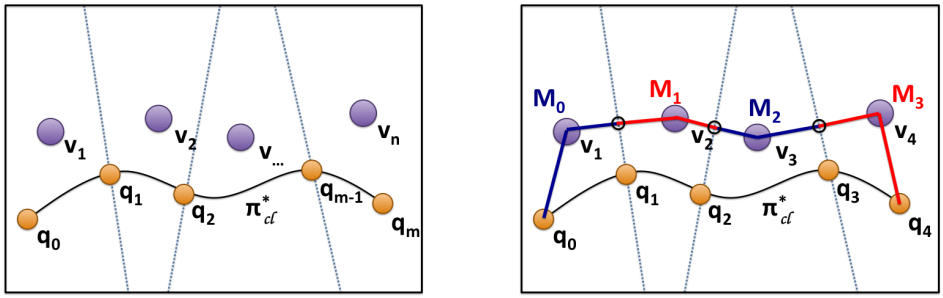

Figure 4.7: (left) All configurations along the optimal path $\pi^*_{\delta_n}(q_0, q_m)$ will be eventually covered by a node in G_S. (right) A path between all spanner nodes covering $\pi^*_{\delta_n}(q_0, q_m)$ will also be created. The figure shows the decomposition of path $\pi_S(q_0, q_m)$ into "midpoint paths" M_{i-1} that exist only in a single visibility region. Each M_{i-1} covers the path $\pi^*_{\delta_n}(q_i, q_{i+1})$.

Proof: In order for the algorithm to detect the existence of an interface between two nodes it has to be that a sample q can connect to its two closest guards v_1 and v_2 as in Figure 4.6 (lines 1-4 of `Add_Interface_Node`). If this is the case, then there must be an interface between v_1 and v_2. Consider a moving configuration q^* along the local path from q to v_2. Moving along this path guarantees that visibility with v_2 is maintained (i.e., $\mathbb{L}(q^*, v_2) \in \mathcal{C}^{\text{free}}$). There are two cases:

(a) either visibility with v_1 is maintained until q^* becomes equidistant with v_1 and v_2

or

(b) at some point visibility with v_1 is lost before q^* becomes equidistant.

In the first case, an interface exists at the point q^* becomes equidistant with v_1 and v_2. It cannot be than any other node v_3 will be closer to q^* than v_1 and v_2. If there were, then some other node v_3 would have been closer to q than v_2, which is not true. In the second case, the interface exists at the point where the visibility with v_1 is lost. For the same reasons, no other node v_3 will be closer at this point than v_1. In order

to be able to sample point q in the first place, it is sufficient that Assumption 1 holds. The above discussion implies that the method is able through a sampling process to detect all pairs of spanner nodes that share an edge but not an interface. The algorithm will try to add an edge between nodes v_1 and v_2. If this fails, then q will be added and will be connected to v_1 and v_2. The addition of q might introduce new interfaces that are not intersected by edges. It will not be the case, however, that these newly created interfaces will always have an obstacle preventing a direct connection between the two vertices which impose it. At some point if vertices are added for this reason, the spanner nodes will be closer than cl and connections between them are guaranteed to be collision-free. In practice, most roadmap nodes will be connected before getting as close as cl. $\square$

4.3.2 Path Quality

The following discussion focuses on showing asymptotic near-optimality properties.

Lemma 12 (Coverage of Optimal Paths by G_S)**.** *Consider an optimal path $\pi^*_{\delta_n}(q_0, q_m)$ in $\mathcal{C}^{\text{free}}$. The probability of having a sequence of nodes in S, $V_\pi = (v_1, v_2, ..., v_n)$ with the following properties approaches 1 as M goes to infinity:*

- $\forall q \in \pi^*_{cl}(q_0, q_m),\ \exists\ v \in V_\pi :\quad \mathbb{L}(q, v) \in \mathcal{C}^{\text{free}}$
- $\mathbb{L}(q_0, v_1) \in \mathcal{C}^{\text{free}}$ *and* $\mathbb{L}(q_m, v_n) \in \mathcal{C}^{\text{free}}$
- $\forall\ v_i, v_{i+1} \in V_\pi, \mathbb{L}(v_i, v_{i+1}) \in E.$

The above lemma is a direct outcome of Theorems 2 and 4 regarding coverage and connected interfaces. Consider now a decomposition of the path $\pi_S(q_0, q_m)$ on G through V_π into sub-paths $\{M_0, M_1, \ldots, M_{m-1}\}$ as shown in Figure 4.7, where M_i is the path between the midpoint $m(v_i, v_{i+1})$ and $m(v_{i+1}, v_{i+2})$, M_0 connects q_0 to v_1 and then $m(v_1, v_2)$ and M_{m-1} is the corresponding last segment. Then the following can be shown:

Lemma 13 (Additive Connection Cost)**.** *The sum of the lengths of segments M_0 and M_{m-1} for a path from q_0 to q_m through G_S is upper bounded by $4 \cdot \Delta$.*

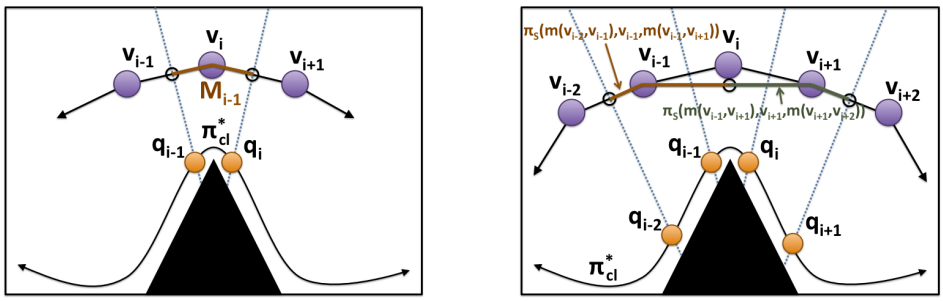

Figure 4.8: (left) The path from $i(v_{i-1}, v_i)$ to $i(v_i, v_{i+1})$ via S must satisfy the spanner property for path $\pi^*_{\delta_n}(q_{i-1}, q_i)$. (right) If there is a path between v_i's neighbors, then the paths $\pi(m(v_{i-2}, v_{i-1}), v_{i-1}, m(v_{i-1}, v_{i+1}))$ and $\pi(m(v_{i+2}, v_{i+1}), v_{i+1}, m(v_{i+1}, v_{i-1}))$ must be checked against $\pi^*_{\delta_n}(q_{i-2}, q_{i-1})$ and $\pi^*_{\delta_n}(q_i, q_{i+1})$.

Proof: The cost for connecting samples to the spanner is at most Δ for both the start and final positions, as we know that q_0 and q_m lie in the visibility region of v_1 and v_m respectively. Note that the remainder of segment M_0 that connects v_1 to v_2 is at most Δ as two spanner nodes that share an interface cannot be further away than $2 \cdot \Delta$ and M_0 terminates at the midpoint between v_1 and v_2. The same is true for M_{m-1}, which implies that each one of these two segments cannot be longer than $2 \cdot \Delta$ and the sum of their lengths is upper bounded by $4 \cdot \Delta$. $\square$

An important property of the midpoint segments, M_i is that they satisfy the spanner property for $\mathcal{C}^{\text{free}}$ paths that go through the corresponding visibility region. Reasoning about the spanner property at a local level allows for the properties to be shown globally.

Lemma 14 (Spanner Property of G_S over $\mathcal{C}^{\text{free}}$). *All segments M_i ($i \in [1, m-2]$) have length bounded by $t \cdot |\pi^*_{\delta_n}(q_{i-1}, q_i)|$, where q_i lies at the intersection of $\pi^*_{\delta_n}(q_0, q_m)$ with interface $i(v_i, v_{i+1})$.*

In order to prove this lemma, it must now be shown separately for `SPARS` and `SPARS2` as they take different approaches to adding nodes for the sake of path quality. A proof is provided first for the `SPARS` method.

Proof for `SPARS`: The following discussion relates to Figures 4.8 and 4.9. Given Lemma 12, the edges $\mathbb{L}(v_{i-1}, v_i)$ and $\mathbb{L}(v_i, v_{i+1})$ are in E_S, at least as M goes to infinity. Assuming $\pi^*_{\delta_n}$ travels through the region of node $v_i \in V_\pi$, there are three possible cases:

(a) $\mathbb{L}(v_{i-1}, v_{i+1}) \notin E_S$ (Figure 4.8(left)),

(b) $\mathbb{L}(v_{i-1}, v_{i+1}) \in E_S$, where $\mathbb{L}(v_{i-2}, v_i) \notin E_S$ and $\mathbb{L}(v_i, v_{i+2}) \notin E_S$ (Figure 4.8(right)), and

(c) $\mathbb{L}(v_{i-1}, v_{i+1}) \in E_S$ and $\mathbb{L}(v_{i-2}, v_i) \in E_S$ or $\mathbb{L}(v_i, v_{i+2}) \in E_S$ (Figure 4.9).

In the first case, the algorithm needs to check $|M_i|$ against $|\pi^*_{\delta_n}(q_{i-1}, q_i)|$. In the case of SPARS, the optimal path $\pi^*_{\delta_n}(q_{i-1}, q_i)$ is asymptotically approximated by the dense graph, so the shortest path through D is used for the true optimum, and is denoted as $\pi^D_{cl}(q_{i-1}, q_i)$. If it were found that $|M_i| > t \cdot \pi^D_{cl}(q_{i-1}, q_i)$, the method would have passed the check on line 10 of Algorithm 4, and a shortcut path would have been added to the graph: either it would have added $\mathbb{L}(v_{i-1}, v_{i+1})$ in which scenario the second or third cases of this proof applies, or a dense path that includes samples from D would have been included in G_S. This means that the representative sequence V_π would have been different for this optimal path.

The second case requires a more careful examination. In particular, the algorithm compares $\pi^D_{cl}(q_{i-1}, q_i)$ not only with the segment M_i but also with all spanner paths from $m(v_{i-1}, v_i)$ to the interfaces of all neighbors of v_i, which are not connected to v_{i-1} but share an interface with v_{i+1}. In the context of Figure 4.8(right) this allows to check whether path $\pi(m(v_{i-2}, v_{i-1}), v_{i-1}, m(v_{i-1}, v_{i+1}))$ satisfies the spanner property for path $\pi^D_{\delta_n}(q_{i-2}, q_{i-1})$ and that path $\pi(m(v_{i+2}, v_{i+1}), v_{i+1}, m(v_{i+1}, v_{i-1}))$ satisfies the spanner property for $\pi^D_{\delta_n}(q_i, q_{i+1})$. If they do, then it does not matter if segment M_{i-1} satisfies the spanner property for path $\pi^D_{\delta_n}(q_{i-1}, q_i)$, because $\pi(m(v_{i-2}, v_{i-1}), v_{i-1}, m(v_{i-1}, v_{i+1}))$ and $\pi(m(v_{i+2}, v_{i+1}), v_{i+1}, m(v_{i+1}, v_{i-1}))$ cover all three consecutive subpaths of the optimum path. Otherwise, the spanner must have been expanded.

The last case is illustrated by Figure 4.9, which has two subcases: the solution returned by SPARS traveling from v_{i-2} to v_{i+2} does so through path $\pi_{a,b,c}$ or through path $\pi_{d,e,f}$. In the case that the path returned is $\pi_{a,b,c}$, the situation is case 2 of the proof and there is no concern. If however, the path returned is $\pi_{d,e,f}$, then it is necessary to show that

$$|\pi_e| \leq t \cdot (|\pi^D_{\delta_n}(q_{i-2}, q_{i-1})| + |\pi^D_{\delta_n}(q_{i-1}, q_i)| + |\pi^D_{\delta_n}(q_i, q_{i+1})|).$$

It is known that $|\pi_b| \leq t \cdot |\pi^D_{\delta_n}(q_{i-2}, q_{i-1})| + |\pi^D_{\delta_n}(q_i, q_{i+1})|$ from case 2. It is also known

that $|\pi_{d,e,f}| \leq |\pi_{a,b,c}|$ or it would otherwise not have been returned by SPARS. Furthermore, the construction of these segments enforces that $|\pi_a| \leq |\pi_d|$ and $|\pi_c| \leq |\pi_f|$, as the endpoints are the intersections with $i(v_{i-2}, v_{i-1})$ for π_a and π_d, and $i(v_{i+1}, v_{i+2})$ for π_c and π_f. By combining and substituting these inequalities, the following expression arises:

$$|\pi_e| \leq |\pi_a| + |\pi_b| + |\pi_c| - |\pi_d| - |\pi_f|.$$

Simplifying this result yields:

$$\begin{gathered} |\pi_e| \leq t \cdot (|\pi^D_{\delta_n}(q_{i-2}, q_{i-1})| + |\pi^D_{\delta_n}(q_i, q_{i+1})|) \leq \\ t \cdot (|\pi^D_{\delta_n}(q_{i-2}, q_{i-1})| + |\pi^D_{\delta_n}(q_{i-1}, q_i)| + |\pi^D_{\delta_n}(q_i, q_{i+1})|). \end{gathered} \tag{4.1}$$

Consequently, this last case is not an issue for the relationship between spanner paths and $\mathcal{C}^{\text{free}}$ paths. □

Proof for SPARS2: There are two points that must be addressed for SPARS2, as it no longer has dense graph information. SPARS2 must (a) use a conservative approximation for $|\pi^*_{\delta_n}(q_{i-1}, q_i)|$, and (b) introduce configurations to approximate dense paths, which would have otherwise been returned by the dense graph.

First, the approximation used for $\pi^*_{\delta_n}(q_{i-1}, q_i)$ must be an underestimation of the true cost. SPARS2 employs $d(q_{i-1}, q_i)$ as the approximation of this cost. It holds that this will always be an underestimation of the true optimal path as given the absence of obstacles, the true optimal paths are the local paths $\mathbb{L}(q_{i-1}, q_i)$. The presence of obstacles will serve only to detour this path and make it longer. Whenever SPARS2 is checking a midpoint path M_i for the spanner property then, it will report that the paths are violating the spanner property more often then they may actually be.

The construction of the approximate optimal paths may be considerably worse than the optimal path that travels through the regions represented by the midpoint paths. In this case, even with powerful smoothing techniques, the resulting path may still not satisfy the spanner property given the conservative approximation of the true optimal path. The argument is that this is not problematic, as the addition of this approximate dense path will create new visibility regions through which new candidate midpoint paths can be evaluated in future iterations of the algorithm. Eventually, the direct

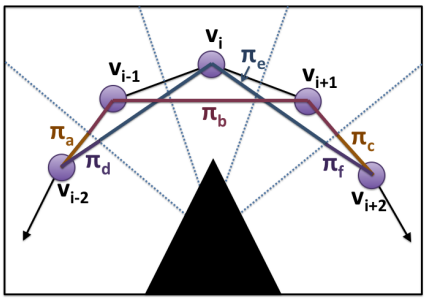

Figure 4.9: In the case that v_i depends on its neighbors for checking the spanner property, for which these neighbors rely on v_i, it must be that $|\pi_e| < |\pi_b|$.

connection between two interfaces will be collision free, and in this case, the conservative approximation of the optimal path length is the true optimal path length. Furthermore, because this path is collision-free, the smoothing operation can simply add $\mathbb{L}(q_{i-1}, q_i)$ as a dense path.

From this reasoning, cases 1 and 2 from the previous proof still hold for `SPARS2`. Case 3 also holds for both algorithms without loss of generality, as it is a property inherent to the planning structure and does not reason about optimal path information. □

Given the above sequence of lemmas, it is then possible to combine them and argue the following theorem:

Theorem 5 (Asymptotic Near-Optimality w/Additive Cost)**.** *As M goes to infinity:* $\forall\ q_0, q_m \in \mathcal{C}^{\text{free}} : |\pi_S(q_0, q_m)| < t \cdot |\pi^*_{\delta_n}(q_0, q_m)| + 4 \cdot \Delta$.

4.3.3 Rate of Node Addition

An important concern with sparse roadmap spanners is whether they will be adding a significant number of nodes in the final data structure. An additional assumption relates to the sampling process used by the algorithm, which is necessary for showing that paths will not be infinitely added to G_S.

Assumption 2. *No sample q is within distance cl from obstacles. No roadmap node $v \in V_S$ is within cl-distance from obstacles.*

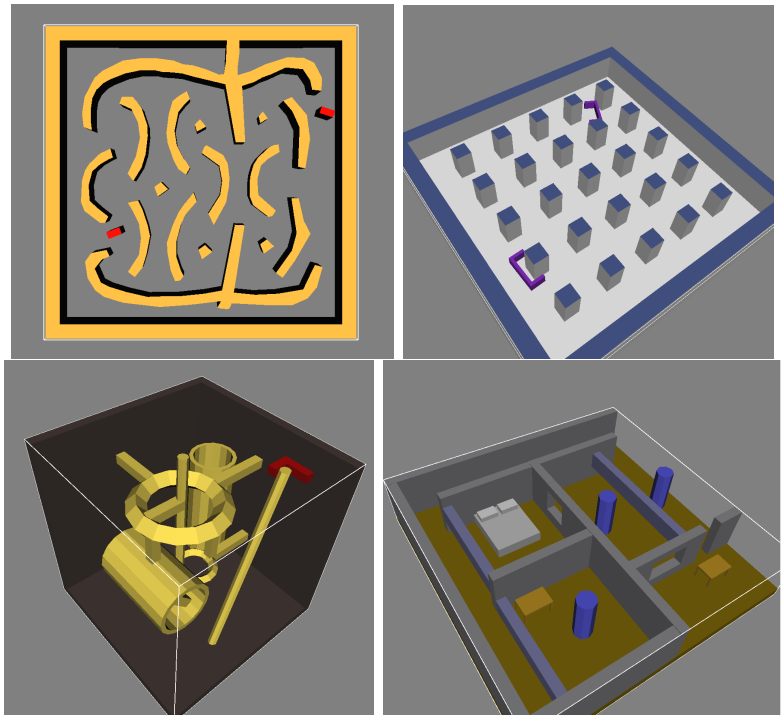

Figure 4.10: The four environments used for benchmarking the algorithms. From left to right are the "Maze", "Pegs", "Abstract" and "Beam Site" environments. Planning was performed in both $SE(2)$ ("Maze","Pegs") and $SE(3)$ ("Abstract","Beam Site") on rigid body systems.

The following theorem argues that the rate of node addition decreases over time, which allows for the reasonable termination criterion the proposed framework employs in line 3 of Algorithm 2.

Theorem 6 (Rate of Node Addition)**.** *As the number of iterations increases in Algorithm 2, the probability of adding a node to* G_S *goes to* 0*.*

Proof Sketch. This proof sketch will rely on Assumption 1, as in certain poorly-behaved spaces, such as fractal spaces, it is impossible to even guarantee coverage with a finite set of samples. There are four possible reasons for promoting nodes to G_S; thus, it must be shown that all four of these criteria will eventually have no reason to add nodes to the planning structure.

Nodes for coverage are added when a sample q lies outside the visibility range of all existing nodes in V. Theorem 2 already argues that the probability of adding guards diminishes to zero as the number of iterations increases.

Nodes for connectivity are added when a sample q connects two disconnected components of the planning structure. Eventually, enough nodes for ensuring coverage will be added in G, and thus, these nodes must be connected. As it is not possible for there to be an infinite number of connected components of $\mathcal{C}^{\text{free}}$, the number of samples needed to connect such disconnected components is finite and will be eventually added. to v'.

Nodes for ensuring interfaces have edges are added when a sample reveals two nodes in the planning structure share an interface but not an edge. It has already been argued in Theorem 4 that the algorithm will connect all interfaces, thus nodes will stop being added at some point for this purpose. The reader might argue that the addition of nodes to reduce the number of interfaces, which do not share an edge, will actually generate new interfaces that do not have edges. This is in fact the case; however, it will not be the case that for all of these newly generated interfaces there will be an obstacle preventing a direct connection from v.

Nodes for ensuring path quality are added when a sample q supports an interface and reveals a path through $\mathcal{C}^{\text{free}}$ that is significantly shorter than the corresponding path in the planning structure. In the case of `SPARS`, these paths correspond to dense paths through D, while in `SPARS2`, they are reconstructed paths. When adding such a path to G, the framework will first try to smooth this path. In some cases, the path cannot be smoothed, which will result in the addition of nodes to G. It is assumed, however, that samples are drawn from the cl-interior of $\mathcal{C}^{\text{free}}$ given Assumption 2. This prevents paths from getting infinitely close to obstacles, and allows direct connections to be made between vertices once they get close enough. Therefore, both methods will eventually stop adding nodes in this fashion. □

4.3.4 Space Requirements

It is important to reason about the space requirements of the proposed `SPARS` algorithms upon construction. While the objective is to return a sparse data structure, this should not take place at the expense of significantly increasing the space requirements of the method during the construction process because of the book-keeping information

utilized by the approaches.

The next section will show that experimentally the algorithms seem to have relatively low memory requirements, at least up to six dimensions. It is well understood that as the dimensionality of a problem increases, the number of potential neighbors for a node increases exponentially. This would indicate that the bookkeeping information stored on the nodes in SPARS2 for detecting interfaces should increase significantly, which would severely limit the practicality of the method in high dimensions. The space requirements for the book-keeping information, however, is also a function of the number of nodes. Since the resulting graphs are sparse by nature and contain a small number of nodes, this does keep the space requirements for the book-keeping information relatively low. This allows to identify a conservative bound on the number of book-keepeing entries needed as $O(n \cdot \log(n)^2)$, where n is the number of nodes and each is expected to be connected to $O(\log(n))$ neighbors. This is because at worst, each of the n nodes in the graph must potentially store interface information between each pair of neighbors. This is obviously higher than the storage requirements for edges in PRM*, which is $O(n \cdot \log(n))$. Neverhtheless, the number of nodes in the sparse representation is significantly lower than in PRM*. Furthermore, the following section is going to show that the average valence of the nodes in the sparse roadmaps is actually significantly smaller than that of PRM*, which implies lower storage costs than $O(n \cdot \log(n)^2)$.

4.4 Simulations, and Experimental Validation

This section provides a series of experimental results, which validate the analysis and the practicality of the proposed sparse roadmap spanners.

4.4.1 Experimental Setup

Experiments were performed on a cluster of the Computer Science department of Rutgers University, which is composed of IBM e-server xSeries 355 machines with 2.8 GHz Intel Xeon quad-core processors and 2GB of memory each. The methods were tested using the Open Motion Planning Library (OMPL) [24] and the environments “Maze”

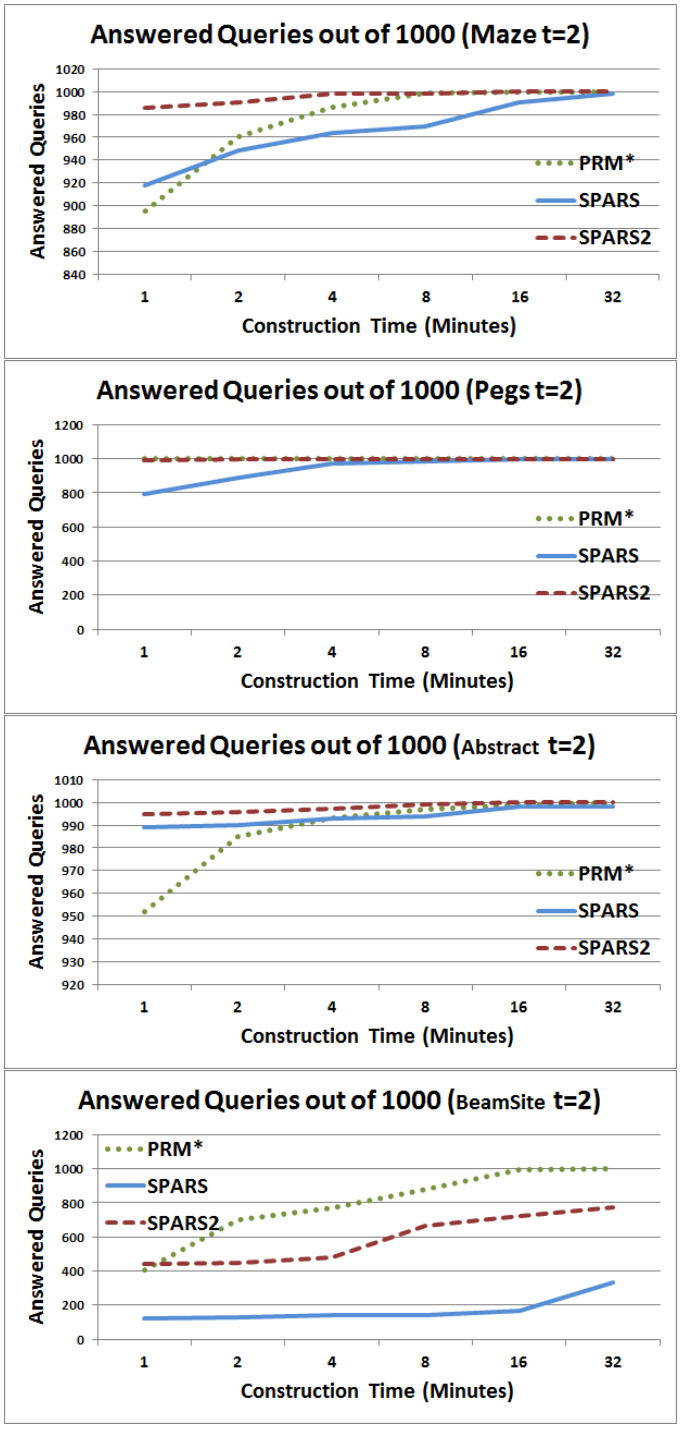

Figure 4.11: The number of successful queries out of 1000 after $\{1, 2, 4, 8, 16, 32\}$ minutes of roadmap construction time. The expected trend is that as iterations increase, all planners should be able to answer all 1000 queries.

($SE(2)$), "Pegs" ($SE(2)$), "Abstract" ($SE(3)$) and "Beam Site" ($SE(3)$), shown in Figure 4.10. Both variations of the proposed framework, SPARS (Section 4.1) and SPARS2 (Section 4.2) were evaluated, and compared against PRM*. Runs were tested with the parameters $\delta = 0.5$, $\Delta = 15$ fixed, and for varying values of the stretch factor t $(2, 3, 5, 9)$. The parameter k was selected to be two times the dimensionality of the $\mathcal{C}$, e.g., 6 for the $SE(2)$ challenges. The parameter M was removed so that it would be possible to observe the behavior of the algorithms as a function of run time. Statistics were collected after 1, 2, 4, 8, 16, and 32 minutes of construction time for the roadmap spanner methods and compared to statistics collected by PRM* for the same time intervals. An effort was made to extend these experiments up to 64 minutes but the space requirements for PRM* did not allow lengthier experiments. If not otherwise specified, the following graphs show results for a stretch factor $t = 2$. In general, the effects of the stretch factor were small in the chosen environments as the first three criteria often correspond to more than 75% of the nodes added in the roadmap. For each combination of environment, algorithm and parameters, 20 experiments were performed and their output was averaged in order to acquire the following statistics.

4.4.2 Query Success Ratio

It is first important to consider the success ratio of the algorithms over time. Given the probabilistically complete nature of the methods, it is expected that they should be able to eventually solve all problems. Figure 4.11 shows the number of successfully answered queries out of 1000 as a function of construction time. It is expected that as construction time increases, that the number of successful queries to increase to 1000. In general, the algorithms tend to converge. Interestingly, the graphs for the "Maze" and "Abstract" environments show that early in the execution of the algorithms, PRM* is unable to answer many queries, while SPARS and SPARS2 answer more. The reason for this can be that PRM* converges slower to a connected structure. This is reasonable as early on SPARS and SPARS2 quickly add nodes for coverage and connectivity purposes with an average iteration cost that is lower (edges are bounded by Δ, fewer connections are attempted than with PRM* and no interfaces are detected early on). Nevertheless, this

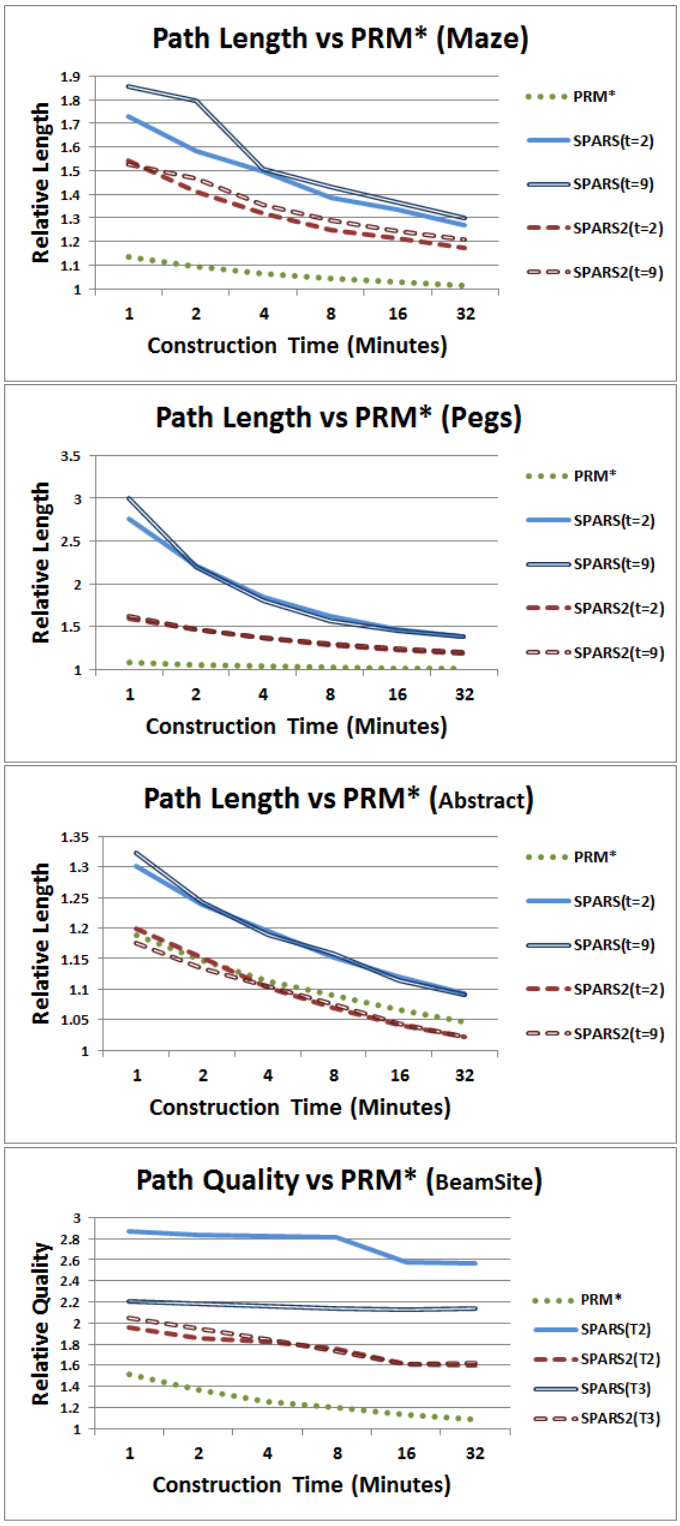

Figure 4.12: Average path quality for solved queries relative to the best paths found by PRM* after 32 minutes. As time increases, the returned path length should be decreasing. The results for the "Beam Site" environment are affected by the lower success ratio of the spanner algorithms.

is not a globally consistent behavior, as in the more complex "Beam Site" environment SPARS and SPARS2 answer fewer queries than PRM*. A possible explanation is the choice of parameters δ and Δ, as the effects of these parameters are still not clear. The path quality statistics in the rest of this section are computed only over successfully solved queries.

4.4.3 Path Quality

The proposed methods guarantee that solutions to queries are within a bound of the optimal solution to these queries. The graphs in Figure 4.12 show the average path quality returned by each of the planners relative to the best paths returned by PRM* after 32 minutes of execution. In these graphs, a value of 1.0 represents a path that has the same cost as the best PRM* path. The PRM* paths asymptotically converge to the optimum, so they are used as a near approximation of the true optimal path cost. Note that the average PRM* relative length after 32 minutes may still be higher than 1.0 and depends on the consistency of the algorithm in returning the best solution. Interestingly, even after a few minutes of execution, the returned path quality by the spanner algorithms is actually much better than what the theoretical bounds would suggest. In the $SE(2)$ environments the results are very consistent, as time increases all planners return solution paths of decreasing length with PRM* closer to the optimum, SPARS2 following and SPARS with a higher degradation. In certain environments, such as the "Abstract" environment, the spanner methods get to within 110% of the optimal length, whereas the stated bounds are relaxed to as much as 200% or 900% for these experiments. Note that for the same environment, the SPARS2 algorithm is more consistent in returning better quality paths than the PRM* itself. Given the results of Figure 4.11, the "Beam Site" environment had fewer answered queries, especially for the SPARS algorithm. As new queries are answered, their path lengths are added to the averages and in some cases the averages are shown to increase as construction time increases. Nevertheless, even in this case the relative path length is lower than the theoretical bounds.

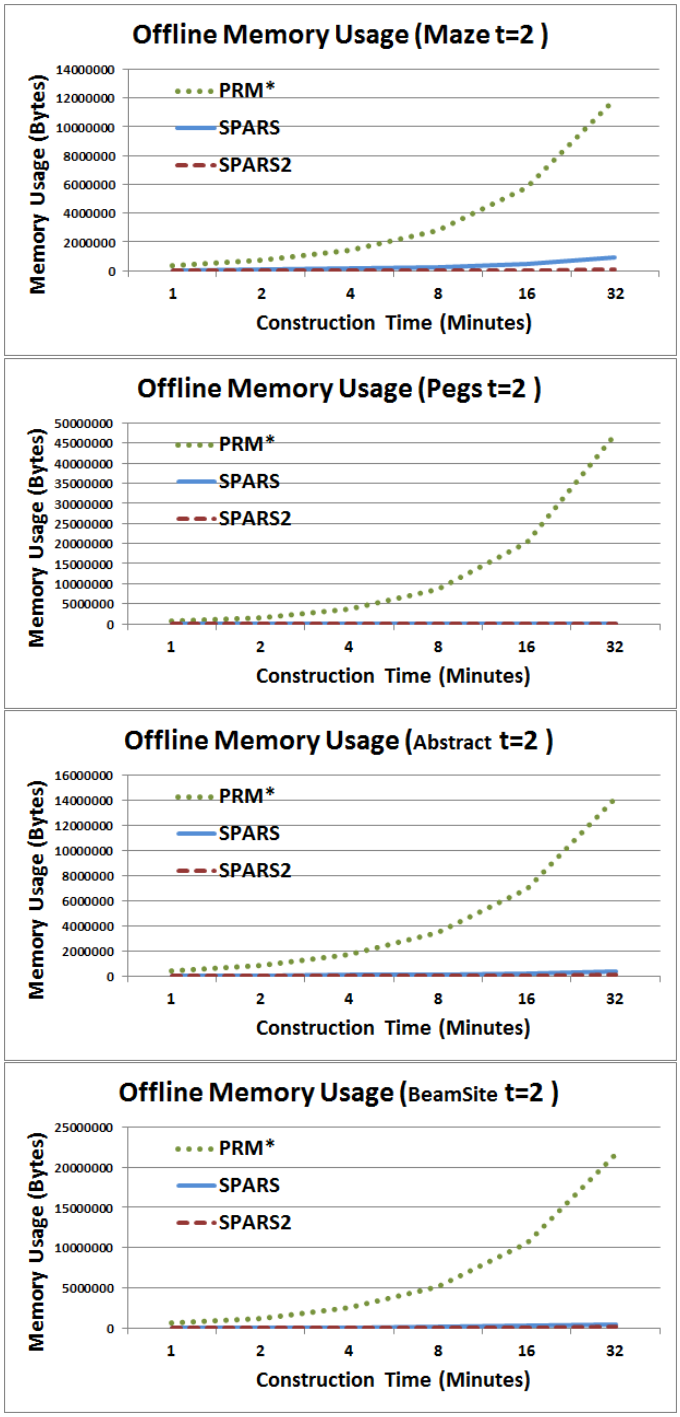

Figure 4.13: The memory usage of the three algorithms while they are preprocessing the space. Both SPARS and SPARS2 provide significant advantages in terms of memory over PRM*.

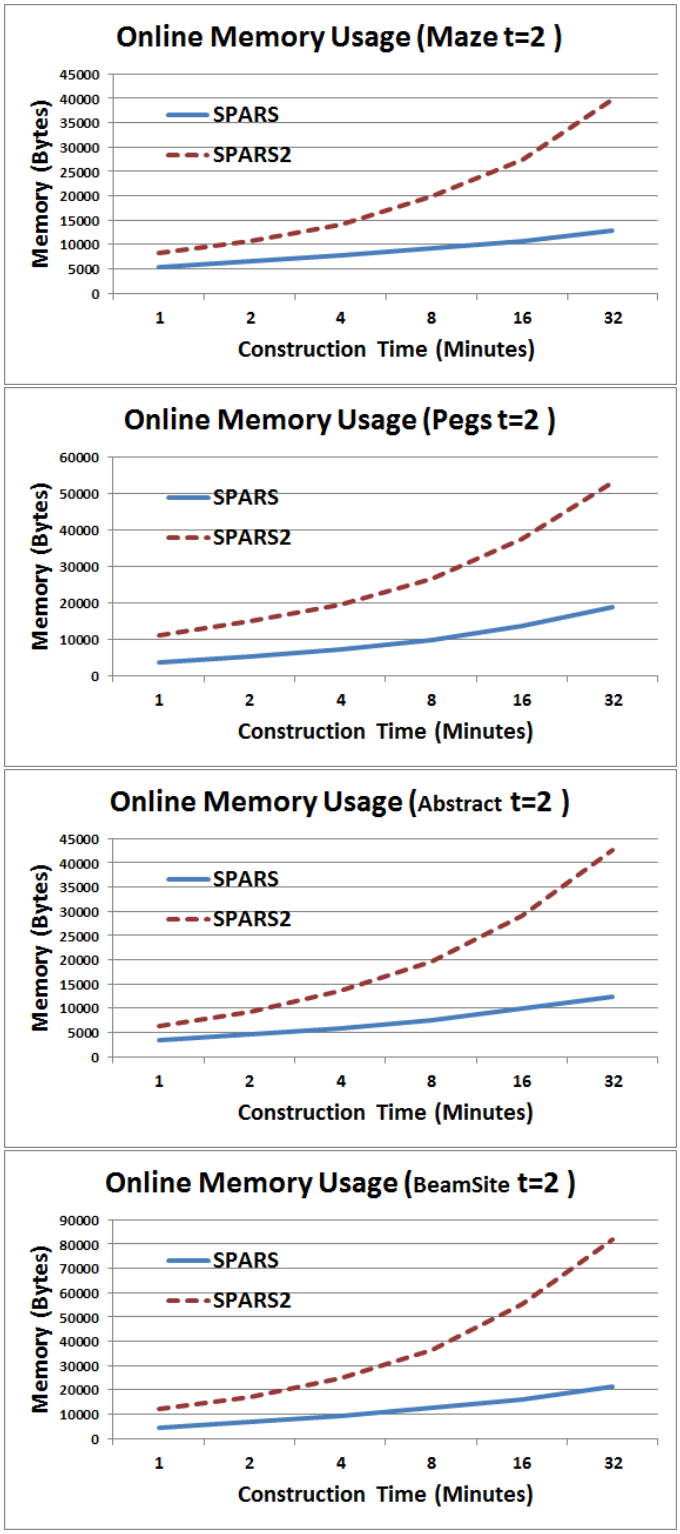

Figure 4.14: The memory required to store the resulting roadmap. PRM* is omitted as the memory requirements are the same as the offline ones (see previous figure) and significantly higher compared to the spanner methods .

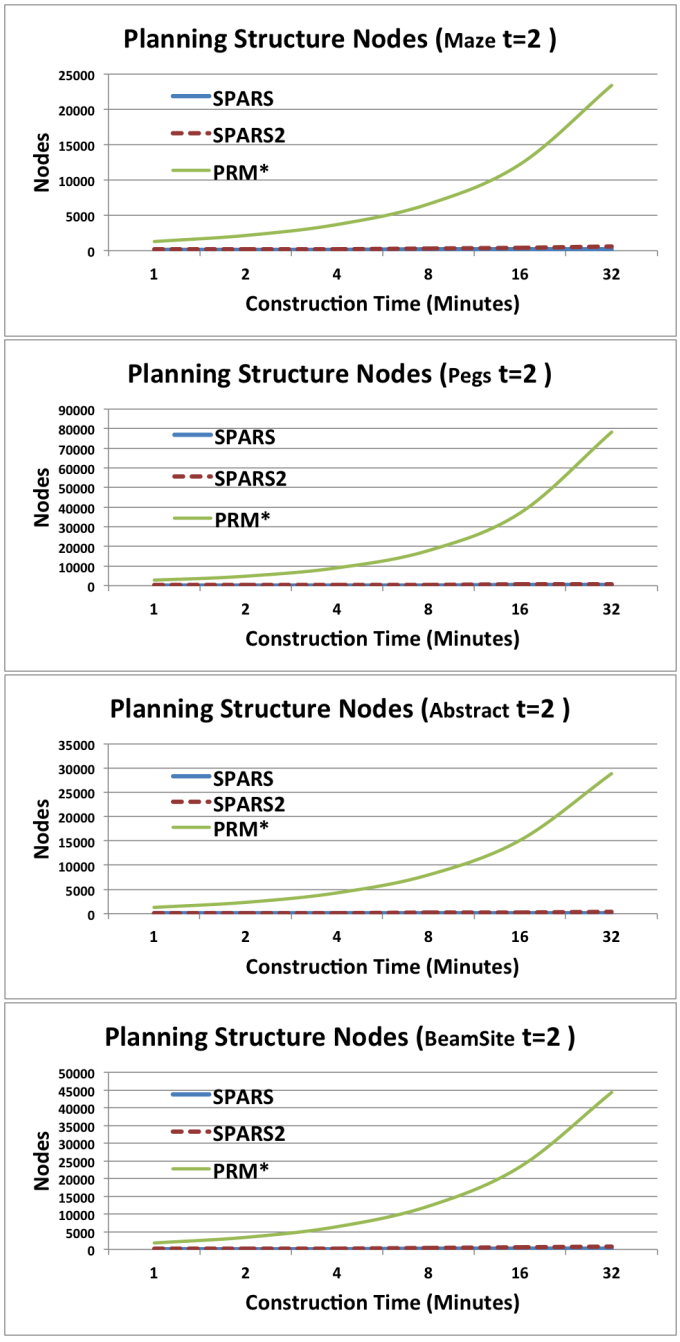

Figure 4.15: The number of nodes generated and stored for the final query structure of the various methods in the four environments.

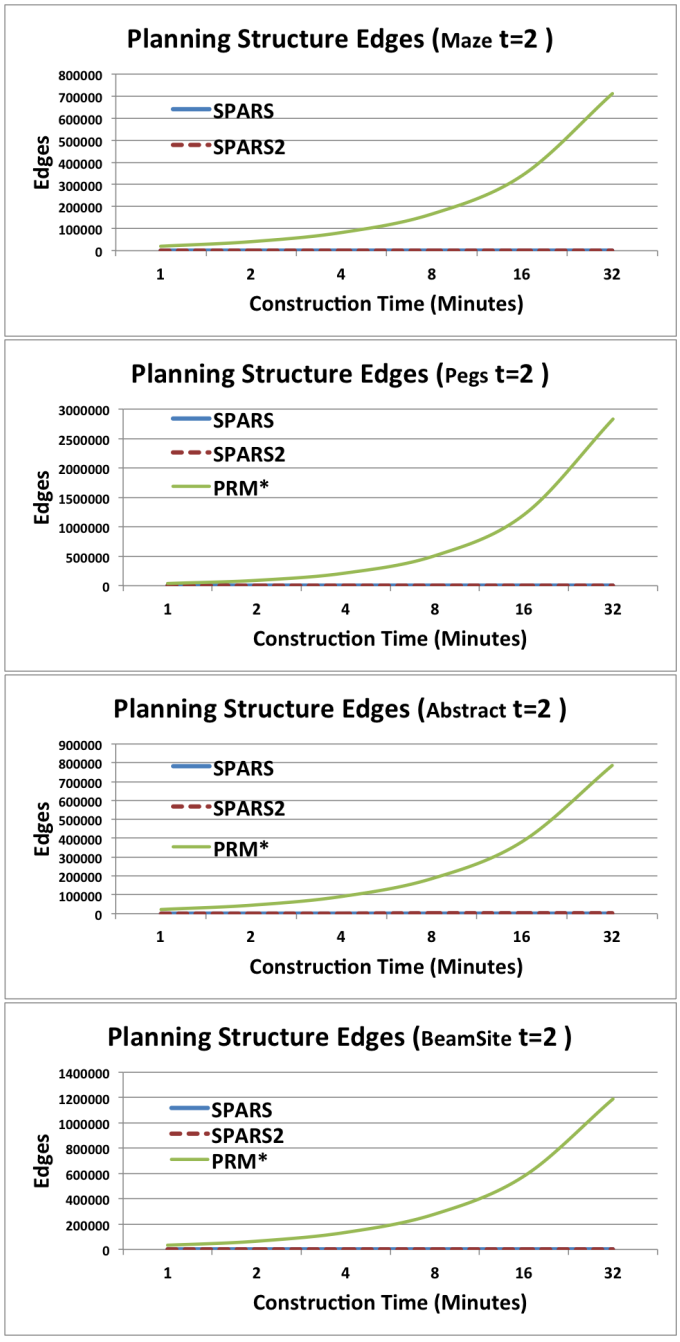

Figure 4.16: The number of edges stored for the final query structure of the various methods in the four environments.

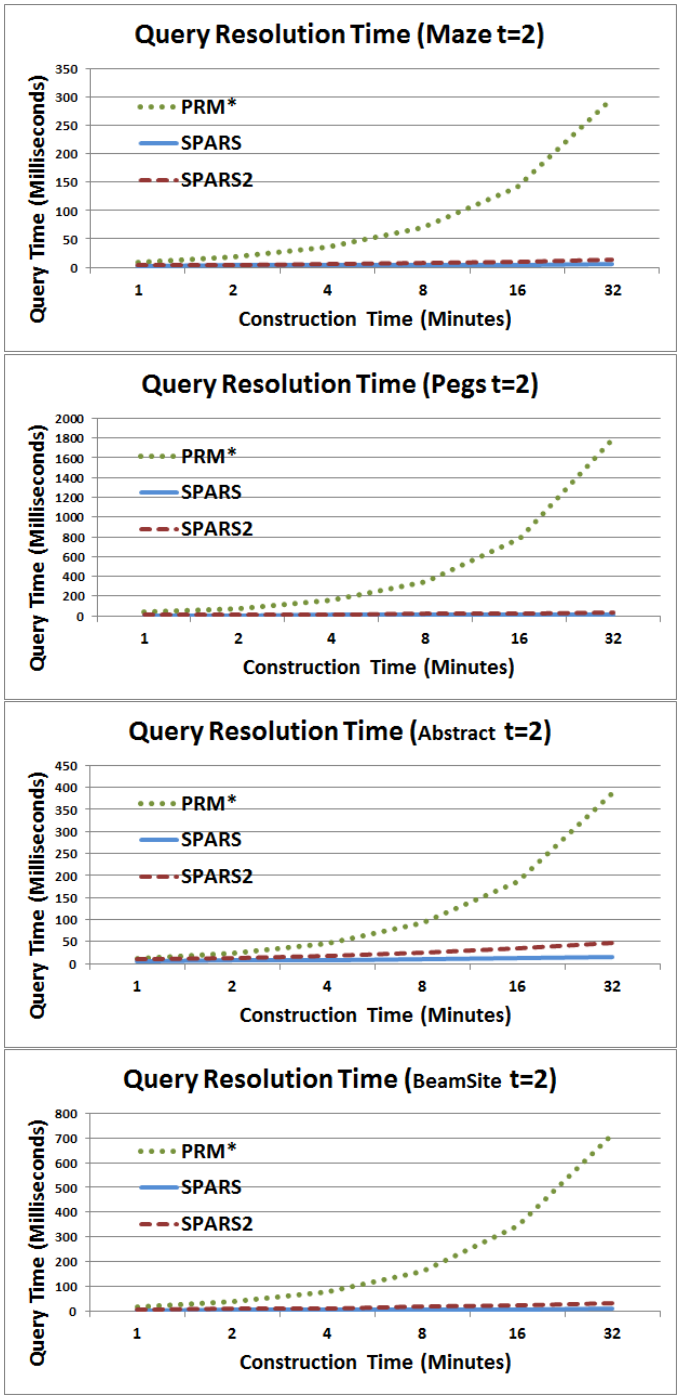

Figure 4.17: Query resolution time as a function of construction time. It is expected that since the size of the planning structures increases, query resolution time also increases.

4.4.4 Offline Memory Requirements

Figure 4.13 provides a comparison of the amount of memory used during the roadmap construction step. These results do include any extra bookkeeping information required by the SPARS and SPARS2 algorithms, as well as the cost of maintaining the sparse and/or dense graph. As expected, PRM* uses a significantly higher amount of memory. As was also expected, however, SPARS uses more memory than SPARS2 during this process, as it maintains the dense graph D. The encouraging result is the significant difference in memory used between SPARS and PRM*. Note, however, that the duration of each iteration in SPARS is much larger than each iteration of PRM*, and as such, the size of the dense graph ends up being smaller than the resulting graph from PRM* given the same amount of preprocessing time. One of the primary objectives of the SPARS2 approach was to reduce the memory footprint during the preprocessing stage while maintaining the guarantees on path quality, and in all cases, SPARS2 takes much less memory than SPARS as the following table suggests:

Environment	SPARS	SPARS2
Maze	94,0373.4	51,125.0
Pegs	104,453.3	54,406.2
Abstract	332,673.9	52,634.6
Beam Site	386,662.6	91,605.6

s

Table 4.1: Memory requirements in bytes for each spanner method after 32 minutes of computation.

4.4.5 Online Memory Requirements

After preprocessing the space, each algorithm returns a graph that is used for answering queries. Larger graphs will have higher memory requirements for storage and transmission. It is expected that as construction time increases, the size of the resulting graphs will be larger. Typically, the larger the graph, the better the path quality. The results shown in Figure 4.14 omit the graphs returned by PRM*. This is because the memory requirements are the same as the offline process, and significantly higher than the roadmaps returned by SPARS and SPARS2, which would render the comparison of

the last two methods impossible. It is important to note that in every case, SPARS2 is returning larger graphs than SPARS. Note that SPARS performs heavier computation when considering adding nodes for path quality. Multiple A^* searches are performed on an increasingly larger dense graph, and representative information must be preserved for every node in the dense graph.

4.4.6 Graph Nodes and Edges

The online memory requirements of the methods are a direct result of the number of nodes and edges in the resulting graphs, shown in Figures 4.15 and 4.16 respectively. Parallel to the online memory requirements, the SPARS and SPARS2 methods use orders of magnitude less nodes and edges in their final query structure.

4.4.7 Query Resolution Time

The amount of time to resolve a query can be very important for certain applications. The query resolution time is directly correlated to the size of the planning structure being queried. Figure 4.17 shows resulting query times for the approaches, and correlates with Figure 4.14. Both SPARS and SPARS2 return queries orders of magnitude faster than querying the roadmap returned by PRM*. SPARS2 tends to return very high quality paths relatively early; however, the query times tend to be higher than those returned by SPARS.

4.4.8 Effects of Smoothing

Smoothing can have a significant effect on the resulting path quality. Figure 4.18 shows the average smoothed solution path length relative to the best smoothed path returned by PRM*. In general, it is expected that PRM* will have less benefit from smoothing than other methods as it already returns paths that are converging to the true optimum. Smoothing provided a significant decrease in relative path length in both $SE(2)$ environments "Maze" and "Pegs". Nevertheless, the effects are less pronounced in the $SE(3)$ environments. In the "Abstract" environment, the smoothing process was advantageous for the PRM* method.

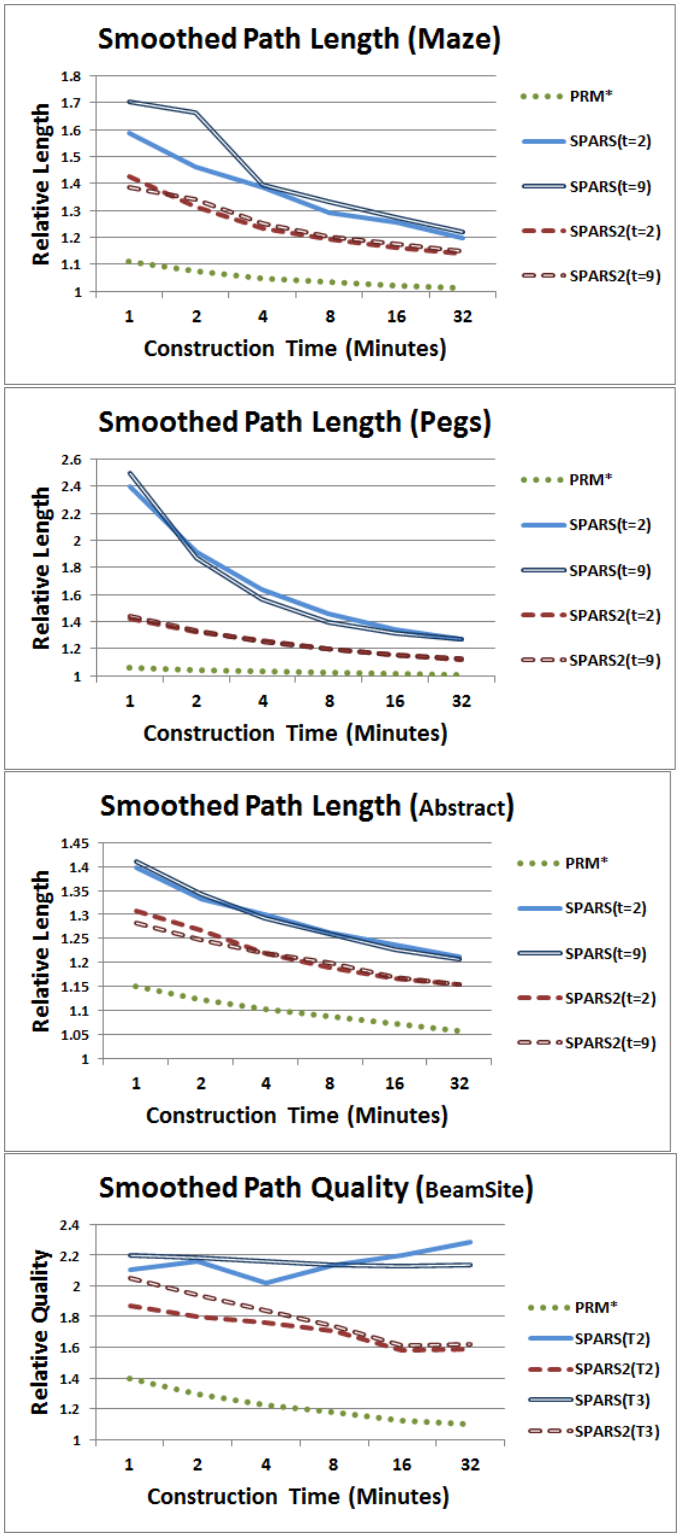

Figure 4.18: Path quality relative to the best smoothed paths from PRM*.

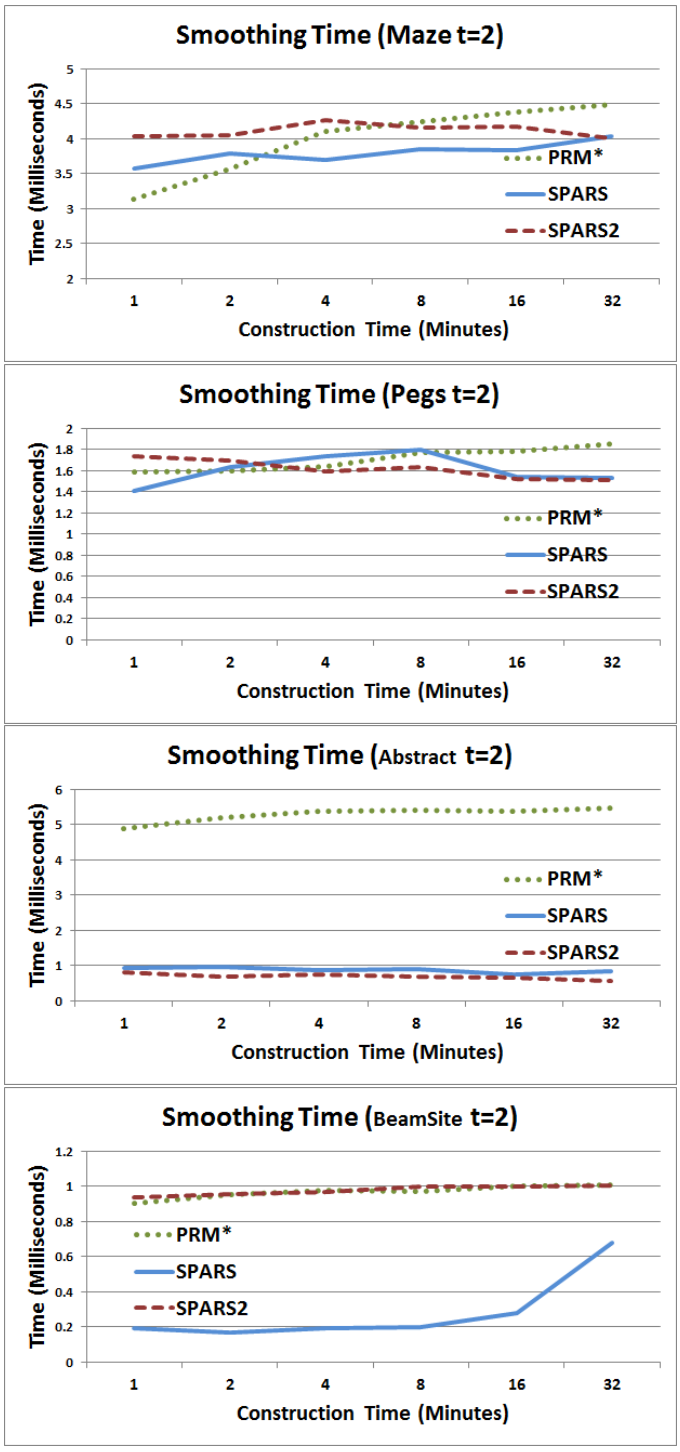

Figure 4.19: Time spent smoothing queries returned by the planning structure. Higher smoothing times represent greater reductions in path length.

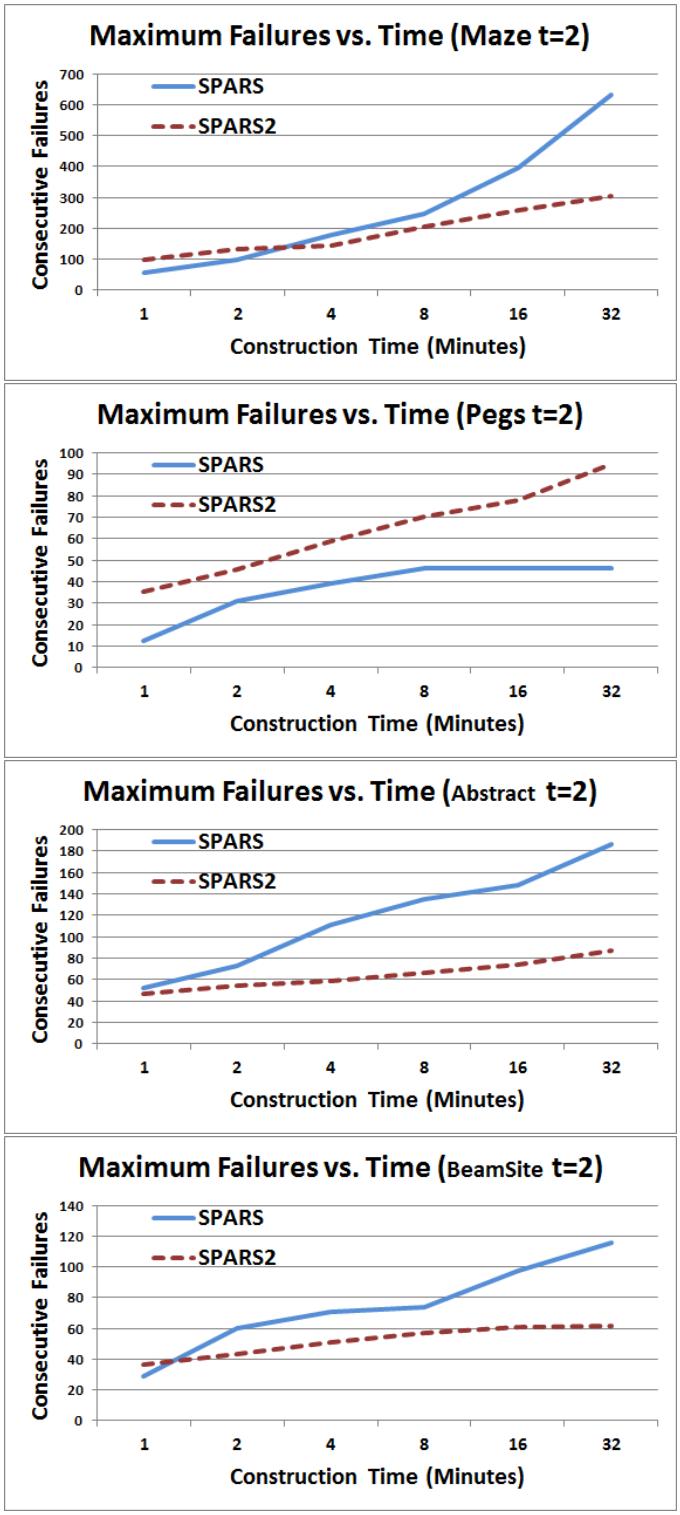

Figure 4.20: The average highest number of consecutive failures reached by the methods. If parameter M were set to a value lower than the graphs, then the algorithms would have automatically terminated.

Smoothing also introduces a time overhead as it involves collision checking, and in certain application areas this overhead can be expensive. Figure 4.19 details the amount of time spent on average for smoothing solutions to the ones shown in Figure 4.18. In general, all of the methods use roughly the same amount of time for smoothing throughout their execution. The smoothing time is also reflective of how much the path is able to improve relative to the original path cost. In the "Abstract" environment, PRM* is shown to have much larger smoothing time than the other methods. This suggests that the paths returned in this environment could be improved significantly by smoothing. This is why in Figure 4.18, the relative path quality shown for SPARS and SPARS2 appear worse than before smoothing. In the case of the "Beam Site" environment, the smoothing time of paths returned by SPARS are very small, suggesting that the smoothing was unable to refine these paths very much. This suggests that it is returning poor quality paths in homotopic classes different than the optimum one. It is apparent that the algorithm was still in the process of converging.

4.4.9 Maximum Consecutive Failures

The approach proposes an automated stopping criterion in the form of a threshold for the maximum number of consecutive failures to add a node to the planning structure. Figure 4.20 shows the rate of growth of the average maximum consecutive failures reached by the methods as they run. As expected and as is desirable, the number of maximum consecutive failures increases over time. In most cases, SPARS reaches a higher number of consecutive failures than SPARS2. A possible explanation is that it is more difficult for the method to identify the need to add dense paths using the dense graph than through a focused sampling process. This correlates with the fact that SPARS returns smaller planning structures than SPARS2.

4.4.10 Average Node Valence

An interesting aspect of the resulting planning structures to study is the average valence of the nodes in the graph, which relates to the sparsity of the resulting data structure. PRM* requires a certain number of attempts to connect to neighbors, which is a function

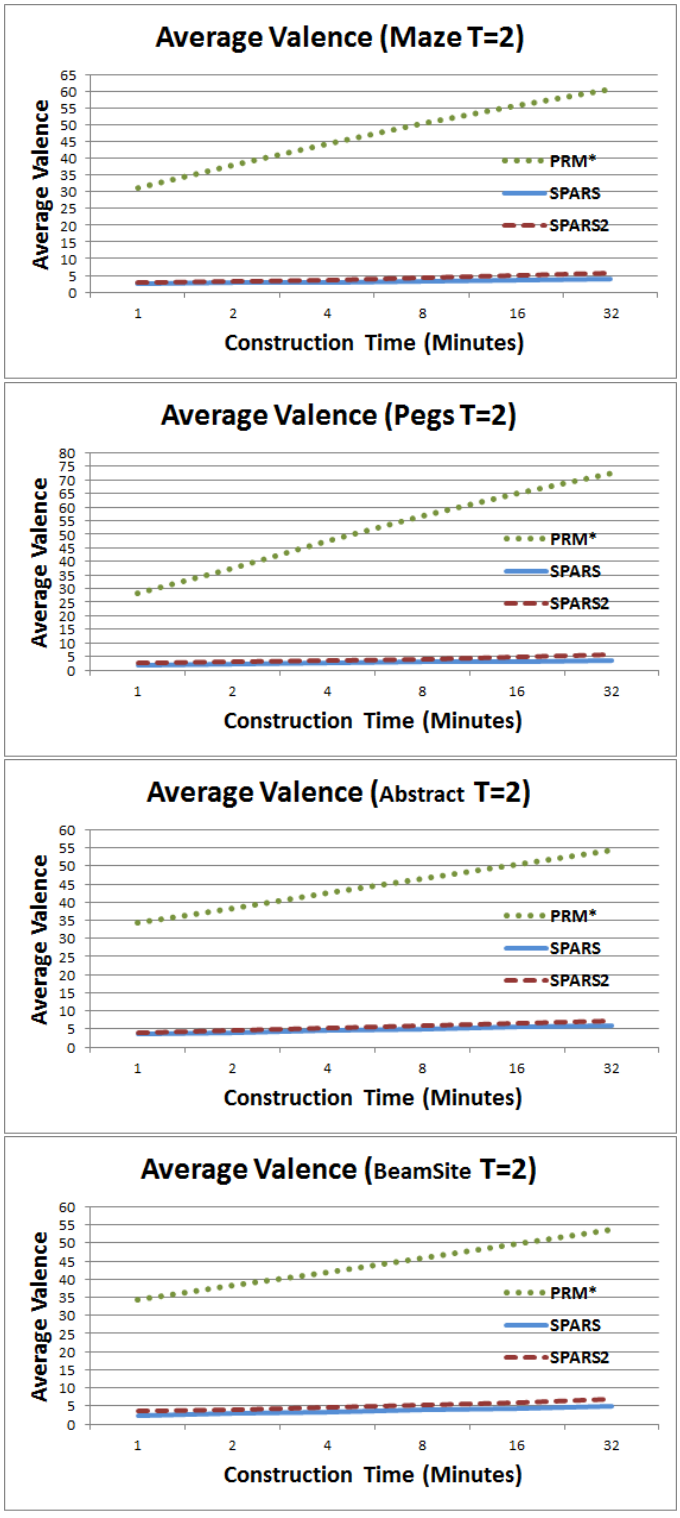

Figure 4.21: The average valence for the nodes in the resulting planning structure at various points during construction.

of the number of nodes in the roadmap, which is reflected in the result. Figure 4.21 shows how the average valence increases for the methods over time and shows that PRM* builds a much denser graph compared to both SPARS and SPARS2.

4.4.11 Online Memory Use vs. Path Quality

A direct correlation can be shown between the size of the resulting planning structure and the path quality that is returned by the structure. Figure 4.22 shows this relationship for SPARS and SPARS2 for stretch factors $t = 2, 9$ at 32 minutes of construction time. The intuition is that larger structures should return better quality paths in general, as they are a closer approximation of the underlying exhaustive graph of $\mathcal{C}^{\text{free}}$. The results, especially in the "Maze" environment, demonstrate this trade-off between memory use and path quality. Note that if these graphs were to include the PRM* algorithm, the corresponding data point would appear orders of magnitude higher on the y axis. For example, Figure 4.23, shows the online memory use over time for the methods, including PRM*. Note that the SPARS techniques do not necessarily reach the same level of path quality as PRM*, with the "Abstract" environment being the exception. Note however that when SPARS and SPARS2 do approach the same relative path length, the methods use far less memory than PRM*.

4.4.12 Query Time vs. Path Quality

Considering the correlation already drawn between resulting planning structure size and query resolution time, it is expected that there would also be a correlation between query time and path quality given the results shown in the previous subsection. The results in Figure 4.24 show this correlation, which is very similar to that provided in Figure 4.22.

4.4.13 Problem of Increasing Complexity

It is not straightforward from the above experiments to determine how well the SPARS methods extend to more difficult problem instances. A straightforward method for increasing problem difficulty is to increase the size of the robot (or equivalently grow

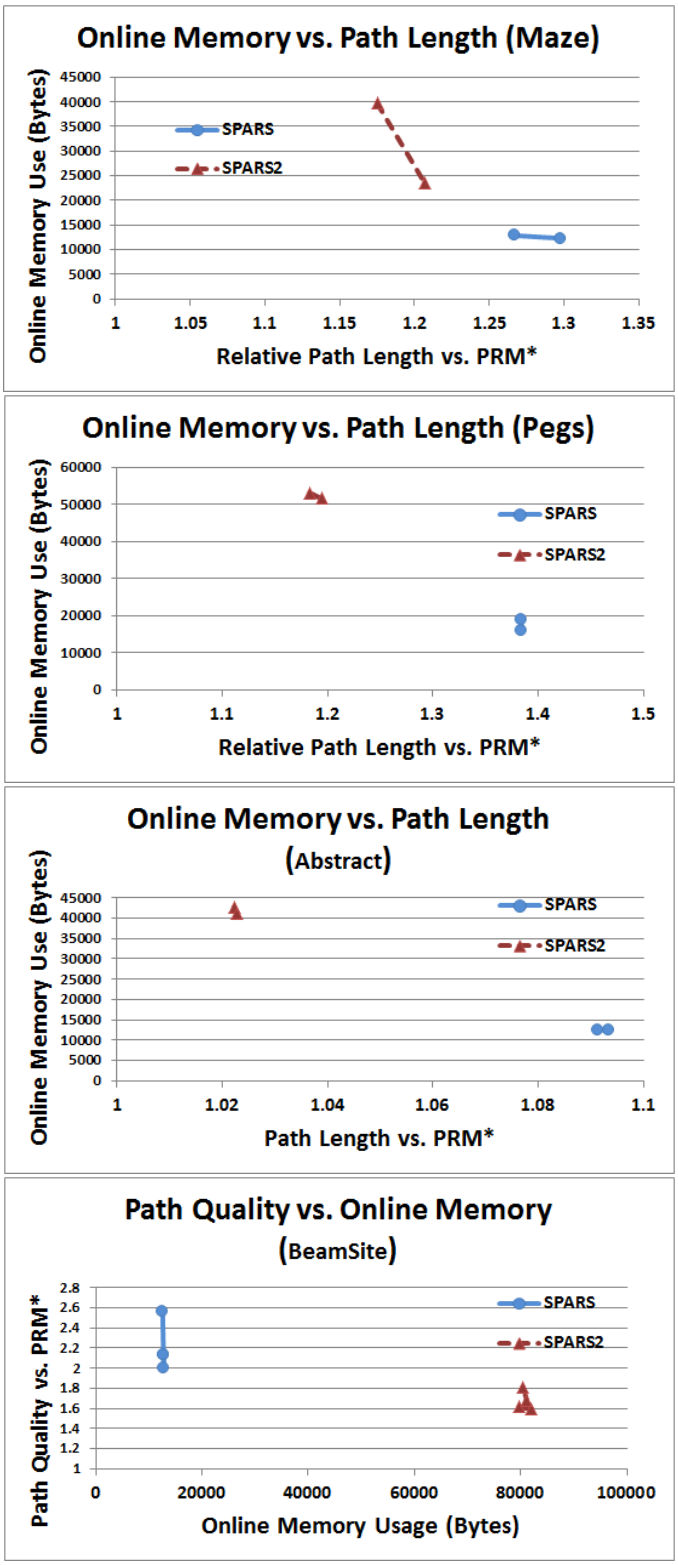

Figure 4.22: A comparison of online memory usage relative to the average returned path quality for the resulting planning structures. Larger structures should return better quality paths. Data points for PRM* are omitted for scaling reasons, but are given as $(1.0140, 11946512.8)$ (Maze), $(1.0080, 47172614.4)$ (Pegs), $(1.0467, 14198567.2)$ (Abstract), and $(1.0852, 21512684.8)$ (BeamSite).

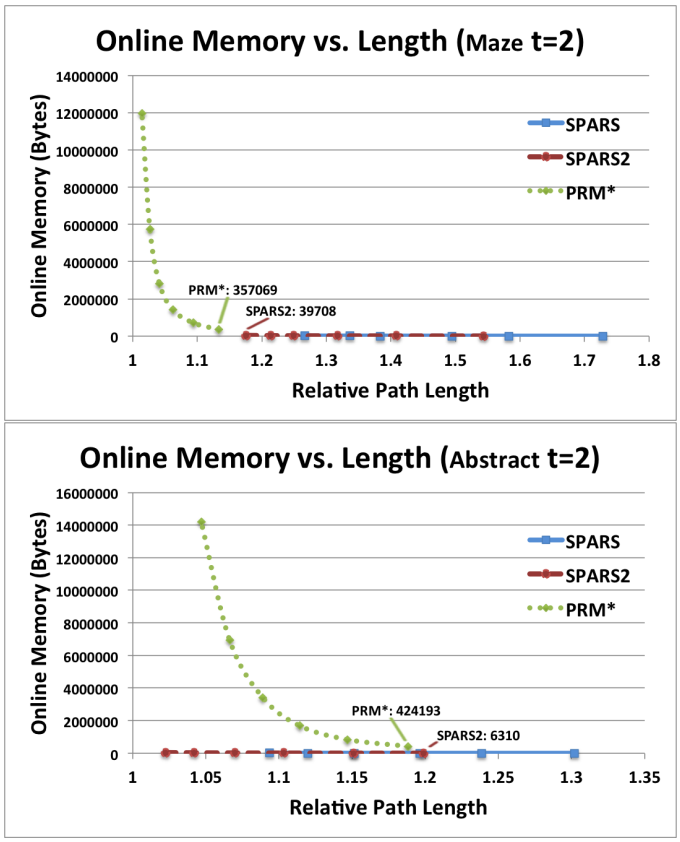

Figure 4.23: A comparison of online memory used against returned path quality through time for stretch factor $t = 2$. Values for the online memory use are highlighted for the lowest memory cost PRM* graph and it's closest neighbor from the SPARS algorithms.

the obstacles), so as to create a more constrained free space. This section examines the "Abstract" environment using an increasingly larger L-shaped robot. Figures 4.25 - 4.28 show interesting aspects of the methods when increasing problem difficulty.

Figure 4.25 shows the relative path length of SPARS and SPARS2 compared to PRM*. The figures indicate that as the problem difficulty increases, both the SPARS and SPARS2 methods show an increase in path length relative to PRM*. Figure 4.26 shows that the offline memory requirements in more complex environments stays relatively the same, with the SPARS methods using slightly more memory and PRM* using slightly less memory as problem complexity increases. As the problem difficulty increases, there are fewer collision-free configurations for PRM* to include in the roadmap. The sparse roadmaps need to include a higher number of nodes to solve the problem. Nevertheless, the difference is not significant over the various problem instances.

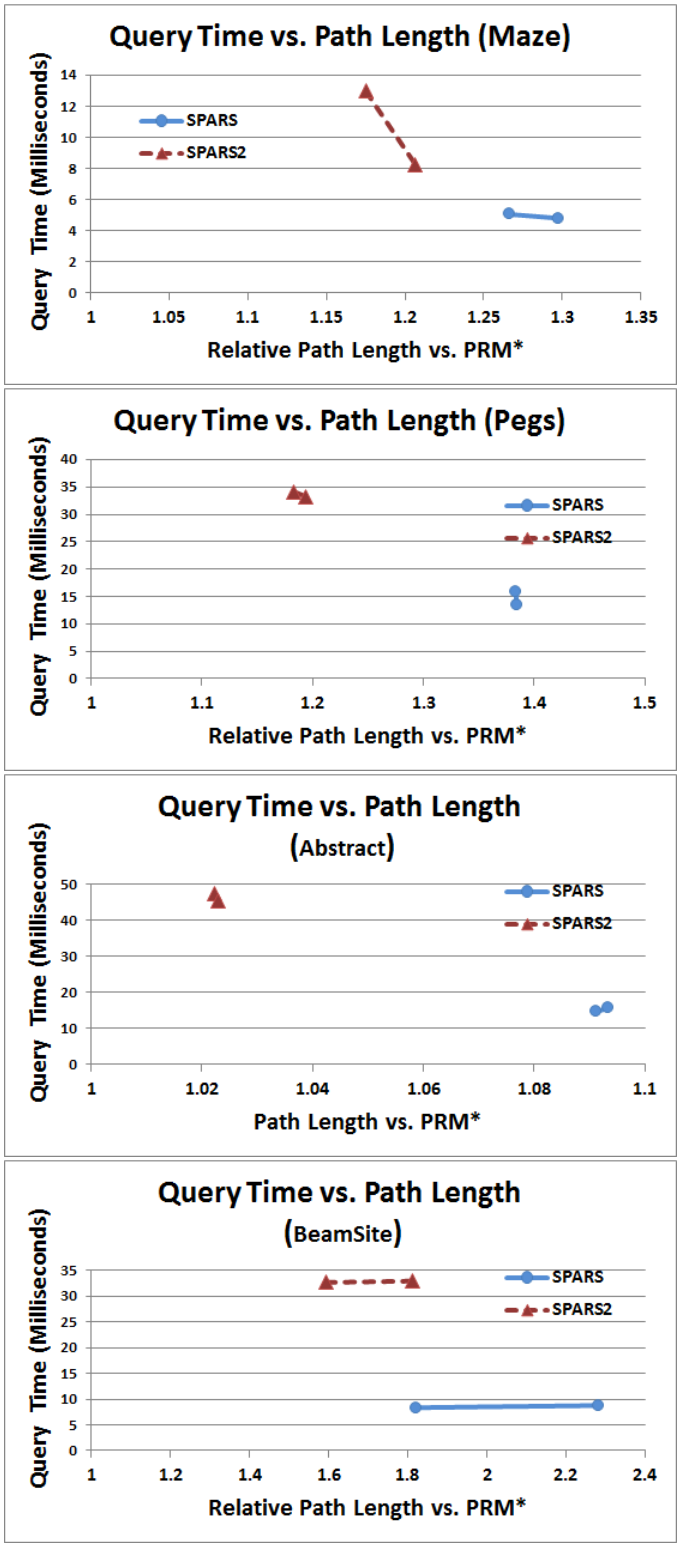

Figure 4.24: A comparison of query time against average path quality. It is expected that paths of high quality take longer to query than those of low quality.

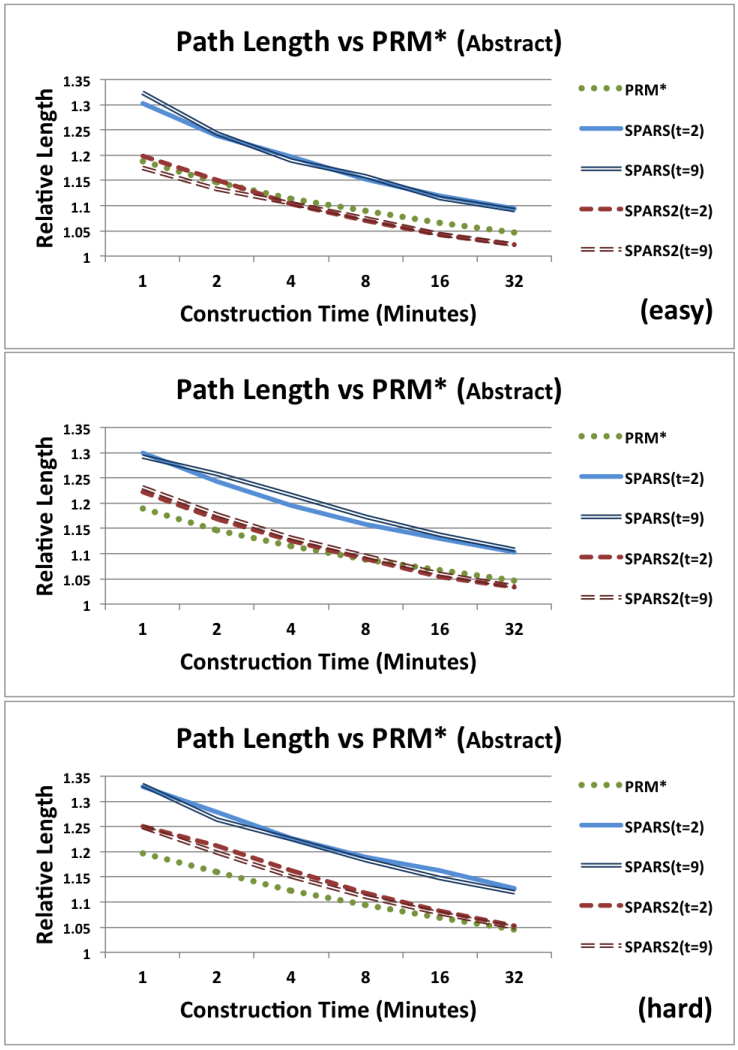

Figure 4.25: Relative path length to best path found for increasing problem difficulty from top to bottom. It is surprising that SPARS2 may even return better path quality than PRM* in easy problems. For harder instances, PRM* will be able to return better paths for longer construction times as expected.

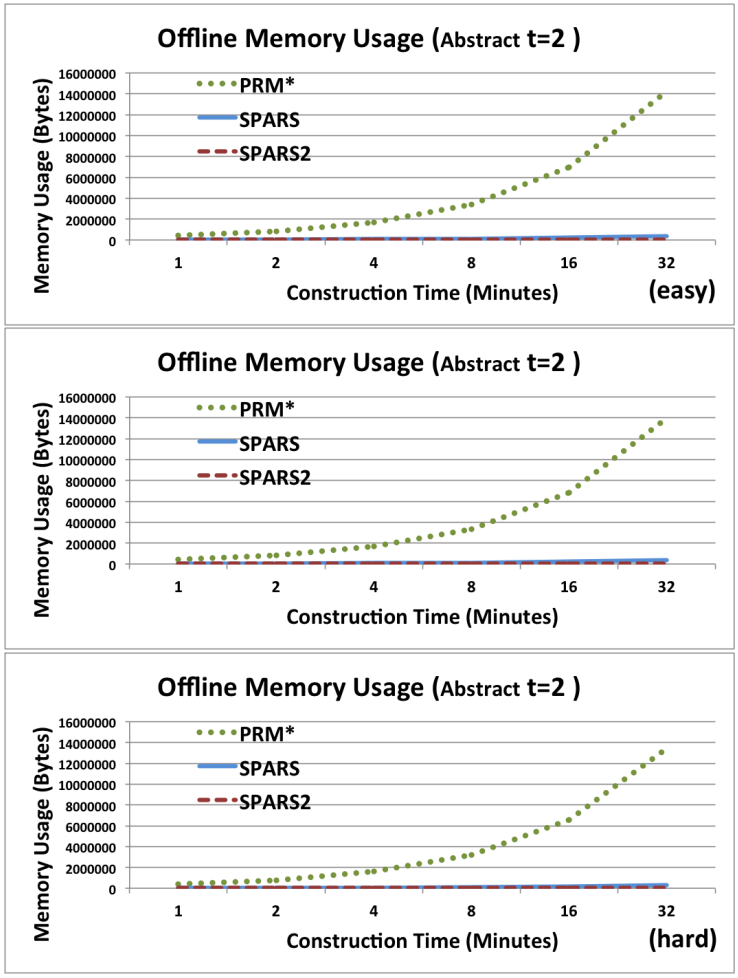

Figure 4.26: Memory usage during offline preprocessing for the various methods for problems of increasing difficulty from left to right. Memory usage remains relatively the same, though PRM* reduces its memory requirements, where SPARS and SPARS2 increase their memory requirements. Both results are expected.

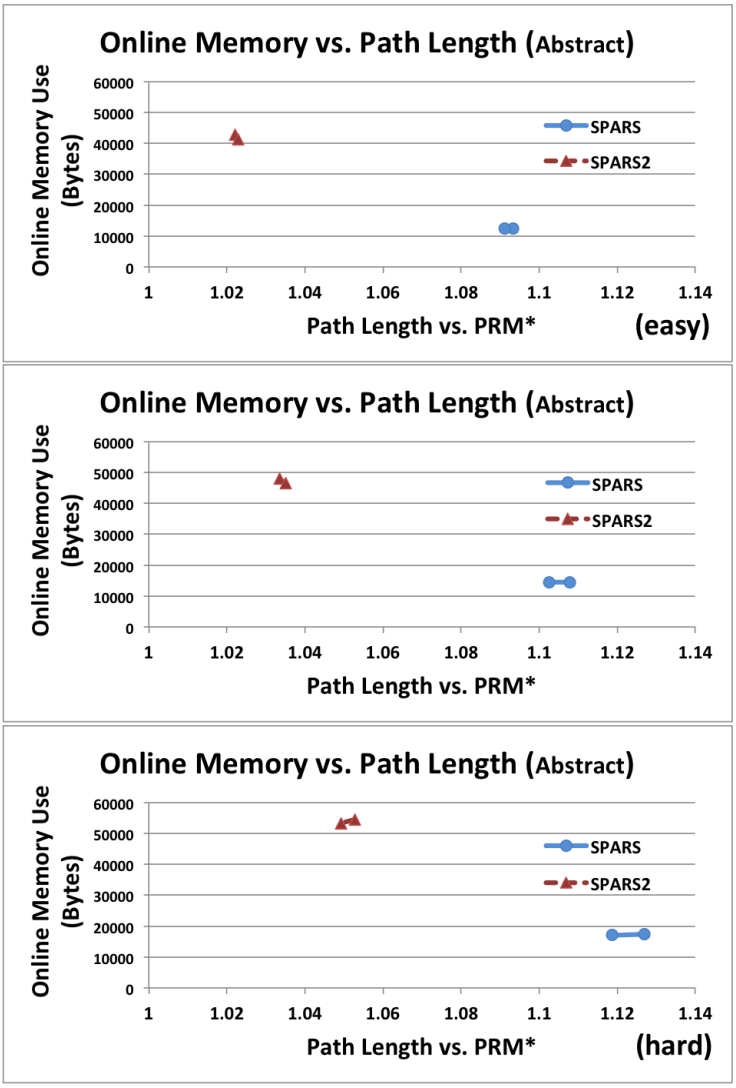

Figure 4.27: Trade-off between online memory usage and path length for the various methods for problems of increasing difficulty from left to right. Both memory usage and path length increase as the problem becomes more difficult. Data points for PRM* are omitted due to scaling issues. The data points for PRM* for the graphs from left to right are $(1.0467, 14198567.2)$, $(1.0460, 13932344)$, and $(1.0453, 13426218.4)$.

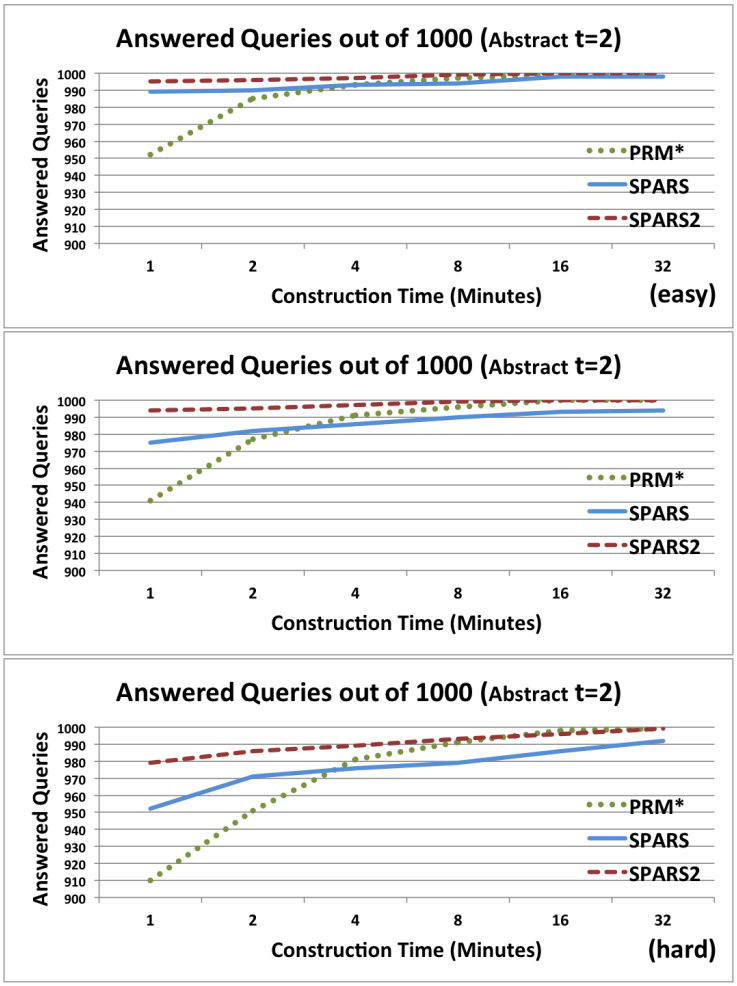

Figure 4.28: Successfully answered queries for the various methods for problems of increasing difficulty from left to right. These graphs indicate the increase in difficulty for solving the problem. In all cases, PRM* has the worst performance early on and then converges to the success ratio of SPARS2.

Figure 4.27 shows the trade-off between memory use and path quality for increasingly difficult problem instances. The trend shows the data points moving up and to the right, indicating that more difficult problem instances require more memory while also yielding lower-quality paths. Finally, Figure 4.28 indicates how many queries are answered for varying difficulty of problems. As expected, the number of answered queries decreases for more difficult problems. An interesting data point is the intersection between the PRM* graph and those of SPARS and SPARS2 methods. These data indicate that the point of intersection may be coming later for more difficult problems, i.e., it takes longer for PRM* to solve a larger percentage of queries.

4.4.14 Types of Nodes Added over Time

Both the SPARS and SPARS2 methods add nodes to the graph only if they satisfy one out of four criteria. It is interesting to see what criteria are satisfied through the run of the algorithm. Figure 4.29 shows averaged data for nodes added to SPARS2 over time. Note that the algorithm starts by quickly covering and connecting the space, followed by a reduction in node addition for these purposes as the sampling process works towards enough sample saturation to begin detecting interfaces.

One question was whether the fourth criterion, which relates to path quality and the spanner property, contributes significantly to the number of nodes added to the roadmap. The graphs in Figure 4.29 indicate that this is indeed the case. Were these types of nodes in fact rarely added to the graph, a simplified version of the algorithm that omits this final, complicated criterion could be potentially sufficient to provide the desired behavior, i.e., a sparse representation that can quickly answer queries with path of sufficient quality. This simplified algorithm would not be providing the desired theoretical properties, but potentially it could still be a practical solution as it would reduce the amount of bookkeeping. The indication, however, from these graphs is that because nodes for upholding the spanner property are a significant percentage of the total number of nodes, removing this criterion would significantly affect path quality.

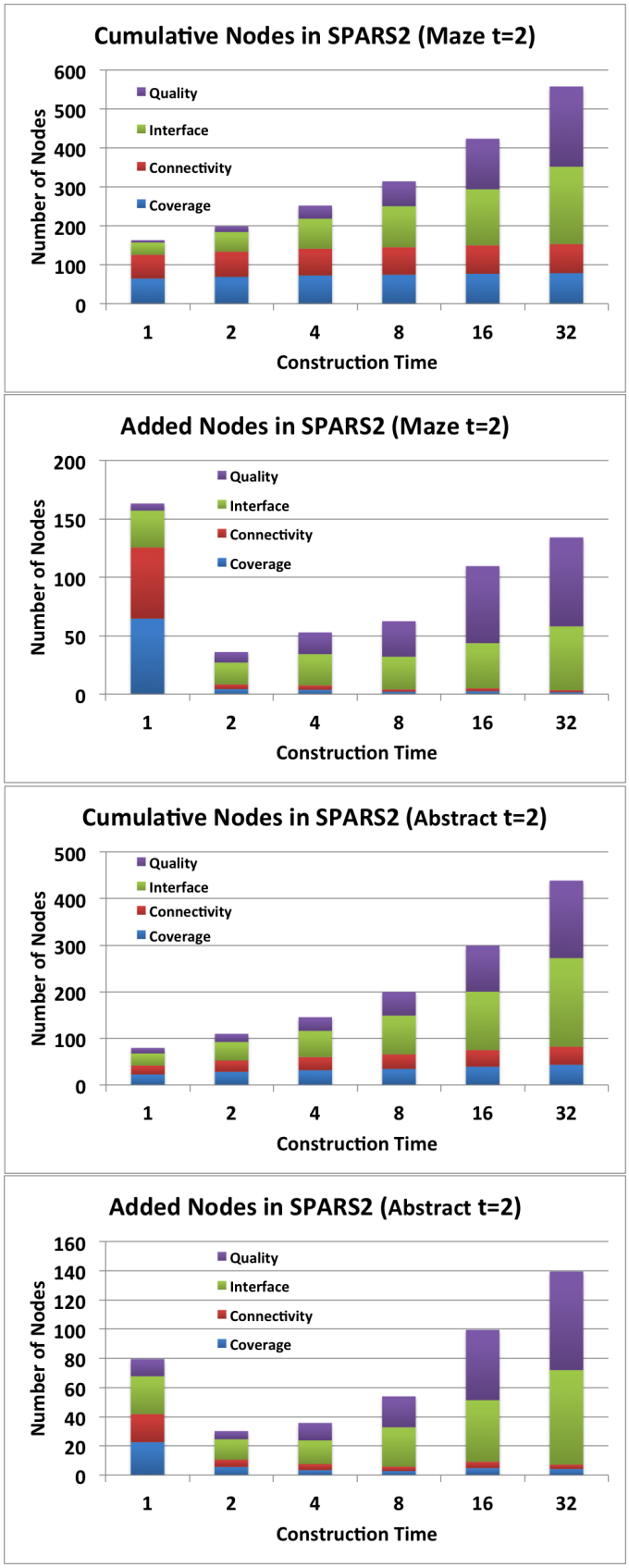

Figure 4.29: Average nodes added due to each criterion in the SPARS2 algorithm for the Maze and Abstract environments. Left: cumulative node totals at given time intervals; Right: node additions between intervals.

4.5 Discussion

A framework for generating sparse roadmap spanners is given in this chapter as a way to solve path planning problems in continuous configuration spaces using compact data structures, while providing the property of asymptotic near-optimality. Two variants of this framework are highlighted, `SPARS` and `SPARS2`. The resulting planning structures from both methods are orders of magnitude sparser and smaller than the corresponding asymptotically optimal structures, while maintaining good quality paths. This results in significantly shorter query resolution times. The resulting graph spanners are shown to provide high quality paths that come much closer to optimal ones than what the theoretical bounds specify. `SPARS2` also reduces memory requirements upon construction of the roadmap versus `SPARS` while returning even better quality paths at the cost of a small increase in the size of the final roadmap. It does so by removing the dependence on maintaining an explicit, dense graph representation of the space. Instead, it relies on properties of visibility that can be computed through localized sampling and smoothing processes to provide the same guarantees.

There are many directions to investigate into the future on this topic: (i) It is interesting to study similar near-optimality challenges in the context of graphs with directed edges, which is a necessary requirement for systems with constraints in their motion. The current line of reasoning relies on being able to directly connect configurations using a bidirectional steering method, which is often unavailable for systems with dynamics. (ii) An important step for the work is to show whether the planning structure converges to a finite-sized roadmap. Showing that the probability of adding nodes goes to 0 is a step in this direction but does not guarantee that the desired properties are provided by finite graphs. (iii) An exciting development would be to compute a confidence value representing what volume of the optimum paths in the space are covered by the planning structure after a finite time execution instead of studying the asymptotic case. This effort could lead to a stopping criterion for the algorithm that would allow the computation in finite time of probably near-optimal paths with a confidence value. (iv) As the method is able to return solutions within a bound of optimal paths, it would be

interesting to show whether the method can guarantee that it finds paths in important homotopic classes of the space, and to see how it compares to methods that attempt to identify these classes explicitly [62]. (v) It is unknown how to select parameters t and Δ to attain an expected average path degradation. Furthermore, it is interesting to evaluate how the results depend on other parameters of the algorithm, such as δ and M. (vi) Finally, many ideas can be exploited to improve computational efficiency, such as using tools that return distance to obstacles to reduce collision-checking calls and possibly quickly identifying nodes that should or should not be considered for addition to the graph.

Chapter 5

Compact and Scalable Multi-robot Motion Planning

This chapter explores methods for performing multi-robot planning in such a way that it is both computationally feasible, and ensures convergence to optimal solutions. Most multi-robot planners sacrifice such guarantees, but in this chapter, an adaptation of an efficient multirobot planning method is presented that provides such guarantees. While this method does not handle the task-planning aspects of the problem, it does provide configuration-to-configuration planning for an arbitrary set or robots.

5.1 Problem Setup and Notation

In this Chapter, the focus is planning for multiple robots in the composite configuration space ($\mathcal{C}$-space) of those arms where $\mathcal{C} = \prod_{i=1}^{R} \mathcal{C}_{\mathcal{A}_i}$ is the Cartesian product of each robot's $\mathcal{C}$-space. A composite configuration $Q = (q_1, \ldots, q_R) \in \mathcal{C}$ is an R-tuple of robot configurations. For two distinct robots $\mathcal{A}_i, \mathcal{A}_j$, denote by $I_i^j(q_j) \subset \mathcal{C}_{\mathcal{A}_i}$ the set of configurations where $\mathcal{A}_i$ and $\mathcal{A}_j$ collide. Then, the composite free space $\mathcal{C}^{\text{free}} \subset \mathcal{C}$ consists of configurations $Q = (q_1, \ldots, q_R)$ subject to:

- $q_i \in \mathcal{C}_{\mathcal{A}_i}^{\text{free}}$ for every $1 \leq i \leq R$;
- $q_i \notin I_i^j(q_j), q_j \notin I_j^i(q_i)$ for every $1 \leq i < j \leq R$.

Each $Q \in \mathcal{C}^{\text{free}}$ requires robots to not collide with obstacles, and each pair to not collide with each other. The composite forbidden space is defined as $\mathcal{C}^{\text{inv}} = \mathcal{C} \setminus \mathcal{C}^{\text{free}}$.

Given $S, T \in \mathcal{C}^{\text{free}}$, where $S = (s_1, \ldots, s_R), T = (t_1, \ldots, t_R)$, a *trajectory* $\Sigma : [0,1] \to \mathcal{C}^{\text{free}}$ is a continuous curve in $\mathcal{C}^{\text{free}}$, such that $\Sigma(0) = S, \Sigma(1) = T$, where the R robots move simultaneously. Σ is an R-tuple $(\sigma_1, \ldots, \sigma_R)$ of robot paths, such that $\sigma_i : [0,1] \to \mathcal{C}_{\mathcal{A}_i}^{\text{free}}$.

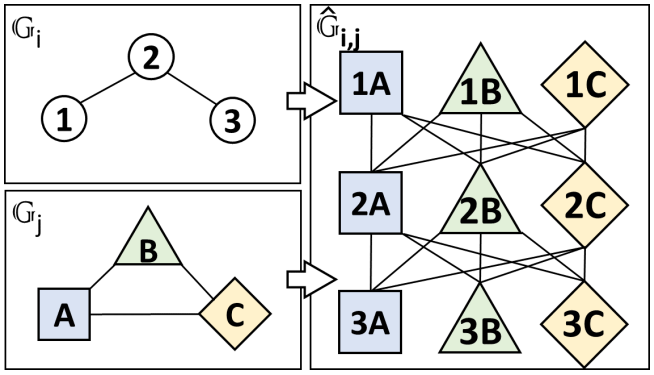

Figure 5.1: An illustration of a two-robot tensor product roadmap $\hat{G}_{i,j}$ between roadmaps G_i and G_j. Two nodes in the tensor-product roadmap share an edge if all the individual robot configurations share an edge in the individual robot roadmaps.

The objective is to find a trajectory that minimizes a *cost function* $c(\cdot)$. The analysis in this chapter assumes the cost is the sum of robot path lengths, i.e., $c(\Sigma) = \sum_{i=1}^{R} |\sigma_i|$, where $|\sigma_i|$ denotes the standard *arc length* of σ_i. The arguments also work for $\max_{i=1:R} |\sigma_i|$. [1] Section 5.3 shows sufficient conditions for the proposed dRRT* method to converge to optimal trajectories over the cost function c.

5.2 Methods for Composite Space Planning

For a fixed $n \in \mathbb{N}_+$, define for every robot $\mathcal{A}_i$ the PRM roadmap $G_i = (V_i, E_i)$ constructed over $\mathcal{C}^{\text{free}}_{\mathcal{A}_i}$, such that $|V_i| = n$ with connection radius $r(n)$. Then, $\hat{\mathbb{G}} = (\mathbb{V}, \mathbb{E}) = G_1 \times \ldots \times G_R$ is the *tensor product roadmap* in space $\mathbb{C}$ (for an illustration, see Figure 5.1). Formally, $\mathbb{V} = \{(v_1, v_2, \ldots, v_R), \forall i, v_i \in V_i\}$ is the Cartesian product of the nodes from each roadmap G_i. For two vertices $V = (v_1, \ldots, v_m) \in \mathbb{V}, V' = (v'_1, \ldots, v'_m) \in \mathbb{V}$ the edge set $\mathbb{E}$ contains edge (V, V') if for every i it is that $v_i = v'_i$ or $(v_i, v'_i) \in E_i$. [2]

As shown in Algorithm 9, dRRT* grows a tree T over $\hat{\mathbb{G}}$, rooted at the start configuration S and initializes path π_{best} (line 1). The method stores the node added each

[1] The types of distances the arguments hold are more general, but proof for alternative metrics is left as future work.

[2] Notice this slight difference from dRRT [139] so as to allow edges where some robots remain motionless while others move.

iteration V (Line 2), as part of an informed process to guide the expansion of T towards the goal. The method iteratively expands T given a time budget (Line 3), as detailed by Algorithm 10, storing the newly added node V (Line 4). After expansion, the method traces the path that connects the source S with the target T (Line 5). If such a path is found, it is stored in π_{best} if it improves upon the cost of the previous solution (Lines 6, 7). Finally, the best path found π_{best} is returned (Line 8).

Algorithm 9: `dRRT`$^*(\hat{\mathbb{G}}, S, T)$

```
1  π_best ← ∅, T.init(S);
2  V ← S;
3  while time.elapsed() < time_limit do
4  |   V ← Expand_dRRT*(Ĝ, T, V, T);
5  |   π ← Trace_Path(T, S, T);
6  |   if π ≠ ∅ and cost(π) < cost(π_best) then
7  |   |   π_best ← π;
8  |   end
9  end
10 return π_best
```

The expansion step is given in Alg. 10. The default initial step of the method is given in Lines 1-4, i.e., when no v^{last} is passed (Line 1), which corresponds to an exploration step similar to `RRT`: a random sample q^{rand} is generated in $\mathbb{C}$ (Line 2), its nearest neighbor v^{near} in T is found (Line 3) and the oracle function $\mathbb{O}_d(\cdot,\cdot)$ returns the implicit graph node v^{new} that is a neighbor of v^{near}on the implicit graph in the direction of q^{rand} (Line 4). If a v^{last}, however, is provided (Line 5)—which happens when the last iteration managed to generate a node closer to the goal relative to its parent—then v^{new} is greedily generated so as to be a neighbor of v^{last} in the direction of the goal T (Line 6).

In either case, the method next finds neighbors N, which are adjacent to v^{new} in $\hat{\mathbb{G}}$ and have also been added to T (Line 7). Among N, the best node v^{best} is chosen, for which the local path $\mathbb{L}(v^{\text{best}}, v^{\text{new}})$ is collision-free and that the total path cost to v^{new} is minimized (Line 8). If no such parent can be found (Line 9), the expansion fails and no node is returned (Line 10). Then, if v^{new} is not in T, it is added (Lines 11-13). Otherwise, if it exists, the tree is rewired so as to contain edge $(v^{\text{best}}, v^{\text{new}})$, and the

cost of v^{new}'s sub-tree (if any) is updated (Lines 14, 15). Then, for all nodes in N (Line 16), the method tests T should be rewired through v^{new} to reach this neighbor. Given that $\mathbb{L}(v^{\text{new}}, v)$ is collision-free and is of lower cost than the existing path to v (Line 17), the tree is rewired to make v^{new} be the parent of v (line 18).

Finally, if in this iteration the heuristic value of v^{new} is lower than its parent node v^{best} (line 19), the method returns v^{new} (Line 20), causing the next iteration to greedily expand v^{new}. Otherwise, $NULL$ is returned so as to do an exploration step. Note that the approach is implemented with helpful branch-and-bound pruning after an initial solution is found, though this is not reflected in the algorithmics.

v^{new} is determined via an oracle function. Using this oracle function and a simple rewiring scheme is sufficient for showing asymptotic optimality for `dRRT`* (see Section 5.3). The oracle function $\mathbb{O}_d$ for a two-robot case is illustrated in Figure 5.2. First, let $\rho(Q, Q')$ be the ray from configuration Q terminating at Q'. Then, denote $\angle_Q(Q', Q'')$ as the minimum angle between $\rho(Q, Q')$ and $\rho(Q, Q'')$. When q^{rand} is drawn in $\mathbb{C}$, its nearest neighbor v^{near} in T is found. Then, project the points q^{rand} and v^{near} into each robot space $\mathcal{C}_i$, i.e., ignore the configurations of other robots.

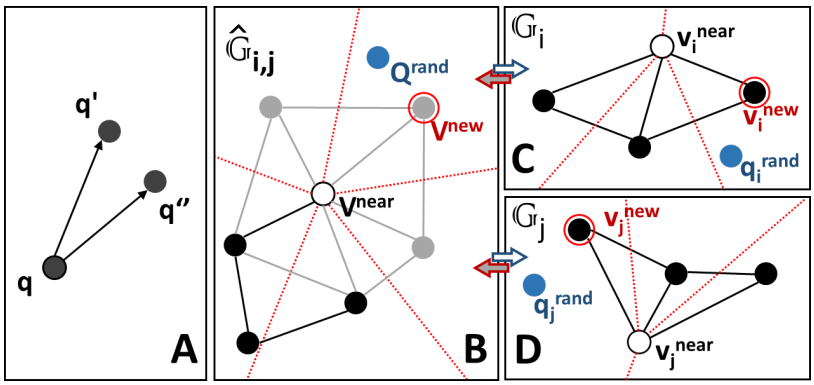

Figure 5.2: (A) The method reasons over all neighbors q' of q so as to minimize the angle $\angle_q(q', q'')$. (B) $\mathbb{O}_d(\cdot, \cdot)$ finds graph vertex $v^{\text{new}}$by minimizing angle $\angle_{v^{\text{near}}}(v^{\text{new}}, q^{\text{rand}})$. (C,D) $v^{\text{near}}$and $q^{\text{rand}}$are projected into each robot's $\mathcal{C}$-space so as to find nodes $v_{\text{i}}^{\text{new}}$ and $v_{\text{j}}^{\text{new}}$, respectively, which minimize angle $\angle_{v_{\text{i/j}}^{\text{near}}}(v_{\text{i/j}}^{\text{new}}, q_{\text{i/j}}^{\text{rand}})$.

The method separately searches the single-robot roadmaps to discover v^{new}. Denote $v^{\text{near}} = (v_1, \ldots, v_R), q^{\text{rand}} = (\tilde{q}_1, \ldots, \tilde{q}_R)$. For every robot i, let $N_i \subset V_i$ be the neighborhood of $v_i \in V_i$, and identify $v_i' = \arg\min_{v \in N_i} \angle_{v_i}(q_i^{rand}, v)$. The oracle function

returns node $v^{\text{new}} = (v'_1, \ldots, v'_R)$.

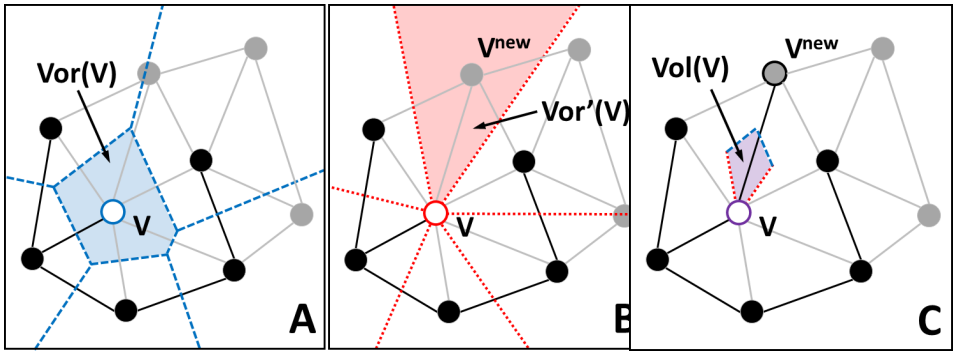

Figure 5.3: (A) The Voronoi region $\text{Vor}(V)$ of vertex V is shown where if q^{rand} is drawn, vertex V is selected for expansion. (B) When q^{rand} lies in the directional Voronoi region $\text{Vor}'(V)$, the expand step expands to v^{new}. (C) Thus, when q^{rand} is drawn within volume $\text{Vol}(V) = \text{Vor}(V) \cup \text{Vor}'(V)$, the method will generate v^{new} via V.

As in the standard RRT as well as in dRRT, the dRRT* approach has a Voronoi-bias property [100]. It is, however, slightly more involved to observe as shown in Figure 5.3. To generate an edge (V, V'), random sample q^{rand} must be drawn within the Voronoi cell of V, denoted $\text{Vor}(V)$ (A) and in the general direction of V', denoted $\text{Vor}'(V)$ (B). The intersection of these two volumes $\text{Vol}(V) = \text{Vor}(V) \cap \text{Vor}'(V)$ is the volume to be sampled generate v^{new} via v^{near}.

5.3 Asypmtotic Optimality of dRRT*

In this section, the theoretical properties of dRRT* are examined, beginning with a study of the asymptotic convergence of the implicit roadmap $\hat{\mathbb{G}}$ to containing a path in $\mathcal{C}^{\text{free}}$ whose cost converges to the optimum. Then, it is shown dRRT* eventually discovers the shortest path in $\hat{\mathbb{G}}$, and that the combination of these two facts proves the asymptotic optimality of dRRT*.

For simplicity, the analysis is restricted to the setting of robots operating in Euclidean space, i.e., $\mathcal{C}_i$ is a d-dimensional Euclidean hypercube $[0,1]^d$ for fixed $d \geq 2$. [3] Additionally, the analysis is restricted to the specific cost function of *total distance*, i.e., $|\Sigma| := \sum_{i=1}^{R} |\sigma_i|$. Discussion on lifting these restrictions is provided in Section 5.5.

[3] For simplicity, it is assumed that all the robots have the same number of degrees of freedom d.

5.3.1 Optimal Convergence of $\hat{\mathbb{G}}$

For each robot, an asymptotically optimal PRM* roadmap $\mathbb{G}_i$ is constructed having n samples and using a connection radius $r(n)$ necessary for asymptotic convergence to the optimum [69]. By the nature of sampling-based algorithms, each graph cannot converge to the true optimum with finite computation, as such a solution may have clearance of exactly 0. Instead, this work focuses on the notion of a robust optimum [4], showing that the tensor product roamdap $\hat{\mathbb{G}}$ converges to this value.

Definition 15. *A trajectory* $\Sigma : [0,1] \to \mathcal{C}^{\text{free}}$ *is* robust *if there exists a fixed* $\delta > 0$*, such that for every* $\tau \in [0,1], X \in \mathcal{C}^{\text{inv}}$ *it holds that* $\|\Sigma(\tau) - X\|_2 \geq \delta$*, where* $\|\cdot\|_2$ *denotes the standard Euclidean distance.*

Definition 16. *A value* $c > 0$ *denotes a path cost is robust if for every fixed* $\epsilon > 0$*, there exists a robust path* Σ*, such that* $|\Sigma| \leq (1+\epsilon)c$*. The* robust optimum c^**, is the infimum over all such values.*

For any fixed $n \in \mathbb{N}^+$, and a specific instance of $\hat{\mathbb{G}}$ constructed from R roadmaps, having n samples each, denote by $\Sigma^{(n)}$ the shortest path from S to T over $\hat{\mathbb{G}}$.

Definition 17. $\hat{\mathbb{G}}$ *is asymptotically optimal (AO) if for every fixed* $\epsilon > 0$ *it holds that* $|\Sigma^{(n)}| \leq (1+\epsilon)c^*$ *a.a.s.*[5]*, where the probability is over all the instantiations of* $\hat{\mathbb{G}}$ *with* n *samples for each PRM.*

Using this definition, the following theorem is proven. Recall that d denotes the dimension of a single-robot configuration space.

Theorem 7. $\hat{\mathbb{G}}$ *is AO when*

$$r(n) \geq r^*(n) = (1+\eta)2\left(\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\log n}{n}\right)^{\frac{1}{d}},$$

where η *is any constant larger than* 0.

---

[4] Note that the given definition of robust optimum is similar to that in previous work [142].

[5] *Let* $A_1, A_2, \ldots$ *be random variables in some probability space and let* B *be an event depending on* A_n*. We say that* B *occurs* asymptotically almost surely *(a.a.s.) if* $\lim_{n\to\infty} \Pr[B(A_n)] = 1$.

Remark. Note that $r^*(n)$ was developed in [65, Theorem 4.1], and guarantees AO of PRM* for a single robot. The proof technique described in that work will be one of the ingredients used to prove Theorem 7. [6]

By the definition of c^*, for any given $\epsilon > 0$ there exists a robust trajectory $\Sigma : [0,1] \to \mathcal{C}^{\text{free}}$, and a fixed $\delta > 0$, such that the cost of Σ is at most $(1 + 1/2 \cdot \epsilon)c^*$ and for every $X \in \mathcal{C}^{\text{inv}}, \tau \in [0,1]$ it holds that $\|\Sigma(\tau) - X\| \geq \delta$. Next, it is shown that $\hat{\mathbb{G}}$ contains a trajectory $\Sigma^{(n)}$ such that:

$$|\Sigma^{(n)}| \leq (1 + o(1)) \cdot |\Sigma|, \tag{5.1}$$

a.a.s.. This immediately implies that $|\Sigma^{(n)}| \leq (1 + \epsilon)c^*$, which will finish the proof of Theorem 7.

Thus, it remains to show that there exists a trajectory on $\hat{\mathbb{G}}$ that satisfies Equation 5.1 a.a.s.. As a first step, it will be shown that the robustness of $\Sigma = (\sigma_1, \ldots, \sigma_R)$ in the composite space implies robustness in the single-robot setting, i.e., robustness along σ_i.

For $\tau \in [0,1]$ define the forbidden space parameterized by τ as

$$\mathcal{C}_i^{\text{inv}}(\tau) = \mathcal{C}_i^{\text{inv}} \cup \bigcup_{j=1, j\neq i}^{R} I_i^j(\sigma_j(\tau)).$$

Claim 1. *For every robot i, $\tau \in [0,1]$, and $q_i \in \mathcal{C}_i^{\text{inv}}(\tau)$, $\|\sigma_i(\tau) - q_i\|_2 \geq \delta$.*

Proof. Fix a robot i, and fix some $\tau \in [0,1]$ and a configuration $q_i \in \mathcal{C}_i^{\text{inv}}(\tau)$. Next, define the following composite configuration

$$Q = (\sigma^1(\tau), \ldots, q_i, \ldots, \sigma^R(\tau)).$$

Note that it differs from $\Sigma(\tau)$ only in the i-th robot's configuration. By the robustness of Σ it follows that

$$\delta \leq \|\Sigma(\tau) - Q\|_2 = \left(\|\sigma_i(\tau) - q_i\|_2^2 + \sum_{j=1, j\neq i}^{R} \|\sigma_j(\tau) - \sigma_j(\tau)\|_2^2 \right)^{\frac{1}{2}} leq \|\sigma_i(\tau) - q_i\|_2.$$

□

[6] Note that $r^*(n)$ can be refined to incorporate the proportion of $\mathcal{C}_i^{\text{free}}$, which would reduce this expression.

The result of claim 1 is that the paths $\sigma_1, \ldots, \sigma_R$ are robust in the sense that there is sufficient clearance for the individual robots to not collide with each other given a fixed location of a single robot. A Lemma is derived using proof techniques from the literature [65], and it implies every G_i contains a single-robot path $\sigma_i^{(n)}$ that converges to σ_i

Lemma 15. *For every robot i, G_i constructed with n samples and a connection radius $r(n) \geq r^*(n)$ contains a path $\sigma_i^{(n)}$ with the following attributes a.a.s.:*

(i) $\sigma_i^{(n)}(0) = s_i$, $\sigma_i^{(n)}(1) = t_i$;

(ii) $|\sigma_i^{(n)}| \leq (1 + o(1))|\sigma_i|$;

(iii) $\forall q \in \mathrm{Im}(\sigma_i^{(n)})$, $\exists \tau \in [0,1]$ *s.t.* $\|q - \sigma_i(\tau)\|_2 \leq r^*(n)$.

Proof. The first property (i) follows from the fact that s_i, t_i are directly added to G_i. The rest follows from the proof of Theorem 4.1 in [65], which is applicable here since $r(n) \geq r^*(n)$. □

Lemma 15 also implies that $\hat{\mathbb{G}}$ contains a path in $\mathcal{C}$, that represents robot-to-obstacle collision-free motions, and minimizes the multi-robot metric cost. In particular, define $\Sigma^{(n)} = (\sigma_1^{(n)}, \ldots, \sigma_R^{(n)})$, where $\sigma_i^{(n)}$ are obtained from Lemma 15. Then

$$|\Sigma^{(n)}| = \sum_{i=1}^{R} |\sigma_i^{(n)}| \leq (1 + o(1)) \sum_{i=1}^{R} |\sigma_i| \leq (1 + o(1))|\Sigma|.$$

However, it is not clear whether this ensures the existence of a path where robot-robot collisions are avoided. That is, although $\mathrm{Im}(\sigma_i^{(n)}) \subset \mathcal{C}_i^{\mathrm{free}}$, it might be the case that $\mathrm{Im}(\Sigma^{(n)}) \cap \mathcal{C}^{\mathrm{inv}} \neq \emptyset$. Next it is shown that $\sigma_1^{(n)}, \ldots, \sigma_R^{(n)}$ can be reparametrized to induce a composite-space path whose image is fully contained in $\mathcal{C}^{\mathrm{free}}$, with length equivalent to $\Sigma^{(n)}$.

For each robot i, denote by $V_i = (v_i^1, \ldots, v_i^{\ell_i})$ the chain of G_i vertices traversed by $\sigma_i^{(n)}$. For every $v_i^j \in V_i$ assign a timestamp τ_i^j of the closest configuration along σ^i, i.e.,

$$\tau_i^j = \arg\min_{\tau \in [0,1]} \|v_i^j - \sigma_i(\tau)\|_2.$$

Also, define $\mathcal{T}_i = (\tau_i^1, \ldots, \tau_i^{\ell_i})$ and denote by $\mathcal{T}$ the ordered list of $\bigcup_{i=1}^{R} \mathcal{T}_i$, according to the timestamp values. Now, for every i, define a global timestamp function $TS_i : \mathcal{T} \to V_i$, which assigns to each global timestamp in $\mathcal{T}$ a single-robot configuration from V_i. It thus specifies in which vertex robot i resides at time $\tau \in \mathcal{T}$. For $\tau \in \mathcal{T}$, let j be the largest index such that $\tau_i^j \leq \tau$. Then simply assign $TS_i(\tau) = \tau_i^j$. From property (iii) in Lemma 15 and Claim 1 it follows that no robot-robot collisions are induced by the reparametrization, concluding the proof of Theorem 7.

5.3.2 Asymptotic Optimality of dRRT*

Finally, dRRT* is shown to be AO. Denote by m the time budget in Algorithm 9, i.e., the number of iterations of the loop. Denote by $\Sigma^{(n,m)}$ the solution returned by dRRT* for n and m.

Theorem 8. *If $r(n) > r^*(n)$ then for every fixed $\epsilon > 0$ it holds that*

$$\lim_{n,m\to\infty} \Pr\left[|\Sigma^{(n,m)}| \leq (1+\epsilon)c^*\right] = 1.$$

Since $\hat{\mathbb{G}}$ is AO (Theorem 7), it suffices to show that for any fixed n, and a fixed instance of $\hat{\mathbb{G}}$, defined over R PRMs with n samples each, dRRT* eventually (as m tends to infinity), finds the optimal trajectory over $\hat{\mathbb{G}}$. This can be shown using the properties of a Markov chain with absorbing states [50, Theorem 11.3]. While a full proof is omitted here, the idea is similar to what is presented in previous work [140, Theorem 3], and examined in an extended version of this manuscript [33]. By restricting the states of the Markov chain to being the graph vertices along the optimal path, setting the target vertex to be an absorbing vertex, and showing that the probability of transitioning along any edge in this path is nonzero (i.e., the probability is proportional to $\frac{\mu(\mathrm{Vol}(V_k))}{\mu(\mathcal{C}^{\mathrm{free}})} > 0$), then the probability that this process does not reach the target state along the optimal path converges to 0 as the number of dRRT* iterations tends to infinity. The final step is to show that the above statements hold when both m and n tend to ∞. A proof for this phenomenon can be found in [140, Theorem 6].

5.4 Experimental Validation

This section provides an experimental evaluation of dRRT* by demonstrating practical convergence, scalability, and applicability to dual-arm manipulation. The approach and alternatives are executed on a cluster with Intel(R) Xeon(R) CPU E5-4650 @ 2.70GHz processors, and 128GB of RAM. [7]

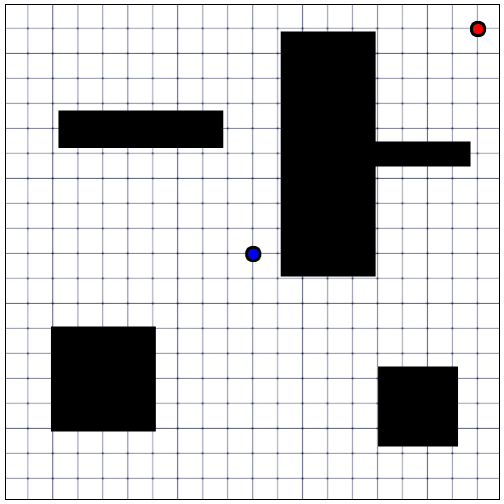

Figure 5.4: The 2D environment where the 2 disk robots operate.

2 Disk Robots among 2D Polygons: This base-case test involves 2 disks ($\mathcal{C}_i := \mathbb{R}^2$) of radius 0.2, in a 10.2×10.2 region, as in Figure 5.4. The disks have to swap positions between $(0, 0)$ and $(9, 9)$. This is a setup where it is possible to compute the explicit roadmap, which is not practical in more involved scenarios. In particular, dRRT* is tested against: a) running A* on the implicit tensor roadmap $\hat{\mathbb{G}}$ (referred to as "Implicit A*") defined over the same individual roadmaps with N nodes each as those used by dRRT*; and b) an explicitly constructed PRM* roadmap with N^2 nodes in the composite space.

Results are shown in Figure 5.5. dRRT* converges to the optimal path over $\hat{\mathbb{G}}$, similar to the one discovered by Implicit A*, while quickly finding an initial solution of high quality. Furthermore, the implicit tensor product roadmap $\hat{\mathbb{G}}$ is of comparable quality to the explicitly constructed roadmap.

Table 5.1 presents running times. dRRT* and implicit A* construct 2 N-sized roadmaps

[7] Additional data are provided in the appendices of an extended version of this paper [33] and the accompanying video.

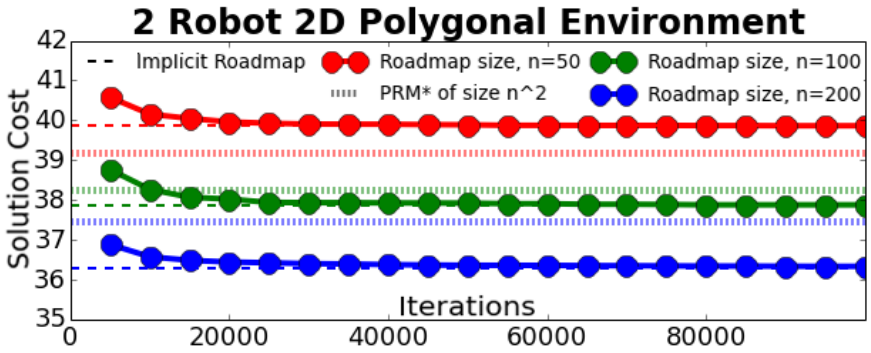

Figure 5.5: Average solution cost over iterations. Data averaged over 10 roadmap pairs. dRRT* (solid line) converges to the optimal path through $\hat{\mathbb{G}}$ (dashed line).

(row 3), which are faster to construct than the PRM* roadmap in $\mathcal{C}$ (row 1). PRM* becomes very costly as N increases. For $N = 500$, the explicit roadmap contains $250,000$ vertices, taking 1.7GB of RAM to store, which was the upper limit for the machine used. When the roadmap can be constructed, it is quicker to query (row 2). dRRT* quickly returns an initial solution (row 5), and converges within 5% of the optimum length (row 6) well before Implicit A* returns a solution as N increases (row 4). The next benchmark further emphasizes this point.

Many Disk Robots among 2D Polygons: In the same environment as above, the number of robots R is increased to evaluate scalability. Each robot starts on the perimeter of the environment and is tasked with reaching the opposite side. An $N = 50$ roadmap is constructed for every robot. It quickly becomes intractable to construct a PRM* roadmap in the composite space of many robots.

Figure 5.6 shows the inability of alternatives to compete with dRRT* in scalability. Solution costs are normalized by an optimistic estimate of the path cost for each case, which is the sum of the optimal solutions for each robot, disregarding robot-robot interactions. Implicit A* fails to return solutions even for 3 robots. Directly executing RRT* in the composite space fails to do so for $R \geq 6$. The original dRRT method (without the informed search component) starts suffering in success ratio for $R \geq 5$ and returns worse solutions than dRRT*. The average solution times for dRRT may decrease as R increases but this is due to the decreasing success ratio, i.e., dRRT begins to only succeed at easy problems.

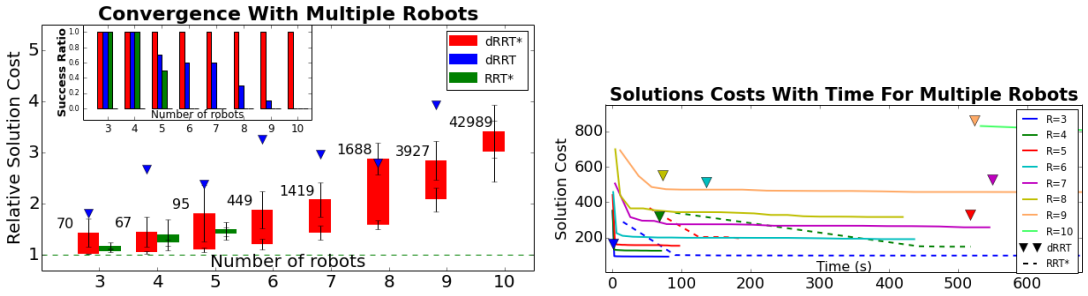

Figure 5.6: Data averaged over 10 runs. (*Top*): Relative solution cost and success ratio of dRRT*, dRRT and RRT* for increasing R. dRRT*: average iteration and variance for initial solution (top of box), and solution cost and variance after 100,000 iterations (bottom). Similar results for RRT*. Single data point for dRRT (no quality improvement after first solution). (*Bottom*): Solution costs over time.

To emphasize the lack of scalability for alternate methods, additional experiments were run in this setup using a minimal roadmap. The tests use a 9-node roadmap for each robot as illustrated in Figure 5.9. Each roadmap is constructed with slight perturbations to the nodes within the shaded regions indicated in the figure.

The data for this modified benchmark (shown in Figure 5.8) indicate that even using a very small roadmap does not allow alternate methods to scale. While the methods do scale better, it is still the case that Implict A* times out for $R \geq 5$, and RRT* times out for $R \geq 6$.

Dual-arm manipulator: This test shows the benefits of dRRT* when planing for two 7-dimensional arms. Figure 5.7 shows that RRT* fails to return solutions within $100K$ iterations. Using small roadmaps is also insufficient for this problem. Both dRRT* and Implicit A* require larger roadmaps to begin succeeding. But with $N \geq 500$, Implicit A* always fails, while dRRT* maintains a 100% success ratio. As expected, roadmaps of increasing size result in higher quality path. The informed nature of dRRT* also allows to find initial solutions fast, which together with the branch-and-bound primitive allows for good convergence. Additional data is presented in Figure 5.10. Here, the data presented in Figure 5.7 is shown again over iterations instead of over time.

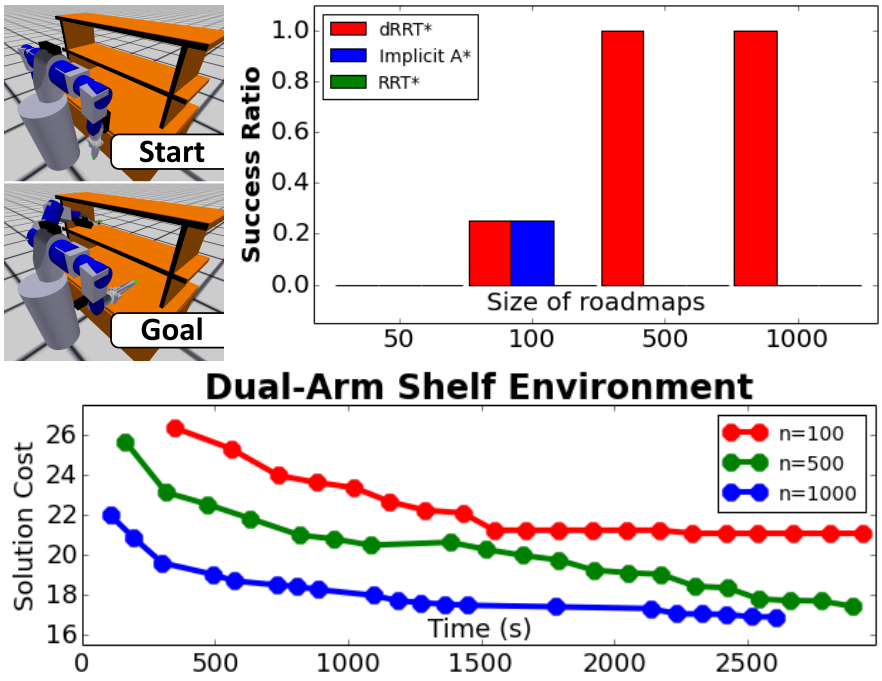

Figure 5.7: (*Top*): dRRT* is run for a dual-arm manipulator to go from its home position (above) to a reaching configuration (below) and achieves perfect success ratio as n increases. (*Bottom*): dRRT* solution quality over time. Here, larger roadmaps provide benefits in terms of running time and solution quality.

5.5 Discussion

The asymptotic optimality properties of implicitly defined planning structures for multi-robot planning are studied in this work. This work shows that implicit planning structures for multi-robot planning maintain asymptotic optimality guarantees given appropriate consideration in the construction of individual robot roadmaps and appropriate search of the tensor product roadmap.

The authors believe that the analysis can be extended to more complex settings, by relying on recent work concerning sampling-based motion-planning with kinodynamic constraints [131, 130]. Furthermore, the analysis should also be applicable to a variety of cost functions other than the total distance, such as $\max_{1 \leq i \leq R}\{|\sigma_i|\}$.

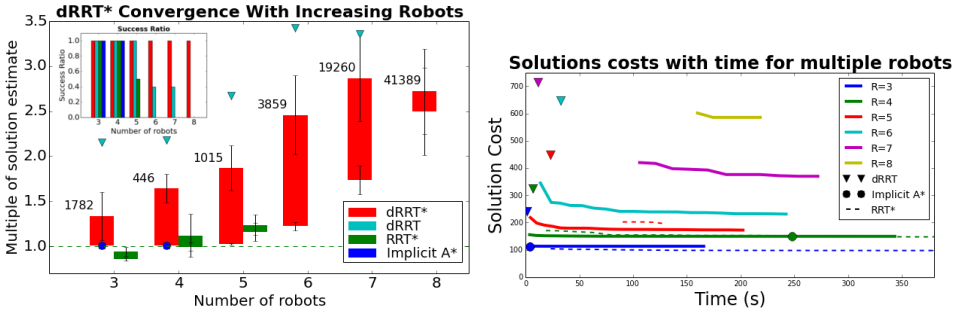

Figure 5.8: (*Top*): Convergence rate and success ratio over the minimal 9-node roadmap (*Bottom*): Solution cost over time when using the minimal roadmap.

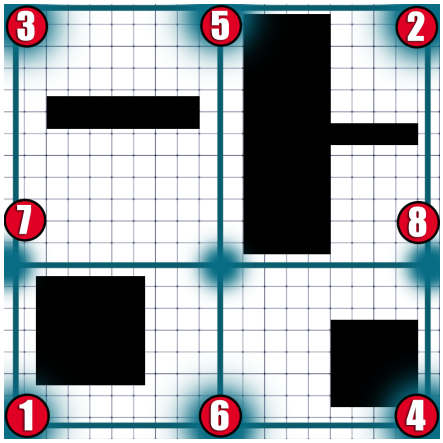

Figure 5.9: Minimal graph for the R-robot case.

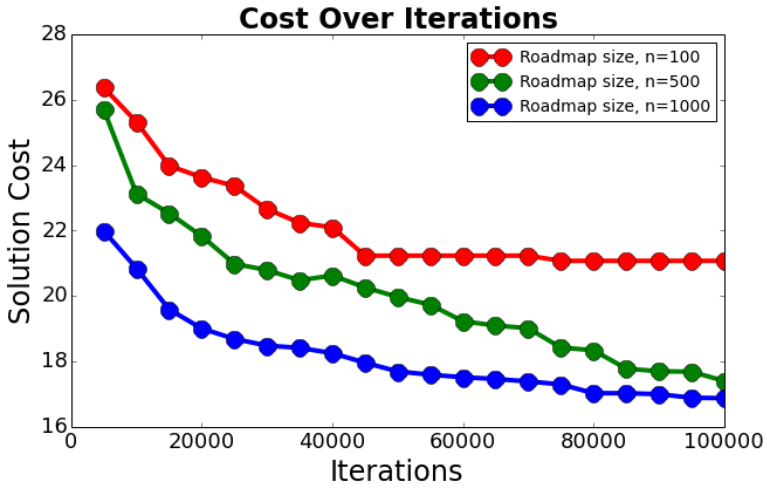

Figure 5.10: Motoman benchmark solution quality over iterations.

Algorithm 10: `Expand_dRRT`$^*(\hat{\mathbb{G}}, T, v^{\text{last}}, T)$

if $v^{\text{last}} == NULL$ **then**
  $q^{\text{rand}} \leftarrow$ `Random_Sample`();
  $v^{\text{near}} \leftarrow$ `Nearest_Neighbor`(T, q^{rand});
  $v^{\text{new}} \leftarrow \mathbb{O}_{\mathtt{d}}(v^{\text{near}}, q^{\text{rand}})$;
end
else
  $v^{\text{new}} \leftarrow \mathbb{O}_{\mathtt{d}}(v^{\text{last}}, T)$;
end
$N \leftarrow$ `Adjacent`$(v^{\text{new}}, \hat{\mathbb{G}}) \cap V_T$;
$V^{\text{best}} \leftarrow \arg\min_{v \in N s.t. \mathbb{L}(v, v^{\text{new}}) \subset \mathcal{C}^{\text{free}}} c(v) + c(\mathbb{L}(v, v^{\text{new}}))$;
if $V^{\text{best}} == NULL$ **then**
  return $NULL$;
end
if $v^{\text{new}} \notin T$ **then**
  T.`Add_Vertex`(v^{new});
  T.`Add_Edge`$(V^{\text{best}}, v^{\text{new}})$;
end
else
  T.`Rewire`$(V^{\text{best}}, v^{\text{new}})$;
end
for $v \in N$ **do**
  if $c(v^{\text{new}}) + c(\mathbb{L}(v^{\text{new}}, v) < c(v)$ *and* $\mathbb{L}(v^{\text{new}}, v) \subset \mathcal{C}^{\text{free}}$ **then**
    T.`Rewire`(v^{new}, v);
  end
end
if $h(v^{\text{new}}) < h(V^{\text{best}})$ **then**
  return v^{new};
end
else
  return $NULL$;
end

Table 5.1: Construction and query times (SECs) for 2 disk robots.

Number of nodes: $N =$	**50**	**100**	**200**
N^2-PRM* construction	3.427	13.293	69.551
N^2-PRM* query	0.002	0.004	0.023
2 N-size PRM* construction	0.1351	0.274	0.558
Implicit A* search over $\hat{\mathbb{G}}$	0.684	2.497	10.184
`dRRT`* over $\hat{\mathbb{G}}$ (initial)	0.343	0.257	0.358
`dRRT`* over $\hat{\mathbb{G}}$ (converged)	3.497	4.418	5.429

Chapter 6

Compact Representations for Multi-arm Manipulation Search

In the previous chapter, the problem of planning motion for multiple robots moving simultaneously was explored, and an efficient, asymptotically optimal method is provided. Using this method, this chapter focuses on solving a multi-arm manipulation problem where multiple robot manipulators have to perform a series of manipulations, including hand-offs, on a target object to bring it from its start position to an intended target. The method provides a minimal search representation and quickly determines a sequence of arm motions to plan for.

6.1 The N-Arm Manipulation Problem

As this work now shifts to focus on manipulation problems, let there be a single rigid-body object, o, with its own configuration space $\mathcal{C}_o \subset SE(3)$ in the workspace. Then, the configuration space for the entire problem is:

$$\mathcal{C} = \prod_{i=1}^{n} \mathcal{C}_{\mathcal{A}_i} \times \mathcal{C}_o.$$

where $\mathcal{C}_{\mathcal{A}_i}$ corresponds to the configuration of the i-th robot manipulator, $\mathcal{A}_i$.

The collision-free subset $\mathcal{C}^{\text{free}} \subset \mathcal{C}$ includes two sets, which allow contacts:

a) *stable configurations* $\mathcal{C}_s$: the object is at a stable pose, where o is in contact with static obstacle geometry, and its pose does not change while no arm acts upon o;

b) *grasping configurations* $\mathcal{C}_g$: these correspond to one or more arms grasping the object.

Let k denote the number of arms required to move the object o, where $1 \leq k \leq n$. The value of k depends on the o; for instance, o may be heavy and require multiple

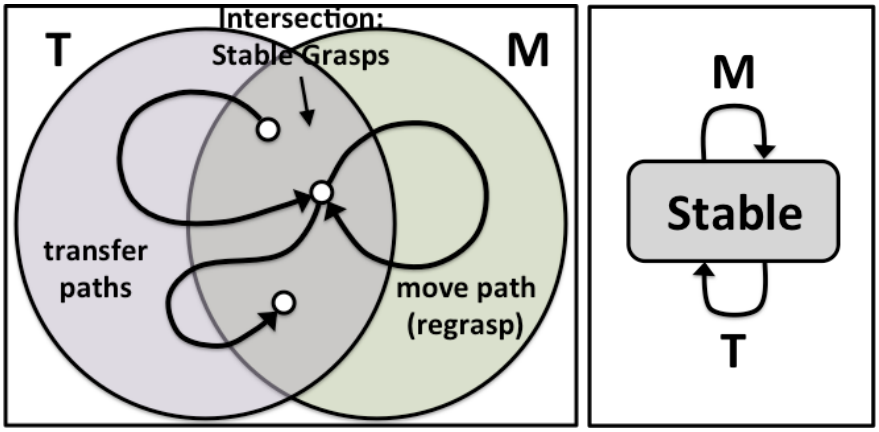

Figure 6.1: Single-arm manipulation graph [136]. *Left:* For the T set, the object is transferred while grasped. For the M set, the arms move and the object rests at a stable pose. At the intersection, the object is grasped at a stable pose. *Right:* Useful paths for manipulation bring the object back to a stable grasp pose.

arms to be transferred. A valid path $\tau : [0,1] \rightarrow \mathcal{C}^{\text{free}}$ is a continuous sequence of stable and grasping configurations, where transitions happen via stable poses or object handoffs.

Definition 18 (Multi-Arm Manipulation Problem)**.** *Given an initial configuration* $q_{init} \in \mathcal{C}_s$ *and a goal configuration* $q_{goal} \in \mathcal{C}_s$*, compute a valid path* $\tau : [0,1] \rightarrow \mathcal{C}^{\text{free}}$*, such that* $\tau(0) = q_{init}$ *and* $\tau(1) = q_{goal}$*.*

The traditional representation for a single arm interacting with an object consists of two modes:

a) transfer configurations T, where the manipulator is grasping and transferring the object, and

b) move configurations M, where the manipulator is moving through the space without holding the object. While typically referred to as "transit" configurations, the term moved is used to follow the M vs. T notation.

This gives rise to the graphs of Fig. 6.1, where a transfer is a trajectory through T connecting different stable poses, and a regrasp goes through M from a stable pose to itself.

This representation was only recently generalized for dual-arm robots [52], which is shown in Fig. 6.2, given the notation used here. Reasoning about the topology of

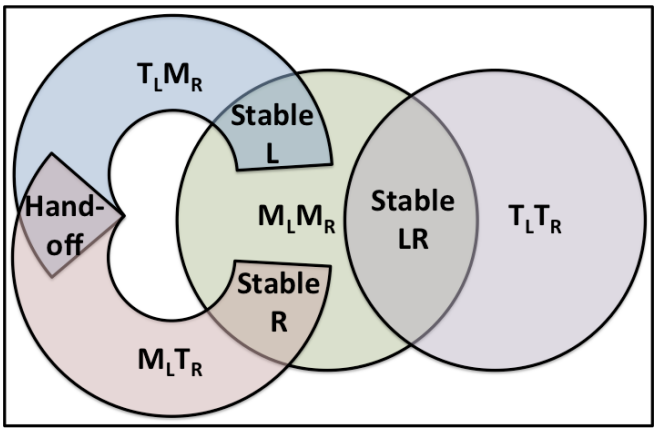

Figure 6.2: The representation for dual-arm manipulation has four sets of configurations in the general case. Note that M_LM_R and T_LT_R correspond directly to M and T for single-arm manipulation (Figure 6.1 Left).

the problem allows to identify whether using both hands is helpful by integrating grasp and object placement planners. In the n-arm case, the planning problem has 2^n modes, and each corresponds to different combinations of arms either moving or transferring. For example, the mode $\mathrm{M}_L\mathrm{M}_R$ corresponds to both arms moving without grasping the object, while $\mathrm{T}_L\mathrm{T}_R$ has both arms transferring the object simultaneously.

In the two arm case, a new mode corresponding to hand-offs arises. A hand-off is a transition where the arm - or, in general, a set of arms - which grasps the object changes while the object is not necessarily in a stable pose. This requires ensuring the object is not dropped; however, this is a physics problem, which is not the focus of this work. Next, this representation is generalized for n arms while pruning away redundant modes.

This work assumes access to four primitives: (i) A **stable pose generation** module provides reachable stable object configurations $q_o \in \mathcal{C}_o$; (ii) A **hand-off generation** module returns object configurations where k arms can grasp the object simultaneously; (iii) A **spatial interaction** module, which is a graph that indicates which subsets of arms can simultaneously grasp an object. This graph is referred to as a Spatial Interaction Graph (`SIG`), and is detailed in section 6.2; (iv) A **grasping** module, which computes grasps to facilitate transfers and hand-offs.

6.2 A Compact Multi-Arm Manipulation Representation

This section presents the reduced representation of n-arm manipulation, providing an algorithm to explicitly construct the graph with an appropriate topology for any n and k. Unsurprisingly, the topology of the graph constructed with n and k is equivalent to one where $n' = h = \binom{n}{k}$ and $k' = 1$.

Assumption 3. *There is a known, fixed number of arms, k, required to transfer the object.*

For $n = 2$, k can either be 1 or 2, as illustrated in Fig. 6.3, noting differences with Fig. 6.2. For $k = 1$, it is unnecessary to have a state where both arms are transferring the object. Conversely, for $k = 2$, single-arm transfer states are infeasible. Note that for $n = k = 2$, the state representation is topologically equivalent to the single-arm case of Fig. 6.1. A dual representation to that of Fig. 6.3 is shown in Fig. 6.4 below.

This representation is general, and Fig 6.5 illustrates the case of $n = 3$ and $k = 2$.

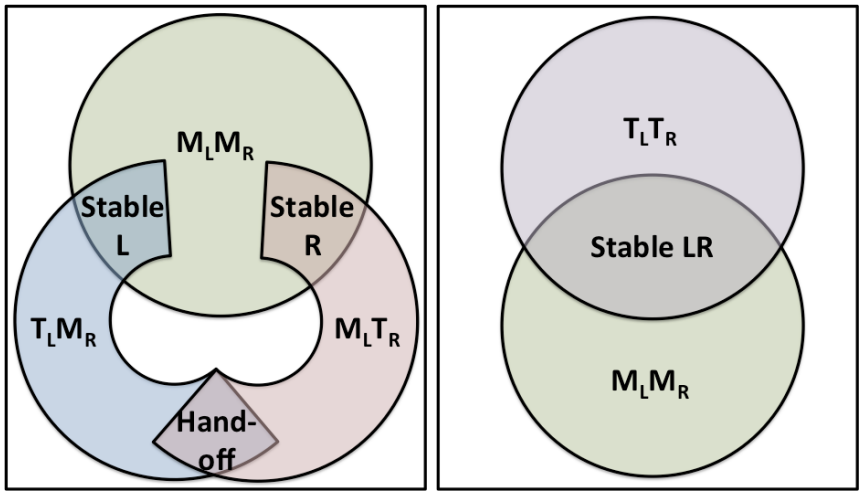

Figure 6.3: The manipulation graphs for $k = 1$ (left) and $k = 2$ (right) using 2 arms: a left (L) and a right arm (R).

The general case has $h = \binom{n}{k}$ stable grasp states, and from each pair of these states, up to $\binom{h}{2}$ handoff states. This results in a total of up to $h + \binom{h}{2}$ states, with the following edges connecting them:

i) Regrasp and Transfer self-edges for each stable grasp state ($2h$ edges).

ii) Move edges between all pairs of stable grasp states ($\binom{h}{2}$ edges).

iii) Transfer edges between handoffs and their corresponding stable grasp states ($2\binom{h}{2}$ edges).

iv) Edges between pairs of handoff states that share a common subset of arms. That is, each handoff state maintains sets of arms, $\mathcal{R}_{in}$ and $\mathcal{R}_{out}$, and there exists an edge between states that share a set ($\binom{h}{2}(h-2)$ edges).

v) Self-edges for each handoff state where (a) k arms keep the object stable and (b) k other arms perform a regrasp ($2\binom{h}{2}$ edges).

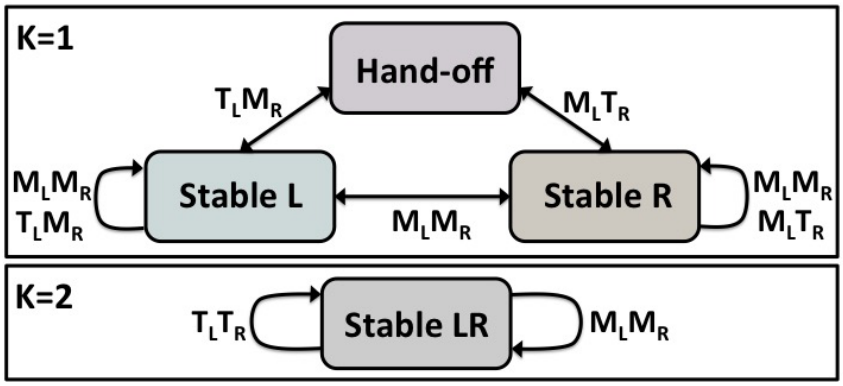

Figure 6.4: *Top:* an object requiring $k = 1$ arms, corresponding to Fig. 6.3 (left). *Bottom:* $k = 2$, corresponding to Fig. 6.3 (right).

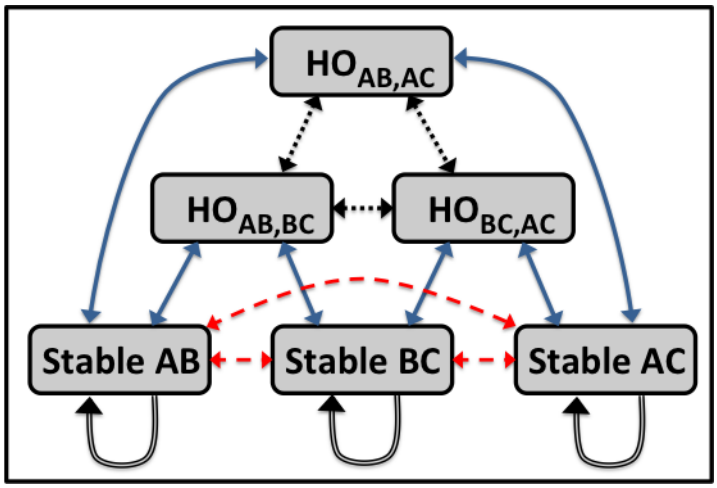

Figure 6.5: Example for $n = 3$ and $k = 2$ (topologically equivalent to $n = 3$ and $k = 1$). Hallow lines represent self-transitions for stable states, light dashed lines are transitions between stable states, dotted lines are passes between different handoffs, while solid lines are transitions between stable and handoff states.

This representation significantly reduces the number of modes compared to the case that allows all combinations of arms to grasp the object. In the exhaustive case, there

are $2^n - 1$ stable grasp states. Then, there are $\binom{2^n-1}{2}$ handoff states, for a grand total of $(2^n-1)+\binom{2^n-1}{2}$ states. This complete representation would also have the same types of edges as described before. This results in a grand total of $2(2^n-1)+5\binom{2^n-1}{2}+\binom{\binom{2^n-1}{2}}{2}$ edges/modes. The relative benefits of the reduced representation for certain choices of n and k is provided in Fig. 6.6.

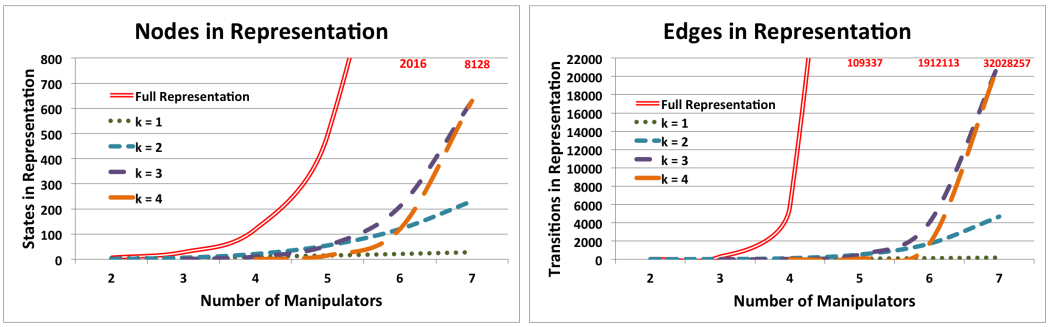

Figure 6.6: A comparison between the full representation and the upper bound for the provided one. For different values of n and k, the provided graph results in considerably fewer modes.

Assumption 3 prunes away states; however, handoffs cannot always be achieved, so another assumption is used:

Assumption 4. *It is known which arms can potentially achieve a handoff and reach the same stable object poses.*

Two input "Spatial Interaction Graphs" are assumed: a $\texttt{SIG}_h$ for feasible handoffs and a $\texttt{SIG}_s$ for transitions between stable grasps. A node in a SIG represents a set of k arms, and an edge between two nodes in $\texttt{SIG}_h$ indicates a handoff is possible, and an edge in $\texttt{SIG}_s$ indicates that there are stable object poses reached by both sets of k arms.

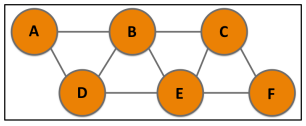

Figure 6.7: An example SIG for $n = 6$ and $k = 1$.

An example SIG is shown in Fig. 6.7. Using the graph for both $\texttt{SIG}_s$ and $\texttt{SIG}_h$ with $n = 6$ and $k = 1$, there will be 15 states and 60 edges, but without the SIGs, there would be 21 states, and 117 edges, reducing the number of modes by nearly half. An algorithm for constructing the appropriate graph is given in the next section.

6.3 Pre-processing and Searchng $G_{\texttt{MAM}}$

Given the outlined representation, this work follows the methodology of prior work [30]. At a high-level, there are two separate steps: a preproccessing phase, and then an online search to answer manipulation queries. This section also details the multi-robot motion planning approach leveraged.

6.3.1 Preprocessing

The preprocessing entails two separate steps.

Automaton Generation

First, the method constructs a high-level representation of the multi-modal task space representation of the problem. This section briefly outlines the reduced representation examined by prior work [30]. This representation is a graph denoted $G_{\texttt{MAM}}$ where nodes correspond to high-level states, which represent transition configurations between manipulation modes, and edges between these nodes represent motion planning problems which transition from one high-level state to another.

Given the number of arms R and the minimum number of arms needed to manipulate an object k, the method first constructs stable grasp states for each set of k arms. Each stable grasp state represents the set of k arms grasping the object while it rests stably on some surface in the environment. These states have two self-edges which represent re-grasp and transfer paths, and each pair of stable grasp states with shared reachability also share an edge. Hand-off states are also generated for each pair of stable grasp states which have shared reachability, and edges are added between the corresponding stable grasp states and the hand-off state. Each hand-off state also has a self-transition which corresponds to a re-grasp. This automaton is constructed given appropriate spatial interaction graphs (`SIG`s) which are optimistic estimations for arm spatial interaction.

Arm Path Precomputation

The next step in preprocessing is generating a set of roadmaps in the configuration space of each of the robot manipulators. For the purpose of analysis, it is assumed that the roadmaps are constructed with the following stipulations:

- Each roadmap is constructed with the asymptotically optimal PRM* method, having n samples.
- Roadmaps are constructed for a model of the robot with an optimisitic estimate of end-effector geometry.

The second point diverges from conventional approaches to the problem and requires the end-effector collision volume to be a subset of the end-effector's true volume, regardless if it is grasping or at rest. As this work will show in Section 6.4, it is an important detail for ensuring asymptotic optimality while maintaining practical efficiency.

These roadmaps must be queried in a lazy fashion to account for different object grasps, end-effector configurations, and configurations of other robots and objects. These roadmaps, denoted $\mathcal{G}$, will be used while performing $G_{\texttt{MAM}}$-search, as detailed next.

6.3.2 Online $G_{\texttt{MAM}}$ Search

Algorithm 11: Solve_MTP$(\mathcal{G}, R, o, n, q_{init})$

```
1 A_k ← Generate_Automaton(R, o.k);
2 G ← Preprocess_Roadmaps(G, R) ;
3 while Not Finished do
4 |   Π ←G_MAM_Search(A_k, n, q_init, G);
5 |   G ← Preprocess_Roadmaps(G, R) ;
6 |   n ← 2 · n;
7 end
8 return Π;
```

This work performs an informed search over $G_{\texttt{MAM}}$, which has similarities to both discrete searches like A* and continuous motion planning methods such as EST. The whole process, including the preprocessing is outlined in Algorithm 11. It begins by

constructing the an appropriate high-level representation (Line 1) and then performing the roadmap precomputation as outlined in Section 6.3.1 (Line 2). The high-level alternates between performing online search (Line 4) and refining the roadmap structures (Line 5) until some pre-specified stopping criterion is satisfied (Line 3), and eventually returns a path.

Algorithm 12: $G_{\texttt{MAM\_Search}}(A_k, n, q_{init}, \mathcal{G})$

```
1  v_s ← Free_State(A_k); v_s.states ← {q_init};
2  O ← {v_s}; Π ← ∅; Π_G ← ∅;
3  while iterations < n do
4      v_min ← peek_min(O);
5      Children, q_select ← expand(v_min, A_k);
6      for c ∈ Children do
7          Q_trans ← Get_Transitions(v_min, c);
8          if Q_trans ≠ ∅ then
9              Π ← Find_Paths(q_select, Q_trans, G);
10             if Π ≠ ∅ then
11                 Add_Paths(Π, q_select, c);
12                 if c ∉ O then
13                     c.heuristic ← h(c);
14                     O.push(c);
15                 end
16                 if c == v_s then
17                     Π_G ← Π_G ∪ {Trace_Path()};
18                 end
19             end
20         end
21     end
22     v_min.heuristic ← (1 + v_min.expansions) · h(v_min);
23     ++iterations;
24 end
25 return Minimum_Cost(Π_G);
```

The $G_{\texttt{MAM\_Search}}$ method outlined in Algorithm 12 keeps track of the high-level free state, v_s (Line 1), which represents a specific set of configurations where all manipulators have returned to their safe configuration and none of them are grasping an object. This allows the search to start and end specifically within v_s. The method will then initialize the open set O with the free state for the initial configuration of the world q_{init} (Line 2). The search continues much like a typical $\texttt{A}^*$ search, extracting the minimum node from the open set and expanding from the corresponding automaton state to find neighboring

Algorithm 13: `Get_Transitions`(q, c)

```
1 if c == v_s and Object_at_Goal(q) == false then
2 |   return ∅;
3 end
4 Q_goal ← Generate_Consistent_Goals(c, q);
5 return Q_goal;
```

automaton nodes to extend toward (Lines 4, 5). Then, it attempts to do forward search to find if there are feasible transitions (determined as shown in Algorithm 13) which can be reached from the expanded configuration q_{select}. The method then attempts to find paths to the generated configurations (Line 9, Algorithm 14) and then if any paths are found, they are added to the planning structure (Line 10, Algorithm 15).

Algorithm 14: `Find_Paths`$(q_{start}, Q_{goal}, \mathcal{G})$

```
1 Π_feas ← ∅;
2 for q ∈ Q_goal do
3 |   Π_feas ← Π_feas ∪ Compute_Plan(q_start, q, G)
4 end
5 return Π_feas;
```

Note that the asymptotically optimal `dRRT`* framework from the previous chapter is employed here. If the neighboring high-level node has not yet been visited by the search, it will be added to the open set for future expansions (Lines 12-14). Then if the neighboring high-level node c is the free state, then the object must be at its target and all manipulators have returned to their safe state, so this search has been successful, and all paths which result in the object resting at the goal pose are retained (Lines 15, 16). The heuristic value for the expanded node v_{min} is then inflated (Line 17), and the method eventually returns the minimum cost path to the goal which has been discovered (Line 19).

Algorithm 15: `Add_Paths`(Π, q, c)

```
1 for π ∈ Π do
2 |   q_reached ← π(1);
3 |   q_reached.cost = q.cost + π.cost;
4 |   c.states ← c.states ∪ {q_reached};
5 end
```

6.4 Analysis: Asymptotic Optimality

This section aims to extend theoretical guarantees to the multi-arm manipulation search $G_{\texttt{MAM}}$-search. Here, analysis from related work showing the asymptotic optimality for a task and motion planning framework under piecewise-analytic constraints [158]. For clarity, the high-level idea of this analysis is reviewed here and then the applicability of this proof to $G_{\texttt{MAM}}$-search is shown. This analysis requires an extra step to show that performing lazy search over the constructed roadmap still fulfills the requirements of the proof.

The proof examines the path returned by the algorithm π^n in relation to the clearance-robust optimal path $\pi^*_{\delta_n}$.

At a high-level, the proof operates by showing the probability of the approach returning a path significantly longer than the optimal path is summable, i.e. that

$$\sum_{n=1}^{\infty} \mathbb{P}(c_n \geq (1+\epsilon)c^*) < \infty \tag{6.1}$$

This is done by showing that this probability is bound by the probability of a few events. First, the method must be able to draw a sample in each of a set of hyperballs at the intersection of modes, an event denoted by $A_{\cap,n}$. Second, the method must also generate sufficient samples within each mode, an event denoted by A_n. Finally, the method also must generate samples within an α fraction of small hyperballs, an event denoted by $A_{n,\alpha,\beta}$. The proof continues by showing that

$$\mathbb{P}(c_n \geq (1+\epsilon)c^*) \leq \mathbb{P}(\overline{A_{\cap,n}}) + \mathbb{P}(\overline{A_n}|A_{\cap,n}) + \mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n})$$

and that each of the probabilities on the right-hand side are summable, i.e. that

$$\sum_{n=1}^{\infty} \mathbb{P}(\overline{A_{\cap,n}}) < \infty, \sum_{n=1}^{\infty} \mathbb{P}(\overline{A_n}|A_{\cap,n}) < \infty,$$
$$\sum_{n=1}^{\infty} \mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n}) < \infty.$$

Combining these results cleary implies the intended result shown in Equation 6.1. It

is simply a matter now of showing that the $G_{\texttt{MAM}}$-search framework also ensures these probabilities are summable.

This will be accomplished by showing an equivalence in the sampling between $G_{\texttt{MAM}}$-search and that of prior work. First, examining the probability $\mathbb{P}(\overline{A_{\cap,n}})$, the argument simply requires the method to asmptotically sample infinitely often in the submanifold corresponding to each intersection of modes. The preprocessing phase of $G_{\texttt{MAM}}$-search explicitly enumerates each such intersection and draws a number of samples from the manifold, and thus it is clear that it will also satisfy the summability condition here.

What is less clear, however, is whether this is true for the other two events. Prior work showed this by assuming for every orbit discovered during the search, the method would construct a $\texttt{PRM}^*$ roadmap of n samples. Then as n tends to infinity, this is sufficient to ensure that $\mathbb{P}(\overline{A_n}|A_{\cap,n})$ and $\mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n})$ are summable. Unlike the related work, $G_{\texttt{MAM}}$-search constructs $\texttt{PRM}^*$ roadmaps for each mode, and then evaluates paths on each orbit through this graph lazily. This work makes the following claim:

Claim 2. *As long as $G_{\texttt{MAM}}$-search generates the $\texttt{PRM}^*$ roadmap for each mode of the graph in a configuration space $\mathcal{C}_{mode}$ such that for any orbit through this mode, its free configuration space $\mathcal{C}^{\text{free}}_{orbit} \subset \mathcal{C}^{\text{free}}_{mode}$, then*

$$\sum_{n=1}^{\infty} \mathbb{P}(\overline{A_n}|A_{\cap,n}) < \infty, \sum_{n=1}^{\infty} \mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n}) < \infty.$$

Proof sketch. Consider the roadmap $\mathcal{G}_i$ constructed for arm $\mathcal{A}_i$. Given the assumptions outlined earlier, when querying this roadmap for a solution, it will be subject to the constraints of an arbitrary orbit such that $\mathcal{C}^{\text{free}}_{orbit} \subset \mathcal{C}^{\text{free}}_{mode}$. Then, consider the subgraph $\mathcal{G}_i^{\text{`}}$ of $\mathcal{G}_i$ which corresponds to removing the set of vertices and edges which are in collision given the new free space constraints $\mathcal{C}^{\text{free}}_{orbit}$. Let $V_i^{\text{`}} \subset V_i$ be the set of vertices in this subgraph, and $n > \hat{n} = card(V_i)$. Furthermore, let $\alpha \in (0, 1]$ be the ratio of samples remaining in $\mathcal{G}_i^{\text{`}}$ after pruning for the new free space, i.e. $\hat{n} = \alpha n$. Given the robustly feasible nature of the problem, it is clear that as n tends to infinity, so too does $\hat{n}$.

Next, consider any arbitrary vertex $v^{\text{`}} \in V^{\text{`}}$. Now, if it were the case that $\mathcal{G}_i^{\text{`}}$ were

directly constructed using the same set of vertices in a PRM*-like fashion, then the neighborhood set of vertex $v^{'}$ would contain the same or fewer possible neighbors than the corresponding node in $\mathcal{G}_i$. Then clearly, $\mathcal{G}_i^{'}$ satisfies the exact conditions required by prior work for asymptotic optimality of the method, as it examined roadmaps which also contain $\hat{n} = \alpha n$ vertices. □

6.5 Experimental Evaluation

The proposed method greatly reduces the number of modes considered in search, and the objective of this section is to highlight the computational and solution length benefits of the proposed framework. The approach was tested in simulation using models of the Baxter robot platform in two environments, and it was compared to a general framework proposed in recent literature called Random-MMP [55]. Since Random-MMP also generates a search tree through the multi-modal search space, the method seemed appropriate to compare against. Another method in the literature performing multi-modal planning could also be considered; however, this method focuses on single-arm problems and does not address the combinatorial challenge that multi-arm manipulation poses [7].

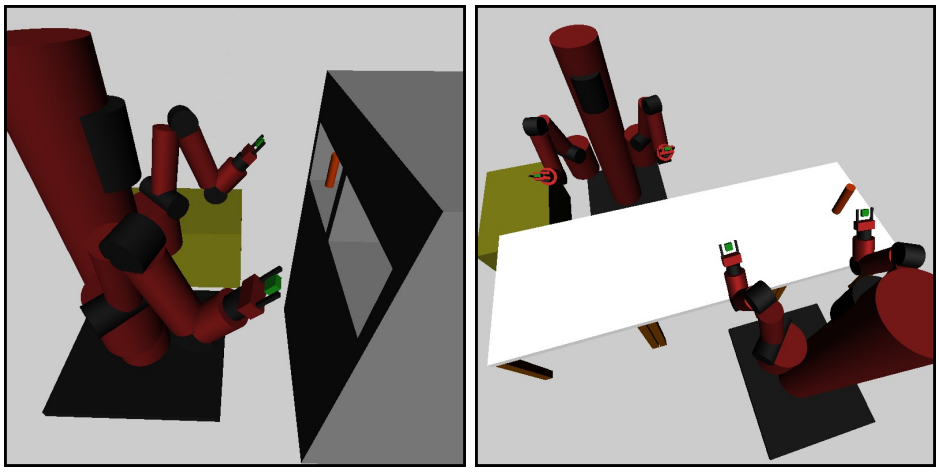

Figure 6.8: The two setups used to evaluate the approach: Shelves environment with 2 arms (left) and Table environment with 4 arms (right). In both setups, the objective is to place the object on top of the yellow bin.

This paper examines two problem setups, seen in Figure 6.8, which are the Shelves environment and the Table environment. For Shelves, $n = 2$ and for Table, $n = 4$, where

both setups use $k = 1$; furthermore, the input SIGs used in each setup are illustrated in Figure 6.9.

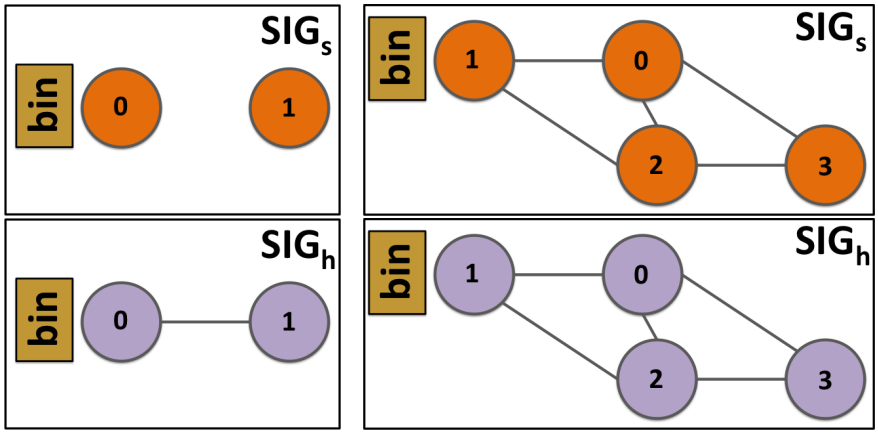

Figure 6.9: The input SIGs used for the two problem setups. In the Shelves (left), the robot cannot pass the object to the other arm through a stable pose ($\texttt{SIG}_s$) but it can via a handoff ($\texttt{SIG}_h$). The Table (right) uses SIGs of the same topology; however, the two right arms of the robots cannot interact.

In the Shelves environment, the arms can perform handoff, but cannot pass the object through a stable pose. In the Table environment, all arms can interact except for arms 1 and 3. The Shelves environment is inspired by applications such as the Amazon Picking Challenge, where using both arms can potentially increase efficiency of the method. The Table environment is used as a prototypical example requiring multiple arm interactions.

The proposed search over $G_{\texttt{MAM}}$ was compared with Random-MMP, and it shows competitive performance, as illustrated in Figure 6.10. Using comparable computation time, the proposed method generates paths up to five times shorter than Random-MMP, and significantly fewer state transitions. The A*-style search is able to appropriately leverage the precomputed heuristics to bias search in such a way to avoid redundant transitions.

Notably, the proposed method has a higher rate of timing out (after 15 minutes) for the harder problem, even though on average it is faster than Random-MMP. This is likely due to heuristics biasing the search into a local minimum, causing the method to occasionally spend too much time exploring regions of the space which do not lend themselves to a solution. This motivates potential future work to determine what

heuristics are appropriate in the context of multi-modal n-arm manipulation.

Method	$G_{\texttt{MAM}}$ SEARCH		Random-MMP	
Environment	Shelves	Table	Shelves	Table
Search Time(s)	7.41s	321.45s	13.31s	379.44s
Expansions	31.12	142.43	38.78	271.45
Path Quality(s)	11.95s	17.08s	50.75s	75.09s
Transitions	4.39	5.70	10.00	14.28
Timeouts(%)	0%	9.82%	0%	1.72%

Figure 6.10: Average statistics for the methods for 350+ experiments, which were solved by both methods. Search time is how long it took to find a path during online query resolution, while path quality is the duration of the solution path. Transitions is the number of high-level state transitions in the automaton. Timeout percentages are given over all attempted problem instances in each environment.

6.6 Discussion

This work describes the topology of n-arm prehensile manipulation, generalizing state-of-the-art results [136, 52]. The given representation prunes redundant modes and yields a general search framework to solve multi-arm problem. A formal algorithm for constructing a manipulation graph ($G_{\texttt{MAM}}$) and useful preprocessing tools are provided, and an online search method following a prior multi-modal motion planning framework is provided [55].

An important future step is evaluating and improving practical performance when $k > 1$. It is possible the greedy nature of the method causes issues with local minima, which should also be addressed. A related issue is to rigorously study the best approach to inflating heuristics in search to achieve good coverage of the search space. The method could benefit from employing $\texttt{M}^*$-style results to ensure planning for multiple arms uses the optimal decoupling.

The framework would be improved by adapting it toward mobile manipulation and non-prehensile manipulation primitives [34, 7]. Furthermore, the method could leverage caging results to practically handle the $k > 1$ case [20, 38]. The approach also needs to be made amenable to other practical issues, such as force control [113], grasp planning [5], sensing [70, 74] and reasoning over uncertainty [82, 81].

Chapter 7

Conclusions

This work is presented as an approach toward solving complex multi-arm manipulation tasks. The focus is on challenging applications where the shape and initial pose of objects is unknown during precomputation, and determining methods for quickly performing the object transfer task in this domain. This work makes contributions towards fully automated recycling centers where the methods provide formal guarantees on the quality of paths produced by the robot systems in order to maximize throughput. This setup requires flexible online methods in order to deal with the unstructured nature of the environment. The final result of this work is a scalable and flexible framework which leverages compact planning representations to provide globally asymptotically optimal task and motion planning in manipulation domains.

7.1 Statement of Contributions

The main contributions of this work are spread across chapters 3, 4, 5, and 6. It begins with expanding the known boundaries regarding properties of sampling-based motion planners, and ends with showing that it is possible to have efficient manipulation planning with formal path quality guarantees for the task as a whole.

Beginning in chapter 3, this work sought out to expand upon the knowledge of what properties sampling-based planners exhibit. It has been known since very early in the field of study that these method exhibit *probabilistic completeness* (definition 4). This property states that these methods return a solution to the motion planning problem, if one exists, with probability approaching 1 as the method is run to infinity. Similarly, recent related work showed that these methods also exhibit *asymptotic optimality* (definition 5). This property states that the methods return the optimal solution with

probability approaching 1 as the method is run to infinity, given sufficient density of the underlying planning structure.

These properties however, are asymptotic in nature. That is, they only truly hold in a theoretical model where the planning is run over an infinite time horizon. This is the motivation behind chapter 3, which formally presents the property of *probabilistic near-optimality* (property 1). This property formally shows that there exists probabilistic bounds on the length of a solution returned by these methods relative to the optimal solution after a finite amount of computation. Formally, the main contributions of chapter 3 can be summarized as follows:

- The probabilistic near-optimality of an asymptotically sparse variant of PRM* is proven, and the drawn bounds are shown to hold closely to the actual results of experimental trials.
- A probabilistic bound on solution quality is formulated, and a principled stopping criterion is identified which probabilistically guarantees low-cost solutions are returned.
- A novel approximation of a previously unsolved variant of the ball-line picking problem is drawn, which may have relevance beyond this work.

The experimental results also recovered the known issue of the *curse of dimensionality*. That is, in order to ensure with high confidence that PRM* returns a solution that is near-optimal in higher-dimensional spaces, the method will require exponentially many samples to be drawn. While this was a known result, it formally motivates the contributions of the rest of this work, which focuses on providing compact representations for planning.

In particular, chapter 4 seeks out ways to still provide formal path quality guarantees for single-robot motion planning without requiring the exponentially large number of samples required to ensure probabilistic near-optimality guarantees. This motivates creating compact planning representations that can solve motion planning problems efficiently. Such a compact representation should have a small memory footprint while returning paths of high quality.

To this end, this chapter introduces the concept of a sparse roadmap spanner: a compact representation of the free configuration space that requires orders of magnitude fewer nodes than the asymptotically optimal PRM* approach and guarantees *asymptotic near-optimality* (definition 9). Asymptotic near-optimality ensures that an algorithm returns a solution to the motion planning problem of length bounded by the length of the true optimal path with probability 1 as the method is run to infinity. Formally, the main contributions of chapter 4 can be summarized as follows:

- It provides a method to construct a sparse roadmap spanner (SPARS), which is proven to be asymptotically near-optimal.
- It proves this approach provides asymptotic sparsity: that the probability of growing the structure tends to 0 as the method is run to infinity.
- It shows the practical speed-ups and orders of magnitude less memory required by the method over asymptotically optimal PRM* graphs.

While directly proving probabilistic near-optimality for the SPARS framework is beyond the scope of this work, it does show that practical sparse roadmap spanners can be employed in practice for robot motion planning. The original problem posed, however, requires the use of multiple robots to solve challenging object transfer tasks.

To this end, chapter 5 focuses on methods for effective multi-robot motion planning. The effort is focused on centralized, coupled planning in order to ensure theoretical properties. While the efforts of chapter 4 provide significant reductions in the memory cost of storing a planning structure, using this compact representation still requires a good deal of computational effort during construction for high-dimensional spaces. This motivates the study of a new kind of compact representation for efficiently solving multi-robot motion planning problems.

An asymptotically optimal variant of dRRT is formulated, which allows practically efficient motion planning for multi-robot systems while maintaining formal guarantees on path quality. Specifically, it ensures that the online search is asymptotically optimal. This is especially challenging since the approach does not explicitly maintain the entire

planning structure in memory, but rather only expands a tree through an asymptotically optimal implicit graph representation. Then, the main contributions of chapter 5 can be summed up as:

- It formalizes the dRRT* method, an asymptotically optimal variant of dRRT with several practical performance improvements.
- It shows asymptotic optimality guarantees can be provided without explicitly constructing a dense graph in composite configuration space.
- It shows via simulated verification that the method scales well to a large number of robots and quickly returns initial solutions of low cost.

This contribution reflects a significant milestone for high-dimensional motion planning in that it is possible to provide formal path quality guarantees without directly constructing and maintaining a planning structure in the composite configuration space of the set of robots being planned over. Furthermore, due to the nature of the method, i.e., it implicitly composes roadmaps for each of the robots, it is amenable to using both probabilistically near-optimal PRM* roadmaps or asymptotically near-optimal sparse roadmaps spanner (SPARS) roadmaps to gain the relative benefits of either of these methods.

The original motivating problem however involves robot manipulation problems using multiple fixed-base robot manipulators. To this end, chapter 6 outlines a framework for performing object transfer tasks using a set of robot manipulators. In line with the theme of this dissertation, it sought out to find a minimal, compact representation for the task planning problem and provide solutions to the problem while providing formal guarantees.

A multi-arm manipulation graph ($G_{\texttt{MAM}}$) is formulated, providing the minimal topology for solving multi-arm manipulation problems involving hand-offs and stable-pose object transfers. The chapter outlines useful preprocessing and shows how to perform the high-level search over this topology in an efficient manner. Formally, chapter 6 provides the following contributions:

- It generalizes the dual-arm manipulation topology to the minimal topology for prehensile multi-arm manipulation problems.
- It provides a preprocessing framework that uses scalable offline roadmap precomputation to adapt to novel objects and object poses.
- A formal proof of the asymptotic optimality for the task and motion planning problem is provided, showing the method converges to returning the optimal path over the optimal sequence of actions.
- Experiments using the approach demonstrate good scalability.

This final set of contributions represents the progress this paper posits toward provably efficient automated recycling tasks in semi-unstructured environments. The flexibility of this approach allows the method to handle novel object placements quickly while performing the object transfer task with provably efficient paths.

7.2 Important Open Questions

While this work provides several contributions toward having compact representations with formal guarantees, there are still many avenues of investigation and open questions remaining unanswered. The general theme of the work includes compact graph representations; however, there is a great body of work focusing on tree-based motion planning approaches. Mathematically, studying the properties of these approaches requires different tools, but given the popularity of the methods, probabilistic near-optimality for these approaches should be investigated. Furthermore, many of the models here, including the `dRRT`* method for multi-robot planning work with undirected graphs, and it would be interesting to investigate how the sparse roadmap spanner and `dRRT`* frameworks can be adapted for systems without symmetric kinematics. This would necessarily require directed graphs. This work would also in general benefit from further efforts to test the methods in physical testbeds to determine practical needs and concerns for the approaches. For instance, these methods may need to be integrated

with online feedback control in order to provide robust paths even under actuation uncertainty.

Much of the work in chapter 3 is also limited in that it reasons over a single robust optimal path, and this framework should be extended to reason over multiple homotopic classes. Furthermore, the automated stopping criterion posed by this chapter suggests a means for finding a principled scheme for performing random restarts, a practical but often heuristic approach used to increase algorithm success rate. This lends itself to a further optimization problem that could be investigated in future work.

There were also open questions left from chapter 4. Two of the most pressing and difficult open questions remain unanswered: (i) can the finite-time properties discussed in chapter 3 be extended to the construction of sparse roadmap spanners, and (ii) under what conditions can the method be shown to truly converge to a finite-sized data structure? It would be desirable to be able to directly drawn a bound on the path length returned by the `SPARS` framework at any finite iteration n, and the fact that the method converges to no longer growing the structure unfortunately does not imply that the structure that is constructed is finite. Furthermore, the method would benefit from finding better, or perhaps automated methods to select the algorithm parameters such as the stretch factor t and the sparse visibility range Δ.

The work in chapter 5 is an important contribution; however, there were still open questions here. Most notably, the proofs of asymptotic optimality of the method assumed a somewhat restrictive class of cost functions, so a more general proof over a wider range of cost metrics would be beneficial. Also, given the restrictions stated above, an extension of this framework to scenes with dynamic obstacles or robots with kinodynamic constraints would greatly improve its practical use for a wide range of physical platforms.

Finally, the work presented in chapter 6 opens up a great deal of possible work directions. The experimental results shown in this work are limited to the $k = 1$ case, as complex manipulation primitives are required for $k > 1$, but the framework is already designed to be amenable to this. It would be good to see if the method practically scales under these conditions. Furthermore, the approach would benefit greatly from a

broader study of heuristics for performing the online search, and more rigorous admissible heuristics should allow the method to quickly converge to the optimal sequence of actions, reducing solution computation time. There are also many practical variations on the problem that the framework currently does not directly address, such as non-prehensile manipulation problems, and mobile manipulation. The results presented are also simulation-based, and practical implementation could benefit by integrating advanced grasping techniques, force control, sensing integration, and reasoning over uncertainty.

7.3 Concluding Remarks

This work is focused on making progress towards solving complex, unstructured object transfer manipulation tasks, and pushes in two directions that the field may benefit from: using compact representations for planning problems, and providing formal guarantees on the solution length of paths returned by these methods after finite computation. The results of chapter 3 represent a principled approach to sampling-based methods, allowing practitioners to make grounded statements about the performance of sampling-based planners in practice. For many years, it was not possible to draw meaningful conclusions if a `PRM` planner did not return a solution, but this framework provides formal machinery to make educated statements about the solvability of the problem. While the exact framework presented here may not end up being the most practical approach for guaranteeing finite-time properties, the author believes these are critical considerations for the advancement of the field.

Furthermore, the practical benefits of compact representations have been illustrated in chapters 4 and 6. Even though the computational budget of computing systems continues to increase, methods which operate over multiple robots simultaneously such as the `dRRT`* approach in chapter 5 show the practical benefit of having compact and efficient planning structures. By pursuing these directions, the future of automation should be able to benefit from practical efficiency and have provably high-quality paths, increasing throughput of these systems even for increasingly unstructured environments. This should allow automated manufacture to push closer to maximum efficiency.

Vita

Andrew Dobson

2012-2017	Ph. D. in Computer Science, Rutgers University
2010-12	M. Sc. in Computer Science, University of Nevada, Reno
2006-10	B. Sc. in Computer Science, University of Nevada, Reno
2006	Graduated from Carson High School, Carson City, Nevada.

2017-2017	Graduate assistant, Department of Computer Science, Rutgers University
2015-2015	Lecturer, Department of Computer Science, Rutgers University
2013-2017	Research fellow, DIMACS, Rutgers University
2012-2013	Teaching assistant, Department of Computer Science, Rutgers University

References

[1] P. Agarwal. Compact Representations for Shortest-Path Queries. In *IROS Workshop on Progress and Open Problems in Motion Planning*, Sept. 2011.

[2] R. Alterovitz, S. Patil, and A. Derbakova. Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[3] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An Obstacle-based PRM for 3D Workspaces. In *WAFR*, pages 155–168, 1998.

[4] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, J.-C. Latombe, and C. Varm. Stochastic Roadmap Simulation: An Efficient Representation and Algorithm for Analyzing Molecular Motion. *Journal of Computational Biology*, 10:257–281, 2003.

[5] B. Balaguer and S. Carpin. A learning method to determine how to approach an unknown object to be grasped. *International Journal of Humanoid Robotics*, 8(3):579–606, 2011.

[6] J. Barraquand and J.-C. Latombe. Robot Motion Planning: A Distributed Representation Approach. *IJRR*, 10(6):628–649, Dec. 1991.

[7] J. Barry, L. Kaelbling, and T. Lozano-Perez. A Hierarchical Approach to Manipulation with Diverse Actions. IEEE Internation Conference on Robotics and Automation, 2013.

[8] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures and Algorithms*, 30(4):532–563, July 2007.

[9] D. Berenson, H. Choset, and J. J. Kuffner. An Optimaztion Approach to Planning for Mobile Manipulation. pages 1187–1192. IEEE International Conference on Robotcis and Automation (ICRA), May 2008.

[10] D. Berenson, S. Srinivasa, and J. Kuffner. Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. 30(12):1435 – 1460, October 2011.

[11] R. Bohlin and L. Kavraki. Path Planning Using Lazy PRM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 521–528, San Francisco, CA, Apr. 2000.

[12] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian Sampling Strategy for Probabilistic Roadmap Planners. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1018–1023, Detroit, MI, May 1999.

[13] M. Branicky, S. M. LaValle, K. Olson, and L. Yang. Quasi-Randomized Path Planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1481–1487, Seoul, Korea, May 2001.

[14] R. Brooks and T. Lozano-Pérez. A Subdivision Algorithm in Configuration Space for Findpath with Rotation. In *IJCAI*, pages 799–803, 1983.

[15] A. Bry and N. Roy. Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty. IEEE International Conference on Robotcis and Automation (ICRA), May 2011.

[16] S. Cambon, R. Alami, and F. Gravot. A Hybrid Approach to Intricate Motion, Manipulation, and Task Planning. *International Journal of Robotics Research*, 28(1):104–126, 2009.

[17] J. Canny. *The Complexity of Robot Motion Planning*. PhD thesis, MIT, Cambridge, MA, 1988.

[18] S. Chaudhuri and V. Koltun. Smoothed Analysis of Probabilistic Roadmaps. *Computational Geometry*, 42(8):731 – 747, 2009.

[19] P. C. Chen and Y. K. Hwang. Practical Path Planning Among Movable Obstacles. In *Proc. of the IEEE Intern. Conf. on Robotics and Automation*, pages 444–449, 1991.

[20] P. Cheng, J. Fink, and V. Kumar. "abstractions and algorithms for cooperative multiple robot planar manipulation". In *Robotics: Science and Systems*, 2008.

[21] H. Chitsaz, S. M. LaValle, D. Balkcom, and M. Mason. Minimum Wheel-Rotation Paths for Differential-Drive Mobile Robots. *IJRR*, 28(1):66–80, 2009.

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

[23] J. B. Cohen, M. Phillips, and M. Likhachev. Planning Single-arm Manipulations with n-Arm Robots. In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA, July 2014.

[24] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, Dec. 2012.

[25] B. Dacre-Wright, J.-P. Laumond, and R. Alami. Motion Planning for a Robot and a Movable Object Amidst Polygonal Obstacles. In *IEEE International Conference on Robotics and Automation*, pages 2474–2480, 1992.

[26] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental Task and Motion Planning: A Constraint-Based Approach. In *Robotics: Science and Systems*, pages 1–6, 2016.

[27] A. Dobson and K. E. Bekris. A Study on the Finite-Time Near-Optimality Properties of Sampling-Based Motion Planners. Tokyo Big Sight, Tokyo, Japan, Nov. 2013. IROS.

[28] A. Dobson and K. E. Bekris. Improving Sparse Roadmap Spanners. In *ICRA*, 2013.

[29] A. Dobson and K. E. Bekris. Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning. *IJRR*, 33, Jan. 2014.

[30] A. Dobson and K. E. Bekris. Planning Representations and Algorithms for Prehensile Multi-Arm Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[31] A. Dobson, T. D. Krontiris, and K. E. Bekris. Sparse Roadmap Spanners. In *WAFR*, Cambridge, MA, June 2012.

[32] A. Dobson, G. Moustakides, and K. E. Bekris. Geometric Probability Results For Bounding Path Quality In Sampling-Based Roadmaps After Finite Computation. In *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2015.

[33] A. Dobson, K. Solovey, R. Shome, D. Halperin, and K. E. Bekris. Scalable Asymptotically-Optimal Multi-Robot Motion Planning. Technical report, arXiv, July 2017.

[34] M. R. Dogar and S. S. Srinivasa. A Framework for Push-Grasping in Clutter. In *Robotics: Science and Systems (RSS)*, 2011.

[35] C. Dornhege, A. Hertle, and B. Nebel. Lazy Evaluation and Subsubmption Caching for Search-based Integrated Task and Motion Planning. In *IROS Workshop on AI-based Robotics*, 2013.

[36] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. 1998.

[37] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[38] J. Fink, M. Ani Hsieh, and V. Kumar. Multi-robot Manipulation via Caging in Environments with Obstacles. In *IEEE International Conference on Robotics and Automation*, 2008.

[39] M. Foskey, M. Garber, M. Lin, and D. Manocha. A Voronoi-based Hybrid Motion Planner. In *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[40] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. FFRob: An Efficient Heuristic for Task and Motion Planning. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[41] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling. Sample-Based Methods for Factored Task and Motion Planning. In *Robotics: Science and Systems (RSS)*, 2017.

[42] S. Ge and Y. Cui. Dynamic Motion Planning for Mobile Robots using Potential Field Method. *Autonomous Robots*, 13:207–222, 2002.

[43] R. Geraerts and M. H. Overmars. A Comparative Study of Probabilistic Roadmap Planners. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V*, pages 43–58. Springer-Verlag, 2003.

[44] R. Geraerts and M. H. Overmars. Creating High-Quality Roadmaps for Motion Planning in Virtual Environments. In *IROS*, pages 4355–4361, Beijing, China, Oct. 2006.

[45] R. Geraerts and M. H. Overmars. Reachability-based Analysis for Probabilistic Roadmap Planners. *Journal of Robotics and Autonomous Systems (RAS)*, 55:824–836, 2007.

[46] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, Oct. 1998.

[47] M. Gharbi, J. Cortés, and T. Siméon. Roadmap Composition for Multi-Arm Systems Path Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[48] F. Gravot and R. Alami. A Method for Handling Multiple Roadmaps and Its Use for Complex Manipulation Planning. In *IEEE Internation Conference on Robotics and Automation (ICRA)*, 2003.

[49] G. Grimmet and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 2001.

[50] C. Grinstead and J. Snell. *Introduction to Probability*. American Mathmatical Society, Providence, RI, 2012.

[51] L. J. Guibas, C. Holleman, and L. E. Kavraki. A Probabilistic Roadmap Planner for Flexible Objects with a Workspace Medial-Axis-Based Sampling Approach. In *IROS*, Oct. 1999.

[52] K. Harada, T. Tsuji, and J.-P. Laumond. A Manipulation Motion Planner for Dual-Arm Industrial Manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 928–934, 2014.

[53] E. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.

[54] K. Hauser and J.-C. Latombe. Multi-modal Motion Planning in Non-Expansive Spaces. In *International Journal of Robotics Research (IJRR)*, volume 29, page 7, 2010.

[55] K. Hauser and V. Ng-Thow-Hing. Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task. *International Journal of Robotics Research*, 30(6):678–698, Feb. 2011.

[56] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J. Artif. Int. Res.*, 14(1):253–302, May 2001.

[57] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The Bridge Test for Sampling Narrow Passages with Probabistic Roadmap Planners. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4420–4426, Taipei, Taiwan, Sept. 2003.

[58] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On Finding Narrow Passages with Probabilistic Roadmap Planners. In *WAFR*, Houston, TX, 1998.

[59] D. Hsu, J.-C. Latombe, and R. Motwani. Path Planning in Expansive Configuration Spaces. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 3, pages 2719–2726, Albuquerque, NM, Apr. 1997.

[60] D. Hsu, J.-C. Latombe, and R. Motwani. Path Planning in Expansive Configuration Spaces. *Int. Journal of Computational Geometry and Applications*, 9(4-5):495–512, 1999.

[61] Y. K. Hwang and N. Ahuja. A Potential Field Approach to Path Planning. *TRA*, 8(1):23–32, Feb. 1992.

[62] L. Jaillet and T. Simeon. Path Deformation Roadmaps. In *WAFR*, New York City, NY, July 2006.

[63] L. Janson, A. Clark, and M. Pavone. Fast Marching Tree: a Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. Technical report, arXiv, Apr. 2014.

[64] L. Janson and M. Pavone. Fast Marching Trees: a Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. International Symposium on Robotics Research, Dec. 2013.

[65] L. Janson, A. Schmerling, A. Clark, and M. Pavone. Fast Marching Tree: a Fast marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. *International Journal of Robotics Research (IJRR)*, 34(7):883–921, 2015.

[66] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical Task and Motion Planning in the Now. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[67] S. Kambhampati and L. S. Davis. Multiresolution Path Planning for Mobile Robots. *JRA*, 2(3):135–145, Sept. 1986.

[68] S. Karaman and E. Frazzoli. Incremental Sampling-based Algorithms for Optimal Motion Planning. In *RSS*, Zaragoza, Spain, 2010.

[69] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *IJRR*, 30(7):846–894, June 2011.

[70] D. Katz, A. Venkatraman, M. Kazemi, D. Bagnell, and A. Stentz. Perceiving, Learning and Exploiting Object Affordances for Autonomous Pile Manipulation. In *Robotics: Science and Systems (RSS)*, 2013.

[71] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of Probabilistic Roadmaps for Path Planning. *IEEE TRA*, 14(1):166–171, 1998.

[72] L. E. Kavraki and J.-C. Latombe. *Probabilistic Roadmaps for Robot Path Planning*, pages 33–53. John Wiley, 1998.

[73] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE TRA*, 12(4):566–580, 1996.

[74] J. Kenney, T. Buckley, and O. Brock. Interactive Segmentation for Manipulation in Unstructured Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1343–1348, 2009.

[75] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *IJRR*, 5(1):90–98, 1986.

[76] A. Kimmel, A. Dobson, Z. Littlefield, A. Krontiris, J. Marble, and K. E. Bekris. PRACSYS: An Extensible Architecture for Composing Motion Controllers and Planners. In *SIMPAR*, Tsukuba, Japan, 11/2012 2012.

[77] D. Koditschek. Robot Planning and Control via Potential Functions. In *The Robotics Review 1*, pages 349–367. MIT Press, 1989.

[78] Y. Koga, K. Kondo, J. J. Kuffner, and J.-C. Latombe. Planning Motions with Intentions. In *Proc. of SIGGRAPH*, pages 395–408, 1994.

[79] Y. Koga and J.-C. Latombe. On Multi-arm Manipulation Planning. In *Prof. of the IEEE Intern. Conference on Robotics and Automation (ICRA)*, 1994.

[80] G. Konidaris, L. Kaelbling, and T. Lozano-Perez. Constructing Symbolic Representations for High-Level Planning. In *Association for the Advancement of Artificial Intelligence (AAAI) conference*, 2014.

[81] M. Koval, N. Pollard, and S. S. Srinivasa. Pose Estimation for Cotact Manipulation with Manifold Particle Filters. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013.

[82] M. Koval, N. Pollard, and S. S. Srinivasa. Pre- and Post-Contact Policy Decomposition for Planar Contact Manipulation Under Uncertainty. In *Robotics: Science and Systems (RSS)*, 2014.

[83] A. Krontiris and K. E. Bekris. Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 05/2016 2016.

[84] A. Krontiris, R. Luna, and K. Bekris. From Feasibility Tests to Path Planners for Multi-agent Pathfinding. pages 114–122, 01 2013.

[85] A. M. Ladd and L. E. Kavraki. Measure Theoretic Analysis of Probabilistic Path Planning. *IEEE TRA*, 20(2):229–242, Apr. 2004.

[86] K. Lakshmanan, A. Sachdev, Z. Xie, D. Berenson, K. Goldberg, and P. Abbeel. *A Constraint-Aware Motion Planning Algorithm for Robotic Folding of Clothes*, pages 547–562. Springer International Publishing, Heidelberg, 2013.

[87] F. Lamiraux and L. E. Kavraki. Planning Paths for Elastic Objects Under Manipulation Constraints. *International Journal of Robotics Research*, 20(3):188–208, 2001.

[88] F. Lamiraux and J.-P. Laumond. On the Expected Complexity of Random Path Planning. In *ICRA*, pages 3306–3311, 1996.

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

[90] S. LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998.

[91] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[92] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. WAFR, 2000.

[93] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *IJRR*, 20:378–400, May 2001.

[94] S. Leroy, J. Laumond, and Siméon. Multiple Path Coordination for Mobile Robots: A Geometric Algorithm. Barcelona, Catalonia, Spain, 1999. International Joint Conference on Artificial Intelligence (IJCAI).

[95] P. Leven and S. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Transactions on Robotics and Automation*, 19(6):1020–1026, 2003.

[96] M. Levinh, J. Scholz, and M. Stilman. Hierarchical Decision Theoretic Planning for Navigation Among Movable Obstacles. In *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, 2012.

[97] Y. Li and K. E. Bekris. Learning Approximate Cost-to-Go Metrics To Improve Sampling-based Motion Planning. In *IEEE ICRA*, Shanghai, China, 9-13 May 2011.

[98] Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically Optimal Sampling-based Kinodynamic Planning. *International Journal of Robotics Research (IJRR)*, 35:528–564, 04/2016 2016.

[99] J.-M. Lien and Y. Lu. Planning Motion in Point-Represented Contact Spaces Using Approximate Star-Shaped Decomposition. IROS, Oct. 2009.

[100] S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 3251–3257, 2004.

[101] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, pages 108–120, 1983.

[102] T. Lozano-Pérez and L. P. Kaelbling. A constraint-based method for solving sequential manipulation planning problems. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3684–3691, 2014.

[103] J. Marble and K. E. Bekris. Asymptotically Near-Optimal Planning with Probabilistic Roadmap Spanners. *IEEE TRO*, 29:432–444, 2013.

[104] J. D. Marble and K. E. Bekris. Asymptotically Near-Optimal is Good Enough for Motion Planning. In *ISRR*, Flagstaff, AZ, Aug. 2011.

[105] J. D. Marble and K. E. Bekris. Computing Spanners of Asympotically Optimal Probabilistic Roadmaps. In *IEEE/RSJ IROS*, San Francisco, CA, Sept. 2011.

[106] J. D. Marble and K. E. Bekris. Towards Small Asymptotically Near-Optimal Roadmaps. In *IEEE ICRA*, Minnesota, MN, May 2012.

[107] Z. McCarthy, T. Bretl, and S. Hutchinson. Proving path non-existence using sampling and alpha shapes. In *ICRA*, May 2012.

[108] O. Nechushtan, B. Raveh, and D. Halperin. Sampling-Diagrams Automata: a Tool for Analyzing Path Quality in Tree Planners. In *WAFR*, Singapore, Dec. 2010.

[109] C. Nielsen and L. E. Kavraki. A Two-Level Fuzzy PRM for Manipulation Planning. In *IEEE/RSJ IROS*, pages 1716–1722, Japan, 2000.

[110] D. Nieuwenhuisen and M. H. Overmars. Using Cycles in Probabilistic Roadmap Graphs. In *IEEE ICRA*, pages 446–452, 2004.

[111] N. Nilsson. Shakey the Robot. Technical Report 323, SRI International, 1984.

[112] J. Ota. Rearrangement Planning of Multiple Movable Objects. In *Prof. of the IEEE Intern. Conference on Robotics and Automation (ICRA)*, 2004.

[113] E. Paljug, T. Sugar, V. Kumar, and X. Yun. Important Considerations in Force Control with Applications to Multi-Arm Manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1270–1275, 1992.

[114] D. Peleg and A. Schäffer. Graph Spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[115] J. Peng and S. Akella. Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths. *The International Journal of Robotics Research*, 24(4):295–310, 2005.

[116] M. Penrose. *Random Geometric Graphs*. 2003.

[117] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-Based Roadmap of Trees for Parallel Motion Planning. *IEEE TRA*, 21(4):587–608, 2005.

[118] E. Plaku and G. Hager. Sampling-based Motion Planning with Symbolic, Geometric, and Differential Constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[119] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[120] S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance. International Symposium of Robotics Research (ISRR), 2008.

[121] B. Raveh, A. Enosh, and D. Halperin. A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm. *IEEE TRO*, 27(2):365–370, 2011.

[122] J. H. Reif. Complexity of the Generalized Mover's Problem. In *FOCS*, pages 421–427, 1979.

[123] E. Rimon and D. Koditschek. Exact Robot Navigation Using Artificial Potential Functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, Oct. 1992.

[124] L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *International Colloquim on Automata, Languages and Programming (ICALP)*, pages 261–272. Springer, 2005.

[125] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[126] O. Salzman and D. Halperin. Asymptotically Near-Optimal RRT for Fast, High-quality Motion Planning. Hong Kong, China, June 2014. ICRA.

[127] G. Sánchez and J.-C. Latombe. On Delaying Collision Checking in PRM Planning: Application to Multi-Robot Coordination. *International Journal of Robotics Research*, 21(1):5–26, 2002.

[128] G. Sanchez and J.-C. Latombe. Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems. In *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, pages 2112–2119, 2002.

[129] L. A. Santalo. *Integral Geometry and Geometric Probability*, volume 1 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley Publishing Company, Reading, Massachusets, 1976.

[130] E. Schmerling, L. Janson, and M. Pavone. Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 2574–2581, 2015.

[131] E. Schmerling, L. Janson, and M. Pavone. Optimal sampling-based motion planning under differential constraints: The driftless case. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 2368–2375, 2015.

[132] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard. Optimal, Sampling-based Manipulation Planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3426–3432. IEEE, 2017.

[133] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of Homotopy Classes with Probabilistic Roadmap. In *IEEE/RSJ IROS*, pages 2317–2322, 2002.

[134] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers. *The International Journal of Robotics Research*, 2(3):46–75, 1983.

[135] F. Schwarzer, M. Saha, and J.-C. Latombe. Adaptive Dynamic Collision Checking for Single and Multiple Articulated Robots in Complex Environments. *IEEE Transactions on Robotics*, 21(3):338–353, 2005.

[136] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. Manipulation Planning with Probabilistic Roadmaps. *International Journal of Robotics Research (IJRR)*, 23(8):729–746, 2004.

[137] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility-based Probabilistic Roadmaps for Motion Planning. *Advanced Robotics Journal*, 41(6):477–494, 2000.

[138] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual-arm Manipulation: A Survey. In *Robotics and Autonomous Systems*, volume 60, pages 1340–1353, 2012.

[139] K. Solovey, O. Salzman, and D. Halperin. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *I. J. Robotics Res.*, 35(5):501–513, 2016.

[140] K. Solovey, O. Salzman, and D. Halperin. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *I. J. Robotics Res.*, 35(5):501–513, 2016.

[141] K. Solovey, O. Salzman, and D. Halperin. New Perspective on Sampling-based Motion Planning via Random Geometric Graphs. *CoRR*, abs/1602.05460, 2016.

[142] K. Solovey, O. Salzman, and D. Halperin. New Perspective on Sampling-based Motion Planning via Random Geometric Graphs. In *Robotics Science and Systems (RSS)*, 2016.

[143] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[144] A. Stentz. The Focused D* Algorithm for Real-Time Replanning. 1995.

[145] M. Stilman and J. J. Kuffner. Planning Among Movable Obstacles with Artificial Constraints. In *Proc. of the Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2006.

[146] M. Stilman, J. Schamburek, J. J. Kuffner, and T. Asfour. Manipulation Planning Among Movable Obstacles. In *IEEE International Conference on Robotics and Automation*, 2007.

[147] S. Sundaram, I. Remmler, and N. M. Amato. Disassembly Sequencing Using a Motion Planning Approach. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1475–1480, Washington, D.C., May 2001.

[148] P. Švestka. *Robot Motion Planning using Probabilistic Road Maps*. PhD thesis, Utrecht University, the Netherlands, 1997.

[149] P. Svestka and M. Overmars. Coordinated Path Planning for Multiple Robots. *Robotics and Autonomous Systems*, 23:125–152, 1998.

[150] S. Tang and V. Kumar. A Complete Algorithm for Generating Safe Trajectories for Multi-Robot Teams. In *International Symposium on Robotics Research (ISRR)*, Sestri Levante, Italy, Sept. 2015.

[151] R. Tedrake, I. Manchester, M. Tobenkin, and J. Roberts. LQR-Trees: Feedback Motion Planning via Sums of Squares Verifcation. *International Journal of Robotics Research*, pages 1038–1052, 2010.

[152] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann. Humanoid Motion Planning for Dual-arm Manipulation and Re-grasping Tasks. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2464–2470, Oct 2009.

[153] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[154] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha. *Reciprocal n-body collision avoidance*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 3–19. Star edition, 2011.

[155] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems V*, 2009.

[156] J. van den Berg, M. Stilman, J. J. Kuffner, M. Lin, and D. Manocha. Path Planning Among Movable Obstacles: A Probabilistically Complete Approach. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2008.

[157] G. Varadhan and D. Manocha. Star-shaped Roadmaps: A Deterministic Sampling Approach for Complete Motion Planning. *RSS*, 2005.

[158] W. Vega-Brown and N. Roy. Asymptotically Optimal Planning under Piecewise-analytic Constraints. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[159] G. Wagner and H. Choset. Subdimensional Expansion for Multirobot Path Planning. *Artificial Intelligence Journal*, 219:1024, 2015.

[160] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space. In *ICRA*, pages 1024–1031, Detroit, MI, May 1999.

[161] J. Wolfe, B. Marthi, and S. Russell. Combined Task and Motion Planning for Mobile Manipulation. In *Proceedings of the Twentieth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'10, pages 254–257. AAAI Press, 2010.

[162] J. Yu and S. M. LaValle. Multi-agent Path Planning and Network Flow. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2012.

[163] J. Yu and S. M. LaValle. Planning Optimal Paths for Multiple Robots on Graphs. 04 2012.

[164] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. Bagnell, and S. S. Srinivasa. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *International Journal of Robotics Research (IJRR)*, 2013.