Informed Anytime Search for Continuous Planning Problems

by

Jonathan D. Gammell

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Informed Anytime Search
for Continuous Planning Problems

Jonathan D. Gammell
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2017

Navigating uncontrolled dynamic environments is a major challenge in robotics. Success requires solving many different technical problems and *path planning* is a key component of most autonomous solutions. Path planning is the task of finding a path through the environment that allows a robot to reach its desired position. This path must avoid obstacles and be executable by the robot while often also reducing a specified cost (e.g., path length). The difficulty of meeting these requirements *reliably* and *quickly* in continuous state spaces has made path planning an active area of research in robotics.

Two popular path planning approaches are informed graph-based search and anytime sampling-based planning. Informed graph-based searches, such as A*, are efficient but use *a priori* approximations of the problem domain. If these approximations are chosen incorrectly, then the algorithms may be unable to find a (suitable) solution or take a prohibitively long time to do so. Anytime sampling-based planners, such as RRT*, continuously improve their approximations of the problem domain but perform inefficient searches. These searches simultaneously search the entire problem domain and can become prohibitively expensive in large or high-dimensional planning problems, such as when planning for high-DOF manipulation arms.

This thesis demonstrates how planning in continuous planning problems can be improved by unifying the informed graph-based search and anytime sampling-based planning approaches. It investigates various ways to use heuristics to focus and order the search of almost-surely asymptotically optimal sampling-based planners. The theoretical and experimental advantages of this informed anytime sampling-based search are demonstrated through the planning algorithms, Informed RRT*, Batch Informed Trees (BIT*), and Sorted RRT* (SORRT*).

Acknowledgements

This Ph. D. thesis would not have been possible without the support of my colleagues, family, and friends. It is my pleasure to take this opportunity to begin repaying my debts by acknowledging their generous contributions.

First of all I would like to thank my supervisor, Prof. Timothy D. Barfoot, for his guidance throughout my Ph. D. This work was only possible because he gave me the freedom to pursue the research wherever it led and for that I will be forever grateful. He has never accepted anything less than my best and his intellectual support, dedication to rigour, and tireless editing prowess are reflected throughout this document. It goes without saying that any remaining hyphenation-errors are solely my own.

I would like to thank the members of my Doctoral Examination Committee, Prof. Christopher J. Damaren and Prof. Gabriele M. T. D'Eleuterio, for sharing their knowledge and insight throughout my degree. I would also like to thank my examiners, Prof. Lydia Kavraki and Prof. Sheila McIlraith, for considering my thesis. This work has been made immeasurably better by their feedback and questions.

Much of the work in this document was done in collaboration with Prof. Siddhartha S. Srinivasa of Carnegie Mellon University. His seemingly unbounded knowledge of the planning literature has been an immeasurable help in situating the work. The experiments on HERB would not have been possible without the generous support of his students in the Personal Robotics Lab, especially Christopher Dellin, Jennifer King, and Michael Koval. It was also through his connections at CMU that I had the opportunity to work with Sanjiban Choudhury of the AIRLab, whose passion and excitement has never failed to motivate me.

There are many exceptional people from my time at UTIAS who also deserve recognition for their help. Thank you Chi Hay Tong for consistently showing me how it should be done. Thank you Chris Ostafew for being such a calm and steady example. Thank you Pete Berczi for lending a helpful and critical ear. Thank you Adam Trischler for being up for anything. Thank you Pete Szabo for being a great housemate and an even better friend. Thank you Vidya Menon for always believing in me and never letting me forget it. Space unfortunately limits further personalized messages, but everyone in ASRL with whom I shared time deserves thanks for making it such an enjoyable environment (i.e., Thank you Andrew, Colin, Paul, Keith, Goran, Hang, Braden, Rehman, Sean, Mike 0, Kirk, Pat, Tyler, Kai, Katarina, François, Mike 1, and all our visitors).

To my friends, thank you Doug, Jeremy, Tim, Kyle, Craig, Scott, Steve, Adam, Kelvin, Hayley and Hai, Andrea, Susie (and Sadie), and everyone else throughout the years who cannot possibly know how much they have helped.

Finally, I would not have made it here without the love and support of my parents, my siblings, Monique and Andrew, and my nieces and nephews, Luc, Kaela, Isabelle, and Nicholas. Thank you all.

Contents

List of Definitions

List of Lemmas, Theorems, and Corollaries

List of Algorithms

List of Figures

Notation

General Notation

a	:	Lower-case variables in this font are scalars.
X	:	Upper-case variables in this font are sets or spaces. (e.g., the search space of the planning algorithm).
$\mathbf{a}$	:	Lower-case variables in this font are column vectors.
$\mathbf{A}$	:	Upper-case variables in this font are matrices.

Specific Symbols

$\emptyset$	:	The empty set.
$\mathbf{1}$	:	The identity matrix.
$\mathbf{1}_p$	:	The p-th column of the identity matrix.
i	:	The number of iterations performed by a planning algorithm.
n	:	The dimension of the search space.
q	:	The number of samples generated by a planning algorithm.
ζ_n	:	The Lebesgue measure of an n-dimensional unit ball.
$\mathcal{Q}$	:	An ordered queue.
V	:	A set of vertices in a graph embedded in the search space, $V \subseteq X$.
E	:	A set of edges in a graph, $E \subseteq (V \times V)$.
$\mathcal{T}$	:	A graph without simple cycles (i.e., a tree) defined by a sets of vertices and edges, $\mathcal{T} = (V, E)$.

Specific Functions & Operators

$P(\cdot)$	:	The probability of an event.
$E[\cdot]$	:	The expectation operator.
$\Gamma(\cdot)$	:	The gamma function, an extension of the factorial function to real numbers (Euler, 1738).
$\|\cdot\|_p$	:	The L^p norm of a vector.
$\|\cdot\|$	:	The cardinality of a set or a graph.
$\lambda(\cdot)$	:	The Lebesgue measure of a set (e.g., the *volume*).
$\mathbf{x} \sim \mathcal{U}(X)$	:	A sample, $\mathbf{x}$, randomly generated from a uniform distribution over a set, X.
$X \xleftarrow{+} \{\mathbf{x}\}$	:	A compact representation of the set compounding operation, $X \leftarrow X \cup \{\mathbf{x}\}$, used in algorithms.
$X \xleftarrow{-} \{\mathbf{x}\}$	:	A compact representation of the set compounding operation, $X \leftarrow X \setminus \{\mathbf{x}\}$, used in algorithms.

Cost Functions

$\widehat{c}(\mathbf{x}, \mathbf{y})$	:	An admissible estimate of the incremental cost to reach a state, $\mathbf{y} \in X$, from another state, $\mathbf{x} \in X$. Bounds the optimal cost from below.
$c(\mathbf{x}, \mathbf{y})$	:	The optimal incremental cost to reach a state, $\mathbf{y} \in X$, from another state, $\mathbf{x} \in X$.
$\widehat{g}(\mathbf{x})$	:	An admissible estimate of the cost to come to a state, $\mathbf{x} \in X$, from the start. Bounds the optimal cost from below.
$g(\mathbf{x})$	:	The optimal cost to come to a state, $\mathbf{x} \in X$, from the start.
$g_{\mathcal{T}}(\mathbf{x})$	:	The cost to come to a state, $\mathbf{x} \in X$, from the start through the current tree, $\mathcal{T}$. If a state is not in the tree, its cost-to-come is taken to be infinity. Bounds the optimal cost from above.
$\widehat{h}(\mathbf{x})$	:	An admissible estimate of the cost to go to the goal from a state, $\mathbf{x} \in X$. Bounds the optimal cost from below.
$h(\mathbf{x})$	:	The optimal cost to go to the goal from a state, $\mathbf{x} \in X$.
$\widehat{f}(\mathbf{x})$	:	An admissible estimate of the cost of a path from the start to the goal constrained to pass through a state, $\mathbf{x} \in X$. Bounds the optimal cost from below.
$f(\mathbf{x})$	:	The optimal cost of a path from the start to the goal constrained to pass through a state, $\mathbf{x} \in X$.

Chapter 1

Introduction

Recent technical advancements have lead to autonomous robotic solutions to a variety of problems, including household cleaning (Fig. 1.1a; Hammill and Hendricks, 2014), warehouse automation (Fig. 1.1b; Wohlsen, 2014), and aspects of extraterrestrial exploration (Fig. 1.2; Bostock et al., 2014). Current research is attempting to extend these successes to more challenging tasks, including transportation (Fig. 1.3a; LaFrance, 2015) and last-mile logistics (Fig. 1.3b; Wingfield, 2015). These tasks pose significant new problems for robotics as they require extended periods of operation in dynamic and uncontrolled environments. Autonomous solutions will require solving a number of difficult subproblems, including the *path-planning problem*.

Path planning is the task of finding a sequence of poses (i.e., a *path*) that allows a robot to move between specified positions (i.e., from a *start* to a *goal*). This path must not only avoid obstacles (i.e., be *collision-free*) but also represent motions the robot can perform (i.e., be *feasible*). Multiple such *valid* paths exist in many planning problems and it is common to seek the path that minimizes a chosen cost function (i.e., the *optimal* path). A common optimization objective is the minimization of path length.

(a)

(b)

Figure 1.1: An iRobot Roomba vacuum, (a), and the Amazon Robotics mobile warehouse platform, (b). Photographs courtesy of iRobot (2015) and Kelly (2014), respectively.

The difficulty of a planning problem depends on the number of robot positions (i.e., *states*) that could possibly belong to a solution (i.e., the size of the *search space*). This search space is often theoretically continuous as most robots can be infinitesimally repositioned. It is often also unbounded (e.g., planning outdoors, Fig. 1.3c) and/or high dimensional (e.g., planning for manipulation, Fig. 1.3d).

Most planning algorithms attempt to simplify this search. Common simplifications include considering only a subset of the problem (i.e., a *local* search) and/or reducing the possible robot poses (e.g., graph-based and sampling-based *approximations*). These simplifications improve real-world performance but reduce the formal properties of the algorithm. Chapter 2 uses these simplifications to classify and evaluate existing planning techniques, focusing on the performance of graph-based and sampling-based approximations in continuous search domains.

Graph-based approximations allow for the use of informed graph-based searches, such as A* (Hart et al., 1968). These algorithms not only find the optimal solution in the chosen representation (i.e., they are *resolution optimal*) but also have formal guarantees on the efficiency of their search. A* is the *optimally efficient* search of a given graph (Hart et al., 1968) for any heuristic.

This efficient search allows graph-based algorithms to be successful on many continuous planning problems despite the difficulty of approximating a problem *a priori*. Low resolution approximations can be searched quickly but result in lower quality *continuous* solutions (if they contain one). High resolution approximations yield high-quality continuous solutions

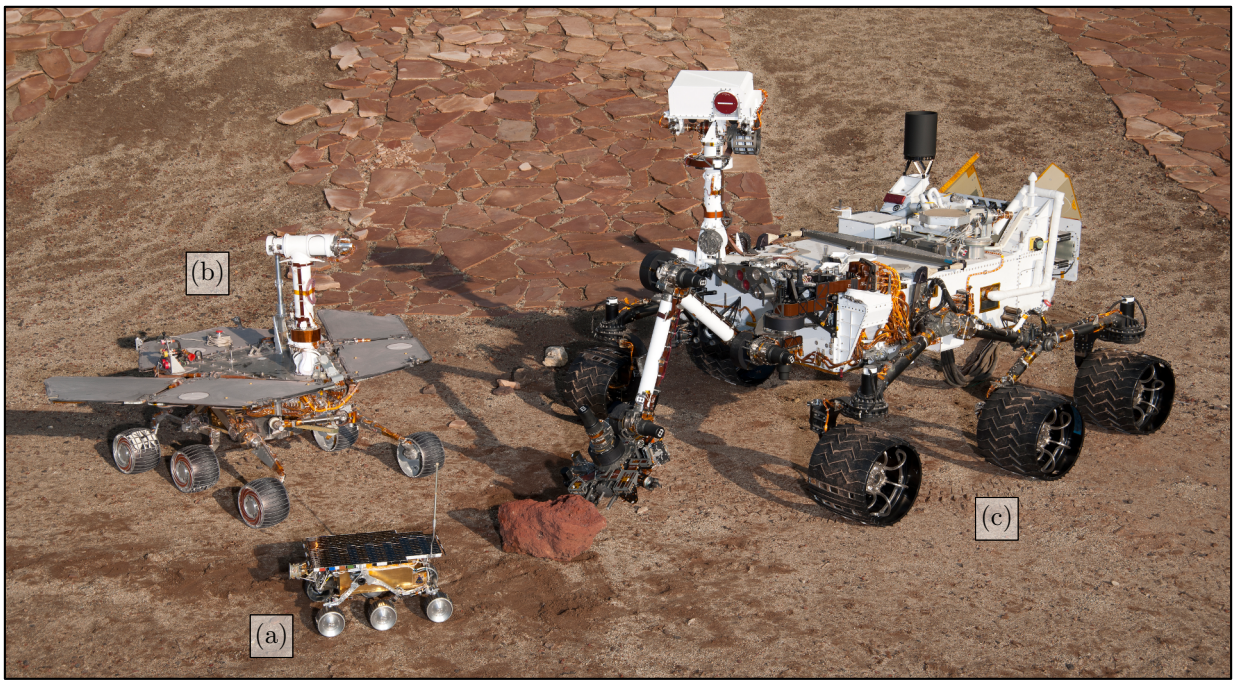
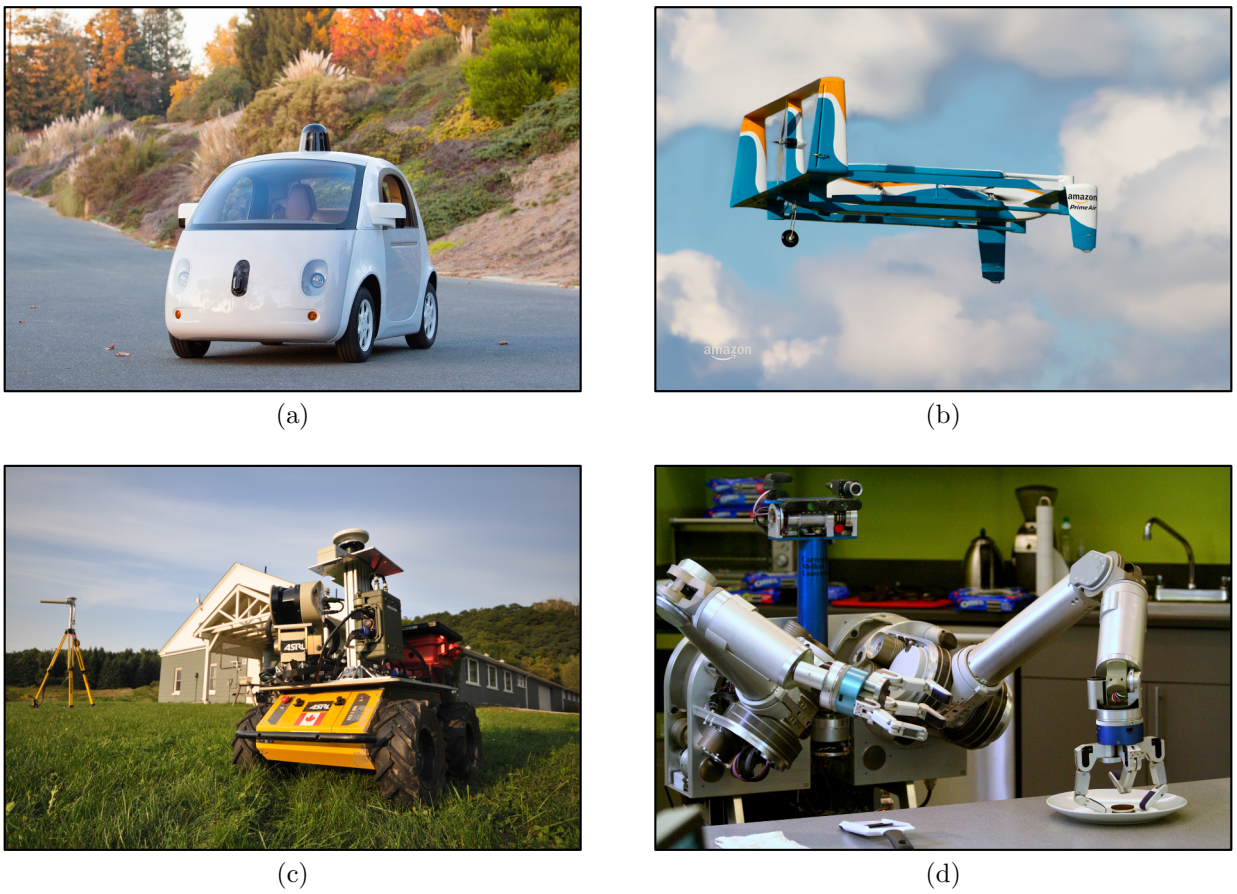


Figure 1.2: Prototypes of three generations of Mars rovers: Sojourner, (a), a Mars Exploration Rover, (b), and the Mars Science Laboratory, (c). Photograph courtesy of NASA JPL, 2012.

(Bertsekas, 1975) but create complex graphs that can be prohibitively expensive to search. These problems are only amplified by the exponential growth of graph size with state dimension, the *curse of dimensionality* (Bellman, 1954, 1957).

Sampling-based approximations allow planners to increase resolution incrementally until a (suitable) solution is found (i.e., they are *anytime*) and avoid the computational cost of unnecessary approximations. These algorithms, such as Probabilistic Roadmaps (PRM; Kavraki et al., 1996), Rapidly exploring Random Trees (RRT; LaValle and Kuffner Jr., 2001), and RRT* (Karaman and Frazzoli, 2011) have probabilistic performance guarantees. With an infinite number of samples, they have a unity probability of finding a solution if one exists (i.e., they are *probabilistically complete*) and PRM and RRT* can have a unity probability of asymptotically converging to the optimal solution, if one exists (i.e., they may be *almost-surely asymptotically optimal*).

This incremental representation allows sampling-based planners to be successful on many continuous planning problems despite the inefficiency of their search. Incremental planners



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

Figure 1.3: The Google Self-Driving Car, (a), Amazon Prime Air delivery drone, (b), the University of Toronto ASRL Husky, (c), a wheeled robot outfitted for extended, long-range traverses (Berczi et al., 2013; Gammell et al., 2013a,b), and the CMU Personal Robotics Lab's HERB, (d), a mobile 14-DOF manipulation platform (Srinivasa et al., 2012). Photographs (a) and (b) courtesy of Google (2015) and Amazon (2015), respectively.

generally search the entire problem domain simultaneously (i.e., they are *space filling*) and waste computational effort searching regions of the problem that are never used to find a solution. This results in algorithms that almost-surely converge asymptotically to the optimal solution by asymptotically finding optimal paths to *every* state in the problem domain. This unordered and unfocused search is prohibitively expensive in many real problems, such as planning for manipulation or in unbounded environments.

This thesis studies how anytime almost-surely asymptotically optimal planners can be unified with informed graph-based searches to improve performance on single-query continuous planning problems. At a high level, it makes the following novel contributions:

1. Investigates the necessary conditions for incremental RRT*-style planners to improve a solution, defining the omniscient set, informed sets, and the precision and recall of informed sampling and provides an upper bound on the probability of these planners improving a solution at any iteration (Chapter 3).
2. Develops techniques to focus the search for improvements in problems seeking to minimize path length, in the process developing bounds on the expected convergence rate of RRT* algorithms, proving that the original RRT* has sublinear convergence, and developing an algorithm that can have linear convergence, Informed RRT* (Chapter 4).
3. Unifies graph-search and sampling-based techniques to develop Batch Informed Trees (BIT*), a sampling-based algorithm that searches a continuous planning problem in order of potential solution quality (as in A*) while maintaining anytime representations and almost-sure asymptotic optimality (as in RRT*) and the related extension, Sorted RRT* (SORRT*) (Chapter 5).

All the algorithms developed in this thesis are available publicly in the Open Motion Planning Library (OMPL; Şucan et al., 2012).

Chapter 3 investigates necessary conditions for anytime almost-surely asymptotically optimal planners, such as RRT*, to improve a solution. It shows that after an initial number of samples it is necessary to sample a state that belongs to a theoretically better solution. While exact knowledge of these necessary samples requires solving the problem, this *omniscient* set can be approximated with an *informed* set using heuristics (Fig. 1.4). The accuracy of an informed set will depend on how completely and tightly it estimates the omniscient set (i.e., its *recall* and *precision*). If an informed set completely contains the omniscient set (i.e., 100% recall) it is *admissible*. The probability of sampling an admissible informed set is an upper bound on the probability of improving a solution with the tightness of the bound depending on the set's precision. An efficient method to directly sample admissible informed sets defined by the L^2 norm is presented for problems seeking to minimize path length.

Chapter 4 develops Informed RRT* as a demonstration of using direct informed sampling for problems seeking to minimize path length. The algorithm maintains RRT*'s probabilistic completeness and almost-sure asymptotic optimality while focusing the search for

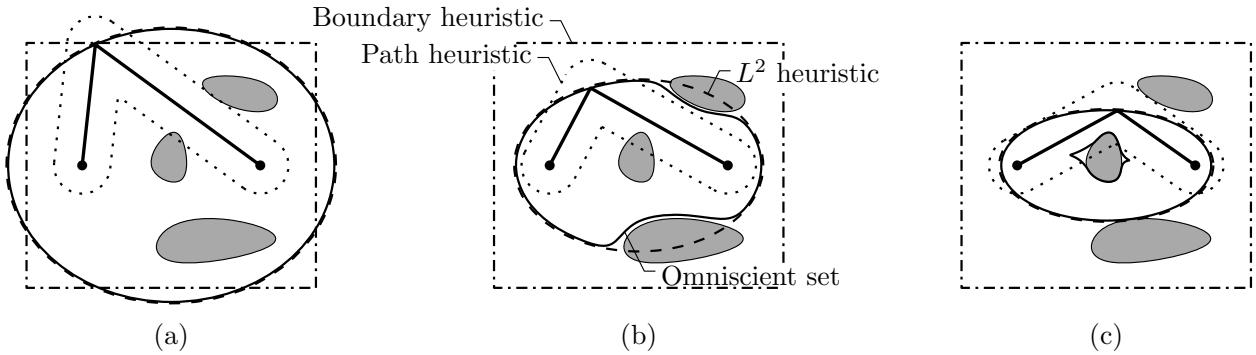
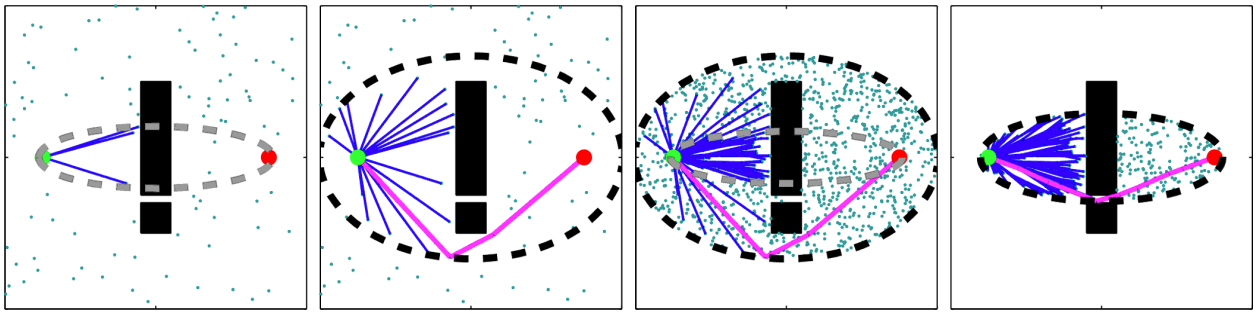


Figure 1.4: An illustration of how the set of states that can improve the current solution (the *omniscient* set) shrinks as better solutions are found. Common search domains used when minimizing path length are labelled by their (implicit) heuristic. Boundary heuristics (*a priori* limits on the state space) commonly contain the entire omniscient set (i.e., high recall) but may include many erroneous states as the omniscient set shrinks with solution improvement (i.e., low precision). Path heuristics (searches limited to the neighbourhood of the current solution) commonly have few erroneous states (i.e., high precision) but do not include the optimum if it belongs to a different homotopy class (i.e., low recall). The L^2 heuristic always contains the entire omniscient set (i.e., 100% recall) and shrinks along with the omniscient set as a function of the current solution (i.e., high precision). It is exactly equal to the omniscient set in the absence of obstacles (i.e., 100% recall *and* precision). The search domain without any heuristic (i.e., the zero heuristic) is infinite for unbounded problems and equivalent to the state-space limits for problems with inherent boundaries.

improvements to the L^2 informed set. Analysis shows that while RRT* always has sublinear convergence to the optimum, Informed RRT* converges linearly (i.e., faster) for some problems. The practical implications of this improved convergence is demonstrated on both abstract problems and experiments for the CMU Personal Robotic Lab's Home Exploring Robot Butler (HERB; Srinivasa et al., 2012), a 14 degree-of-freedom (DOF) mobile manipulation platform. These experiments show that Informed RRT* outperforms existing techniques as state dimension increases, especially in problems with large planning domains.

These experiments also highlight the *limitations* of only using heuristics to focus the search for improvements. Searching in the order given by a sequence of samples causes RRT-based algorithms to consider many different paths simultaneously and often results in circuitous initial solutions. These solutions define large informed sets and for many planning problems provide little opportunity to focus the search. In comparison, informed graph-based searches such as A* are able to use heuristics to focus their *entire* search. Considering states in the order of their potential solution quality allows these algorithms to avoid *unnecessary* search effort and quickly find high-quality solutions.

Chapter 5 combines sampling-based planners and graph-search techniques to develop BIT* as an informed anytime sampling-based search (Fig. 1.5). BIT* approximates a search space with an *edge-implicit* random geometric graph (RGG; Penrose, 2003) defined by batches of samples. The accuracy of this RGG improves incrementally as new samples are added and provides an anytime approximation of the continuous planning problem. This changing



The search expands outwards from the minimum solution propsed by a heuristic. (a)

The search stops when a solution is found or the batch of samples has been completely searched. (b)

A batch of new samples is then added and the search is restarted. (c)

This process repeats indefinitely, focusing both the approximation and search of the domain. (d)

Figure 1.5: A simple example of the ordered sampling-based search performed by BIT*. The start and goal states are shown as green and red, respectively, while the current solution is highlighted in magenta. The informed set is shown as a black dashed line, while the progress of the current batch is illustrated as a grey dashed line. Fig. (a) shows the growing search of the first batch of samples, and (b) shows the first search ending when a solution is found. By ordering the search an initial solution is found without considering states that are outside the resulting informed set. Fig. (c) shows the search continuing after pruning the representation and increasing its accuracy with a new batch of samples while (d) shows this second search ending when an improved solution is found.

approximation is searched efficiently in order of potential solution quality by using incremental graph-search techniques that reuse information, such as Lifelong Planning A* (LPA*; Koenig et al., 2004). The result is a sampling-based planner that searches in order of potential solution quality while almost-surely converging asymptotically to the optimal solution in an anytime manner.

In addition to being a unification of graph-based and sampling-based approaches (Fig. 2.2), BIT* can be viewed as a generalization of existing almost-surely asymptotically optimal planners. With a batch size of one sample it is equivalent to a version of Informed RRT* and with a single batch of samples it is equivalent to a version of Fast Marching Tree (FMT*; Janson et al., 2015b). It can also be extended to modify existing algorithms, as illustrated with SORRT*, an ordered version of Informed RRT*.

The benefits of performing an ordered search on an anytime approximation are demonstrated on both abstract problems and experiments for HERB. The results show that BIT* finds better solutions faster than existing almost-surely asymptotically optimal planners and also RRT, especially in high dimensions. The only tested planner that found solutions faster than BIT* was RRT-Connect (Kuffner Jr. and LaValle, 2000), a bidirectional version of RRT that is *not* almost-surely asymptotically optimal.

Chapters 3 and 4 consist of ideas first published as Gammell et al. (2014b) at the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). This work has been combined with supporting technical reports (Gammell and Barfoot, 2014; Gammell et al., 2014c) and expanded in preparation for submission as a journal paper (Gammell

et al., 2017b). Chapter 5 consists of ideas first presented as Gammell et al. (2014a) at the Information-based Grasp and Manipulation Planning Workshop at the 2014 Robotics: Science and Systems Conference (RSS). This work was published as Gammell et al. (2015) at the 2015 IEEE International Conference on Robotics and Automation (ICRA) and has been expanded in preparation for submission as a journal paper (Gammell et al., 2017a).

Chapter 2

Background

or: The difficulty of finding good solutions fast

This chapter reviews planning algorithms as they relate to robotics and *feasible* and *optimal* planning problems in continuous state spaces (Section 2.1). Feasible problems seek a path through a search space that connects a start to a goal while avoiding obstacles and obeying the differential constraints of the robot. Optimal problems seek the feasible path that minimizes a chosen cost function (e.g., path length). By definition, solving an optimal planning problem requires solving the underlying feasible problem.

Planning problems in robotics are often in search spaces that are continuous and/or without strict boundaries. In order to reduce search complexity many algorithms use simplified approximations of the search space (Fig. 2.2). This facilitates solving problems in real time but negatively affects the formal guarantees of the algorithms. Popular approximations include graph-based (Section 2.2) and sampling-based (Section 2.3) discretizations.

Another method to reduce search complexity is to restrict the search space artificially. These *local* search algorithms find good solutions quickly in many practical applications; however, they provide no *global* guarantees and will not be discussed in detail in this thesis. Examples include Covariant Hamiltonian Optimization for Motion Planning (CHOMP; Ratliff et al., 2009; Zucker et al., 2013), Stochastic Trajectory Optimization for Motion Planning (STOMP; Kalakrishnan et al., 2011), and constrained trajectory optimization (Schulman et al., 2013).

This thesis specifically focuses on planning algorithms with global optimality guarantees. It combines informed graph searches, such as A* (Hart et al., 1968), with sampling-based planners, such as RRT* (Karaman and Frazzoli, 2011), to develop informed asymptotically optimal searches with anytime performance for continuous planning problems. While this work ultimately unifies the two approaches, it is presented as extensions to sampling-based planning (Section 2.4).

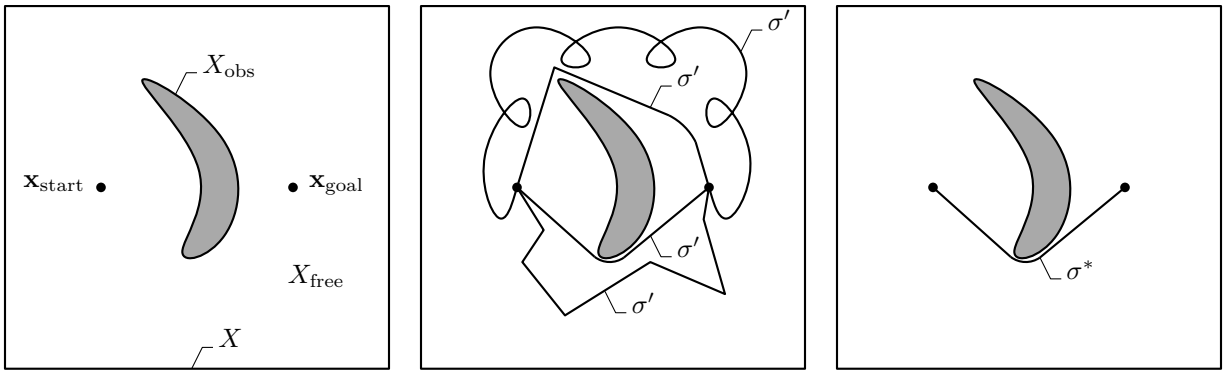


Figure 2.1: An illustration of a simple planning problem, (a), defined by a search space, X, a start, $\mathbf{x}_{\text{start}}$, a goal, $\mathbf{x}_{\text{goal}}$, and obstacles, $X_{\text{obs}} \subset X$. Solutions must pass solely through the set of states not in collision with an obstacle, $X_{\text{free}} \coloneqq \text{cl}\,(X \setminus X_{\text{obs}})$. For this problem there exists an uncountable number of feasible solutions, σ', (b), but only one optimal solution with respect to path length, σ^*, (c).

2.1 Planning Problems

Formal definitions of the feasible and optimal planning problems (Fig. 2.1) are given in Definitions 2.1 and 2.2. These definitions are expressed in the state space of a robot but can be posed in other representations including configuration space (Lozano-Pérez, 1983). The goal of these problems may be a single goal state (e.g., a pose for a mobile robot) or any state in a goal *region* (e.g., the set of joint angles that give a redundant manipulator a desired end-effector position).

Definition 2.1 (The feasible planning problem)**.** *Let $X \subseteq \mathbb{R}^n$ be the n-dimensional state space of the planning problem, $X_{\text{obs}} \subset X$ be the states in collision with obstacles, and $X_{\text{free}} = \text{cl}\,(X \setminus X_{\text{obs}})$ be the resulting set of permissible states, where $\text{cl}\,(\cdot)$ represents the closure of a set. Let $\mathbf{x}_{\text{start}} \in X_{\text{free}}$ be the initial state and $X_{\text{goal}} \subset X_{\text{free}}$ be the set of desired goal states. Let $\sigma : [0, 1] \to X_{\text{free}}$ be a map to a sequence of states through collision-free space that can be executed by the robot (i.e., a collision-free, feasible path) and Σ be the set of all such nontrivial paths.*

The feasible planning problem is then formally defined as the search for any path from this set, $\sigma' \in \Sigma$, that connects $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{goal}} \in X_{\text{goal}}$,

$$\sigma' \in \{\sigma \in \Sigma \mid \sigma(0) = \mathbf{x}_{\text{start}},\, \sigma(1) \in X_{\text{goal}}\}.$$

Definition 2.2 (The optimal planning problem)**.** *Let $X \subseteq \mathbb{R}^n$ be the state space of the planning problem, $X_{\text{obs}} \subset X$ be the states in collision with obstacles, and $X_{\text{free}} = \text{cl}\,(X \setminus X_{\text{obs}})$ be the resulting set of permissible states, where $\text{cl}\,(\cdot)$ represents the closure of a set. Let $\mathbf{x}_{\text{start}} \in X_{\text{free}}$ be the initial state and $X_{\text{goal}} \subset X_{\text{free}}$ be the set of desired goal states. Let σ :*

$[0, 1] \to X_{\text{free}}$ *be a map to a sequence of states through collision-free space that can be executed by the robot (i.e., a collision-free, feasible path) and* Σ *be the set of all such nontrivial paths.*

The optimal planning problem is then formally defined as the search for a path, $\sigma^* \in \Sigma$*, that minimizes a given cost function,* $c : \Sigma \to \mathbb{R}_{\geq 0}$*, while connecting* $\mathbf{x}_{\text{start}}$ *to* $\mathbf{x}_{\text{goal}} \in X_{\text{goal}}$*,*

$$\sigma^* = \underset{\sigma \in \Sigma}{\arg\min} \left\{ c(\sigma) \mid \sigma(0) = \mathbf{x}_{\text{start}}, \sigma(1) \in X_{\text{goal}} \right\},$$

where $\mathbb{R}_{\geq 0}$ *is the set of non-negative real numbers.*

Some robotic systems require solving multiple planning problems in a single search environment (i.e., different $\mathbf{x}_{\text{start}}$ and/or X_{goal} with constant X and X_{obs}). Algorithms can reduce the total search effort in these *multiquery* planning scenarios by using the same information to solve multiple problems. This reusable information is often calculated with a general search of the problem domain that is performed before solving individual planning problems.

There is no benefit to *preprocessing* problem domains if each planning problem is posed in a different search environment. Algorithms can reduce the total search effort in these *single-query* planning scenarios by attempting to solve each problem as efficiently as possible. This thesis focuses on planning algorithms for single-query scenarios.

2.2 Graph-based Searches

A common way to simplify continuous planning problems is to approximate the search space as a graph of discrete states (i.e., vertices) connected by edges. Many cost functions in robotics (e.g., path length) satisfy Bellman's principle of optimality (Bellman, 1954, 1957) and the resulting discrete problem can be solved with dynamic programming (Bellman, 1954; Larson, 1967) or graph-search techniques (Fig. 2.2).

Formal guarantees of these algorithms in relation to the original *continuous* feasible or optimal planning problem will depend on the chosen *a priori* discretization. If an algorithm is guaranteed to find a solution to a given problem at the chosen resolution, if one exists, it is described as *resolution complete*. If it is guaranteed to find the optimal solution at the chosen resolution, if one exists, it is described as *resolution optimal*. By definition, resolution optimality implies resolution completeness.

Dynamic programming techniques solve optimal planning problems by iteratively calculating vertex costs. The Floyd-Warshall algorithm (Floyd, 1962; Ingerman, 1962; Roy, 1959; Warshall, 1962) calculates costs between every pair of vertices in a graph. This information can be used to find resolution-optimal solutions efficiently in multiquery planning scenarios that consist of problems with many different starts and goals. The Bellman-Ford-Moore algorithm (Bellman, 1958; Ford Jr., 1956; Moore, 1957) calculates the costs from a single state to every other vertex in a graph. This information can be used to find resolution-optimal

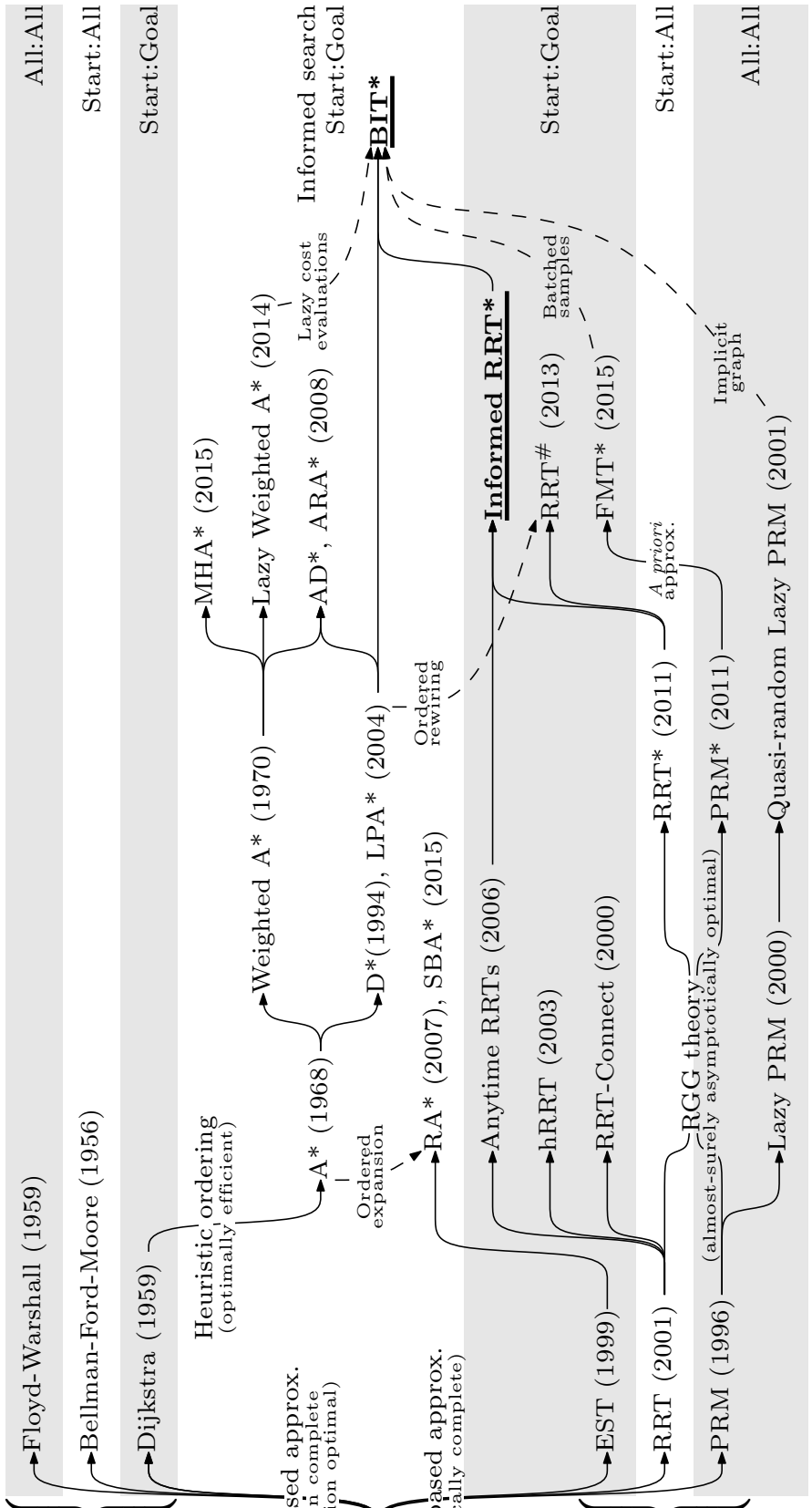


Figure 2.2: A simplified history of planning algorithms for continuous problems. The diagram is arranged to highlight two major methods of representing the search space, graph-based and sampling-based approximations; however, a similar diagram could be made dividing the algorithms into anytime and nonanytime approximations. Full descriptions of the algorithms are presented with citations in Chapter 2. [Referenced: pp. 6, 8, 10, 18, 29, 52, 77, 110]

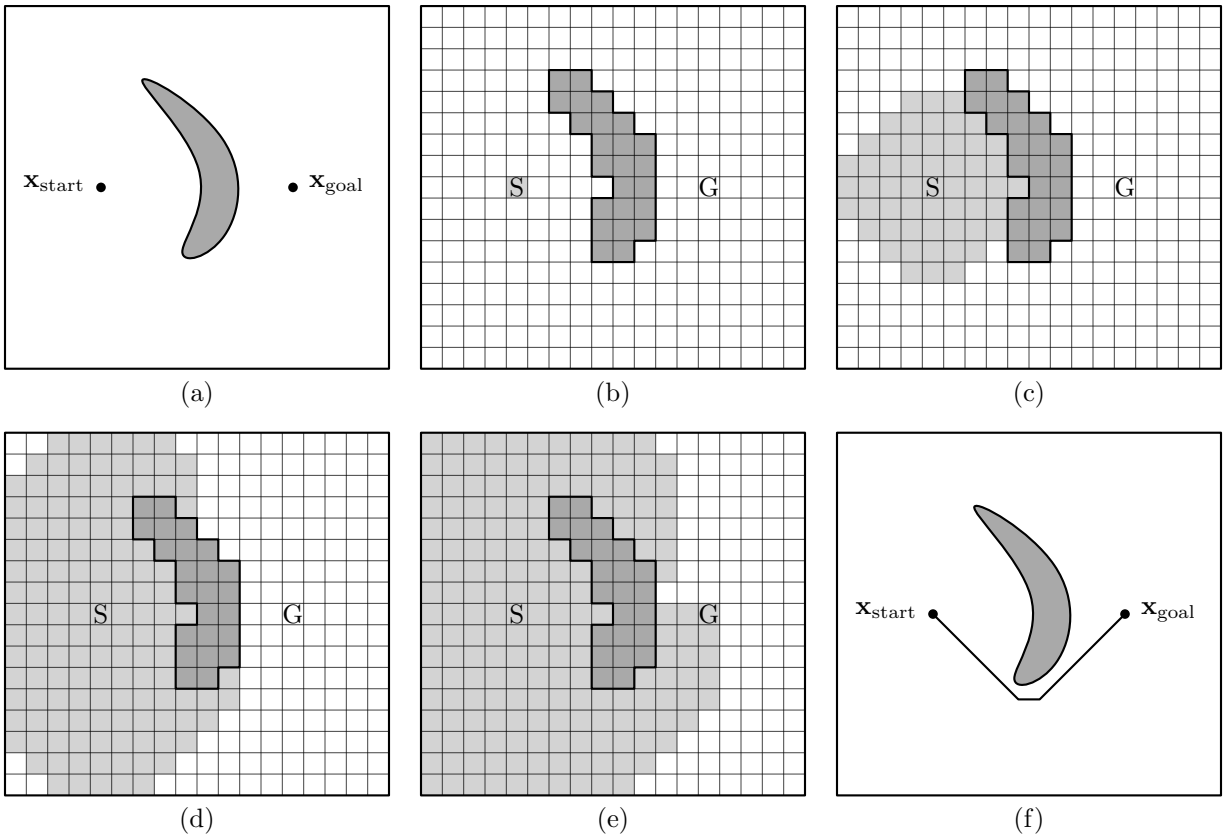


Figure 2.3: An illustration of Dijkstra's algorithm (Dijkstra, 1959). The continuous problem, (a), is approximated by a discrete set of states, (b). This discretization is searched outwards from the start in order of increasing cost-to-come, (c) and (d). The search terminates when no unexpanded vertices have a cost-to-come less than the goal, (e). The resulting solution is resolution optimal, (f).

solutions efficiently in multiquery planning scenarios that consist of multiple problems with a common start but different goals.

Both of these techniques search the *entire* graph and find resolution-optimal paths between many different vertices. This may be cost effective in multiquery planning scenarios but is inefficient in single-query scenarios that only require one path. Searching the entire graph is often unnecessary in these situations as the resolution-optimal solution is found as soon as (i) a solution is found, and (ii) all possibly better solutions are shown to be infeasible.

These conditions can be met simultaneously by considering potential solutions in order of cost. Such an *ordered search* assures that solutions are found only after all possibly better paths have been considered. This avoids unnecessarily searching the entire graph and can allow algorithms to find the resolution-optimal solution efficiently in single-query planning scenarios.

Dijkstra's algorithm (Dijkstra, 1959) searches a graph in order of increasing vertex cost (Fig. 2.3). This finds resolution-optimal paths to vertices in order of the *cost to come* to them from the start. By stopping the search once a solution is found, this finds the resolution-optimal solution by only considering vertices with lower costs-to-come than the goal.

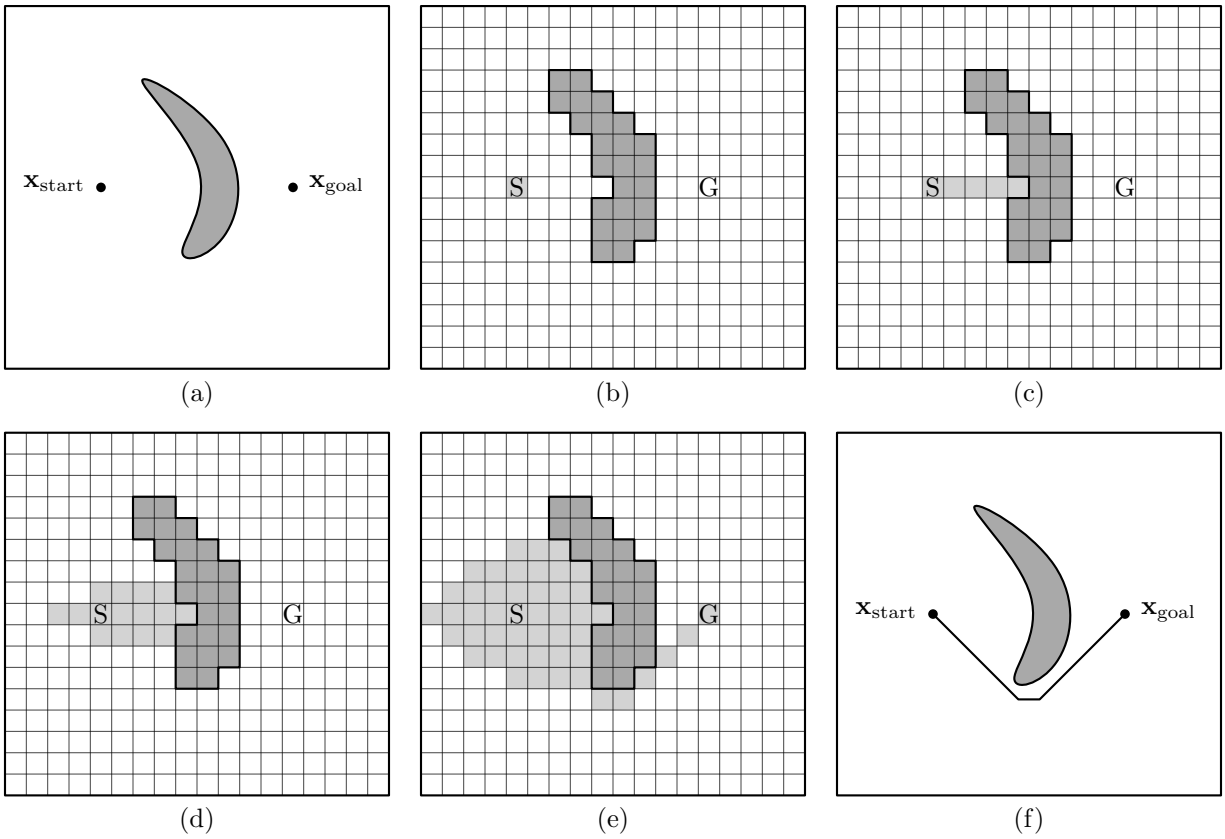


Figure 2.4: An illustration of the A* algorithm (Hart et al., 1968). The continuous problem, (a), is approximated by a discrete set of states, (b). This discretization is searched outwards from the start in order of increasing solution cost as estimated by an *admissible* heuristic, (c) and (d). The search terminates when no unexpanded vertices have an estimated solution cost less than the cost to the goal, (e). The resulting solution is resolution optimal, (f), and for the chosen heuristic the search is optimally efficient in the number of vertices expanded.

Algorithmically, Dijkstra's algorithm uses a queue of vertices ordered by cost-to-come. At each iteration the lowest-cost vertex in the queue is removed, the costs-to-come of its descendants are calculated, and these new vertices are added to the queue. This process continues until the goal reaches the front of the queue at which point all vertices with lower costs-to-come have been considered. This avoids vertices with higher costs than the goal but expands equally in every direction and may still consider vertices that are known to be unable to provide a better solution.

The A* algorithm (Hart et al., 1968) avoids low-cost vertices that cannot provide better solutions by searching a graph in order of potential solution quality (Fig. 2.4). Vertices are expanded in the order given by their combined cost-to-come and a heuristic estimate of the *cost to go* from them to the goal. These combined costs will provide an *admissible* estimate of the cost of a solution passing through the vertex for any heuristic that does not overestimate the true cost-to-go. This admissible heuristic allows A* to exploit problem information to find the resolution-optimal solution by only considering vertices believed to be capable of providing a better solution.

A* is algorithmically the same as Dijkstra's algorithm with the queue of vertices ordered by their heuristically estimated solution cost. The resulting search is *optimally efficient* in the number of vertices expanded since any other resolution-optimal algorithm using the same heuristic will expand at least as many vertices as A* (Hart et al., 1968). The practical meaning of this efficiency depends on how accurately the heuristic approximates the true solution cost and with the trivial *zero heuristic* A* is equivalent to Dijkstra's algorithm.

A* was first developed for Shakey (Nilsson, 1969), an early *mobile automaton*, and it has remained popular in robotics. An incomplete list of interesting uses includes nonholonomic multibody systems (Barraquand and Latombe, 1993), systems that can reconfigure their environments (Lynch and Mason, 1996), and systems that consider interactions between physical models of the robot and terrain (Cherif, 1999).

A* has also been extended to specific applications, including relaxing resolution optimality (Section 2.2.1), handling changing graphs efficiently (Section 2.2.2), and finding solutions in an *anytime* manner (Section 2.2.3). These techniques improve performance in many situations but their solutions are still ultimately limited by the *a priori* discretization (Section 2.2.4).

2.2.1 Near-resolution-optimal Search

A* expands a prohibitively large number of vertices to find a solution to many practical planning problems. As this solution is only an approximation of the continuous optimum (i.e., it is only *resolution* optimal), it may be acceptable to reduce search effort by relaxing optimality. If an algorithm is guaranteed to find a solution *near* the optimum at the chosen resolution, if one exists, it is described as *near resolution optimal*.

Weighted A* (Pohl, 1970a,b, 1973) finds near-resolution-optimal solutions by inflating an admissible heuristic by $\varepsilon > 1$ (Fig. 2.5). This prioritizes expanding vertices closer to the goal and finds solutions to many problems with fewer vertex expansions than A*. The resulting solution is guaranteed to be no more than ε worse than the resolution-optimal solution. Lazy Weighted A* (Cohen et al., 2014b) further accelerates the search by delaying cost calculations until absolutely necessary. This can be beneficial in problems where states have many descendants and calculating their cost-to-come is computationally expensive, such as multiarmed manipulation planning problems.

Multi-Heuristic A* (MHA*; Aine et al., 2014, 2015) finds near-resolution-optimal solutions by combining multiple arbitrarily inadmissible heuristics with a single admissible heuristic. This allows a wide variety of information to be incorporated into searches, including dynamically generated heuristics (Islam et al., 2015), while maintaining a bound on resolution quality. The resulting solution is guaranteed to be no more than a user-controlled parameter worse than the resolution-optimal solution.

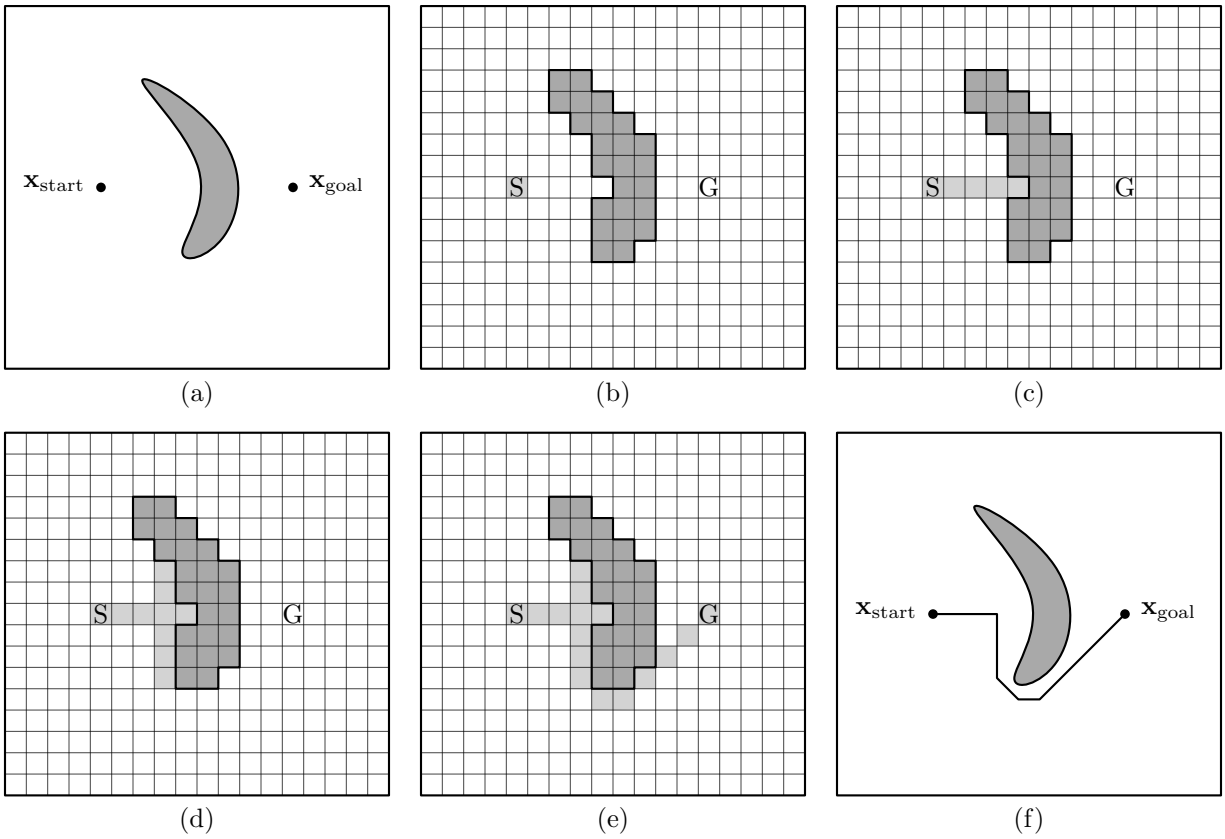


Figure 2.5: An illustration of a simple Weighted A* algorithm (Pohl, 1970a,b, 1973). The continuous problem, (a), is approximated by a discrete set of states, (b). This discretization is searched outwards from the start in order of increasing solution cost as estimated by an *inadmissible* heuristic, (c) and (d). The search terminates when no unexpanded vertices have a heuristic value less than the cost to the goal, (e). The use of an inflated heuristic results in a resolution-suboptimal solution, (f), but can reduce the total number of vertices expanded.

These and other similar techniques may find a solution to specific planning problems faster than A* but they do so by sacrificing solution quality and in general are not guaranteed to find the resolution-optimal solution.

2.2.2 Replanning or Incremental Search

Many robotic applications plan and operate in environments that are (partially) unknown or change during path planning and/or path execution. In these scenarios, A* must be restarted each time the graph changes to guarantee resolution optimality. This replanning is expensive and may be unnecessary depending on what regions of the graph have changed. *Replanning*, or *incremental*, search techniques handle graph changes efficiently by reusing previous search information and only updating the search where necessary.

Dynamic A* (D*; Stentz, 1994) and derived algorithms (e.g., Focussed D*; Stentz, 1995; D* Lite; Koenig and Likhachev, 2005; and Delayed D*; Ferguson and Stentz, 2005) are designed for robotic planning in unknown environments. They search a planning problem

by assuming that it is obstacle free wherever information is unavailable (i.e., a *free-space* assumption). The algorithms then incorporate new information as it becomes available (e.g., by executing the proposed plan) into an updated plan. This allows a robot to navigate an unknown environment efficiently by planning and executing a series of paths. The final route driven by the robot may be suboptimal but each path will be the resolution-optimal solution from the current robot position to the goal given the available information.

Lifelong Planning A* (LPA*; Koenig et al., 2004) adapts A* to handle graphs with changing edge weights. The algorithm searches for *inconsistent* vertices any time the graph is changed and reinserts them into the search queue. A vertex is inconsistent if its cost-to-come is different than the best value that could be calculated from its immediate neighbours (the *lookahead* value). The search queue is processed in order of potential solution quality as in A*. This efficiently maintains formal guarantees on resolution completeness and optimality in the presence of a changing graph (including the addition/removal of vertices) by reusing unchanged information.

D* and LPA* propagate changes through their entire search in order to maintain resolution optimality. This can be computationally expensive and it may be desirable to reduce computational cost by relaxing optimality, as in Section 2.2.1. Truncated D* Lite and Truncated LPA* (TD* Lite and TLPA*; Aine and Likhachev, 2016) reduce computational cost by prematurely stopping the propagation of changes. The resulting search is guaranteed to find a near-resolution-optimal solution no more than a user-controlled parameter worse than the resolution-optimal solution.

All these techniques require fewer vertex expansions than A* to find a solution to a *modified* planning problem but they still only find the resolution-optimal solution. On *unmodified* planning problems these algorithm will require the same number of vertex expansions as A* to find the resolution optimum.

2.2.3 Anytime Search

Many robotic systems have a finite amount of computational time available to solve planning problems. A* will only find *a* solution in these situations if the given time is sufficient to find the *resolution-optimal* solution. This behaviour is often undesirable and it would be preferred to have algorithms find the *best-possible* solution in the available time. This can be accomplished by finding an initial suboptimal solution and then spending the remaining available time improving it. Such algorithms can be stopped at any time to retrieve intermediate solutions and are often described as *anytime* searches.

Anytime Repairing A* (ARA*; Likhachev et al., 2008, 2003) and Anytime Dynamic A* (AD*; Likhachev et al., 2005, 2008) achieve anytime search by combining near-resolution-optimal (Section 2.2.1) and incremental (Section 2.2.2) search techniques. They attempt to

find an initial suboptimal solution quickly and then incrementally improve it while maintaining bounds on its resolution suboptimality. They will eventually find the resolution-optimal solution but can be stopped at any time to retrieve the current near-resolution-optimal solution. AD* also handles dynamic graphs and has been extended with Anytime Truncated D* (ATD*; Aine and Likhachev, 2016) to include the performance benefits of prematurely stopping change propagation.

Finding an initial near-resolution-optimal solution allows these techniques to often find solutions to planning problems with fewer vertex expansions than A*. With further computational effort, these solutions can be improved to the resolution optimum but doing so will require more vertex expansions than A*.

2.2.4 The Limitations of *A Priori* Discretizations

Graph-based search performance ultimately depends on the accuracy with which the graph approximates the underlying continuous problem and as a result there has been a significant amount of research in robotics on different discretizations. Techniques include regular graphs (Lozano-Pérez and Wesley, 1979), hierarchical, nonuniform, and mulitresolution graphs (e.g., SANDROS; Chen and Hwang, 1992), graphs with random perturbations (Sallaberger and D'Eleuterio, 1995), implicit graphs (Bohlin, 2001), graphs with kinodynamic velocity and acceleration constraints (Donald et al., 1993), and graphs of precomputed feasible motions whose representation may be implicit (*state lattices*; Pivtoraiko and Kelly, 2005; Pivtoraiko et al., 2009; and *manipulation lattice graphs*; Cohen et al., 2014a).

These discretizations all require selecting *a priori* resolutions. Low resolutions create simple graphs that can be searched quickly but often result in low-quality *continuous* solutions (if they contain one). High resolutions approximate the problem accurately and yield high-quality continuous solutions (Bertsekas, 1975) but create complex graphs that are often expensive to search. These effects are only exacerbated by the exponential growth of graph size with state dimension, which Bellman (1954, 1957) refers to as the *curse of dimensionality*.

Selecting an appropriate resolution is a fundamental challenge of using *a priori* discretizations to represent continuous problems. In practice the *ideal* approximation of a problem will be simple enough to search quickly and complex enough to contain a good solution. This depends on the specific planning problem and is often expensive and time-consuming to find since changing the resolution typically requires restarting the search. Practical systems often use a conservatively complex *a priori* discretization to guarantee a solution exists and then use advanced graph-search techniques to find a suboptimal solution quickly.

This thesis seeks methods that avoid these limitations by improving their approximation continuously with computational time (i.e., they build an *anytime representation* of the problem). Such searches often use sampling-based techniques (Section 2.3).

2.3 Sampling-based Planners

Another way to simplify continuous planning problems is by randomly (Barraquand et al., 1996) or deterministically (Branicky et al., 2001; Janson et al., 2015a) sampling the search space (Fig. 2.2). This can allow algorithms to build and search representations that increase in accuracy with additional computational time (i.e., *anytime representations*). These anytime representations avoid dense *a priori* discretizations and can reduce the curse of dimensionality and improve performance on systems with differential constraints (Hsu et al., 2002; LaValle and Kuffner Jr., 2001).

Formal guarantees of these algorithms in relation to the original *continuous* feasible or optimal planning problem are often probabilistic. If with an infinite number of samples an algorithm has a unity probability of finding a solution, if one exists, it is described as *probabilistically complete* (Definition 2.3). If with an infinite number of samples it has a unity probability of converging asymptotically to the optimum, if one exists, it is described as *almost-surely asymptotically optimal* (Definition 2.4). By definition, almost-sure asymptotic optimality implies probabilistic completeness.

Definition 2.3 (Probabilistic completeness)**.** *A planner is said to be* probabilistically complete *if with an infinite number of samples the probability of finding a feasible solution (Definition 2.1), if one exists, is one,*

$$\liminf_{q\to\infty} P\left(\sigma_q \in \Sigma,\, \sigma_q\left(0\right) = \mathbf{x}_{\text{start}},\, \sigma_q\left(1\right) \in X_{\text{goal}}\right) = 1,$$

where q is the number of samples, σ_q is the path found by the planner from those samples, and Σ is the set of all feasible, collision-free paths.

Definition 2.4 (Almost-sure asymptotic optimality)**.** *A planner is said to be* almost-surely asymptotic optimal *if with an infinite number of samples the probability of converging asymptotically to the optimum (Definition 2.2), if one exists, is one,*

$$P\left(\limsup_{q\to\infty} c\left(\sigma_q\right) = c\left(\sigma^*\right)\right) = 1,$$

where q is the number of samples, σ_q is the path found by the planner from those samples, σ^* is the optimal solution to the planning problem, and $c\left(\cdot\right)$ is the cost of a path.

Early sampling-based planners include Probabilistic Roadmaps (PRM; Kavraki et al., 1996), Expansive Space Trees (EST; Hsu et al., 1997, 1999a) and Rapidly exploring Random Trees (RRT; LaValle, 1998; LaValle and Kuffner Jr., 2001).

PRM uses a two-stage process to solve continuous planning problems. First, a *learning* stage uses a *local planner* to connect samples into a *roadmap* (i.e., a graph) that approximates

the collision-free space. A *query* stage attempts to solve the specific planning problem by using the local planner to add the start and goal to the roadmap. A feasible solution to the continuous planning problem can then be found by extracting a path through the roadmap between the start and goal. PRM is probabilistically complete and with an appropriate connection scheme (e.g., simplified PRM; Kavraki et al., 1998; or PRM*; Karaman and Frazzoli, 2011) almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

Roadmaps can be reused indefinitely and are generally built to span the entire search space. This makes PRM an efficient algorithm in multiquery planning scenarios but inefficient for single-query scenarios as individual solutions rarely uses the entire roadmap. Techniques exist to reduce the computational cost of the unused roadmap (e.g., Lazy PRM; Bohlin and Kavraki, 2000; and Quasi-random Lazy PRM; Branicky et al., 2001) but in single-query scenarios it is often more efficient to search incrementally until a solution is found.

EST solves continuous planning problems by incrementally growing trees from the start and goal of a planning problem. The trees are grown by randomly selecting a vertex and expanding it to nearby new samples. A feasible solution to the continuous planning problem is found once the two trees are connected. This technique can be extended to handle kinodynamic constraints by expanding vertices with a motion model and random control inputs (Hsu et al., 2002). EST is probabilistically complete but no claims are made about its asymptotically optimality.

The probability of both expanding a vertex and adding a new sample in EST is inversely proportional to the number of nearby states. This avoids repeatedly searching the same regions of the problem domain but requires calculating the number of states nearby to *each vertex* in the tree at *each iteration*. This can be computationally expensive and other single-query planning algorithms seek simpler methods to prioritize new expansion.

RRT solves continuous planning problems by incrementally growing a tree through collision-free space (Fig. 2.6). Instead of calculating expansion probabilities for each vertex in the tree, it draws a random sample from the problem domain and extends the tree towards it. This guides the search to unexplored regions of the planning domain through a *Voronoi bias* (see Voronoi tessellations; Dirichlet, 1850; Voronoi, 1908). A feasible solution to the continuous planning problem is found once the goal is added to the tree. RRT is probabilistically complete but not almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

Algorithmically, RRT draws a random sample from the search domain at each iteration. This sample is used to extend the nearest vertex in the tree no more than a user-tuned distance towards it to create a potential new state. This avoids solving two-point boundary-value problems (two-point BVPs) between the vertex and the random sample which can be expensive for systems with differential constraints. If the new state and connection are collision free, they are added to the tree as a new vertex and edge. In practice the sampling distribution often has a small likelihood of directly sampling the goal to bias the search towards a solution.

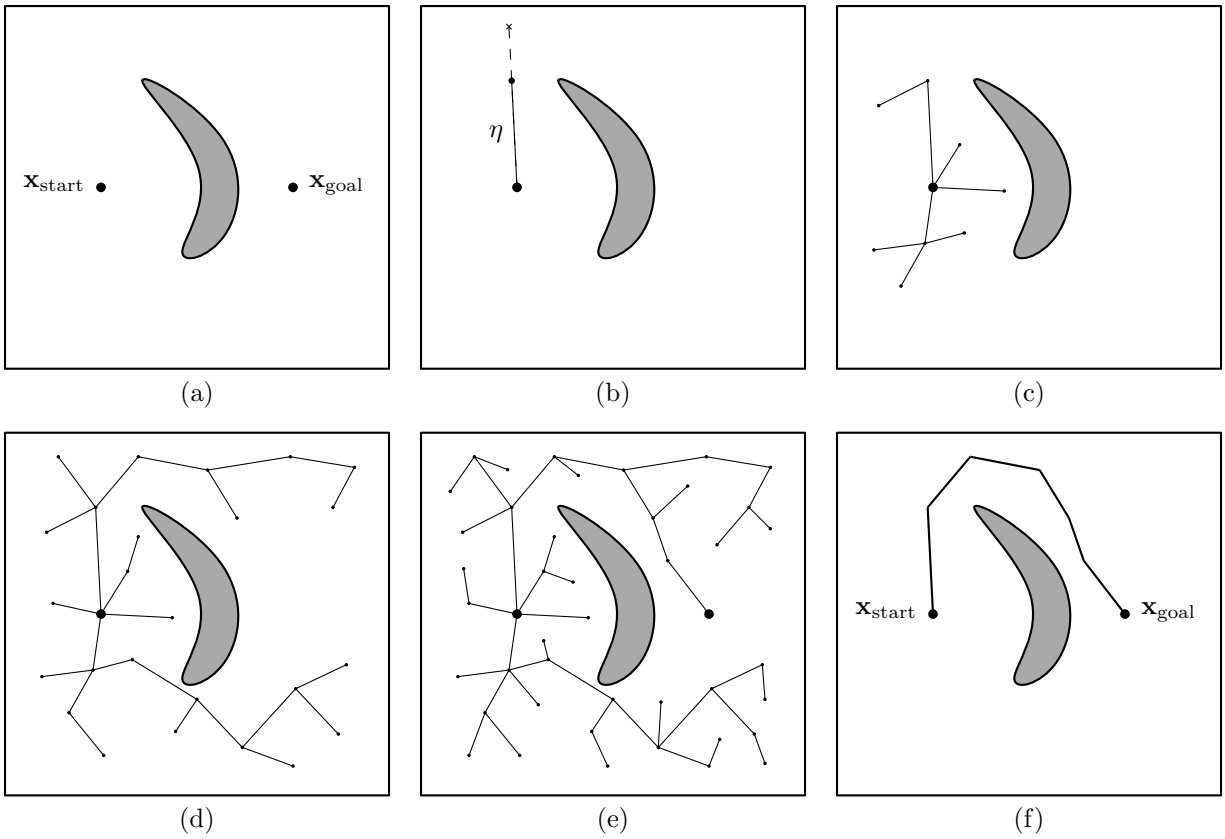


Figure 2.6: An illustration of the RRT algorithm (LaValle, 1998; LaValle and Kuffner Jr., 2001). The continuous problem, (a), is approximated with a sample-based tree rooted at the start. The tree is grown towards samples, (b), in increments no larger than a user-selected distance, η. Adding collision-free vertices and edges incrementally searches the collision-free space in an anytime fashion, (c) and (d). The search is often guided towards a solution by including a low probability of directly sampling the goal, (e). With an infinite number of samples RRT has a unity probability of finding a solution, if one exists; however, it will *not* almost-surely converge to the optimum with additional computational time, (f).

RRT-Connect (Kuffner Jr. and LaValle, 2000) is a bidirectional version of RRT that expands trees from both the start and goal (Fig. 2.7). The trees are alternately grown towards the newest state in the other tree and expanded into unexplored regions of the problem domain through sampling. Using multiple trees to explore the state space from different directions allows RRT-Connect to find solutions to many problems faster than RRT, especially in the presence of directional features (e.g., 'bug traps'). RRT-Connect is also probabilistically complete and not almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

RRT and RRT-Connect are designed to expand quickly into unexplored regions of the problem domain (i.e., they are *space filling*). They continue to expand with additional sampling but their existing edges and the resulting costs-to-come of vertices are not changed. Vertices may be optimally connected given the *current* tree (e.g., k-nearest RRT; Urmson and Simmons, 2003) but they are almost-surely suboptimally connected with respect to the continuous planning

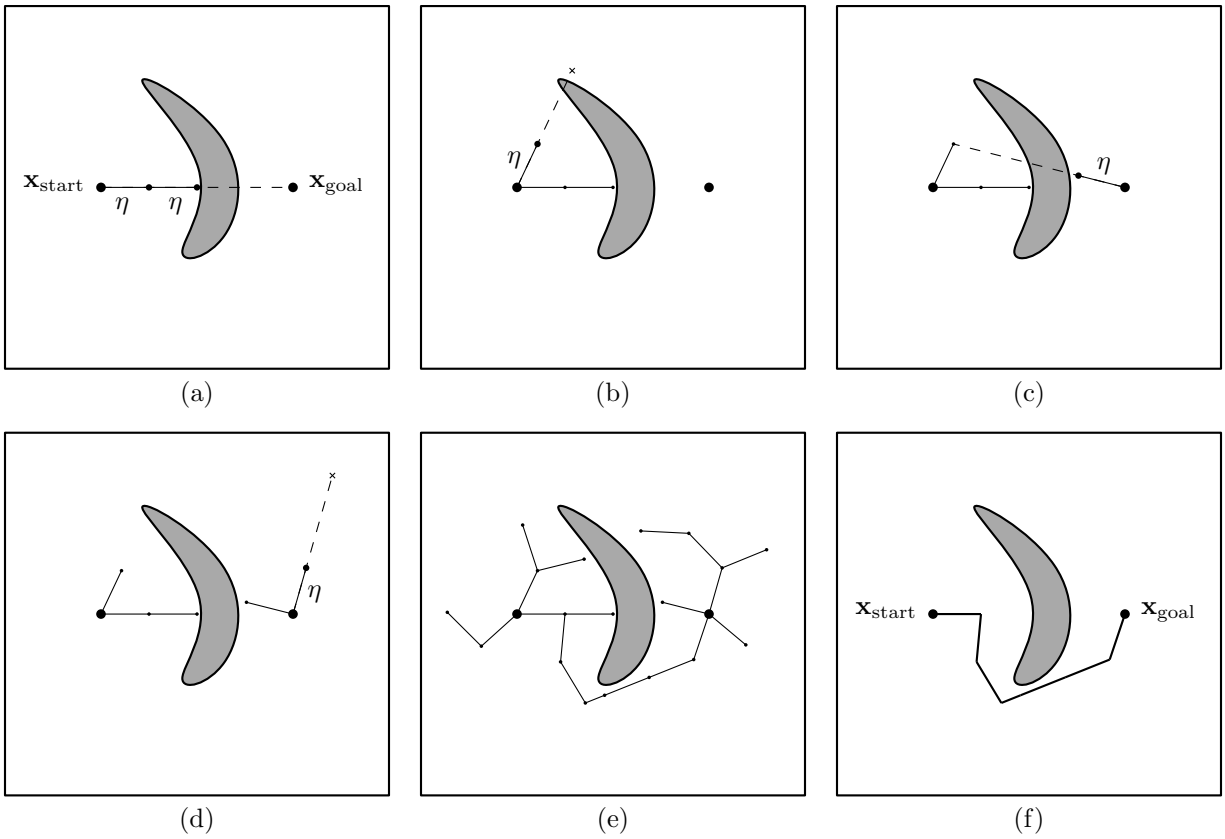


Figure 2.7: An illustration of the RRT-Connect algorithm (Kuffner Jr. and LaValle, 2000). The continuous problem is approximated with two sample-based trees rooted at the start and goal. The trees are alternately grown towards each other, (a) and (c), and expanded with the RRT search behaviour, (b) and (d). This quickly navigates directional obstacles, such as 'bug traps', while maintaining a search bias for a solution (e). With an infinite number of samples, RRT-Connect has a unity probability of finding a solution, if one exists; however, it will *not* almost-surely converge to the optimum with additional computational time, (f).

problem (Karaman and Frazzoli, 2011). RRT solution quality can be improved by modifying existing connections during the search to improve the costs-to-come of vertices incrementally.

Solution quality can also be improved by postprocessing solutions. Techniques such as path pruning or path shortcutting (Berchtold and Glavina, 1994; Geraerts and Overmars, 2007; Hauser and Ng-Thow-Hing, 2010; Hsu et al., 1999b; Sekhavat et al., 1998) simplify and improve paths found by sampling-based planners. These are local techniques that provide no global guarantees and can be applied to the output of any sampling-based planner and will not be discussed further in this thesis.

Rapidly exploring Random Graph (RRG) and RRT* (Karaman and Frazzoli, 2010, 2011) modify RRT to consider both the incoming and outgoing edges of new vertices in order to improve the costs-to-come of existing vertices. RRG uses all these edges to build a graph while RRT* finds those that locally minimize cost-to-come and maintains a tree (Fig. 2.8). Both algorithms are probabilistically complete and almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

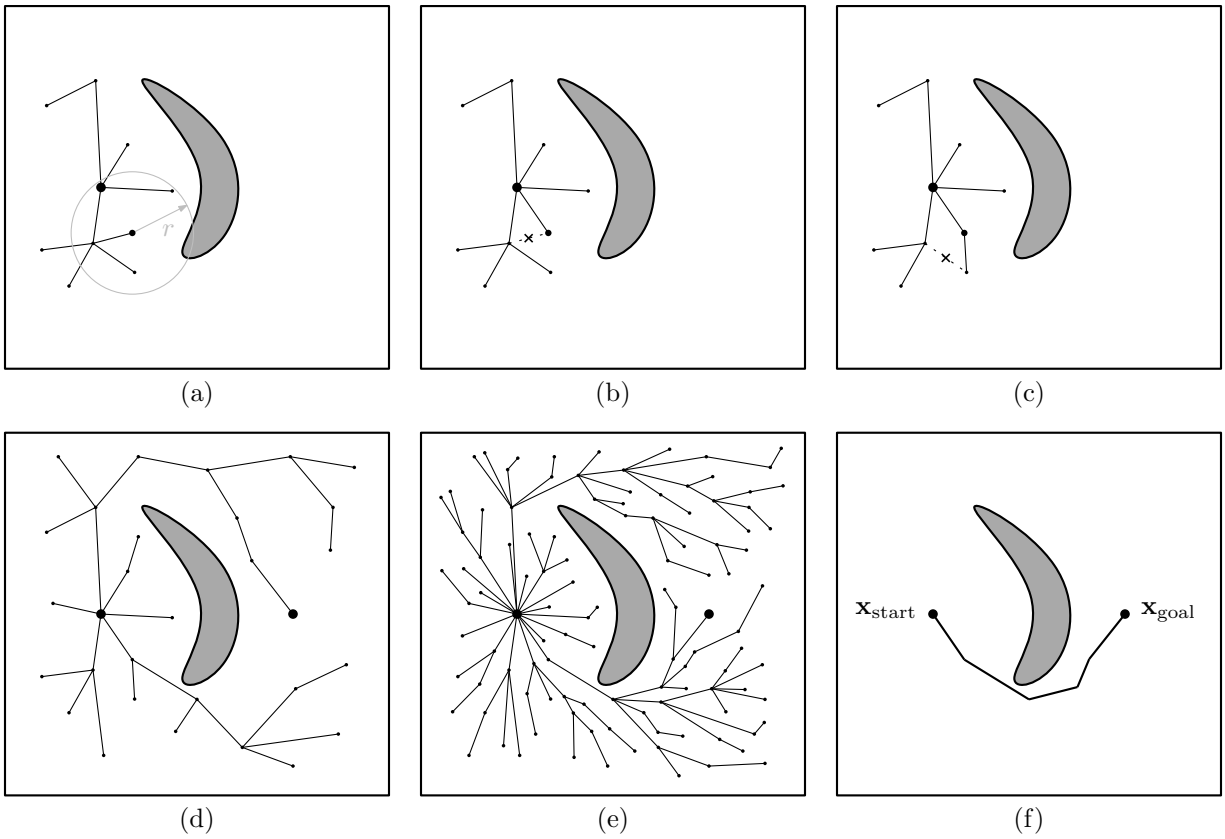


Figure 2.8: An illustration of the RRT* algorithm (Karaman and Frazzoli, 2010, 2011). The continuous problem is approximated with a sample-based tree rooted at the start as in RRT. States are optimized over a local neighbour defined by RGG theory, (a). Each new state is both connected to the local state that minimizes its cost-to-come, (b) and used to rewire existing local states, (c). The resulting anytime search fills the space while almost-surely asymptotically converging towards optimal paths, (d) and (e). With an infinite number of samples, RRT* has a unity probability of both finding a solution and converging asymptotically to the optimum, if one exists, (f).

RRT* maintains a tree through *local rewiring*. Instead of connecting new vertices to the *nearest* vertex in the tree, it connects them to the *nearby* vertex that minimizes their cost-to-come. It then considers whether this new vertex can lower the costs-to-come of existing vertices. Any nearby vertices that can be improved are rewired to be descendants of the new vertex. This allows existing vertices to benefit from future samples and simultaneously improves the tree while searching the problem domain.

RRT* performance depends on the size of the neighbourhood used for rewiring. It must be small enough to be searched efficiently at each iteration but large enough to actually improve the tree. Karaman and Frazzoli (2011) provide two separate definitions for this rewiring neighbourhood, either as the k-nearest vertices or all the vertices within a distance, r. They use RGG theory to provide theoretical lower bounds for these values such that the resulting solution almost-surely converges asymptotically to the optimum. These bounds are a function of the number of samples, the dimension of the state space, and its measure (e.g., volume).

Incremental sampling-based planners build and search anytime approximations of continuous planning problems and can be stopped once a suitable solution is found. This avoids computational effort that is unnecessary to solve the given planning problem and has made them popular choices in single-query planning scenarios. While algorithms such as EST, RRT, and RRT* grow their approximations from the start and/or goal, they asymptotically find paths to *every* state in the problem domain and their searches are generally agnostic to the specific problem. This behaviour is inefficient in single-query scenarios and more efficient algorithms can be designed by exploiting information about the specific planning problem, as in graph-based searches. Common techniques include different sampling methods (Section 2.3.1), ordered approximations (Section 2.3.2) and ordered searches (Section 2.3.3). These approaches often improve performance but they do so without addressing all the challenges of simultaneously approximating and searching a continuous planning problem (Section 2.3.4).

The real-time performance of almost-surely asymptotically optimal algorithms can also improved by relaxing the formal guarantee to asymptotic *near* optimality (Dobson and Bekris, 2014; Marble and Bekris, 2013, 2017; Salzman and Halperin, 2014; Wang et al., 2015). These ideas are analogous to near-resolution-optimal graph-based search (Section 2.2.1) and will not be discussed in detail in this thesis.

2.3.1 Sample Distributions

Sampling-based planners search continuous planning problems by building and connecting sets of samples. Their performance in finite time depends on the number and distribution of samples relative to the features of the planning problem. Uniform distributions are effective on a wide range of problems and easy to analyze (e.g., Karaman and Frazzoli, 2011; Kavraki et al., 1998) but agnostic to the search domain. One way to improve real-time performance of sampling-based planners is to use nonuniform sample distributions that prioritize regions of the problem domain believed to be useful in finding (high-quality) solutions.

Many different sample distributions have been used with PRM (Geraerts and Overmars, 2004), RRT, and RRT*. An incomplete list includes Gaussian distributions (Boor et al., 1999), obstacle-based sampling (Amato and Wu, 1996; Sun et al., 2005; Yeh et al., 2012) including near the medial axis of free space (Holleman and Kavraki, 2000; Wilmarth et al., 1999; Yang and Brock, 2004), visibility (Nissoux et al., 1999; Siméon et al., 2000), manipulability (Leven and Hutchinson, 2003), and many other techniques to prioritize areas of predicted high utility (Akgun and Stilman, 2011; Baldwin and Newman, 2010; Burns and Brock, 2005a,b; Jaillet et al., 2008, 2010; Kiesel et al., 2012; Kim et al., 2014; Nasir et al., 2013; Rickert et al., 2008; Thomas et al., 2007; van den Berg and Overmars, 2004; Yershova et al., 2005; Zucker et al., 2008).

These nonuniform distributions increase the probability of sampling the regions believed to be of high utility by decreasing the probability of sampling other regions. These undersampled regions are *expected* to be less useful but they may still contain solutions to planning problems. If these are the *only* solutions to a given problem then their reduced sampling probability can decrease performance or even preclude finding a solution.

A nonzero sampling probability can be maintained over all possible solutions by combining nonuniform and uniform sample distributions. This guarantees probabilistic completeness but reduces nonuniformity by increasing the sampling of otherwise low-probability regions. The relative ratio of these distributions is often an extra user-specified parameter whose optimal value can vary widely between different problems.

Nonuniform sample distributions also affect the almost-sure asymptotically optimality of planners such as RRT*. These algorithms assume uniformly distributed samples to calculate their rewiring neighbourhoods and violating this assumption invalidates their associated proofs of asymptotic performance. Almost-sure asymptotic optimality can be maintained by defining these neighbourhoods in terms of the subset of uniformly distributed samples (Janson et al., 2015b); however, this results in larger neighbourhoods that rewire more vertices and increases computational cost.

2.3.2 Heuristically Ordered Approximations

Sampling-based planners are often used to approximate and search continuous planning problems incrementally. Being able to stop these anytime algorithms once a suitable solution is found avoids unnecessary computational cost, especially in single-query scenarios, but often results in searches that are ordered by the sequence of approximating samples (e.g., RRT*). This is inefficient and more effective algorithms can be created by using heuristic concepts (e.g., A*) to order the approximation and the resulting search.

Heuristically Guided RRT (hRRT; Urmson and Simmons, 2003) uses rejection sampling to create a "probabilistic implementation of heuristic search concepts." At each iteration a uniformly distributed sample is probabilistically kept or rejected as a function of its heuristic value relative to the existing tree. This iteratively biases RRT expansion towards regions of the problem domain believed to contain high-quality solutions and often finds better solutions than RRT, especially on problems with continuous cost functions (e.g., path length; Urmson and Simmons, 2003).

Randomized A* (RA*; Diankov and Kuffner Jr., 2007), Hybrid Randomized A* (HRA*; Teniente and Andrade-Cetto, 2013), and Sampling-based A* (SBA*; Persson and Sharf, 2014) independently develop A*-based methods to order the sampling-based approximation. Each algorithm is unique but they all incrementally build trees by heuristically selecting vertices to expand with local sampling. This prioritizes sampling regions of the problem domain believed to contain high-quality solutions and can result in better performance than RRT/RRT*.

Heuristics predict an infinite number of potential high-quality solutions in continuous planning problems. To solve these problems quickly while maintaining probabilistic completeness, algorithms with ordered approximations must balance *exploiting* heuristic information with *exploring* the entire problem domain. This can be accomplished in a variety of ways including a minimum sampling probability for each region (e.g., hRRT), a maximum local sample density (e.g., RA* and HRA*), and an inverse correlation between local sample density and expansion priority (e.g., HRA* and SBA*). All these techniques affect the algorithm performance and require additional user-specified parameters.

Using a minimum sampling probability or including local density in sample ordering allows exploratory states to be sampled at any iteration and results in a search that simultaneously expands multiple independent paths regardless of their relative cost. This is inefficient compared to informed graph-based searches, such as A*, which completely consider each potential solution and immediately extend paths to the goal whenever possible. Comparatively, algorithms such hRRT, HRA*, and SBA* may continue to search other regions of the problem domain even when a state can be immediately connected to the goal.

Using a maximum local sample density limits the number of possible descendants for each vertex. This reduces anytime performance compared to algorithms such as RRT, which consider samples indefinitely until a suitable solution is found. Comparatively, algorithms such as RA* sample regions of the problem domain in order of potential solution quality until the *a priori* maximum resolution has been reached. If this resolution is insufficient to find a (suitable) solution, then the search must be restarted.

2.3.3 Heuristically Ordered Search

Heuristics can also be used to order the search of sampling-based planners without modifying their representation of the underlying continuous problem. This separation between the approximation of a problem and the search of a representation is common in multiquery planners (e.g., the independent learning and query stages of PRM) but can also be used in single-query scenarios.

RRT$^{\#}$ (Arslan and Tsiotras, 2013, 2015) uses LPA* to update an RRG search. This allows it to incorporate new samples efficiently, propagate the required changes through the entire graph, and avoid computations that cannot improve a solution once one is found. This results in better solutions faster than RRT* but still searches for an initial solution in the random order given by the sequence of samples.

Fast Marching Tree (FMT*; Janson and Pavone, 2013; Janson et al., 2015b) uses RGG theory to approximate the continuous planning problem with a batch of samples (Fig. 2.9). It searches this representation with a fast-marching method (Sethian, 1996) that expands outward from the start in order of increasing cost-to-come, as in Dijkstra's algorithm (see

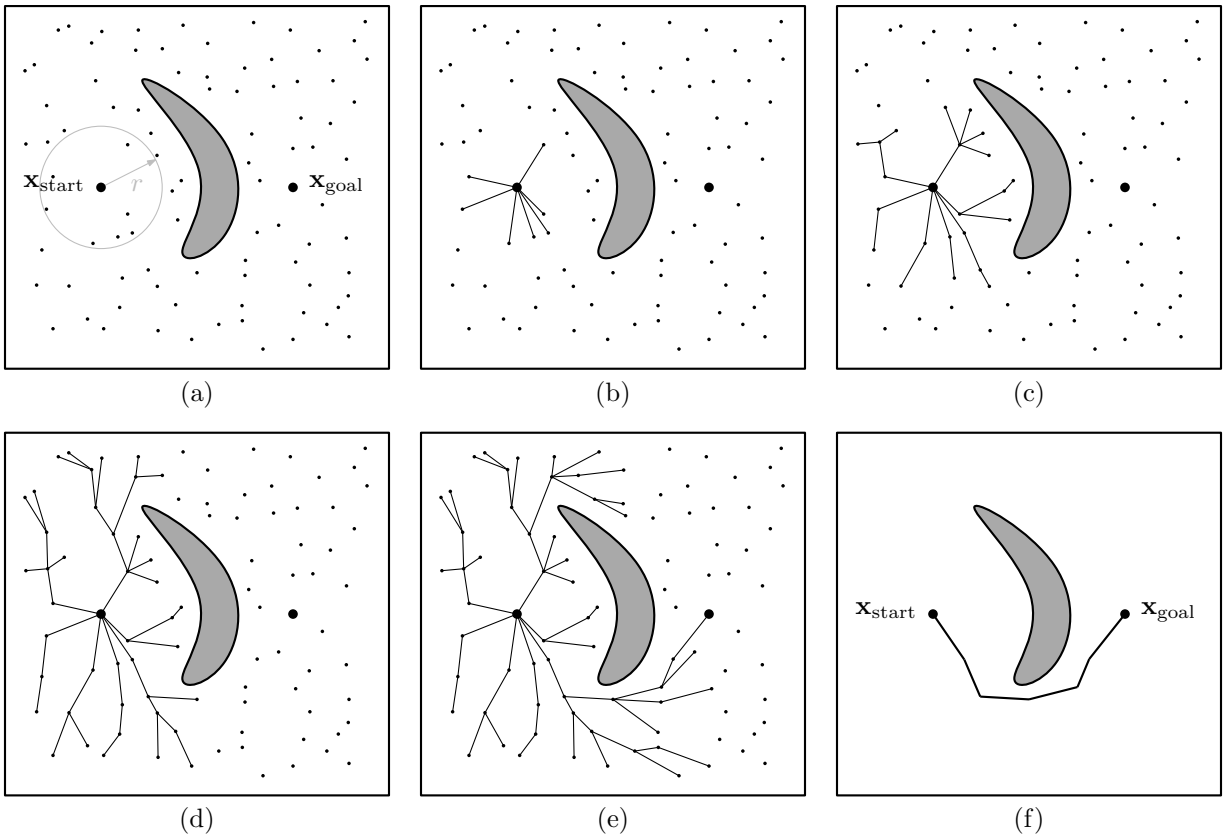


Figure 2.9: An illustration of the FMT* algorithm (Janson and Pavone, 2013; Janson et al., 2015b). The continuous planning problem is searched with a sample-based tree rooted at the start. The tree is grown outwards from the start using a marching method and a radius of connection defined by RGG theory, (a). In a process reminiscent of Dijkstra's algorithm (see Fig. 2.3), the search proceeds in order of increasing cost-to-come, (b)–(d), and terminates when no unexpanded samples have a cost-to-come less than the goal, (e). With an infinite number of samples, FMT* *converges in probability* to asymptotic optimality, (f).

Fig. 2.3). This separates the approximation of a planning problem from its search and *converges in probability* to asymptotic optimality (Janson et al., 2015b) but does so in a manner that is not anytime. The search does not return a solution until it is finished and must be restarted with more samples if a (suitable) solution is not found. A quasi-anytime version that does so with heuristics (Salzman and Halperin, 2015) and a bidirectional version (Starek et al., 2015) have recently been presented.

These two algorithms illustrate the challenges of heuristically ordering the search of a sampling-based approximation independent of its construction. Searching the representation after every change, as in RRT$^{\#}$, maintains good anytime performance but searches in the order given by the sequence of samples and limits the ability for heuristics to improve efficiency. Searching the representation infrequently/once, as in FMT*, allows for an efficient ordered search but reduces the number of intermediate solutions and decreases anytime performance. This again illustrates the need to balance *exploiting* problem information to perform an efficient search and *exploring* the entire problem domain quickly.

2.3.4 The Challenges of Simultaneous Approximation and Search

Anytime sampling-based planners simultaneously approximate and search a continuous planning problem. This allows them to continue indefinitely until a (suitable) solution is found and avoid the challenges of selecting *a priori* approximations (Section 2.2.4). To do so effectively, they must balance the accuracy of the approximation with the computational cost required to search it. This challenge is only amplified by the incremental nature of anytime sampling-based planners which must regularly consider new samples.

Repeatedly updating the search of a dense representation is prohibitively expensive in most real-world situations. Anytime sampling-based planners often avoid this cost by building approximations that do not require an updated search (e.g., RRT only includes *one* incoming edge to each vertex) or not updating the entire search (e.g., RRT* only considers connections to/from the *new* vertex). These approaches reduce the quality of solutions found from a given set of samples and increase the real-time rate at which new samples can be added. This allows algorithms to achieve high sample densities and find (suitable) solutions to many planning problems in finite computational time.

Requiring high sample density to offset a representation with few connections or an incomplete search is not scalable. Exponentially more samples will be required to achieve equivalent sample densities as state dimension increases (i.e., the curse of dimensionality). Other anytime sampling-based planners avoid this limitation by increasing the accuracy of the representation given by a set of samples or improving the search. These approaches require more computational effort (e.g., RRT$^{\#}$ *updates* the search after each iteration) or decrease anytime performance (e.g., FMT* searches an *a priori* set of samples) but they do so efficiently with heuristics and often find better solutions.

Other anytime sampling-based planners attempt to improve performance by ordering the approximation of the problem domain by potential solution quality. These approaches may unnecessarily consider low-quality solutions (e.g., hRRT uses a probabilistic ordering) or decrease anytime performance (e.g., RA* searches *a priori* resolution) but can also improve performance.

This thesis seeks effective ways to balance the approximation and search of a continuous planning problem while maintaining the benefits of both informed graph-based search and anytime sampling-based planning. It does so by unifying these two approaches and performing an informed search on incrementally built sampling-based approximations (Section 2.4).

2.4 Discussion

Path planning is a fundamental task for robotic systems operating autonomously in dynamic and uncontrolled environments. Successful algorithms must be able to find high-quality

solutions in a reasonable amount of time to problems that are often continuous valued, high dimensional, and/or poorly bounded. Most algorithms achieve this by approximating the continuous search domain with either an *a priori* graph-based or an anytime sampling-based discretization. There are advantages and disadvantages to both of these approaches.

Using an *a priori* discretization requires defining the approximation before performing the search. If the chosen resolution is too low then any solutions that exist in the resulting representation will be of insufficient quality. If the chosen resolution is too high then the resulting representation will be prohibitively expensive to search.

Graph-based techniques are effective despite these limitations because of the efficiency of their search. Informed graph-search techniques, such as A*, search in order of potential solution quality and are optimally efficient in the number of vertices expanded (Hart et al., 1968). This makes them popular choices in situations where an appropriate representation can be chosen *a priori*.

Using sampling-based techniques requires approximating the planning problem with sets of samples. These approximations are more accurate and contain better solutions when they consist of many different connections; however, this makes them computationally expensive to search. RGG theory can be used to define sparse approximations that are less expensive to search and have probabilistic asymptotic guarantees (Penrose, 2003).

Building these representations incrementally (i.e., building *anytime* representations) avoids the need to define approximations *a priori* but often requires incremental search. Completely searching the *entire* representation after each new vertex always finds the best solution in the current approximation but can be prohibitively expensive. Only searching *parts* of the representation reduces the computational cost of new vertices but makes the planner performance dependent on sampling order and often results in *unordered* searches.

Sampling-based techniques are effective despite these limitations because of the efficiency of their anytime representations. Incremental sampling-based planners, such as RRT*, almost-surely converge asymptotically to the optimal solution of a problem in an anytime manner (Karaman and Frazzoli, 2011). This makes them popular choices for situations where an unknown amount of computational time is available to find a solution and/or an appropriate representation cannot be chosen *a priori*.

Unordered sampling-based planners simultaneously search towards every state in the problem domain. This becomes prohibitively expensive as state dimension increases and eventually outweighs the advantages of anytime representations in incremental sampling-based planners. Previous work has used heuristics to improve the performance of anytime sampling-based planning but has failed to fully realize the benefits of graph-based search. Existing techniques apply heuristics to only some aspects of the search (e.g., RRT$^{\#}$), sacrifice anytime performance (e.g., RA*, FMT*), or sacrifice efficiency by including other metrics in the ordering (e.g., hRRT, HRA*, SBA*).

This thesis focuses on using heuristics to improve sampling-based planners without sacrificing the benefits of either informed search or anytime approximation. It does so by carefully applying techniques from graph-based search to sampling-based planners and in the process shows how the two approaches can be unified (Fig. 2.2). It makes use of a variety of techniques from both fields, including efficient informed search (e.g., A*; Hart et al., 1968; Lazy Weighted A*; Cohen et al., 2014b; and LPA*; Koenig et al., 2004), efficient representations (e.g., Anytime RRTs; Ferguson and Stentz, 2006; and Quasi-random Lazy PRM; Branicky et al., 2001), almost-surely asymptotically optimal approximations (e.g., RRT*; Karaman and Frazzoli, 2011), and batched processing (e.g., FMT*; Janson et al., 2015b).

This work culminates in the development of informed anytime sampling-based planners that outperform existing techniques, especially in high state dimensions. Informed RRT* uses heuristics to focus RRT* after an initial solution is found. This allows for an algorithm that shrinks the search domain to only the set of states that could possible belong to a better solution. BIT* separates the approximation of the planning problem from its search and directly applies incremental graph-based searches (e.g., LPA*) on an anytime approximation built through sampling (e.g., RRT*). This allows for the search to proceed in order of potential solution quality and reduces the amount of search effort spent considering unnecessarily poor solutions. These concepts are also extended with SORRT* to the algorithmic simplicity of Informed RRT*.

Chapter 3

Informed Sampling

or: The counterintuitive nature of high-dimensional shapes

This chapter investigates ways to improve the rate at which anytime almost-surely asymptotically optimal planners improve solutions. It shows that one reason RRT* almost-surely converges slowly to the optimal solution, especially in high state dimensions, is because it asymptotically finds optimal paths to *every* state in the problem domain. This behaviour is inconsistent with single-query planning and can be improved by modifying the search domain.

Search domains are the set of states that could belong to a solution to a planning problem. Searching the problem domain is logical when seeking an initial solution (e.g., RRT) but not necessarily when seeking to improve an existing solution (e.g., RRT*). In these situations, algorithms only need to consider states that could belong to a better solution, an *omniscient set* that decreases in size as the solution improves (Ferguson and Stentz, 2006).

Section 3.2 shows that the sampling rate of the omniscient set is an upper bound on the probability of RRT* improving a solution. Exact knowledge of the omniscient set requires solving the planning problem but it can be estimated with *informed sets*. The performance of these informed sets will depend on their definition and can be described in terms of *precision* and *recall* (Definitions 3.3 and 3.4). Precision is the percentage of states in the informed set that can improve the solution (i.e., how much of the informed set is in the omniscient set) and recall is the percentage of states that could improve the solution that are in the informed set (i.e., how much of the omniscient set is in the informed set). An informed set defined by an admissible heuristic will be a superset of the omniscient set (i.e., 100% recall) and may be referred to as an *admissible informed set*.

A common admissible heuristic for problems seeking to minimize path length is the L^2 norm (i.e., Euclidean distance) between a state and the start and goal. The resulting informed set is a symmetric multidimensional hyperellipsoid (i.e., a prolate hyperspheroid) whose transverse axis is equal to the length of the current solution (Ferguson and Stentz, 2006; Otte and Correll, 2013). This heuristic is universally admissible and exactly describes the

omniscient set in the absence of obstacles (i.e., it is *sharp*[1]). This heuristic is well known but its insights on the minimum-path-length problem are poorly understood.

Section 3.3 shows that common versions of the minimum-path-length problem suffer from a curse of dimensionality. The difficulty of sampling the omniscient set makes the probability of improving a solution go to zero *factorially* (i.e., faster than exponentially) as state dimension increases. A method to improve this performance by directly sampling prolate hyperspheroids is presented. This efficiently generates uniform samples in the L^2 informed set regardless of its size or the state dimension and outperforms rejection sampling techniques by orders of magnitude as state dimension increases. Section 3.4 discusses how these results can be used to improve almost-surely asymptotically optimal planners.

In summary, this chapter makes the following novel contributions:

- Formally defines omniscient and informed sets (Definitions 3.1 and 3.2).
- Defines precision and recall as methods to quantify different informed sets (Definitions 3.3 and 3.4).
- Provides upper bounds on the probability of anytime sampling-based planners, such as RRT*, improving a solution at any iteration (Theorems 3.3 and 3.6).
- Shows that common formulations of the minimum-path-length problem have a probability of improving solutions that decreases factorially with state dimension (Theorem 3.7).
- Develops a method to avoid this curse of dimensionality for problems seeking to minimize path length to a goal or a countable set of goals by directly sampling the L^2 informed set (Algs. 3.2–3.6).

3.1 RRT*

A version of RRT* as it appeared in Karaman and Frazzoli (2010, 2011) is presented in Alg. 3.1 to motivate the remaining discussion. Necessary notation and subfunctions are detailed in Sections 3.1.1 and 3.1.2, the requirements for almost-sure asymptotic optimality are presented in Section 3.1.3, and its performance as a single-query planner is discussed in Section 3.1.4.

RRT* searches for the optimal path to a planning problem by building a tree through collision-free space (Fig. 2.8). The tree is constructed incrementally from the start state (Alg. 3.1, Line 1) by drawing samples randomly from the problem domain (Alg. 3.1, Line 3). These random samples are converted into potential new vertices with a `Steer` function that enforces expansion limits and any differential constraints (Alg. 3.1, Lines 4–5). If the connection from the nearest vertex to the potential new vertex passes solely through free space then the potential vertex is added to the graph (Alg. 3.1, Lines 6–7). This is the same incremental search as performed by RRT.

[1] A bound or estimate is said to be sharp if it is exactly equal to the true value for at least one element of the domain.

Algorithm 3.1: RRT*($\mathbf{x}_{\text{start}} \in X_{\text{free}},\ X_{\text{goal}} \subset X$)

```
1  V ← {x_start}; E ← ∅; T = (V, E);
2  for i = 1 ... q do
3      x_rand ← Sample;
4      v_nearest ← Nearest (V, x_rand);
5      x_new ← Steer (v_nearest, x_rand);
6      if CollisionFree (v_nearest, x_new) then
7          V ←+ {x_new};
8          V_near ← Near (V, x_new, r_rewire);
9          v_min ← v_nearest;
10         forall v_near ∈ V_near do
11             c_new ← g_T (v_near) + c (v_near, x_new);
12             if c_new < g_T (v_min) + c (v_min, x_new) then
13                 if CollisionFree (v_near, x_new) then
14                     v_min ← v_near;
15         E ←+ {(v_min, x_new)};
16         forall v_near ∈ V_near do
17             c_near ← g_T (x_new) + c (x_new, v_near);
18             if c_near < g_T (v_near) then
19                 if CollisionFree (x_new, v_near) then
20                     v_parent ← Parent (v_near);
21                     E ←− {(v_parent, v_near)};
22                     E ←+ {(x_new, v_near)};
23 return T;
```

RRT* achieves almost-sure asymptotic optimality by incrementally rewiring the tree in the local neighbourhood around each new vertex (Alg. 3.1, Line 8). It finds the best parent for the vertex (Alg. 3.1, Lines 9–15) and then checks if the vertex can improve any of its neighbours (Alg. 3.1, Lines 16–22).

RRT* finds the best parent for the newly added vertex by incrementally considering the cost to come to local vertices and the edge cost to the new state (Alg. 3.1, Line 11). The vertex that minimizes this sum becomes the parent of the newly added vertex (Alg. 3.1, Lines 14–15). A similar search is performed to improve the cost-to-come of existing vertices by incrementally considering the cost to come to the newly added vertex plus the edge cost to each existing local vertex (Alg. 3.1, Line 17). If this path improves the cost-to-come of a vertex and the new edge is collision free then the tree is rewired to use the newly added vertex (Alg. 3.1, Lines 20–22). This entire search procedure continues for a user-defined number of samples, q (Alg. 3.1, Line 2).

3.1.1 Notation

RRT*'s tree, $\mathcal{T} \coloneqq (V, E)$, is defined by a set of vertices, $V \subset X_{\text{free}}$, and edges, $E = \{(\mathbf{v}, \mathbf{w})\}$, for some $\mathbf{v}$, $\mathbf{w} \in V$. The function $g_{\mathcal{T}}(\mathbf{v})$ represents the cost to reach a vertex, $\mathbf{v} \in V$, from the start given the current tree, $\mathcal{T}$ (the cost-to-come). The function $c(\mathbf{v}, \mathbf{w})$ represents the cost of a path connecting the states $\mathbf{v}, \mathbf{w} \in X_{\text{free}}$, and corresponds to the edge cost between those two states if they are connected as vertices in the tree. The notation $X \xleftarrow{+} \{\mathbf{x}\}$ and $X \xleftarrow{-} \{\mathbf{x}\}$ is used to compactly represent the compounding set operations $X \leftarrow X \cup \{\mathbf{x}\}$ and $X \leftarrow X \setminus \{\mathbf{x}\}$, respectively.

3.1.2 Subfunctions

`CollisionFree` Given two states, $\mathbf{x}, \mathbf{y} \in X$, the function $\texttt{CollisionFree}(\mathbf{x}, \mathbf{y})$ returns `True` if and only if the path from $\mathbf{x}$ to $\mathbf{y}$, σ, lies solely in free space, otherwise it returns `False`, i.e.,

$$\texttt{CollisionFree}(\mathbf{x}, \mathbf{y}) \coloneqq \begin{cases} \texttt{True} & \text{if } \forall t \in [0, 1],\ \sigma(t) \in X_{\text{free}} \\ \texttt{False} & \text{otherwise} \end{cases}$$

`Near` Given a set of states, $Y \subset X$, a state, $\mathbf{x} \in X$, and a distance, $r \in \mathbb{R}_{\geq 0}$, the function $\texttt{Near}(Y, \mathbf{x}, r)$ returns the set of states, X_{near}, that are within a radius r of the state,

$$X_{\text{near}} \coloneqq \{\mathbf{y} \in Y \mid \|\mathbf{x} - \mathbf{y}\|_2 \leq r\},$$

where $\|\cdot\|_2$ is the L^2 norm (i.e., Euclidean distance).

`Nearest` Given a set of states, $Y \subseteq X$, and a state, $\mathbf{x} \in X$, the function $\texttt{Nearest}(Y, \mathbf{x})$ returns the state, $\mathbf{v}_{\text{nearest}}$, that is the nearest neighbour to the state, e.g., for geometric planning,

$$\mathbf{v}_{\text{nearest}} \coloneqq \underset{\mathbf{y} \in Y}{\arg\min} \{\|\mathbf{x} - \mathbf{y}\|_2\}. \tag{3.1}$$

`Parent` Given a tree, $\mathcal{T} = (V, E)$, and a vertex, $\mathbf{v} \in V$, the function $\texttt{Parent}(\mathcal{T}, \mathbf{v})$ returns the one and only direct ancestor of $\mathbf{v}$ in the tree,

$$\mathbf{v}_{\text{parent}} \coloneqq \mathbf{u} \text{ s.t. } (\mathbf{u}, \mathbf{v}) \in E.$$

`Sample` The function `Sample` returns independent and identically distributed (i.i.d.) samples from free space, e.g., $\mathbf{x}_{\text{rand}} \sim \mathcal{U}(X_{\text{free}})$.

`Steer` Given two states, $\mathbf{x}, \mathbf{z} \in X$, the function `Steer` $(\mathbf{x}, \mathbf{z})$ returns a new state, $\mathbf{x}_{\text{new}} \in X$, that is towards $\mathbf{z}$ from $\mathbf{x}$ and satisfies the expansion and any differential constraints. The expansion constraint takes the form of a user-selected maximum edge length between states, η, in geometric planning,

$$\mathbf{x}_{\text{new}} \coloneqq \underset{\mathbf{y} \in X}{\arg\min} \left\{ \|\mathbf{z} - \mathbf{y}\|_2 \;\middle|\; \|\mathbf{y} - \mathbf{x}\|_2 \leq \eta \right\}. \tag{3.2}$$

3.1.3 Almost-sure Asymptotic Optimality

RRT* must use sufficiently large rewiring neighbourhoods to almost-surely converge asymptotically to the optimum. Karaman and Frazzoli (2011) present two definitions of the neighbourhood for uniform sample distributions corresponding to r-disc and k-nearest variants of RRT*. They show that the r-disc variant of RRT* almost-surely converges asymptotically to the optimum when the local neighbourhood is the set of states within a radius, r_{rewire}, of the new state,

$$r_{\text{rewire}} \coloneqq \min \left\{ \eta, r_{\text{RRT}^*} \right\}, \tag{3.3}$$

where η is the maximum allowable edge length of the tree (see `Steer` in Section 3.1.2) and r_{RRT^*} is a function of the problem measure and the number of vertices in the tree. Specifically,

$$\begin{aligned} &r_{\text{RRT}^*} > r^*_{\text{RRT}^*}, \\ &r^*_{\text{RRT}^*} \coloneqq \left(2 \left(1 + \frac{1}{n} \right) \left(\frac{\lambda\left(X\right)}{\zeta_n} \right) \left(\frac{\log\left(|V|\right)}{|V|} \right) \right)^{\frac{1}{n}}, \end{aligned} \tag{3.4}$$

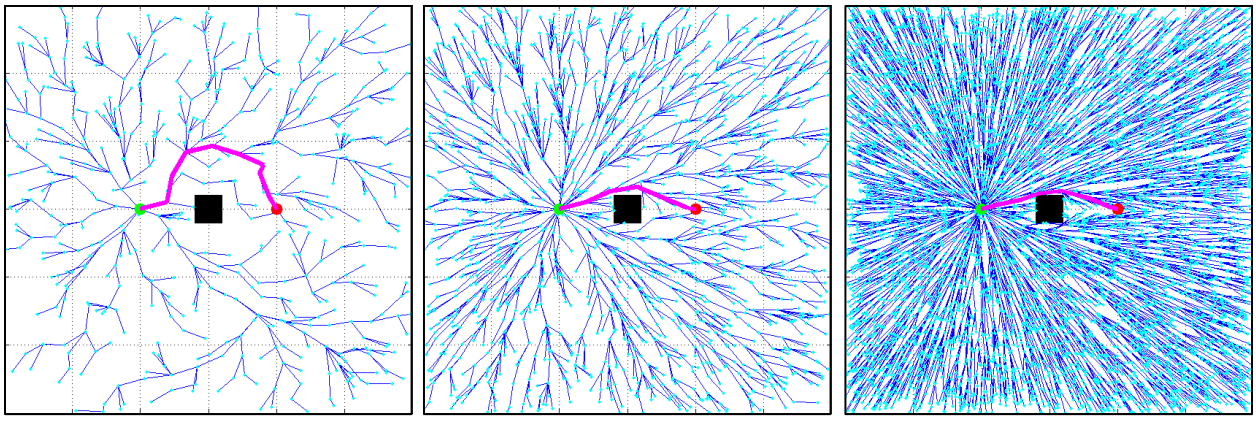
where $\lambda\left(\cdot\right)$ is the Lebesgue measure of a set (e.g., the *volume*), ζ_n is the Lebesgue measure of an n-dimensional unit ball, and $|\cdot|$ is the cardinality of a set.

They show that the k-nearest variant of RRT* almost-surely converges asymptotically to the optimum when the local neighbourhood is the k_{RRT^*}-nearest states, where

$$\begin{aligned} &k_{\text{RRT}^*} > k^*_{\text{RRT}^*}, \\ &k^*_{\text{RRT}^*} \coloneqq e \left(1 + \frac{1}{n} \right) \log\left(|V|\right). \end{aligned} \tag{3.5}$$

3.1.4 Single-query Performance

RRT* incrementally builds a tree out from the start through the free space of a problem domain. This allows it to search the entire domain for a solution (i.e., it is space filling); however, outside of a small sampling bias it leaves the search independent of the goal. This results in a search that almost-surely converges asymptotically to the optimum by



(a) (b) (c)

Figure 3.1: An example of RRT* asymptotically finding the optimal path to every state in the problem domain. RRT* approximates and searches planning problems through incremental sampling. New samples are added to a tree rooted at the start to both expand its search and improve existing connections. This process continues indefinitely and almost-surely converges asymptotically to the optimal solution by asymptotically improving every path in the tree, (a)–(c). This is inefficient in single-query planning scenarios.

asymptotically finding the optimal paths to *every* state in the problem domain (Fig. 3.1). This is inefficient in single-query planning scenarios and can be improved by exploiting the start and goal states to focus the search.

Problem domains are the set of states that could provide a solution to a given problem. It is not necessary to search this entire set once a solution is found as only states that could belong to a better solution need to be considered. This is exploited by Anytime RRTs (Ferguson and Stentz, 2006) to improve the quality of solutions found by RRT. A series of shrinking problem domains defined by the previous solution are searched independently in an attempt to find increasingly better solutions in an anytime fashion. These results are not asymptotically optimal but similar techniques can be adapted for almost-surely asymptotically optimal planners, such as RRT*, by formally defining the required search domains (Section 3.2).

3.2 Omniscient and Informed Sets

RRT* only needs to search the set of states that could belong to a better solution once a solution is found. This set is a function of the current solution cost and the optimal cost-to-come and cost-to-go of every state in the problem domain (Definition 3.1).

Definition 3.1 (Omniscient set)**.** *Let $g(\mathbf{x})$ be the cost of the optimal path from the start to a state, $\mathbf{x} \in X$, the optimal cost-to-come,*

$$g(\mathbf{x}) \coloneqq \min_{\sigma \in \Sigma} \{c(\sigma) \mid \sigma(0) = \mathbf{x}_{\text{start}}, \sigma(1) = \mathbf{x}\},$$

and $h(\mathbf{x})$ *be the cost of the optimal path from* $\mathbf{x}$ *to the goal region, the optimal cost-to-go,*

$$h(\mathbf{x}) \coloneqq \min_{\sigma \in \Sigma} \{c(\sigma) \mid \sigma(0) = \mathbf{x}, \sigma(1) \in X_{\text{goal}}\}.$$

The cost of the optimal path from $\mathbf{x}_{\text{start}}$ *to* X_{goal} *constrained to pass through* $\mathbf{x}$ *is then given by* $f(\mathbf{x}) \coloneqq g(\mathbf{x}) + h(\mathbf{x})$*. This defines the subset of states that can belong to a solution better than the current solution,* c_i*, as*

$$X_f \coloneqq \{\mathbf{x} \in X_{\text{free}} \mid f(\mathbf{x}) < c_i\}. \tag{3.6}$$

Exact knowledge of X_f *requires solving the entire planning problem so it will be referred to as the* omniscient set*.*

RRT* incrementally builds and improves a tree by adding states from the problem domain. A necessary condition for this process to improve a solution at any iteration is to add a state to the tree from the omniscient set (Lemma 3.1).

Lemma 3.1 (The necessity of adding states in the omniscient set)**.** *Adding a state from the omniscient set,* $\mathbf{x}_{\text{new}} \in X_f$*, is a necessary condition for RRT* to improve the current solution,* c_i*,*

$$c_{i+1} < c_i \implies \mathbf{x}_{\text{new}} \in X_f.$$

This condition is not sufficient *to improve the solution as the ability of states in* X_f *to provide better solutions at any iteration depends on the optimality of the tree.*

Proof. The proof of Lemma 3.1 is presented in Appendix A.1. □

RRT* generates potential new states by applying expansion and differential constraints between the randomly generated samples and existing states in the tree (see `Steer` in Section 3.1.2). The number of states in the tree increases as RRT* expands through the planning domain and after an initial κ iterations the entire problem will be no more than η away from the tree. This removes the expansion constraint and makes sampling the omniscient set a necessary condition to improving the solution (Lemma 3.2).

Lemma 3.2 (The necessity of sampling states in the omniscient set)**.** *Sampling the omniscient set,* $\mathbf{x}_{\text{rand}} \in X_f$*, is a necessary condition for RRT* to improve the current solution,* c_i*, after an initial* κ *iterations,*

$$\forall i \geq \kappa,\ c_{i+1} < c_i \implies \mathbf{x}_{\text{rand}} \in X_f,$$

for any sample distribution that maintains a nonzero probability over the entire omniscient set.

Proof. The proof of Lemma 3.2 is presented in Appendix A.2. □

This need to sample the omniscient set to improve a solution provides an upper limit on the probability of RRT* improving a solution at any iteration (Theorem 3.3).

Theorem 3.3 (An upper bound on the probability of improving a solution given knowledge of the omniscient set)**.** *The probability that an iteration of RRT* improves the current solution, c_i, is bounded by the probability of sampling the omniscient set, X_f,*

$$\forall i \geq \kappa,\, P\left(c_{i+1} < c_i\right) \leq P\left(\mathbf{x}_{\text{rand}} \in X_f\right),$$

for any iteration, i, after a sufficient state density is achieved in the initial κ iterations.

Proof. Proof of Theorem 3.3 follows directly from Lemma 3.2. Sampling a state in X_f is a necessary condition to improve the solution after κ iterations; therefore, the probability of sampling such a state provides an upper bound on the probability of improving the solution. □

Knowledge of an omniscient set requires solving the planning problem and these results can be extended to estimates of the omniscient set. These estimates are often defined in terms of solution cost heuristics (Definition 3.2).

Definition 3.2 (Informed set)**.** *Let $\widehat{f}(\mathbf{x})$ represent a heuristic estimate of the solution cost constrained to go through a state, $\mathbf{x} \in X$, then a heuristic estimate of the omniscient set can be defined as*

$$X_{\widehat{f}} \coloneqq \left\{\mathbf{x} \in X \;\middle|\; \widehat{f}(\mathbf{x}) < c_i\right\}.$$

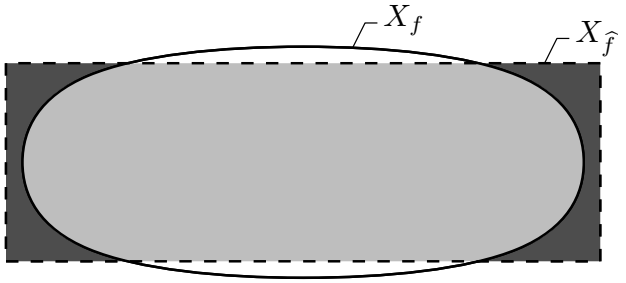
Such a set will be referred to as an informed set*.*

There are an infinite number of potential informed sets for every planning problem and choosing the 'best' set requires methods to quantify their performance. Estimates of a set are evaluated in binary classification in terms of their precision and recall (Fig. 3.2). Analogue terms can be defined in sampling-based planning to quantify the ability of informed sets to estimate the omniscient set (Definitions 3.3 and 3.4).

Definition 3.3 (Precision for informed sampling techniques)**.** *The precision of an informed sampling technique is the probability that random samples from the informed set could also be drawn from the omniscient set (e.g., the percentage of states drawn from the informed set, $X_{\widehat{f}}$, that belong to the omniscient set, X_f). For uniform sampling of an informed set, this is a ratio of measures,*

$$\text{Precision}\left(X_{\widehat{f}}\right) \coloneqq \frac{\lambda\left(X_{\widehat{f}} \cap X_f\right)}{\lambda\left(X_{\widehat{f}}\right)}.$$

Any subset of the omniscient set with nonzero sampling probability will have 100% *precision.*



$$\text{Precision}\left(X_{\widehat{f}}\right) := \frac{\lambda\left(X_{\widehat{f}} \cap X_f\right)}{\lambda\left(X_{\widehat{f}}\right)} \equiv \frac{\square}{\square + \blacksquare}$$

$$\text{Recall}\left(X_{\widehat{f}}\right) := \frac{\lambda\left(X_{\widehat{f}} \cap X_f\right)}{\lambda\left(X_f\right)} \equiv \frac{\square}{\square + \square}$$

Figure 3.2: An illustration of the precision and recall of a rectangular informed estimate, $X_{\widehat{f}}$, of an oblong omniscient set, X_f. The informed set is rendered in grey with shades depending on whether it correctly (light grey) or incorrectly (dark grey) estimates the omniscient set. The sections of the omniscient set that are not estimated by the informed set are shown in white. Precision is the likelihood of correctly sampling the omniscient set by sampling the informed set. Recall is the coverage of the omniscient set by the informed set. These terms are both ratios of Lebesgue measures for uniform distributions.

Definition 3.4 (Recall for informed sampling techniques)**.** *The recall of an informed sampling technique is the probability that random states drawn from the omniscient set could also be sampled from the informed set (e.g., the percentage of states that belong to the omniscient set, X_f, with a nonzero probability of being sampled from the informed set, $X_{\widehat{f}}$). For uniform sampling of an informed set, this is a ratio of measures,*

$$\text{Recall}\left(X_{\widehat{f}}\right) \coloneqq \frac{\lambda\left(X_{\widehat{f}} \cap X_f\right)}{\lambda\left(X_f\right)}.$$

Any superset of the omniscient set with nonzero sampling probability will have 100% *recall.*

Sampling informed sets with high precision can rapidly improve solutions but may prevent finding the optimal solution if the set has less than 100% recall (e.g., sampling near an existing solution). Alternatively, sampling informed sets with 100% recall maintains almost-sure asymptotic convergence to the optimum but may not improve convergence if the set has low precision it (e.g., sampling the entire problem domain). Almost-surely asymptotically optimal sampling-based planners can therefore be improved by finding high-precision informed sets with 100% recall (Definition 3.5).

Definition 3.5 (Admissible informed set)**.** *A heuristic is said to be* admissible *if it never overestimates the true value of the function,*

$$\forall \mathbf{x} \in X,\ \widehat{f}\left(\mathbf{x}\right) \leq f\left(\mathbf{x}\right).$$

Any informed set defined by such an admissible heuristic will contain all possibly better

solutions and have 100% *recall, i.e.,*

$$X_{\widehat{f}} \supseteq X_f.$$

This set will be referred to as an admissible estimate of the omniscient set, or an admissible informed set*. If the heuristic is an admissible estimate of the cost function for all possible problems then the set will be referred to as a* universally *admissible informed set.*

These definitions of informed sets, precision, and recall allow Lemmas 3.1 and 3.2 and Theorem 3.3 to be extended to any admissible estimate of the omniscient set (Lemmas 3.4 and 3.5 and Theorem 3.6). These bound the probability of improving a solution at any iteration by the probability of sampling an admissible informed set. The tightness of this bound will depend on the precision of the chosen estimate and motivates the need to define high-precision universally admissible informed sets for common planning problems (Section 3.3).

Lemma 3.4 (The necessity of adding states in an admissible informed set)**.** *Adding a state from an admissible informed set,* $\mathbf{x}_{\text{new}} \in X_{\widehat{f}} \supseteq X_f$*, is a necessary condition for RRT* to improve the current solution,* c_i*,*

$$c_{i+1} < c_i \implies \mathbf{x}_{\text{new}} \in X_{\widehat{f}} \supseteq X_f.$$

Proof. Proof of Lemma 3.4 follows directly from Lemma 3.1 given that $X_{\widehat{f}} \supseteq X_f$. □

Lemma 3.5 (The necessity of sampling states in an admissible informed set)**.** *Sampling an admissible informed set,* $\mathbf{x}_{\text{rand}} \in X_{\widehat{f}} \supseteq X_f$*, is a necessary condition for RRT* to improve the current solution,* c_i*, after an initial* κ *iterations,*

$$\forall i \geq \kappa,\, c_{i+1} < c_i \implies \mathbf{x}_{\text{rand}} \in X_{\widehat{f}} \supseteq X_f,$$

for any sample distribution that maintains a nonzero probability over the entire informed set.

Proof. Proof of Lemma 3.5 follows directly from Lemma 3.2 given that $X_{\widehat{f}} \supseteq X_f$. □

Theorem 3.6 (An upper bound on the probability of improving a solution given knowledge of an admissible informed set)**.** *The probability that an iteration of RRT* improves the current solution,* c_i*, is bounded by the probability of sampling an admissible informed set,* $X_{\widehat{f}} \supseteq X_f$*,*

$$\forall i \geq \kappa,\, P\left(c_{i+1} < c_i\right) \leq P\left(\mathbf{x}_{\text{rand}} \in X_f\right) \leq P\left(\mathbf{x}_{\text{rand}} \in X_{\widehat{f}}\right),$$

for any iteration, i*, after an initial* κ *iterations.*

Proof. Proof of Theorem 3.6 follows directly from Theorem 3.3 given that $X_{\widehat{f}} \supseteq X_f$. □

3.3 The L^2 Informed Set

A universally admissible heuristic is well defined for problems seeking to minimize path length in $\mathbb{R}^n$ to a single goal (Ferguson and Stentz, 2006; Otte and Correll, 2013). The cost of a solution constrained to pass through any state is bounded from below by the L^2 norm (i.e., Euclidean distance) between the state, $\mathbf{x} \in X$, and the start, $\mathbf{x}_{\text{start}}$, and goal, $\mathbf{x}_{\text{goal}}$,

$$\begin{aligned}\widehat{f}(\mathbf{x}) &= \widehat{g}(\mathbf{x}) + \widehat{h}(\mathbf{x}) \\ &= \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2 .\end{aligned} \tag{3.7}$$

The set of states that could provide a better solution than the current solution cost, c_i, is then the L^2 informed set,

$$X_{\widehat{f}} = \left\{\mathbf{x} \in X_{\text{free}} \;\middle|\; \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2 < c_i\right\}.$$

This informed set is a universally admissible estimate of the omniscient set and is exact in the absence of obstacles (i.e., it is sharp over all minimum-path-length problems). The size of this informed set will decrease as solutions improve.

The L^2 informed set is the intersection of the free space, X_{free}, and a n-dimensional hyperellipsoid symmetric about its transverse axis (i.e., a prolate hyperspheroid),

$$X_{\widehat{f}} = X_{\text{free}} \cap X_{\text{PHS}},$$

where

$$X_{\text{PHS}} \coloneqq \left\{\mathbf{x} \in \mathbb{R}^n \;\middle|\; \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2 < c_i\right\}.$$

The prolate hyperspheroid has focal points at $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_{\text{goal}}$, a transverse diameter of c_i, and conjugate diameters of $\sqrt{c_i^2 - c_{\min}^2}$, where

$$c_{\min} \coloneqq \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\|_2 ,$$

is the theoretical minimum cost (Fig. 3.3). The measure of the informed set is

$$\lambda\left(X_{\widehat{f}}\right) \leq \lambda\left(X_{\text{PHS}}\right) = \frac{c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}} \zeta_n}{2^n}, \tag{3.8}$$

where ζ_n is the Lebesgue measure of an n-dimensional unit ball,

$$\zeta_n \coloneqq \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)},$$

and $\Gamma(\cdot)$ is the gamma function, an extension of factorials to real numbers (Euler, 1738).

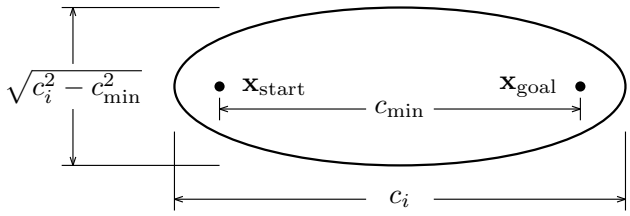


Figure 3.3: The L^2 informed set, $X_{\widehat{f}}$, for a problem seeking to minimize path length in $\mathbb{R}^2$ is an ellipse with the initial state, $\mathbf{x}_{\text{start}}$, and the goal state, $\mathbf{x}_{\text{goal}}$ as focal points. The shape of the ellipse depends on both the initial and goal states, the theoretical minimum cost between the two, $c_{\min}$, and the cost of the best solution found to date, c_i. The eccentricity of the ellipse is given by $c_{\min}/c_i$.

The probability of uniformly sampling this informed set by sampling any superset (e.g., a bounding box), $X_{\text{sampling}} \supseteq X_{\widehat{f}}$, can be written as a ratio of measures,

$$P\left(\mathbf{x}_{\text{rand}} \in X_{\widehat{f}} \;\middle|\; \mathbf{x}_{\text{rand}} \sim \mathcal{U}\left(X_{\text{sampling}}\right)\right) \leq \frac{\lambda\left(X_{\text{PHS}}\right)}{\lambda\left(X_{\text{sampling}}\right)} = \frac{\pi^{\frac{n}{2}} c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right) \lambda\left(X_{\text{sampling}}\right)}. \quad (3.9)$$

This provides a bound on the probability of improving the solution (Theorem 3.6),

$$\forall i \geq \kappa,\; P\left(c_{i+1} < c_i \;\middle|\; \mathbf{x}_{\text{rand}} \sim \mathcal{U}\left(X_{\text{sampling}}\right)\right) \leq \frac{\pi^{\frac{n}{2}} c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right) \lambda\left(X_{\text{sampling}}\right)}. \quad (3.10)$$

The probability in (3.10) becomes arbitrarily small for (i) costs, c_i, near the theoretical limit, $c_{\min}$, (ii) large sampling domains, $\lambda\left(X_{\text{sampling}}\right)$, or (iii) high state dimensions, n. The solution cost and the sampling domain size may vary with iteration but the state dimension is a constant of the planning problem. This motivates investigating the effect of state dimension on the most common formulations of the minimum-path-length planning problem (Theorem 3.7).

Theorem 3.7 (The minimum-path-length curse of dimensionality)**.** *The probability that RRT* improves a solution to a problem seeking to minimize path length decreases* factorially *(i.e., faster than exponentially) as state dimension increases,*

$$\forall i \geq \kappa,\; P\left(c_{i+1} < c_i \;\middle|\; \mathbf{x}_{\text{rand}} \sim \mathcal{U}\left(X_{\text{rect}}\right)\right) \leq \frac{\pi^{\frac{n}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right)}, \quad (3.11)$$

when uniformly sampling a (hyper)rectangle bounding the L^2 informed set, $X_{\text{rect}} \supset X_{\text{PHS}} \supseteq X_{\widehat{f}} \supseteq X_f$.

Proof. Theorem 3.7 is proven for RRT* but the theorem and proof holds for any planning algorithm for which an equivalent to Theorem 3.6 exists.

The smallest possible X_{rect} that completely contains X_{PHS} is a (hyper)rectangle with widths that correspond to the diameters of the prolate hyperspheroid (Fig. 3.4). The measure

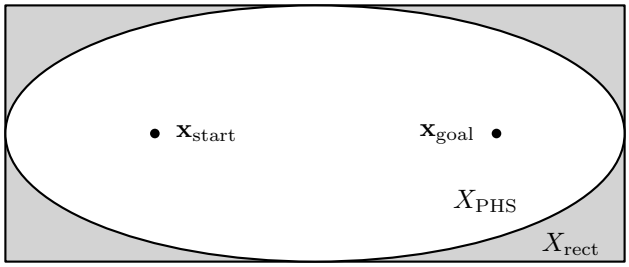


Figure 3.4: The minimal rectangular sampling domain, X_{rect}, that completely contains the prolate hyperspheroid set, $X_{\text{PHS}} \supseteq X_{\hat{f}}$, as defined by the start, $\mathbf{x}_{\text{start}}$, goal, $\mathbf{x}_{\text{goal}}$, and the current solution, c_i, of a planning problem. This is the best-case sampling domain for the algorithm proposed by Otte and Correll (2013). The rectangle tightly bounds X_{PHS}; however, the probability of finding a state in X_{PHS} by sampling X_{rect} decreases factorially with state dimension due to the nature of high-dimensional space. The probability of RRT* improving a solution with samples generated from any admissible bounding rectangle therefore decreases rapidly with state dimension.

of any $X_{\text{rect}} \supset X_{\text{PHS}}$ is therefore bounded from below as

$$\lambda\left(X_{\text{rect}}\right) \geq c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}}. \tag{3.12}$$

When substituted into (3.10) this gives

$$\forall i \geq \kappa,\ P\left(c_{i+1} < c_i \ \mid\ \mathbf{x}_{\text{rand}} \sim \mathcal{U}\left(X_{\text{rect}}\right)\right) \leq \frac{\pi^{\frac{n}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right)},$$

proving Theorem 3.7 for all rectangular sets, X_{rect}, such that $X_{\text{rect}} \supset X_{\text{PHS}} \supseteq X_{\hat{f}} \supseteq X_f$. □

Theorem 3.7 is an upper bound on the utility of rectangular rejection sampling and is illustrated by plotting (3.11) versus state dimension (Fig. 3.5). The results show that rectangular rejection sampling can be up to a 78.6% successful in $\mathbb{R}^2$ but that its success decreases factorially as the state dimension increases, becoming 1.6% in $\mathbb{R}^8$, and 3.6×10^{-4}% in $\mathbb{R}^{16}$. These numbers represent the *best-case* probability of improvement with rectangular rejection sampling and actual performance will depend on the size and orientation of the informed set relative to the sampling domain. This motivates a need for a method to directly sample the prolate hyperspheroid regardless of size, orientation, or state dimension (Section 3.3.1).

3.3.1 Direct Sampling

A method to generate uniformly distributed samples in the L^2 informed set is adapted from techniques to generate samples from hyperellipsoids (Sun and Farooq, 2002).

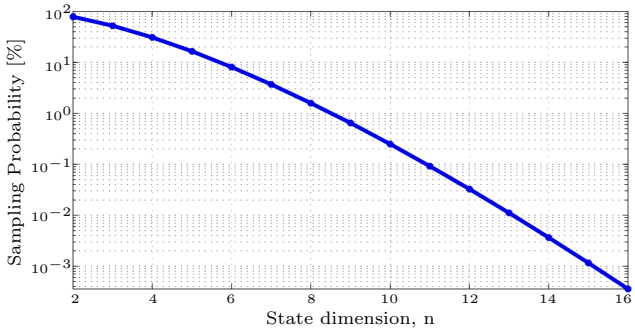


Figure 3.5: The upper-bound probability of sampling a prolate hyperspheroid, $X_{\text{PHS}} \supseteq X_{\hat{f}}$, from a larger hyperrectangular sampling domain, $X_{\text{rect}} \supseteq X_{\text{PHS}}$, versus state dimension, n. The sampling probability decreases factorially (i.e., faster than exponentially) as state dimension increases (Theorem 3.7). This provides an upper-bound on the probability of RRT* improving a solution at any iteration and shows that common formulations of RRT* do not scale effectively to high state dimensions.

Let $\mathbf{S} \in \mathbb{R}^{n \times n}$ be a symmetric, positive-definite matrix (the hyperellipsoid matrix) such that the interior of a hyperellipsoid, X_{ellipse}, is defined as

$$X_{\text{ellipse}} \coloneqq \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; (\mathbf{x} - \mathbf{x}_{\text{centre}})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{x}_{\text{centre}}) < 1 \right\}, \tag{3.13}$$

where $\mathbf{x}_{\text{centre}}$ is the centre point of the hyperellipsoid. Uniformly distributed samples in the hyperellipsoid, $\mathbf{x}_{\text{ellipse}} \sim \mathcal{U}(X_{\text{ellipse}})$, can be generated from uniformly distributed samples in the interior of a unit n-dimensional ball, $\mathbf{x}_{\text{ball}} \sim \mathcal{U}(X_{\text{ball}})$, by

$$\mathbf{x}_{\text{ellipse}} = \mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}, \tag{3.14}$$

where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is a lower-triangular transformation matrix calculated from the Cholesky decomposition of the hyperellipsoid matrix,

$$\mathbf{L}\mathbf{L}^T \equiv \mathbf{S},$$

and

$$X_{\text{ball}} \coloneqq \{ \mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_2 < 1 \}.$$

For hyperellipsoids with orthogonal axes, there exists a coordinate frame in which the hyperellipsoid matrix is diagonal,

$$\mathbf{S}' \coloneqq \text{diag}\left(r_1^2, r_2^2, \ldots, r_n^2\right),$$

where r_j is the radius of j-th axis of the hyperellipsoid and $\text{diag}(\cdot)$ denotes a diagonal matrix. A rotation from this hyperellipsoid-aligned frame to the world frame, $\mathbf{C}_{\text{we}} \in SO(n)$, can be

used to write (3.13) in terms of $\mathbf{S}'$ as

$$X_{\text{ellipse}} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; (\mathbf{x} - \mathbf{x}_{\text{centre}})^T \mathbf{C}_{\text{we}} \mathbf{S}'^{-1} \mathbf{C}_{\text{we}}^T (\mathbf{x} - \mathbf{x}_{\text{centre}}) < 1 \right\},$$

and (3.14) as

$$\mathbf{x}_{\text{ellipse}} = \mathbf{C}_{\text{we}} \mathbf{L}' \mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}, \tag{3.15}$$

where $\mathbf{L}'\mathbf{L}'^T \equiv \mathbf{S}'$ due to the orthogonality of rotation matrices, $\mathbf{C}_{\text{we}}^{-1} \equiv \mathbf{C}_{\text{we}}^T$.

The rotation between frames can be solved directly as a general Wahba problem (Wahba, 1965) which has a valid solution even when underspecified (de Ruiter and Forbes, 2013). Generally, the rotation matrix from one set of axes, $\{\mathbf{a}_j\}$, to another set of axes, $\{\mathbf{b}_j\}$, is given by

$$\mathbf{C}_{\text{ba}} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^T, \tag{3.16}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is

$$\mathbf{\Lambda} \coloneqq \text{diag}\left(1, \ldots, 1, \det(\mathbf{U}) \det(\mathbf{V})\right),$$

and $\det(\cdot)$ is the matrix determinant. The terms $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are unitary matrices such that $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \equiv \mathbf{M}$ via singular value decomposition, and $\mathbf{M} \in \mathbb{R}^{n \times n}$ is given by the outer product of the $j \leq n$ corresponding axes,

$$\mathbf{M} \coloneqq [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_j] [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_j]^T. \tag{3.17}$$

The hyperellipsoid for problems seeking to minimize path length is a prolate hyperspheroid described as,

$$\mathbf{x}_{\text{centre}} \coloneqq \frac{\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}}}{2}, \tag{3.18}$$

$$\mathbf{S}' \coloneqq \text{diag}\left(\frac{c_i^2}{4}, \frac{c_i^2 - c_{\min}^2}{4}, \ldots, \frac{c_i^2 - c_{\min}^2}{4} \right),$$

and,

$$\mathbf{L}' = \text{diag}\left(\frac{c_i}{2}, \frac{\sqrt{c_i^2 - c_{\min}^2}}{2}, \ldots, \frac{\sqrt{c_i^2 - c_{\min}^2}}{2} \right). \tag{3.19}$$

The local coordinate system is underspecified in the conjugate directions due to symmetry, making (3.17) just

$$\mathbf{M} = \mathbf{a}_1 \mathbf{1}_1^T, \tag{3.20}$$

where $\mathbf{1}_1$ the first column of the identity matrix and the transverse axis in the world frame is

$$\mathbf{a}_1 = (\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}) / \left\| \mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}} \right\|_2.$$

Algorithm 3.2: `Sample` $(\mathbf{x}_{\text{start}} \in X,\ \mathbf{x}_{\text{goal}} \in X,\ c_i \in \mathbb{R}_{\geq 0})$

```
1 repeat
2     if λ(X_PHS) < λ(X) then
3         x_rand ← SamplePHS(x_start, x_goal, c_i);
4     else
5         x_rand ← SampleProblem(X);
6 until x_rand ∈ X_free ∩ X_PHS;
7 return x_rand;
```

Samples distributed uniformly in the L^2 informed set, $X_{\hat{f}} = X_{\text{PHS}} \cap X_{\text{free}}$, can therefore be generated by using (3.15) to transform samples drawn uniformly from a unit n-ball. These samples are uniformly distributed in the prolate hyperspheroid through scaling, (3.19), rotation, (3.16) & (3.20), and translation, (3.18).

Sun and Farooq (2002) investigate various methods to generate samples in hyperellipsoids and provide the following lemma regarding uniform sample density.

Lemma 3.8 (The uniform distribution of samples transformed into a hyperellipsoid from a unit n-ball. Originally Lemma 1 in Sun and Farooq, 2002)**.** *If the random points distributed in a hyperellipsoid are generated from the random points uniformly distributed in a hypersphere through a linear invertible nonorthogonal transformation, then the random points distributed in the hyperellipsoid are also uniformly distributed.*

Proof. For brevity, Sun and Farooq (2002) only present anecdotal proofs of Lemma 3.8. A full proof is presented in Appendix B. □

3.3.1.1 Algorithm

The L^2 informed set is an arbitrary intersection of the prolate hyperspheroid and the problem domain. It can be sampled efficiently by considering the relative measure of the two sets and sampling the smaller set until a sample belonging to both sets is found. These procedures are presented algorithmically in Algs. 3.2 and 3.3 and are publicly available in OMPL. Note that Alg. 3.3, Lines 1–6 are constant for most problems and only need to be calculated once.

The function `SVD` $(\cdot)$ denotes the singular value decomposition of a matrix and `SampleUnitBall` (n) returns i.i.d. uniformly distributed samples from the interior of an n-dimensional unit ball. The measure of the prolate hyperspheroid, $\lambda(X_{\text{PHS}})$, is given by (3.8) and `SampleProblem` returns i.i.d. samples uniformly distributed over the entire planning domain. Implementations of `SVD` and `SampleUnitBall` can be found in common C++ libraries, such as Eigen and Boost, respectively.

Algorithm 3.3: `SamplePHS` $(\mathbf{x}_{\text{start}} \in X,\ \mathbf{x}_{\text{goal}} \in X,\ c_i \in \mathbb{R}_{\geq 0})$

$c_{\min} \leftarrow \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\|_2$;
$\mathbf{x}_{\text{centre}} \leftarrow (\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}})/2$;
$\mathbf{a}_1 \leftarrow (\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}})/c_{\min}$;
$\{\mathbf{U}, \mathbf{V}\} \leftarrow \texttt{SVD}\left(\mathbf{a}_1 \mathbf{1}_1^T\right)$;
$\mathbf{\Lambda} \leftarrow \text{diag}\left(1, \ldots, 1, \det(\mathbf{U}) \det(\mathbf{V})\right)$;
$\mathbf{C}_{\text{we}} \leftarrow \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$;
$r_1 \leftarrow c_i/2$;
$\{r_j\}_{j=2,\ldots,n} \leftarrow \left(\sqrt{c_i^2 - c_{\min}^2}\right)/2$;
$\mathbf{L} \leftarrow \text{diag}\left(r_1, r_2, \ldots, r_n\right)$;
$\mathbf{x}_{\text{ball}} \leftarrow \texttt{SampleUnitBall}\,(n)$;
$\mathbf{x}_{\text{rand}} \leftarrow \mathbf{C}_{\text{we}}\mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}$;
return $\mathbf{x}_{\text{rand}}$;

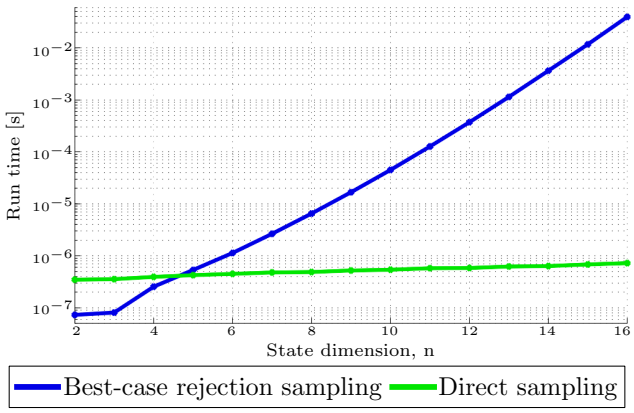


Figure 3.6: The time required to generate a sample inside the L^2 informed set using informed sampling (Alg. 3.2) and the *best-case* results for rectangular rejection sampling. Rejection sampling times were calculated from a planning problem where the domain tightly bounds the L^2 informed set (e.g., Fig. 3.4). This underestimates the computational cost of both global and tightly-bounded versions of rejection sampling. In global sampling, the problem domain generally does not tightly bound the L^2 informed set and a higher percentage of samples will be rejected. In focused rejection sampling, the start and goal are generally not aligned with the global coordinate system and either the bounding box must be relaxed, increasing the number of rejected samples (Otte and Correll, 2013), or a coordinate rotation must be applied to align the problem, increasing computational cost. Samples for Alg. 3.3 were generated from the unit n-ball with Boost 1.58.

3.3.1.2 Practical Performance

Direct informed sampling (Alg. 3.2) is compared to rejection sampling by measuring the computational time required for each algorithm to find samples in the L^2 informed set. The average time is calculated by generating 10^6 samples in the informed set at each dimension (Fig. 3.6). The results show that rejection sampling slightly outperforms direct informed sampling in low state dimensions (e.g., 7.3×10^{-8} seconds vs. $\mathbb{R}^2$: 3.5×10^{-7} seconds) but becomes orders of magnitude slower as state dimension increases (e.g., $\mathbb{R}^{16}$: 4.0×10^{-2} seconds vs. 7.2×10^{-7} seconds).

These per-sample times may appear small but RRT* and similar sampling-based planners require high sample densities to find high-quality solutions. This need is only exacerbated by the curse of dimensionality that sees the number of samples necessary to achieve equivalent sample densities increase in high state dimensions (e.g., when planning for manipulation). In $\mathbb{R}^{16}$ direct informed sampling can generate 10^5 samples in less than a second (7.2×10^{-2} seconds) while rectangular rejection sampling requires over an hour (3953 seconds).

The initial samples used for rejection sampling in this experiment were drawn from the rectangle that tightly bounds the informed set (Fig. 3.4). This represents the *best case* for admissible rectangular rejection sampling and real-world performance will be worse as sampling domains will either be larger relative to the informed set or require additional computational cost.

Constant sample domains (e.g., the problem domain) are inexpensive to sample but are independent of the informed set. The relative size of the informed set decreases as solutions improve and further reduces performance compared to the experiment. Dynamic sampling domains (e.g., a rectangle approximating the L^2 informed set; Otte and Correll, 2013) adapt to the shrinking informed set but must deal with its alignment to the problem coordinate frame. If the informed set is not aligned, then these algorithms must either use an expanded bounding box that contains the entire informed set or rotate a bounding box to tightly fit the prolate hyperspheroid, as in direct informed sampling. This either decreases the relative size of the informed set or increases the computational cost of the initial sampling and further reduces performance compared to the experiment.

Experimentally, the main cost of Alg. 3.2 appears to be the cost of sampling the unit n-ball. This is outside the scope of this thesis but suggests potential areas for further improvement.

3.3.2 Extension to Multiple Goals

Many planning problems seek the minimum-length path that connects a start to any state in a goal region, X_{goal}. The omniscient set in these situations consists of all states that could provide a better solution to any goal and the associated multigoal L^2 informed set is given by

$$X_{\hat{f}} \coloneqq \left\{ \mathbf{x} \in X_{\text{free}} \;\middle|\; \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal},j} - \mathbf{x}\|_2 < c_i \text{ for any } \mathbf{x}_{\text{goal},j} \in X_{\text{goal}} \right\}.$$

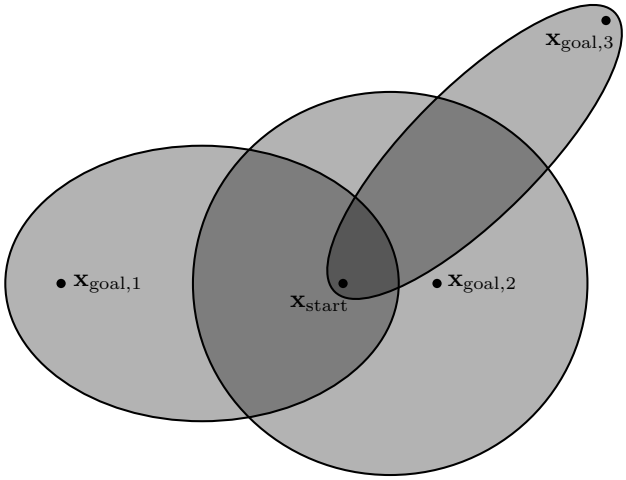


Figure 3.7: An illustration of the multigoal L^2 informed set for a problem seeking to minimize path length from a start, $\mathbf{x}_{\text{start}}$, at $[0, 0]^T$, to any of three goals, $\mathbf{x}_{\text{goal},j}$, at $[-0.75,\ 0]^T$, $[0.25,\ 0]^T$, and $[0.7,\ 0.7]^T$, and a current solution cost of $c_i = 1.05$. Each ellipse illustrates the L^2 informed set of a start-goal pair. Individual uniform distributions (light grey) create a *nonuniform* distribution over the multigoal L^2 informed set (dark grey) when the ellipses overlap.

This set for a countable goal region, $X_{\text{goal}} = \{\mathbf{x}_{\text{goal},j}\}_{j=1}^{z}$, is the union of the individual informed sets of each goal,

$$X_{\widehat{f}} = \bigcup_{j=1}^{z} X_{\widehat{f},j},$$

where z is the number of goals and

$$X_{\widehat{f},j} \coloneqq \left\{\mathbf{x} \in X_{\text{free}} \;\middle|\; \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal},j} - \mathbf{x}\|_2 < c_i\right\},$$

is the L^2 informed set of an individual $(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal},j})$ pair.

If the individual informed sets are *independent* and do not intersect then a uniform sample distribution over the multigoal informed set can be generated by randomly selecting an individual subset, j, in proportion to its relative measure,

$$p\left(1 \leq j \leq z\right) \coloneqq \frac{\lambda\left(X_{\widehat{f},j}\right)}{\sum_{k=1}^{z} \lambda\left(X_{\widehat{f},k}\right)},$$

and then generating a sample with uniform distribution inside the selected subset, $X_{\widehat{f},j}$.

If the informed sets are not independent and do intersect then this technique will oversample states that belong to multiple sets (Fig. 3.7). Uniform sample density can be maintained in these situations by probabilistic rejecting samples in proportion to their membership in individual sets. This results in a uniform sample distribution over multigoal L^2 informed sets defined by arbitrarily overlapping individual informed sets (Fig. 3.8).

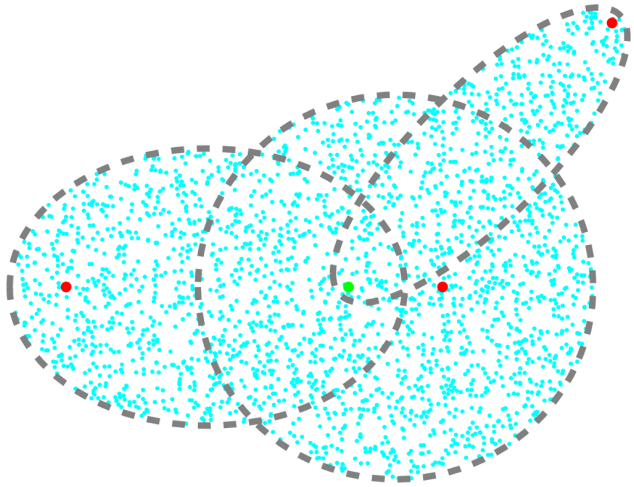


Figure 3.8: An example of using Alg. 3.4 to create 2500 random samples uniformly distributed over a complex multigoal informed set. The multigoal L^2 informed set is defined by one start at $[0,\ 0]^T$, three goals at $[0.25,\ 0]^T$, $[-0.75,\ 0]^T$, and $[0.7,\ 0.7]^T$, and a current solution cost of $c_i = 1.05$.

Algorithm 3.4: `Sample` $(\mathbf{x}_{\text{start}} \in X,\ X_{\text{goal}} \subset X,\ c_i \in \mathbb{R}_{\geq 0})$

```
repeat
    if (1/z) Σ_{j=1}^{z} λ(X_{PHS,j}) < λ(X) then
        x_{goal,j} ← RandomGoal(x_start, X_goal, c_i);
        x_rand ← SamplePHS(x_start, x_{goal,j}, c_i);
    else
        x_rand ← SampleProblem(X);
until x_rand ∈ X_free ∩ (⋃_{j=1}^{z} X_{PHS,j}) and KeepSample(x_rand, x_start, X_goal, c_i);
return x_rand;
```

3.3.2.1 Algorithm

The resulting procedure is described in Algs. 3.4–3.6 as modifications of the method to sample a single-goal L^2 informed set, with changes highlighted in red (cf. Alg. 3.2). This method can be directly extended to problems with a countable start region. The measure of the informed set defined for a specific start-goal pair, $\lambda(X_{\text{PHS},j})$, is calculated from (3.8) using the appropriate goal, $\mathbf{x}_{\text{goal},j}$.

3.4 Discussion

Most anytime almost-surely asymptotically optimal planners use the same procedure to find initial solutions as to improve them. This results in algorithms, such as RRT*, that almost-surely converge asymptotically to the optimum by asymptotically finding the optimal paths to every state in the problem domain. This is inefficient in single-query scenarios and better performance can be attained by focusing the search.

Algorithm 3.5: `RandomGoal` $(\mathbf{x}_{\text{start}} \in X,\ X_{\text{goal}} \subset X,\ c_i \in \mathbb{R}_{\geq 0})$

```
1  a ← 0;
2  forall x_goal,k ∈ X_goal do
3  |  a ← a + λ(X_PHS,k);
4  p ← U[0, 1];
5  j ← 0;
6  repeat
7  |  j ← j + 1;
8  |  p ← p − λ(X_PHS,j)/a;
9  until p ≤ 0;
10 return x_goal,j;
```

Algorithm 3.6: `KeepSample` $(\mathbf{x}_{\text{rand}} \in X,\ \mathbf{x}_{\text{start}} \in X,\ X_{\text{goal}} \subset X,\ c_i \in \mathbb{R}_{\geq 0})$

```
1  a ← 0;
2  forall x_goal,k ∈ X_goal do
3  |  if ‖x_rand − x_start‖_2 + ‖x_goal,k − x_rand‖_2 < c_i then
4  |  |  a ← a + 1;
5  p ← U[0, 1];
6  return p ≤ 1/a;
```

There exists a set of states that could improve any suboptimal solution, the omniscient set (Definition 3.1). Algorithms such as RRT* need to add states from within this set to improve solutions (Lemma 3.1) and can only do so after a finite number of iterations by sampling the set (Lemma 3.2). This bounds the probability of improving a solution in the remainder of the search by the probability of sampling the omniscient set (Theorem 3.3).

The omniscient set is unknowable but can be estimated with informed sets (Definition 3.2), the performance of which are quantified by their precision and recall (Definitions 3.3 and 3.4). A sufficient condition to maintain almost-sure asymptotic optimality while focusing the search is to consider *admissible* informed sets that completely contain the omniscient set (i.e., have 100% recall; Definition 3.5).

Admissible informed sets provide the same requirements as omniscient sets for algorithms to improve their solutions (Lemmas 3.4 and 3.5). The probability of sampling an admissible informed set bounds the probability of almost-sure asymptotically optimal algorithms, such as RRT*, improving their solution at any iteration (Theorem 3.6). This motivates a need for high-precision admissible informed sets for common planning problems.

A universally admissible informed set for problems seeking to minimize path length is given by the L^2 norm (i.e., Euclidean distance). The shape of this L^2 informed set, a prolate hyperspheroid, provides insight into the properties of the minimum-path-length problem (Theorem 3.7). The probability of improving a solution by sampling any rectangular superset of the omniscient set (e.g., the problem domain) decreases *factorially* (i.e., faster than exponentially) as state dimension increases.

This curse of dimensionality can be avoided by directly sampling the L^2 informed set. A method to do so by directly sampling a prolate hyperspheroid is presented in Algs. 3.2 and 3.3. It is shown to perform better than the best case of rectangular rejection sampling by orders of magnitude in high state dimensions. This technique is extended to multigoal problems in Algs. 3.4–3.6.

It is satisfying to note the importance of informed sets being recognized in the literature. Kunz et al. (2016) use the ideas to develop a hierarchical rejection sampling method for systems with differential constraints. They can efficiently generate samples from complex informed sets that cannot be expressed in closed form and for which no direct sampling method exists.

Omniscient and informed sets provide insight into improving the performance of anytime almost-surely asymptotically optimal planners. Direct informed sampling can be used to improve existing algorithms (Informed RRT*; Chapter 4). Informed sets can also provide guidance for new anytime almost-surely asymptotically optimal sampling-based planners that unify informed graph-based search and sampling-based planning (BIT*; Chapter 5).

In summary, this chapter presents the following specific contributions:

- Formally defines omniscient and informed sets and the concepts of precision and recall (Definitions 3.1–3.4).
- Provides upper bounds on the probability of anytime sampling-based planners, such as RRT*, improving a solution at any iteration (Theorems 3.3 and 3.6).
- Shows that common formulations of the minimum-path-length problem have a probability of improving solutions that decreases factorially with state dimension (Theorem 3.7).
- Presents a method to address this curse of dimensionality for problems seeking to minimize path length to a goal or a countable set of goals by directly sampling the L^2 informed set (Algs. 3.2–3.6).

Chapter 4

Informed RRT*

or: The benefits of ignoring bad solutions

This chapter illustrates how informed sets can be used to improve anytime almost-surely asymptotically optimal planners. It uses the ideas presented in Chapter 3 to develop a method that focuses RRT* search and avoids states known to be unable to improve the current solution. This increases the convergence rate of RRT* compared to existing work and reduces the effects of the minimum-path-length curse of dimensionality (Theorem 3.7).

Section 4.1 reviews the existing work to accelerate RRT* and shows that it can generally be described as some combination of sample biasing, sample rejection, and graph pruning. These techniques are effective on some problems but reduce theoretical optimality, fail to address the limitations of high state dimensions, and/or do not effectively accelerate almost-sure asymptotic convergence on all problems.

Section 4.2 demonstrates how these limitations can be addressed through direct informed sampling in problems seeking to minimize path length. Informed RRT* is an anytime almost-surely asymptotically optimal sampling-based planner designed for single-query planning scenarios (Fig. 4.1). It uses graph pruning and direct informed sampling to search only the shrinking subproblem that could contain better solutions (i.e., the L^2 informed set). This process can be viewed as a continuous, almost-surely asymptotically optimal version of Anytime RRTs that does not require restarting the search (Fig. 2.2).

Unlike sample biasing, Informed RRT* considers all homotopy classes that could provide a better solution (i.e., 100% recall) while maintaining uniform sample distribution over a planning subproblem. Unlike sample rejection or graph pruning, it is effective regardless of the relative size of the informed set or the state dimension (i.e., high precision). It performs identically to RRT* when the heuristic does not provide substantial information (e.g., small planning problems and/or large informed sets). A version of Informed RRT* is publicly released in OMPL.

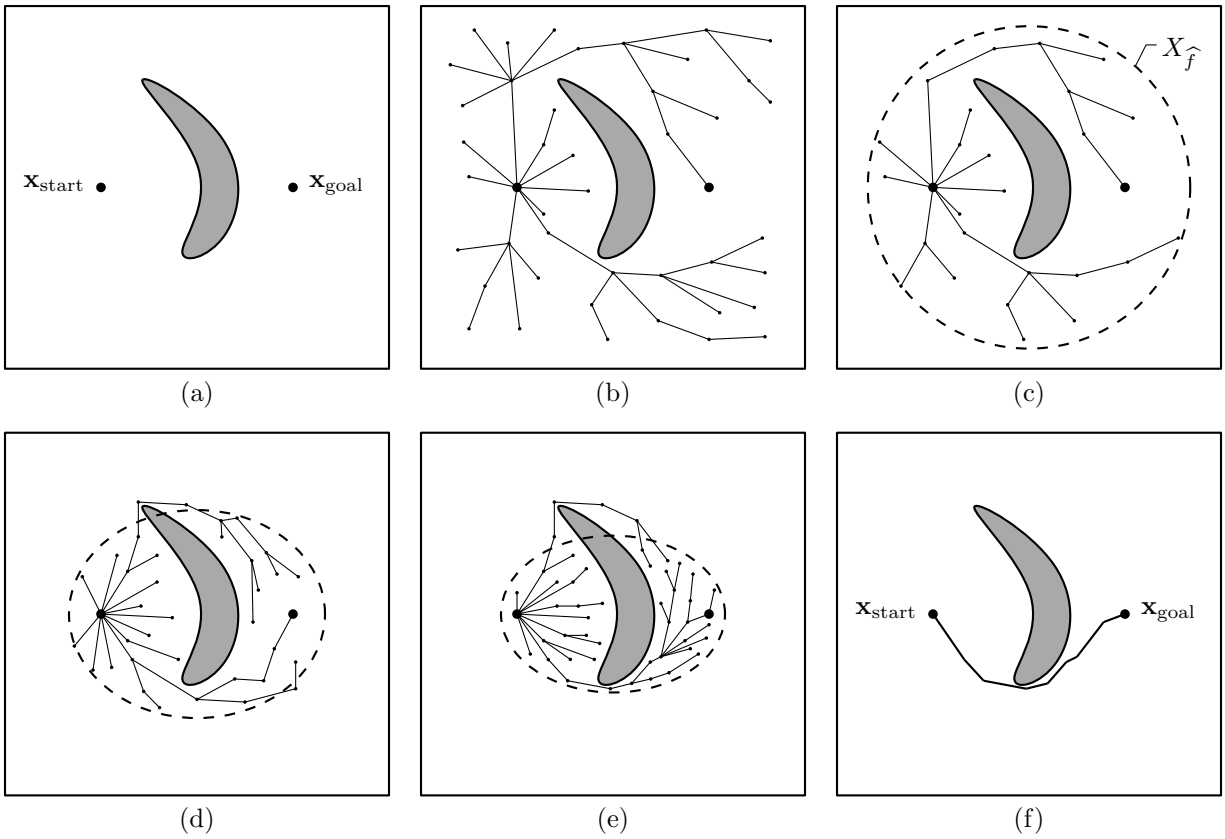
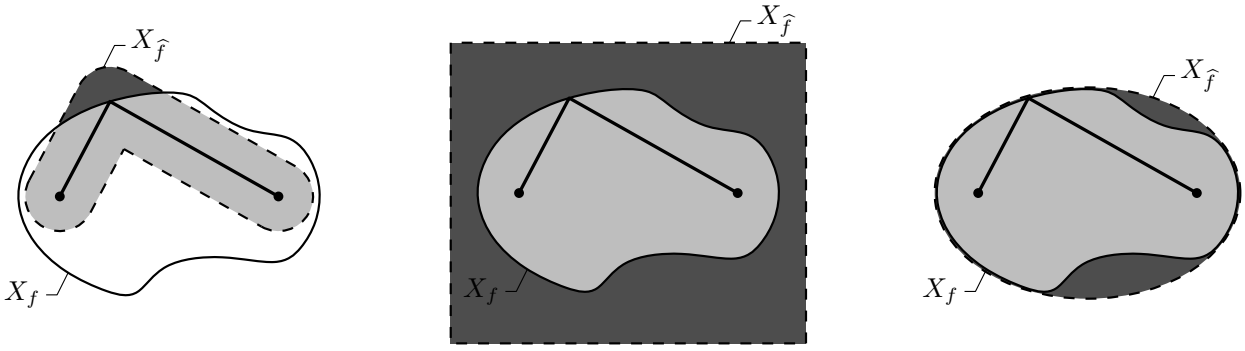


Figure 4.1: An illustration of the Informed RRT* algorithm. The continuous problem, (a), is approximated with a sample-based tree rooted at the start and initially built using RRT*, (b). Once a solution is found, Informed RRT* prunes and focuses the search to the informed set that contains all possibly better solutions, (c). This set shrinks and focuses the search as better solutions are found, (d) and (e), and accelerates the almost-sure asymptotic convergence towards the optimal solution, (f).

Section 4.3 analyzes the theoretical performance of RRT* by deriving sharp bounds on the expected solution costs at any iteration. It shows that planners focused to the informed set (e.g., Informed RRT*) can achieve linear convergence to the optimum but that the original RRT* has provably sublinear convergence (i.e., it converges slower than linear). It also shows that Informed RRT* is the most effective focusing technique and that rectangular rejection sampling factorially approaches sublinear convergence as state dimension increases.

Section 4.4 demonstrates the benefits of Informed RRT* experimentally on both abstract problems and on the CMU Personal Robotic Lab's HERB, a 14-DOF mobile manipulation platform. The abstract problems show that Informed RRT* increasingly outperforms existing techniques as state dimension increases, especially in problems with large planning domains. The experiments on HERB demonstrate that Informed RRT* outperforms other RRT* algorithms on some high-dimensional planning problems with tight search limits (i.e., $\mathbb{R}^{14}$).

Section 4.5 reviews this work and discusses the implications of using heuristics to focus the search for improvements and not the entire search.



(a) (b) (c)

Figure 4.2: A illustration of the precision and recall of various informed sampling techniques. The informed set is rendered in grey with shades depending on when it correctly (light grey) or incorrectly (dark grey) estimates the omniscient set. The sections of the omniscient set that are not estimated by the informed set are shown in white. Path-biasing, (a), generally has high precision but low recall, especially in the presence of multiple homotopy classes. Global or bounded sampling, (b), generally has full recall but low precision, especially in high state dimensions or when improved solutions shrink the omniscient set. Direct sampling of the L^2 informed set, (c), has full recall and high precision, regardless of the size of the omniscient set. It is exactly equal to the omniscient set in the absence of obstacles (i.e., it is *sharp*).

In summary, this chapter makes the following novel contributions:

- Reviews existing techniques to accelerate RRT* and shows how they reduce theoretical optimality, fail to address the problems of high state dimension, and/or do not effectively accelerate convergence on all problems (Section 4.1).
- Develops Informed RRT* as an example of using informed sets to accelerate almost-surely asymptotically optimal planning through direct informed sampling and admissible graph pruning (Algs. 4.1 and 4.2).
- Derives sharp bounds for the expected next-iteration cost of RRT* on problems seeking to minimize path length (Lemma 4.1).
- Proves that RRT* has sublinear convergence on these problems but that focused variants (e.g., Informed RRT*) can have linear convergence (Theorems 4.2–4.4).

4.1 Prior Work Accelerating RRT* Convergence

The majority of existing work to improve RRT*'s rate of convergence attempts to increase the real-time rate of searching the omniscient set. Sections 4.1.1–4.1.4 use results of Chapter 3 to discuss the advantages and disadvantages of these techniques viewed as combinations of sample biasing, sample rejection, and graph pruning (Fig. 4.2).

4.1.1 Sample Biasing

Sample biasing attempts to increase the rate at which RRT* improves a solution by biasing sampling to an informed set. A common informed set for these techniques is the current solution (i.e., path biasing). This increases the likelihood of sampling a state that can improve the current solution but reduces the likelihood of finding a solution in other homotopy classes (i.e., it increases precision by decreasing recall; Fig 4.2a). The ratio of path biasing to global search is frequently a user-chosen parameter that must be tuned for each problem.

RRT* uses local rewiring neighbourhoods to almost-surely converge asymptotically to the optimum. These neighbourhoods are defined assuming a uniform sample distribution and sampling biasing invalidates this assumption and the resulting formal guarantees. One method to maintain performance guarantees is to use both uniform and nonuniform sample distributions and calculate the neighbourhoods from only the uniformly distributed samples (Janson et al., 2015b). This maintains almost-sure asymptotic optimality but may result in a prohibitively high number of rewirings for heavily biased sample distributions.

Akgun and Stilman (2011) use path biasing in their dual-tree version of RRT*. Once an initial solution is found the algorithm spends a user-specified percentage of its iterations refining the current solution. It does this by explicitly sampling near a randomly selected state along the current solution. This increases the probability of improving the current path at the expense of exploring other homotopy classes. Their algorithm also employs sample rejection in exploring the state space (Section 4.1.2).

Nasir et al. (2013) combine path biasing with smoothing in their RRT*-Smart algorithm. RRT*-Smart smooths solution paths and reduces its number of states and then uses these states as biases for further sampling around the solution. Path smoothing improves the current solution but path biasing decreases the likelihood of finding a solution in a different homotopy class.

Kiesel et al. (2012) use a two-stage sampling process in their *f-biasing* technique. Samples are generated by randomly selecting a region of the planning problem and then sampling a uniform distribution inside it. The regions and their probabilities are calculated by solving a discretization of the planning problem using Dijkstra's algorithm. The regions belonging to the discrete solution are given a higher selection probability but a nonzero probability must be maintained over all regions to compensate for the incompleteness of the discretization. This technique provides a sampling bias for the entire RRT* search but once a solution is found it continues to sample states that cannot provide a better solution. They state that almost-sure asymptotic optimality is maintained but do not discuss how to modify the rewiring neighbourhood to do so.

Kim et al. (2014) also use a two-stage sampling process in their Cloud RRT* algorithm. They use a generalized Voronoi graph (Choset and Burdick, 1995) to define a series of collision-free, possibly overlapping, spheres in the planning domain from which they generate

uniform samples. New spheres are added along solutions as they are found and the probability of selecting each sphere is updated so that samples in the homotopy class of a solution are biased around the current path without decreasing the probability of sampling other homotopy classes. Cloud RRT* successfully finds better solutions faster than other algorithms but continues to sample states that cannot improve the solution. They also do not discuss the effect of their sampling scheme on almost-sure asymptotic optimality.

Unlike sample biasing methods, Informed RRT* does not sample states that are known to be unable to improve a solution. It does have a nonuniform sample distribution over the problem domain but its uniform distribution over the informed set easily maintains almost-sure asymptotic optimality.

4.1.2 Sample Rejection

Sample rejection attempts to increase the real-time rate at which RRT* improves a solution by ignoring samples that do not belong to an informed set. This decreases the computational cost associated with samples that cannot improve a solution but does not increase the probability of sampling states in the informed set (Fig. 4.2b). Rejection sampling will spend an increasing amount of time on rejected samples as this probability decreases (i.e., as precision decreases and the informed set shrinks relative to the sampling domain). The effectiveness of rejection sampling the L^2 informed set decreases factorially (i.e., faster than exponentially) with state dimension for problems seeking to minimize path length (Theorem 3.7).

Akgun and Stilman (2011) use global rejection sampling in addition to sample biasing in their dual-tree algorithm. Samples are drawn from the entire problem domain and performance will decrease rapidly as the solution improves or when used on large or high-dimensional planning problems.

Otte and Correll (2013) use rejection sampling in their parallelized Coupled Forest of Random Engrafting Search Trees (C-FOREST) algorithm for problems seeking to minimize path length. Samples are generated from a rectangular subset of the planning domain that bounds the L^2 informed set and rejected using the L^2 heuristic. This increases precision and improves performance in large planning problems but its effectiveness still decreases factorially with state dimension (Theorem 3.7).

Arslan and Tsiotras (2013) combine the RRG algorithm (Karaman and Frazzoli, 2011) with incremental graph search techniques in their RRT$^{\#}$ algorithm. This work is extended in Arslan and Tsiotras (2015) with rejection sampling using three proposed criteria. This focuses the search but its performance will decrease rapidly as the solution improves or when used on large or high-dimensional planning problems as the samples are drawn from the entire planning domain. Some of the rejection criteria also use the *current* cost-to-come of vertices, an inadmissible estimate of the optimal cost-to-come that may reject samples that could later improve the solution.

Unlike sample rejection methods, Informed RRT* maintains high precision and 100% recall regardless of the size of the informed set relative to the problem domain. Its focused search shrinks in response to solution improvements and does not decrease in effectiveness on large or high-dimensional planning problems.

4.1.3 Graph Pruning

Graph pruning limits the RRT* tree to an informed set. When applied to an existing graph it helps increase the real-time rate of solution improvement by reducing the computational cost of basic operations (e.g., nearest neighbour searches). When applied to potential new states it increases the real-time rate of solution improvement by rejecting states not in the informed set. After a sufficient number of iterations this is functionally equivalent to rejection sampling (Lemma 3.5) but with the additional computational cost of converting samples into potential states (e.g., the `Steer` function).

Karaman et al. (2011) use graph pruning to implement an online version of RRT* that improves solutions during path execution. They remove vertices from the tree whose current cost-to-come plus a heuristic estimate of cost-to-go is higher than the current solution. The current cost-to-come of a vertex overestimates its optimal cost-to-come (i.e., it is an inadmissible heuristic) and this may erroneously remove vertices that could provide a better solution.

Unlike graph pruning methods, Informed RRT* expends no computational effort on states that are known to be unable to improve the solution. Samples are generated directly from the current informed set of states and existing vertices in the tree are removed with an admissible pruning algorithm if and only if doing so does not negatively affect the search.

4.1.4 Other Techniques

Other important techniques in the literature do not fit neatly into the previous categories. Many of these methods improve solution quality in ways that could be further accelerated through direct informed sampling.

Ferguson and Stentz (2006) recognize that a solution can be used to estimate the states that could provide better solutions (i.e., an informed set). Their Anytime RRTs method incrementally shrinks the planning problem by independently searching a series of informed sets defined by the previous solutions. Restricting the search domain encourages RRT to find a better solution; however, the algorithm cannot guarantee one is found.

Alterovitz et al. (2011) add path refinement to RRT* in their Rapidly exploring Roadmap (RRM) algorithm. Once an initial solution is found, each iteration of RRM either samples a new state or selects an existing state from the current solution and refines it. Path refinement occurs by connecting the selected state to its neighbours and results in a graph instead of a tree. The ratio of refinement to exploration is a user-tuned parameter.

Shan et al. (2014) use RRT to bootstrap RRT*. An initial solution found by RRT is simplified and rewired using their RRT*_S algorithm before the search is continued with RRT*. This can find better solutions faster than RRT* alone but the resulting search is not focused and continues to consider states that cannot provide better solutions.

Salzman and Halperin (2014) relax almost-sure asymptotic optimality to almost-sure asymptotic *near* optimality in their Lower Bound Tree RRT (LBT-RRT) algorithm. They improve search performance by only rewiring connections as necessary to maintain the desired tolerance to the optimum.

Devaurs et al. (2015) use ideas from stochastic optimization to explore complex cost functions efficiently in their Transition-based RRT* (T-RRT*) and Anytime Transition-based RRT (AT-RRT) algorithms. Transition tests accept or reject a potential new state depending on its cost relative to its parent. These tests help reduce the *integral* or *mechanical work* of the path through the cost space; however, for problems seeking to minimize path length would be equivalent to graph pruning.

These algorithms, and those designed for more advanced purposes (e.g., RRT$^{\mathrm{X}}$; Otte and Frazzoli, 2016), can be improved with the techniques illustrated in Informed RRT*. Their search can be focused more efficiently to the informed set by using direct informed sampling and/or admissible pruning.

4.2 Focusing Search with Informed RRT*

Informed RRT* demonstrates how informed sets can be used to improve the performance of incremental almost-surely asymptotically optimal sampling based planners. It uses RRT* to find an initial solution and then focuses the remaining search to the informed set (Fig. 4.3). This avoids states that cannot improve the solution and improves the convergence rate towards the optimum regardless of the relative size of the informed set (e.g., near-minimum solutions or large problem domains) or the problem state dimension.

Informed RRT* focuses the search through direct informed sampling (Alg. 3.4), admissible graph pruning (Section 4.2.1), and an updated calculation of the rewiring neighbourhood (Section 4.2.2). The complete algorithm is presented in Algs. 4.1 and 4.2 as simple modifications to RRT*, with changes highlighted in red (cf. Alg. 3.1). This version is publicly available in OMPL.

At each iteration, Informed RRT* calculates the current best solution (Alg. 4.1, Line 4) from the vertices in the goal region (Alg. 4.1, Lines 2, 9–10). This defines a shrinking L^2 informed set that is used to both focus sampling (Alg. 4.1, Line 5; Alg. 3.4) and prune the graph (Alg. 4.1, Line 27; Alg. 4.2). This process continues for as long as time allows. As is customary, the minimum of an empty set is taken to be infinity and a prolate hyperspheroid defined by an infinite transverse diameter is taken to have infinite measure.

Algorithm 4.1: Informed RRT*$(\mathbf{x}_{\text{start}} \in X_{\text{free}},\ X_{\text{goal}} \subset X)$

$V \leftarrow \{\mathbf{x}_{\text{start}}\};\ E \leftarrow \emptyset;\ \mathcal{T} = (V, E);$
$V_{\text{sol'n}} \leftarrow \emptyset;$
for $i = 1 \dots q$ **do**
  $c_i \leftarrow \min_{\mathbf{v}_{\text{goal}} \in V_{\text{sol'n}}} \{g_{\mathcal{T}}(\mathbf{v}_{\text{goal}})\};$
  $\mathbf{x}_{\text{rand}} \leftarrow \texttt{Sample}(\mathbf{x}_{\text{start}}, X_{\text{goal}}, c_i);$
  $\mathbf{v}_{\text{nearest}} \leftarrow \texttt{Nearest}(V, \mathbf{x}_{\text{rand}});$
  $\mathbf{x}_{\text{new}} \leftarrow \texttt{Steer}(\mathbf{v}_{\text{nearest}}, \mathbf{x}_{\text{rand}});$
  if $\texttt{CollisionFree}(\mathbf{v}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
    if $\mathbf{x}_{\text{new}} \in X_{\text{goal}}$ **then**
      $V_{\text{sol'n}} \xleftarrow{+} \{\mathbf{x}_{\text{new}}\};$
    $V \xleftarrow{+} \{\mathbf{x}_{\text{new}}\};$
    $V_{\text{near}} \leftarrow \texttt{Near}(V, \mathbf{x}_{\text{new}}, r_{\text{rewire}});$
    $\mathbf{v}_{\text{min}} \leftarrow \mathbf{v}_{\text{nearest}};$
    forall $\mathbf{v}_{\text{near}} \in V_{\text{near}}$ **do**
      $c_{\text{new}} \leftarrow g_{\mathcal{T}}(\mathbf{v}_{\text{near}}) + c(\mathbf{v}_{\text{near}}, \mathbf{x}_{\text{new}});$
      if $c_{\text{new}} < g_{\mathcal{T}}(\mathbf{v}_{\text{min}}) + c(\mathbf{v}_{\text{min}}, \mathbf{x}_{\text{new}})$ **then**
        if $\texttt{CollisionFree}(\mathbf{v}_{\text{near}}, \mathbf{x}_{\text{new}})$ **then**
          $\mathbf{v}_{\text{min}} \leftarrow \mathbf{v}_{\text{near}};$
    $E \xleftarrow{+} \{(\mathbf{v}_{\text{min}}, \mathbf{x}_{\text{new}})\};$
    forall $\mathbf{v}_{\text{near}} \in V_{\text{near}}$ **do**
      $c_{\text{near}} \leftarrow g_{\mathcal{T}}(\mathbf{x}_{\text{new}}) + c(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}});$
      if $c_{\text{near}} < g_{\mathcal{T}}(\mathbf{v}_{\text{near}})$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}})$ **then**
          $\mathbf{v}_{\text{parent}} \leftarrow \texttt{Parent}(\mathbf{v}_{\text{near}});$
          $E \xleftarrow{-} \{(\mathbf{v}_{\text{parent}}, \mathbf{v}_{\text{near}})\};$
          $E \xleftarrow{+} \{(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}})\};$
    $\texttt{Prune}(V, E, c_i);$
return $\mathcal{T}$;

Algorithm 4.2: $\texttt{Prune}(V \subseteq X,\ E \subseteq V \times V,\ c_i \in \mathbb{R}_{\geq 0})$

repeat
  $V_{\text{prune}} \leftarrow \left\{\mathbf{v} \in V \;\middle|\; \widehat{f}(\mathbf{v}) > c_i, \text{ and } \forall \mathbf{w} \in V,\ (\mathbf{v}, \mathbf{w}) \notin E\right\};$
  $E \xleftarrow{-} \{(\mathbf{u}, \mathbf{v}) \in E \mid \mathbf{v} \in V_{\text{prune}}\};$
  $V \xleftarrow{-} V_{\text{prune}};$
until $V_{\text{prune}} = \emptyset;$

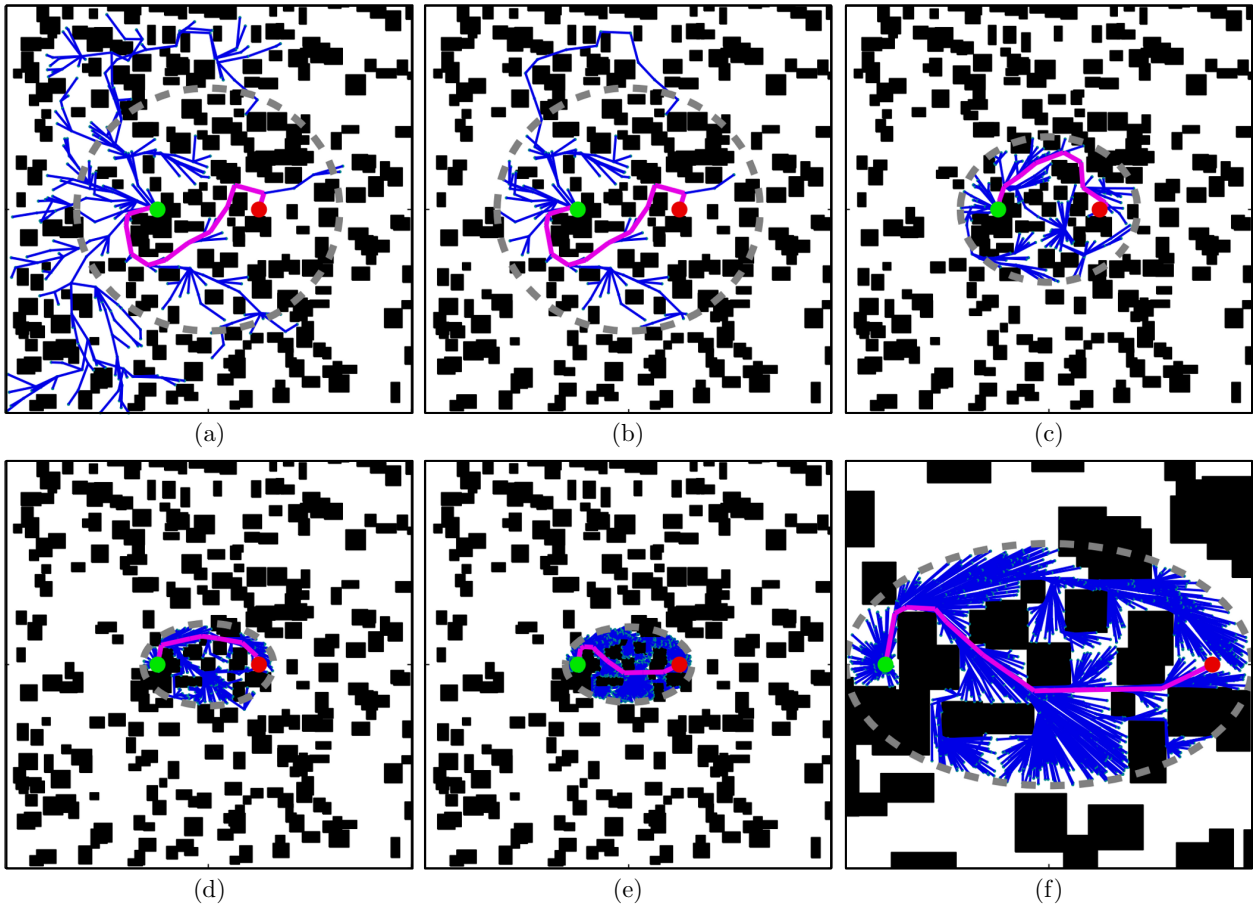
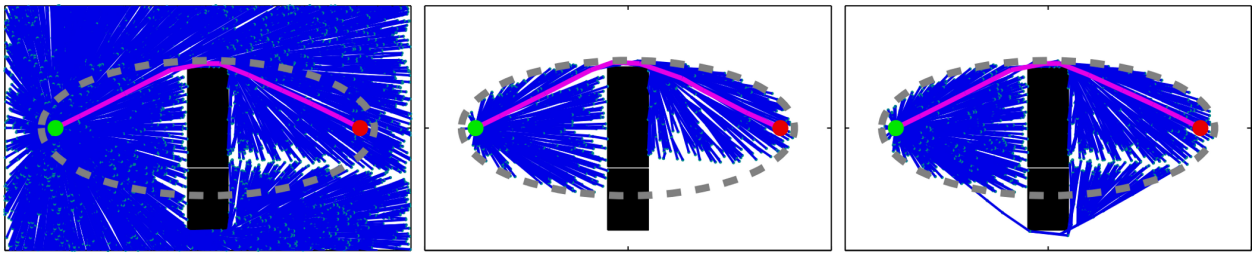


Figure 4.3: An example of how Informed RRT* uses the current solution to focus search to the L^2 informed set. After an unfocused search for an initial solution, (a), Informed RRT* prunes the graph of unnecessary states and redefines the search domain to the current L^2 informed set, (b). Samples are then generated directly from this informed set, avoiding those that are known to be unable to improve the current solution. This reduced search space increases the likelihood of finding an improved solution, which in turn further reduces the search space, (c)–(e). This results in an algorithm that focuses the search to the subproblem given by the current solution, shown enlarged in (f), even as the subproblem continues to shrink as the solution is improved.



(a) (b) (c)

Figure 4.4: A comparison of inadmissible and admissible (Alg. 4.2) pruning algorithms that demonstrates the effect of considering vertex descendants during graph pruning. A graph found by RRT*, (a), is pruned using two different methods. In the first, (b), vertices are removed if they cannot belong to a better solution regardless of their descendants. In the second, (c), vertices are removed if they cannot belong to a better solution *and* neither can their descendants (Alg. 4.2). It is clear that while (c) maintains a uniform sample density in the subset, (b) does not. Furthermore, the pruning strategy illustrated in (b) has greedily removed vertices that will provide a better solution through the narrow passage once the graph is further improved.

4.2.1 Graph Pruning (Alg. 4.2)

Graph pruning is a way to reduce the computational complexity of basic RRT* operations, such as nearest neighbour searches, by removing unnecessary vertices from the tree. A common technique is to remove all vertices with heuristic values larger than the current solution cost (i.e., all vertices that are not members of the informed set). This is a sufficient condition to identify vertices that cannot provide a better solution but an insufficient condition to remove them. These vertices may have descendants that could provide better solutions (i.e., vertices that are members of the informed set) and removing them would negatively affect performance by decreasing vertex density in areas being searched (i.e., the informed set; Fig. 4.4b).

An *admissible* pruning method that does not remove vertices from the informed set is presented in Alg. 4.2. It iteratively removes leaves of the tree that cannot provide a better solution until no such leaves exist. This results in a pruning algorithm that only removes vertices if they *and all their descendants* cannot belong to a better solution (i.e., only removes vertices that are not members of the informed set; Fig. 4.5). This retains all vertices that may later be beneficial regardless of their current connections and does not alter the vertex distribution in areas being searched (i.e., the informed set; Fig. 4.4c).

4.2.2 The Rewiring Neighbourhood

Informed RRT* can be viewed as a method to define and search a shrinking planning problem continuously. This problem contains all possibly better solutions and attaining almost-sure asymptotic optimality in this focused search domain is sufficient to guarantee almost-sure asymptotic optimality in the original problem. This motivates revisiting the definitions of the local rewiring neighbourhoods used by RRT*.

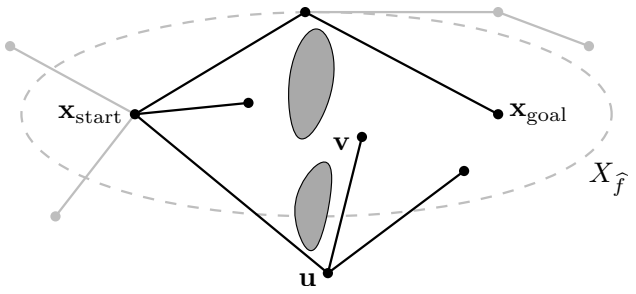


Figure 4.5: An illustration of Alg. 4.2 on a graph containing a solution passing above two obstacles. Elements of the graph retained by the algorithm are coloured in black, pruned elements in grey, and the L^2 informed set is illustrated as a dashed grey line. Vertices are pruned if and only if they cannot improve the current solution (i.e., they are not members of the L^2 informed set) and neither can any of their descendants. This pruning condition avoids removing promising vertices (e.g., $\mathbf{v}$) that are currently descendants of vertices outside the subset (e.g., $\mathbf{u}$) and maintains the vertex distribution of the L^2 informed set.

The bounds presented by Karaman and Frazzoli (2011) depend on graph cardinality and, in the r-disc variant, the measure of the problem domain. In Informed RRT* the informed set is the search domain and these values can be limited to the number of vertices in the informed set, $\left|V \cap X_{\hat{f}}\right|$, and its measure, $\lambda\left(X_{\hat{f}}\right)$. The L^2 informed set is not known in closed form (it is an intersection between the prolate hyperspheroid and free space) but its measure can be bounded from above by the minimum measure of the prolate hyperspheroid and problem domain,

$$\lambda\left(X_{\hat{f}}\right) \leq \min\left\{\lambda\left(X\right), \lambda\left(X_{\text{PHS}}\right)\right\}.$$

This updates (3.4) and (3.5) to

$$r^*_{\text{RRT}^*} = \left(2\left(1+\frac{1}{n}\right)\left(\frac{\min\left\{\lambda\left(X\right), \lambda\left(X_{\text{PHS}}\right)\right\}}{\zeta_n}\right)\left(\frac{\log\left(\left|V \cap X_{\hat{f}}\right|\right)}{\left|V \cap X_{\hat{f}}\right|}\right)\right)^{\frac{1}{n}} \tag{4.1}$$

and

$$k^*_{\text{RRT}^*} = e\left(1+\frac{1}{n}\right)\log\left(\left|V \cap X_{\hat{f}}\right|\right), \tag{4.2}$$

where $\lambda\left(X_{\text{PHS}}\right)$ is given by (3.8) and is a function of the current solution.

These rewiring neighbourhoods can be smaller than the definitions provided by Karaman and Frazzoli (2011) since they may be calculated from fewer vertices (i.e., only those in the informed set) and/or a smaller problem measure (i.e., the measure of the informed set). This reduces the number of rewiring candidates considered for a set of samples, the associated computational cost at each iteration, and improves the real-time performance of Informed RRT* while maintaining almost-sure asymptotic optimality.

4.3 Rates of Convergence

Almost-sure asymptotic optimality provides no insight into the rate at which an algorithm improves a suboptimal solution. Previous work has found probabilistic rates for PRM* (Dobson et al., 2015) and FMT* (Janson et al., 2015b), but only estimated the expected length of an RRT* solution after finite computational time for use as an automated stopping criterion (Dobson et al., 2015).

One method to quantify incremental planning performance is to evaluate the rate at which the sequence of solution costs converges to the optimum, this rate can be classified as sublinear, linear, or superlinear (Definition 4.1).

Definition 4.1 (Rate of convergence)**.** *A sequence of numbers, $(a_i)_{i=1}^{\infty}$, that monotonically and asymptotically approaches a limit, a_∞, has a rate of convergence,*

$$\mu \coloneqq \lim_{i \to \infty} \frac{|a_{i+1} - a_\infty|}{|a_i - a_\infty|}.$$

The sequence is said to converge linearly *if the rate is in the range $0 < \mu < 1$,* superlinearly *(i.e., faster than linear) when $\mu = 0$, and* sublinearly *(i.e., slower than linear) when $\mu = 1$.*

The expected convergence rate of an algorithm depends on its specific tuning and the planning problem. These rates are calculated for RRT* with and without sample rejection and Informed RRT* (Theorems 4.2–4.4) by first calculating sharp bounds on the expected next-iteration cost (Lemma 4.1).

Lemma 4.1 (Expected next-iteration cost in geometric planning)**.** *The expected value of the next solution, $E[c_{i+1}]$, is bounded for geometric planning by*

$$p_f \frac{n c_i^2 + c_{\min}^2}{(n+1)\, c_i} + (1 - p_f)\, c_i \leq E[c_{i+1}] \leq c_i, \tag{4.3}$$

where c_i is the current solution cost, $c_{\min}$ is the theoretical minimum solution cost, n is the state dimension of the planning problem, and $p_f = P(\mathbf{x}_{\text{new}} \in X_f)$ is the probability of adding a state that is a member of the omniscient set (i.e., that can belong to a better solution). While not explicitly shown, the subset, X_f, and the probability of improving the solution, p_f, generally are functions of the current solution cost.

The lower bound is sharp over the set of all possible planning problems and algorithm configurations and occurs for versions of RRT using an infinite rewiring radius (i.e., $\eta = \infty$, and $r_{\text{RRT}^*} = \infty$) in an obstacle-free environment.*

Proof. The proof of Lemma 4.1 is presented in Appendix C. □

This result allows for the calculation of sharp bounds on the convergence rates of RRT* (with and without rejection sampling) and Informed RRT* in any configuration or planning

problem. These bounds will be exact in problems without obstacles and with an infinite rewiring neighbourhood (i.e., $\eta = \infty$, and $r_{\text{RRT}^*} = \infty$) and show that RRT* has sublinear convergence to the optimum (Theorem 4.2).

Theorem 4.2 (Sublinear convergence of RRT* in geometric planning)**.** *RRT* converges* sublinearly *towards the optimum of geometric planning problems,*

$$E\left[\mu_{\text{RRT}^*}\right] = 1. \tag{4.4}$$

Proof. The proof of Theorem 4.2 follows directly from Lemma 4.1 when p_f is given by (3.9), and is presented in Appendix D.1. □

The convergence rate of RRT* can be improved with adaptive rectangular rejection sampling. The biggest improvement will occur when samples are generated from a rectangle that tightly bounds the informed set (Fig. 3.4). This shows that RRT* with adaptive rectangular rejection sampling (e.g., Otte and Correll, 2013) can converge linearly with an infinite rewiring neighbourhood in the absence of obstacles but will quickly (i.e., factorially) approach sublinear convergence with increasing state dimension (Theorem 4.3).

Theorem 4.3 (Linear convergence of RRT* with adaptive rectangular rejection sampling in geometric planning)**.** *RRT* with adaptive rectangular rejection sampling converges at best* linearly *towards the optimum of geometric planning problems but factorially approaches sublinear convergence as state dimension increases,*

$$1 - \frac{\pi^{\frac{n}{2}}}{(n+1)\,2^{n-1}\Gamma\left(\frac{n}{2}+1\right)} \leq E\left[\mu_{\text{Reject}}\right] \leq 1. \tag{4.5}$$

Proof. The proof of Theorem 4.3 follows directly from Lemma 4.1 when p_f is given by (3.9) and (3.12) and is presented in Appendix D.2. □

Direct informed sampling avoids the minimum-path-length curse of dimensionality and can further improve the convergence rate of RRT*. Informed RRT* can converge linearly with an infinite rewiring neighbourhood in the absence of obstacles and will approach sublinear convergence significantly slower than rejection sampling as state dimension increases (Theorem 4.4).

Theorem 4.4 (Linear convergence of Informed RRT* in geometric planning)**.** *Informed RRT* converges at best* linearly *towards the optimum of geometric planning problems,*

$$\frac{n-1}{n+1} \leq E\left[\mu_{\text{Informed}}\right] \leq 1, \tag{4.6}$$

where the lower-bound occurs exactly with an infinite rewiring neighbourhood in the absence of obstacles.

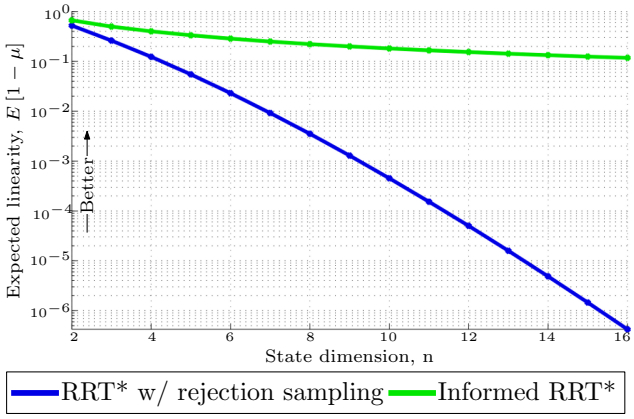


Figure 4.6: An illustration of the lower-bounds on linearity, $E[1-\mu^*]$, of RRT* with rejection samping and Informed RRT* (Corollary 4.5). The convergence rates bounds of RRT* with rejection sampling and Informed RRT* diverge as state dimensions increase and rejection sampling factorially approaches sublinear convergence (Theorems 4.3 and 4.4).

Proof. The proof of Theorem 4.4 follows directly from Lemma 4.1 when $p_f = 1$, and is presented in Appendix D.3 □

Theorems 4.2–4.4 result in the following corollary regarding the relative convergence rates of the algorithms.

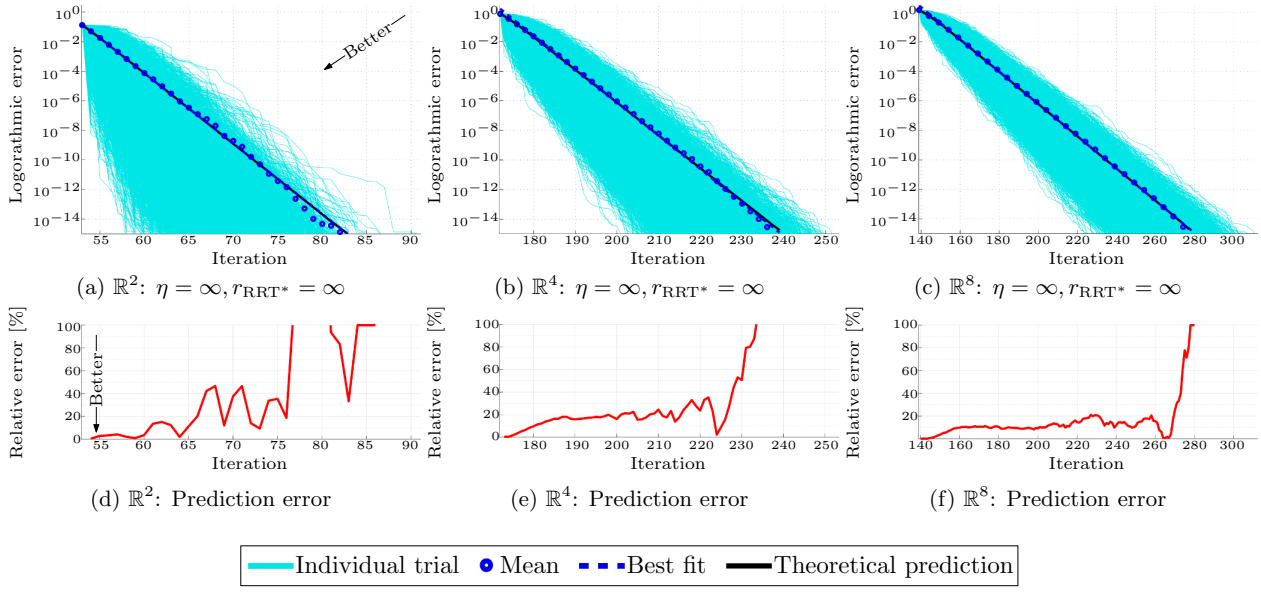
Corollary 4.5 (The faster convergence of Informed RRT* in geometric planning)**.** *The best-case convergence rate of Informed RRT*, μ^*_{Informed}, is always better than that of RRT*, with or without rejection sampling in geometric planning,*

$$\forall n \geq 2,\ \frac{n-1}{n+1} \leq E\left[\mu^*_{\text{Informed}}\right] \leq E\left[\mu^*_{\text{Reject}}\right] \leq E\left[\mu^*_{\text{RRT*}}\right] = 1.$$

Proof. The proof follows immediately from the lower bounds in (4.4), (4.5), and (4.6). It is illustrated in Fig. 4.6. □

4.3.1 Experimental Validation and Extension

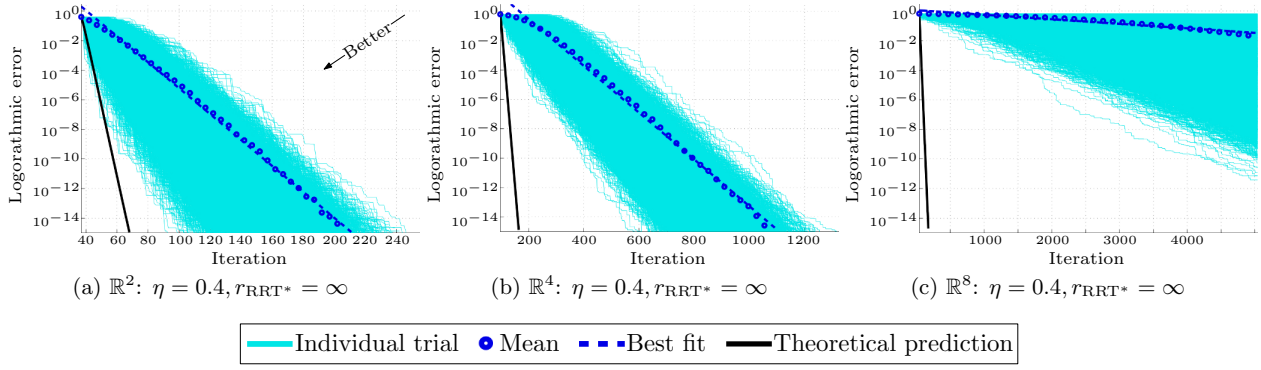
Convergence rates are investigated experimentally for infinite, constant finite, and decreasing finite rewiring radii. Informed RRT* was run in each configuration for 10^4 trials in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ on obstacle-free problems to isolate the effects of the rewiring parameters. Each trial started from the same initial solution but used different pseudo-random seeds to search for improvements. The logarithmic error, $\log(c_i - c_{\min})$, and the resulting mean were calculated at each iteration of each trial and compared to theoretical predictions. The results validate Theorem 4.4 and illustrate the effects of rewiring parameters on the convergence rate.



(a) $\mathbb{R}^2$: $\eta = \infty, r_{\text{RRT}^*} = \infty$ (b) $\mathbb{R}^4$: $\eta = \infty, r_{\text{RRT}^*} = \infty$ (c) $\mathbb{R}^8$: $\eta = \infty, r_{\text{RRT}^*} = \infty$

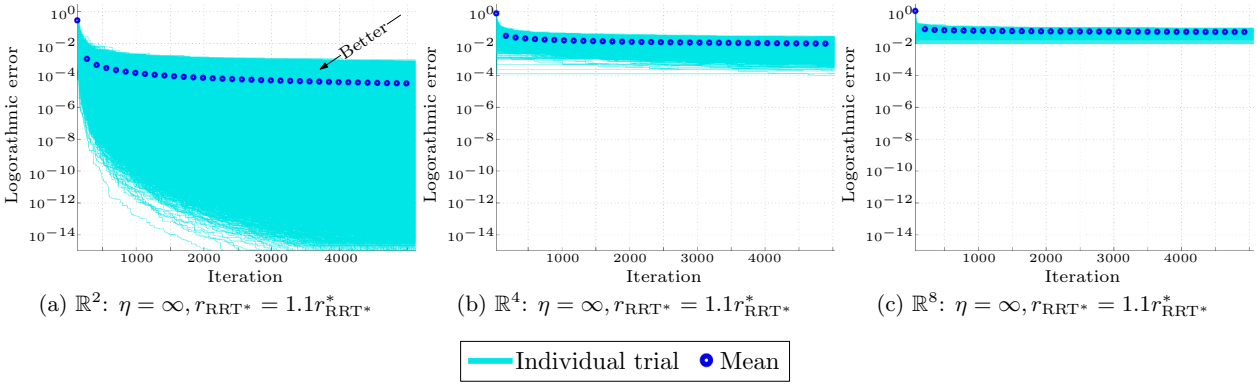
(d) $\mathbb{R}^2$: Prediction error (e) $\mathbb{R}^4$: Prediction error (f) $\mathbb{R}^8$: Prediction error

Figure 4.7: Experimental validation of Lemma 4.1 and Theorem 4.4 showing the linear best-case convergence rate of Informed RRT* in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$. Informed RRT* was run 10^4 times with different pseudo-random seeds from a common initial solution in each dimension. The solution cost found by each planner was recorded at each iteration and used to calculate the logarithmic error (cyan lines), $\log(c_i - c^*)$, and the mean logarithmic error (blue circles) in (a)–(c). The best-case solution cost predicted by Lemma 4.1 (black line) demonstrates the linear convergence predicted by Theorem 4.4 and shows excellent agreement with a line of best fit calculated from the mean error (blue dashed line). The difference between the best-case predicted by Lemma 4.1 and the mean error from the experiments is further highlighted in (d)–(f) by plotting the relative error, $|(c_{\text{mean},i} - c_{\text{theory},i}) / (c_{\text{mean},i} - c^*)|$, with the results showing excellent agreement up to the limitations of numerical precision.



(a) $\mathbb{R}^2$: $\eta = 0.4, r_{\text{RRT}^*} = \infty$ (b) $\mathbb{R}^4$: $\eta = 0.4, r_{\text{RRT}^*} = \infty$ (c) $\mathbb{R}^8$: $\eta = 0.4, r_{\text{RRT}^*} = \infty$

Figure 4.8: An experimental investigation on the effect of a finite but *constant* rewiring neighbourhood on the convergence rate of Informed RRT* in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$. As in the previous experiment, Informed RRT* was run 10^4 times with different pseudo-random seeds from a common initial solution for each dimension. The solution cost found by each planner was recorded at each iteration and used to calculate the logarithmic error (cyan lines), $\log(c_i - c^*)$, and the mean logarithmic error (blue circles). As expected, the resulting mean error is higher than the lower bound of Lemma 4.1 (black line); however, a line of best fit (blue dashed line) suggests that the convergence remains linear. Combined with the results of Fig. 4.9, this motivates further research on the effects of the RRT* rewiring neighbourhood.



(a) $\mathbb{R}^2$: $\eta = \infty, r_{\text{RRT}^*} = 1.1 r^*_{\text{RRT}^*}$ (b) $\mathbb{R}^4$: $\eta = \infty, r_{\text{RRT}^*} = 1.1 r^*_{\text{RRT}^*}$ (c) $\mathbb{R}^8$: $\eta = \infty, r_{\text{RRT}^*} = 1.1 r^*_{\text{RRT}^*}$

Figure 4.9: An experimental investigation on the effect of a finite and *decreasing* rewiring neighbourhood on the convergence rate of Informed RRT* in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$. As in the previous experiments, Informed RRT* was run 10^4 times with different pseudo-random seeds from a common initial solution for each dimension. The solution cost found by each planner was recorded at each iteration and used to calculate the logarithmic error (cyan lines), $\log(c_i - c^*)$, and the mean logarithmic error (blue circles). Unlike experiments with a constant rewiring neighbourhood, the results suggest that the convergence is sublinear. Combined with the results of Fig. 4.8, this motivates further research on the effects of the RRT* rewiring neighbourhood.

The experimental results for an infinite rewiring neighbourhood (i.e., $\eta = \infty$ and $r_{\text{RRT}^*} = \infty$) show excellent agreement with the theoretical predictions in Theorem 4.4 (Fig. 4.7). The mean solution cost converges linearly towards the optimum and closely matches the lower-bound predicted by Lemma 4.1 (Fig. 4.7d–f).

The experimental results for a *constant finite* rewiring neighbourhood (e.g., $\eta = 0.4$ and $r_{\text{RRT}^*} = \infty$) show that the convergence rate is slower than predicted by Theorem 4.4 (Fig. 4.8). The rate appears to start nonlinearly but quickly stabilize to linear convergence. It is hypothesized that this change in rate is related to the distribution of samples relative to the maximum edge length.

The experimental results for a *decreasing finite* rewiring neighbourhood (e.g., $\eta = \infty$ and $r_{\text{RRT}^*} = 1.1 r^*_{\text{RRT}^*}$) show that the convergence rate appears to be sublinear (Fig. 4.9). It is hypothesized that this is a result of the rewiring neighbourhood shrinking 'too' fast for the rate at which sample density increases.

These experiments suggest that further research is necessary to understand the tradeoff between per-iteration cost and the number of iterations needed to find a solution. A shrinking rewiring neighbourhood limits the number of rewirings but its apparent sublinear convergence requires significantly more iterations to find high-quality solutions. Alternatively, the apparent linear convergence of a constant rewiring radius may find equivalent solutions in significantly fewer iterations but the number of rewirings required at each iteration increases rapidly.

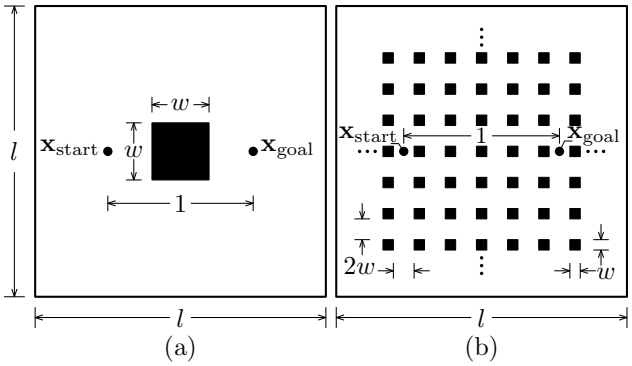


Figure 4.10: Illustrations of the planning problems used for Sections 4.4.1, 4.4.2, and 5.5.1.1. The toy problem shown in (a) is used to both investigate the ability to find solutions within a specified tolerance of the known optimum and the effect of map width, l, on the ability to find high quality solutions. The width of the obstacle is a random variable uniformly distributed over the range $[0.25, 0.5]$. The problem in (b) is used to investigate the algorithm performance in more complex problems with many possible homotopy classes. The use of regularly spaced obstacles allows the problem to scale efficiently to high dimensions. The width of the obstacle is chosen such that the start and goal states are 5 'columns' apart.

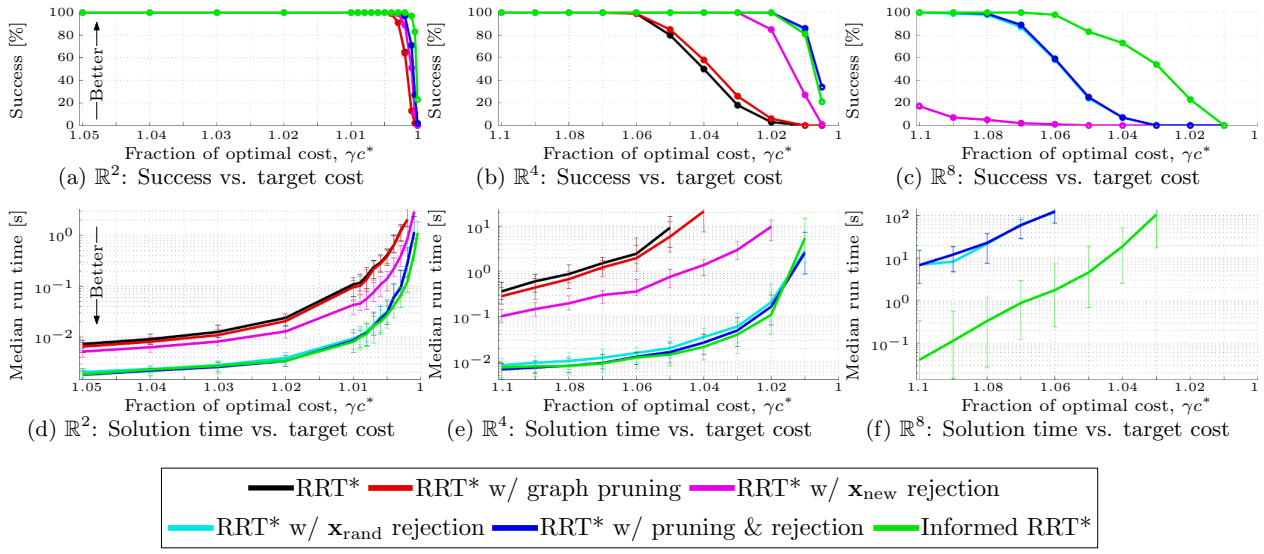
4.4 Experiments

The benefits of focusing the search for improvements in RRT* are demonstrated on simulated problems in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ (Sections 4.4.1 and 4.4.2) and for HERB (Section 4.4.3) using OMPL[1]. Informed RRT* is compared to the original RRT* and versions that focus the search with graph pruning (e.g., Alg. 4.2), heuristic rejection on $\mathbf{x}_{\text{new}}$, heuristic rejection on $\mathbf{x}_{\text{rand}}$, and all three techniques combined.

All planners used the same tuning parameters and the ordered rewiring technique presented in Perez et al. (2011). Planners used a goal-sampling bias of 5% and an RRT* radius of $r_{\text{RRT}^*} = 2r^*_{\text{RRT}^*}$. The maximum edge length was selected experimentally to reduce the time required to find an initial solution on a training problem, with values of $\eta = 0.3$, 0.5, 0.9, and 1.3 used in $\mathbb{R}^2$, $\mathbb{R}^4$, $\mathbb{R}^8$, and on HERB ($\mathbb{R}^{14}$), respectively. Available planning time was limited for each state dimension to 3 seconds, 30 seconds, and 150 seconds, and 600 seconds, respectively. Planners with heuristics used the L^2 norm as estimates of cost-to-come and cost-to-go while those with graph pruning delayed its application until solution cost changed by more than 5%.

These experiments were designed to investigate admissible methods of focusing search. More advanced extensions of RRT* were not considered as they commonly include some combination of the investigated techniques and could instead use direct informed sampling.

[1]The experiments were run on a laptop with 16 GB of RAM and an Intel i7-4810MQ processor. The abstract experiments were run in Ubuntu 12.04 (64-bit) with Boost 1.58, while the HERB experiments were run in Ubuntu 14.04 (64-bit).



(a) $\mathbb{R}^2$: Success vs. target cost (b) $\mathbb{R}^4$: Success vs. target cost (c) $\mathbb{R}^8$: Success vs. target cost

(d) $\mathbb{R}^2$: Solution time vs. target cost (e) $\mathbb{R}^4$: Solution time vs. target cost (f) $\mathbb{R}^8$: Solution time vs. target cost

Figure 4.11: The time required to find near-optimal solutions for the problem illustrated in Fig. 4.10a with $l = 2$. The results show that in high dimensions direct informed sampling finds better solutions faster than other methods. Each planner was run 100 different times in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ for up to 3, 30, and 150 seconds, respectively. The time required to find solutions within various tolerances of the optimum, c^*, were recorded for each trial. The percentage of trials that found a solution within the specified tolerances is presented in (a)–(c). The median time to find a solution within the specified tolerances is presented in (d)–(f), with unsuccessful trials assigned infinite time. The error bars denote a nonparamentric 99% confidence interval on the median value. Informed RRT* performs similarly to rejection sampling methods of RRT* in low dimensions ($\mathbb{R}^2$ and $\mathbb{R}^4$); however, outperforms all other methods as state dimension increases as predicted by the theoretical results ($\mathbb{R}^8$).

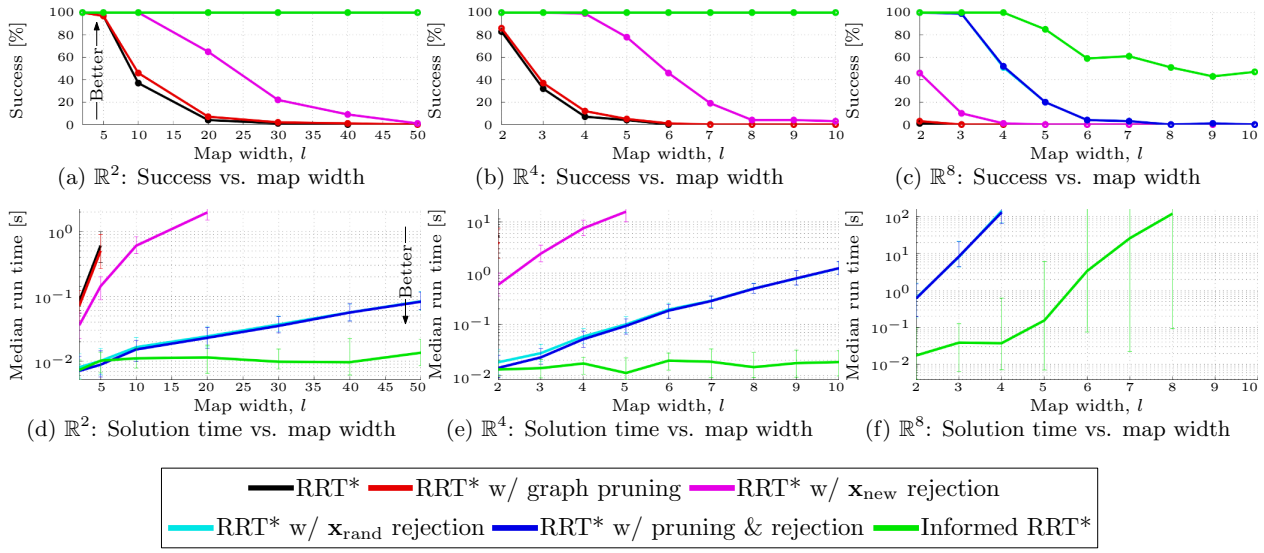
4.4.1 Toy Problems

Two separate experiments were run in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ on randomized variants of the toy problem depicted in Fig. 4.10a to investigate the effects of obstacles on (3.10),

$$\forall i \geq \kappa,\ P\left(c_{i+1} < c_i \ \middle|\ \mathbf{x}_{\text{rand}} \sim \mathcal{U}\left(X_{\text{sampling}}\right)\right) \leq \frac{\pi^{\frac{n}{2}} c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right) \lambda\left(X_{\text{sampling}}\right)}. \qquad \text{(3.10 redux)}$$

The problem consists of a (hyper)cube of width l with a single start and goal located at $[-0.5, 0, \ldots, 0]^T$ and $[0.5, 0, \ldots, 0]^T$, respectively. A single (hyper)cube obstacle of width $w \sim \mathcal{U}\left[0.25, 0.5\right]$ sits between the start and goal in the centre of the problem domain.

These experiments require assigning values to the problem parameters not being investigated. The choice of these values affect the tests but the combined results of the two experiments show that increasing problem size and state dimension decreases the ability of nondirect sampling methods to find near-optimal solutions, as predicted by (3.10). Using direct informed sampling to focus its search to the informed allows Informed RRT* to limit these effects and outperforms the other techniques on these problems.



(a) $\mathbb{R}^2$: Success vs. map width (b) $\mathbb{R}^4$: Success vs. map width (c) $\mathbb{R}^8$: Success vs. map width

(d) $\mathbb{R}^2$: Solution time vs. map width (e) $\mathbb{R}^4$: Solution time vs. map width (f) $\mathbb{R}^8$: Solution time vs. map width

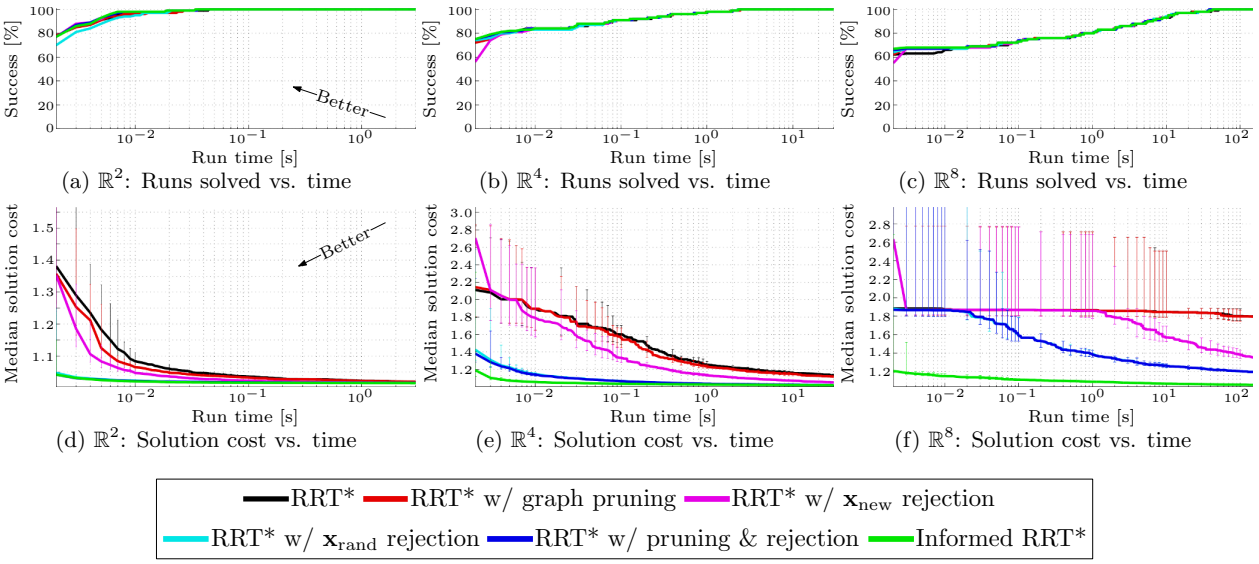
Figure 4.12: The time required to find a solution within a specified fraction of the known optimum for the problem illustrated in Fig. 4.10a for various map widths. Each planner was run 100 different times on maps of various widths in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ to find solutions better than $1.01c^*$, $1.05c^*$, and $1.15c^*$, respectively. The time required to find the target solution was recorded for each trial, with times limited to 3, 30, and 150 seconds, respectively. The percentage of trials that found the target solution is presented in (a)–(c). The median time to find the solution is presented in (d)–(f), with unsuccessful trials assigned infinite time. The error bars denote a nonparamentric 99% confidence interval on the median value. Informed RRT* performs better than the other algorithms on larger problems, with the difference becoming particularly evident in high state dimensions ($\mathbb{R}^8$).

4.4.1.1 Target Solution Quality

The first experiment investigates finding near-optimal solutions in the presence of obstacles. The time required for each planner to find a solution within various fractions of the known optimum, c^*, was recorded over 100 trials with different pseudo-random seeds for maps of width $l = 2$. The percentage of trials that found a solution within the target tolerance of the optimum and the median time necessary to do so are presented for each planner in Fig. 4.11. Trials that did not find a suitable solution were treated as having infinite time for the purpose of calculating the median. The results show that Informed RRT* performs equivalently to sample-rejection-based algorithms in low state dimensions but outperforms all techniques in high dimensions.

4.4.1.2 Map Width

The second experiment investigates finding near-optimal solutions in large planning problems. The time required for each planner to find a near-optimal solution was recorded over 100 trials with different pseudo-random seeds for maps of increasing width, l. Planners sought a solution better than $1.01c^*$, $1.05c^*$, and $1.15c^*$ in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$, respectively. The percentage of trials that found a sufficiently near-optimal solution and the median time necessary to do



(a) $\mathbb{R}^2$: Runs solved vs. time (b) $\mathbb{R}^4$: Runs solved vs. time (c) $\mathbb{R}^8$: Runs solved vs. time

(d) $\mathbb{R}^2$: Solution cost vs. time (e) $\mathbb{R}^4$: Solution cost vs. time (f) $\mathbb{R}^8$: Solution cost vs. time

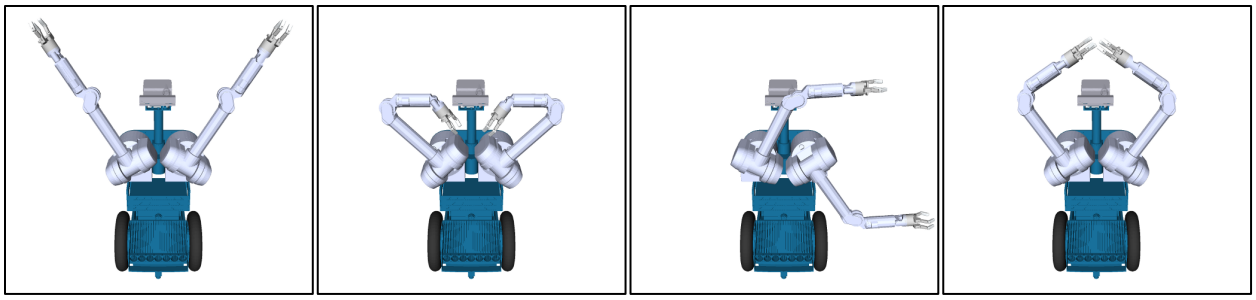
Figure 4.13: Planner performance versus time for the problem illustrated in Fig. 4.10b. Each planner was run 100 different times in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$ with $l = 4$ and run times limited to 3, 30, and 150 seconds, respectively. The percentage of trials solved is plotted versus run time for each planner and presented in (a)–(c). The median path length is plotted versus run time for each planner and presented in (d)–(f), with unsuccessful trials assigned infinite length. The error bars denote a nonparamentric 99% confidence interval on the median value. Informed RRT* has similar performance to rejection sampling methods in $\mathbb{R}^2$, but finds better solutions faster in $\mathbb{R}^4$ and $\mathbb{R}^8$. This demonstrates the benefits of direct informed sampling even in the presence of high numbers of obstacles and homotopy classes.

so are presented for each planner in Fig. 4.12. Trials that did not find a suitable solution were treated as having infinite time for the purpose of calculating the median. The results show that Informed RRT* outperforms all techniques in large-domain planning problems and that the difference increases in high state dimensions.

4.4.2 Worlds with Many Homotopy Classes

The algorithms were tested on more complicated problems with many homotopy classes in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$. The worlds consisted of a (hyper)cube of width $l = 4$ with the start and goal located at $[-0.5, 0, \ldots, 0]^T$ and $[0.5, 0, \ldots, 0]^T$, respectively. The problem domain was filled with a regular pattern of axis-aligned (hyper)rectangular obstacles with a width such that the start and goal were 5 'columns' apart (Fig. 4.10b).

The planners were tested with 100 different pseudo-random seeds on each world and state dimension. The solution cost of each planner was recorded every 1 millisecond by a separate thread and the median was calculated from the 100 trials by interpolating each trial at a period of 1 millisecond. The absence of a solution was considered an infinite cost for the purpose of calculating the median.



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

Figure 4.14: A motion planning problem for HERB inspired by Morali and Willis (1978). Planners must find a collision-free path between each pair of subsequent poses, e.g., (a) to (b). HERB's 14 DOFs and high number of potential self-collisions make this a nontrivial planning problem for RRT*. The planners were given 600 seconds for each phase of the planning problem and the results are presented in Fig 4.15.

The results are presented in Fig. 4.13, where the percent of trials solved and the median solution cost are plotted versus run time. They demonstrate how Informed RRT* uses direct sampling of the L^2 informed set to improve the real-time convergence of the solution towards the optimum compared to the other techniques.

4.4.3 Motion Planning for HERB

Informed RRT* was demonstrated on a high-dimensional problem using HERB, a 14-DOF mobile manipulation platform (Srinivasa et al., 2012). Poses were defined for the two arms ($\mathbb{R}^{14}$) to create a sequence of three planning problems (Fig. 4.14) inspired by Morali and Willis (1978). RRT*, RRT* with pruning and rejection, and Informed RRT* were each run for 50 trials on each problem of the cycle. The resulting median path lengths are presented in Fig. 4.15. Trials that did not find a solution were considered to have infinite length for the purpose of calculating the median. This only occurred for the problem from (a) to (b), where the planners found a solution on 94% of the trials.

Manipulation planning in joint space results in problem domains with strict limits and the joints on HERB all have between π and and 2π radians of travel. This limited problem domain can place large sections of the sampled prolate hypershperoid outside the search domain. This limits the effectiveness of direct informed sampling but, as reflected in the results, does not prevent Informed RRT* from outperforming existing algorithms on some problems. RRT* with and without pruning and rejection sampling both fail to improve the initial solutions on all three planning problems but Informed RRT* is able to improve the path length in joint space by 3.9%, 7.9%, and 28.2%, respectively. The improvement in length from (a) to (b) is not statistically significant but the improvements from (b) to (c) and (c) to (d) demonstrate the benefits of accounting for the relative sizes of the informed set and problem domain in high state dimensions.

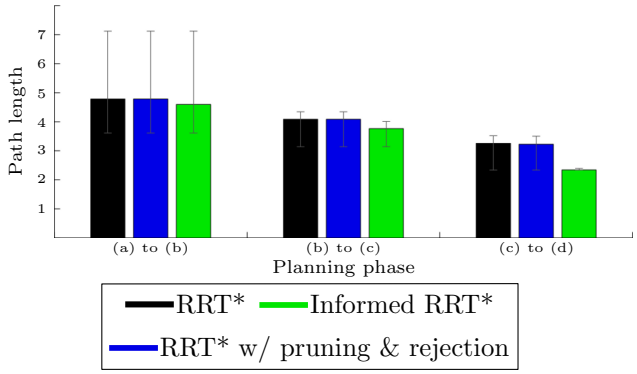


Figure 4.15: Results from 50 trials on the motion planning problems depicted in Fig. 4.14. The median path length between each pose found by each planner was calculated after 600 seconds of planning time. Planners found a solution between each pose in every trial other than the transition from (a) to (b), where solutions were only found in 94% of the trials. These unsolved trials were assigned an infinite cost Ffor the purpose of calculating the median. Error bars denote a nonparamentric 99% confidence interval on the median value. The results show that Informed RRT* can outperform rejection sampling in high-dimensional problems even in the presence of strict state-space limits.

4.5 Discussion

RRT* almost-surely converges asymptotically to the optimal solution to planning problems by asymptotically finding the optimal paths to every state in the problem domain. This is inefficient given its single-query nature as once a solution is found it only needs to consider states that could provide improvements (i.e., an informed set). Existing techniques to focus the search to informed sets are insufficient for problems seeking to minimize path length (Section 4.1). The majority either reduce the ability to find solutions in other homotopy classes (i.e., reduce recall) or fail to account for the reduction of the L^2 informed set in response to solution improvement (i.e., have decreasing precision). Even adaptive techniques that address these problems (e.g.,; Otte and Correll, 2013) fail to account for the factorial decrease in measure with state dimension (i.e., the minimum-path-length curse of dimensionality; Theorem 3.7).

Informed RRT* is presented as an example of how direct informed sampling and admissible graph pruning addresses these limitations (Algs. 4.1 and 4.2; Section 4.2). It focuses the search for improvements to the L^2 informed set regardless of its size relative to the problem domain or the state dimension of the planning problem (i.e., it has 100% recall and high precision). This allows it to ignore states that cannot provide a better solution and improves convergence towards the optimum, especially in planning problems with large domains or high state dimensions.

RRT* performance can be analyzed theoretically for problems seeking to minimize path length by using the shape of the L^2 informed set (Section 4.3). The resulting bounds on the expected solution cost at any iteration (Lemma 4.1) are sharp over the set of all possible

planning problems and algorithm configurations, with the lower bound being exact for an infinite rewiring radius in the absence of obstacles. This allows for an analysis of the expected convergence rates for minimizing path length and shows that RRT* provably converges sublinearly (i.e., slower than linear; Theorem 4.2) when not focused to the informed set. Focusing the search can result in linear convergence (Theorems 4.3 and 4.4) but adaptive rectangular rejection sampling approaches sublinear convergence *factorially* with state dimension.

These analytical results are verified and extended experimentally to different algorithm configurations. The experimental results confirm the theoretical limits and suggest that obstacle-free convergence remains linear with a finite constant maximum edge length, but becomes sublinear with a finite decreasing RRT* radius. Previous analysis of these rewiring terms have focused on their effect on iteration complexity but this result should motivate future research into the trade off between per-iteration cost and convergence rate.

Focusing techniques are also tested experimentally on a variety of planning problems (Section 4.4). These experiments demonstrate how the better theoretical convergence rate of Informed RRT* corresponds to finding better solutions faster that other techniques on real planning problems. This may not occur in small problem domains and/or with long circuitous solutions but the designs of Algs. 3.2 and 3.4 assure that in these problems Informed RRT* performs no worse than other methods to exploit the L^2 heuristic (e.g., rejection sampling).

Designing these experiments demonstrates the importance of the maximum edge length, η, on the experimental performance of the tested algorithms. It not only affected the time required to find an initial solution but also the quality of the solution found in finite time as a result of (3.3). Specifically, large values of η appeared to decrease the difference between algorithms; however, also resulted in order of magnitude increases in the time required to find initial solutions. Given that anytime improvement of a solution is a major feature of RRT*, η was tuned to minimize the initial-solution time on a series of test problems. When coupled with the results of Section 4.4, this result should further motivate more research into the effects of the RRT* tuning parameters, η and r_{RRT^*}, on real-time performance.

This chapter demonstrates how direct informed sampling can be used to focus RRT*'s search for improvements to an estimate of the omniscient set. Searching this omniscient set is the fundamental task of anytime asymptotically optimal planning and the performance of any informed techniques will ultimately depend on the precision and recall of its estimate. The L^2 informed set is a uniformly admissible estimate of the omniscient set for problems seeking to minimize path length and is exact in the absence of obstacles. It represents a *sharp* bound on the omniscient set even in the presence of differential constraints. This suggests that any informed set that is more precise for minimum-path-length problems must either (i) exploit additional information about the problem domain, and/or (ii) be inadmissible for some minimum-path-length planning problems. Investigating how problem-specific information (e.g., obstacles) can be used to define new universally admissible heuristics could potentially allow algorithms such as Informed RRT* to converge linearly in the presence of obstacles.

It is satisfying to note Informed RRT* being improved in the literature. Kim and Song (2015) extend Informed RRT* by applying a path smoother to the existing solution. This helps locally optimize the path and reduce the size of the L^2 informed set, further focusing the remaining search. Lee et al. (2016) modify Informed RRT* to also include path biasing. This helps improve performance in online scenarios where path execution must start before planning is finished. Primatesta et al. (2016) adapt Informed RRT* to allow for path repair in changing environments. This helps avoid excessive replanning in the presence of dynamic obstacles. Burget et al. (2016) extend Informed RRT* to include bidirectional search (see RRT-Connect) in their Bidirectional Informed RRT* (BI^{2}RRT*) algorithm. This helps quickly find an initial solution and focus the search to an informed set.

Informed RRT* is also used to find initial collision-free paths in various multistage planning algorithms. Oleynikova et al. (2016) use it in their real-time trajectory optimization method for unmanned aerial vehicles (UAVs). Bevilacqua et al. (2016) use it in their algorithm to generate plans that account for user comfort in service and assistive robotics.

This chapter also illustrates the limitations of only applying heuristics to focus the search for improvements. Informed RRT*'s search is still unordered and considers states regardless of their potential solution quality. This can result in spending a significant amount of computational effort on areas of the problem domain that are never used in a solution.

In comparison, anytime graph-based search techniques, such as ARA*, use heuristics to both order and focus their search. Doing so avoids unnecessary search effort and reduces the time required to find both the initial solution and subsequent improvements. Chapter 5 applies these ideas to develop BIT*, an informed anytime sampling-based planner that uses heuristics throughout its search and almost-surely converges asymptotically to the optimum in an anytime manner. It can be viewed as a unification of graph-based search and sampling-based planning techniques.

In summary, this chapter presents the following specific contributions:

- Reviews techniques to accelerate RRT* and shows that they either reduce theoretical optimality, fail to address the problems of high state dimension, and/or generally do not effectively improve performance (Section 4.1).
- Develops Informed RRT* as an illustration of how informed sets can accelerate almost-surely asymptotically optimal planning through direct informed sampling and admissible graph pruning (Algs. 4.1 and 4.2).
- Derives sharp bounds for the expected next-iteration cost of RRT* on problems seeking to minimize path length (Lemma 4.1).
- Proves that RRT* has sublinear convergence on these problems but that focused variants (e.g., Informed RRT*) can have linear convergence (Theorems 4.2–4.4).

Chapter 5

Batch Informed Trees (BIT*)

or: The benefits of trying good solutions first

This chapter demonstrates how informed graph-based search can be used to address the limitations of anytime sampling-based planning. Unifying these approaches results in algorithms that build anytime approximations of continuous planning problems and search this representations in order of potential solution quality. This avoids unnecessary computational costs throughout the search and results in finding better solutions faster than existing algorithms, especially in high state dimensions.

Single-query anytime sampling-based planners, such as RRT*, often approximate continuous planning problems incrementally. This allows them to stop once a suitable solution is found but makes their search dependent on the underlying sequence of samples. Such an *unordered* search simultaneously expands towards every state in the problem domain and performs work that is not needed to find the eventual solution. This wasted computational effort can become prohibitively expensive in large problem domains and high state dimensions. The effort can be reduced by focusing the search once a solution is found (e.g., Informed RRT*; Chapter 4) but the search itself will still be inefficient.

Informed graph-based searches, such as A*, avoid unnecessary computational effort by ordering their entire search by potential solution quality. This assures that solutions are found by only considering states that could have provided a better solution and effectively solves a problem by searching the informed set that the solution *will* define (the *future informed set*). These approaches have previously been applied to *a priori* graph-based discretizations.

Section 5.1 presents a review of previous work to incorporate ordered search concepts into sampling-based planning. The review is roughly divided into efforts to add sampling to informed graph-based search (e.g., to provide anytime representations for A*) and to add heuristics to anytime sampling-based planners (e.g., to order the search of RRT*). This work either applies heuristics partially (e.g., RRT#), sacrifices anytime resolution (e.g., RA* and FMT*), or unnecessarily searches outside the future informed set due to metrics other than solution cost (e.g., SBA*).

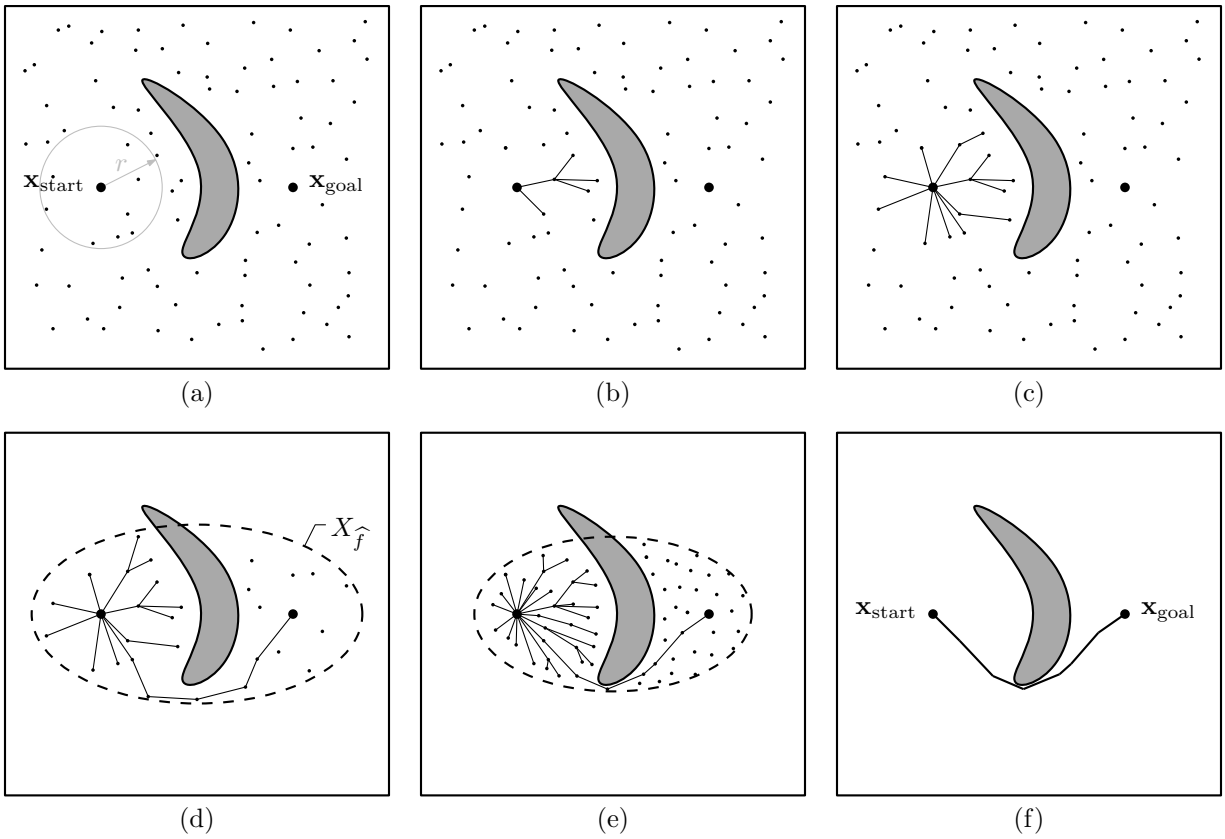


Figure 5.1: An illustration of the BIT* algorithm. The continuous planning problem is approximated with a sample-based tree rooted at the start. The tree is grown outwards using a radius of connection defined by RGG theory, (a). The search proceeds in order of increasing solution cost as estimated by an *admissible* heuristic, (b) and (c), as in A* (see Fig. 2.4) until no edges in can provide a (better) solution, (d). A batch of new samples is then added to the informed set and the search continues efficiently with incremental search techniques, (e). The ordered search assures that BIT* finds solutions without searching outside the future informed set of a batch. With an infinite number of samples, BIT* has a unity probability of both finding a solution and converging asymptotically to the optimum, if one exists, (f).

Section 5.2 presents Batch Informed Trees (BIT*) as an example of a single-query sampling-based planner that is both anytime and ordered only on potential solution quality (Fig. 5.1). It unifies informed graph-based search and sampling-based planning to efficiently search an increasingly dense implicit random geometric graph (RGG) defined by *batches* of samples with incremental search techniques (Fig. 2.2). Processing batches of multiple samples allows it to search the current approximation in order of potential solution quality, as in A*. Processing multiple batches allows it to increase the approximation accuracy of the continuous planning problem until it contains a suitable solution, as in RRT*. A version of BIT* is publicly released in OMPL.

BIT* uses heuristics in all aspects of its search to improve its efficiency. It uses informed sets to focus its approximation of the continuous planning problem to the set of states that could belong to a better solution, as in Informed RRT*. It uses RGG theory to reduce

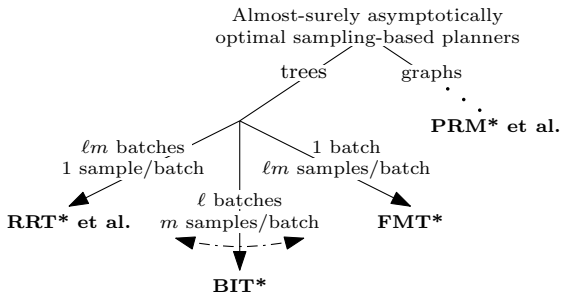


Figure 5.2: A simplified taxonomy of almost-surely asymptotically optimal sampling-based planners that demonstrates the relationship between RRT*, FMT*, and BIT*. When using a batch size of a single sample, BIT* is a version of RRT*. When using a single batch consisting of multiple-samples, BIT* is a version of FMT*.

the number of edges in its approximation while still maintaining almost-sure asymptotic convergence to the optimum, as in RRT* or FMT*. It uses incremental search to order the search of a changing approximation by potential solution quality and reuse information, as in LPA*. It uses an implicit representation of the RGG to avoid the computational cost of explicitly representing every edge, as in Bohlin (2001) and Quasi-random Lazy PRM (Branicky et al., 2001). It uses lazy collision checking to avoid considering unnecessary edges, as in Lazy Weighted A* (Cohen et al., 2014b), lazy versions of PRM (Bohlin and Kavraki, 2000; Branicky et al., 2001; Hauser, 2015), and other techniques (Helmert, 2006; Sánchez and Latombe, 2002).

BIT* only requires three user-defined options, the RGG constant, the heuristic function, and the number of samples per batch and can be be viewed as a generalization of existing sampling-based planners. With no heuristic and multiple batches of one sample, it is a version of RRT*. With no heuristic and a single batch of m samples, it is a version of FMT* (Fig. 5.2).

Section 5.3 presents a theoretical analysis of BIT* that proves it to be probabilistically complete and almost-surely asymptotically optimal. It also shows its search order to be analytically equivalent to a version of LPA* that does not propagate rewirings.

Section 5.4 presents simple extensions of BIT* that may further improve performance in some planning situations. Extensions include delaying the rewiring of connected vertices until an initial solution is found, generating samples when necessary *just in time* (JIT), and removing unconnected samples while maintaining almost-sure asymptotic optimality. Extending the idea of removing unconnected samples also results in Sorted RRT* (SORRT*), a variation of Informed RRT* that is also publicly available in OMPL.

Section 5.5 demonstrates the benefits of BIT* on both abstract problems and experiments for the CMU Personal Robotic Lab's HERB, a 14-DOF mobile manipulation platform. The results show that BIT* outperforms other almost-surely asymptotically optimal planners while maintaining anytime performance, especially in high dimensions. The only tested planner that consistently found solutions faster than BIT* was RRT-Connect, a bidirectional algorithm that does not converge towards the optimum.

Section 5.6 reviews the benefits of unifying informed graph-based search and sampling-based planning and discusses potential future work.

In summary, this chapter makes the following novel contributions:

- Reviews existing methods to combine informed search and sampling-based representations and shows that they either provide incomplete/inefficient ordering or sacrifice anytime performance (Section 5.1).
- Develops BIT* as a unification of informed graph-based search and sampling-based planning that is almost-surely asymptotically optimal (Algs. 5.1–5.3).
- Presents extensions to BIT* (Section 5.4), including JIT sampling to allow for the direct search of unbounded problems (Alg. 5.5) and SORRT* (Alg. 5.6) to apply ordered search concepts directly to RRT*.
- Demonstrates experimentally the benefits of using ordered search to combat the curse of dimensionality (Section 5.5).

5.1 Prior Work Ordering Sampling-based Planners

Prior work to combine graph-based search and sampling-based planning can be loosely classified as either incorporating sampling into informed A*-style searches (Section 5.1.1) or adding heuristics to incremental RRT/RRT*-style searches (Section 5.1.2).

5.1.1 A*-based Approaches

A* is the optimally efficient search for a given graph and any other algorithm using the same heuristic to find the resolution-optimal solution will expand at least as many vertices (Hart et al., 1968). This is achieved by ordering the search by potential solution quality such that the optimal solution is found by only considering states that could provide a better solution (i.e., by only searching the future informed set). Applying A* to continuous planning problems requires discretizing the search domain and significant work has incorporated sampling into this process, often to avoid the need to do so *a priori*.

Sallaberger and D'Eleuterio (1995) demonstrate the advantages of including stochasticity in the representation of a continuous state space by adding random perturbations to a regular discretization. They solve planning problems for spacecraft and multilink robotic arms using dynamic programming and show that their technique finds better solutions than regular discretizations alone; however, the approximation is still defined *a priori* to the search.

Randomized A* (RA*; Diankov and Kuffner Jr., 2007) uses random sampling to apply A* directly to continuous planning problems. Vertices are expanded by randomly sampling a user-specified number of possible descendants in a local neighbourhood. If a sample is sufficiently different than the existing states in the tree and can be reached without collision it is added

as child of the expanded vertex. The resulting sampling-based search expands outwards from the start in order of potential solution quality but is not anytime. If the search does not find a (suitable) solution then it must be restarted with a different number of samples per vertex.

Hybrid Randomized A* (HRA*; Teniente and Andrade-Cetto, 2013) performs a similar search for systems with differential constraints by sampling control inputs instead of states. Vertices are expanded by generating a user-specified number of random control inputs and propagating them forward in time with a motion model. These states are used to expand the tree and, when they are sufficiently similar to an existing state, rewire the tree to improve the cost of existing vertices. The search expands vertices with a hybrid cost policy that considers path cost, the number of nearby vertices, and the distance to obstacles. This biases sampling into unexplored regions of the problem domain but may extend the search outside the future informed set and waste computational effort on states that are unnecessary to find the eventual solution.

Sampling-based A* (SBA*; Persson and Sharf, 2014) performs an ordered search by *iteratively* expanding vertices in a heuristically weighted version of EST (Hsu et al., 1997, 1999a). At each iteration a *single* random sample is generated in the local neighbourhood of the vertex at the front of a priority queue. This queue is ordered on the potential solution quality of vertices and the likelihood of sampling a *unique* and *useful* sample in their neighbourhoods. These likelihoods bias sampling into unexplored space and, as vertices are never removed from the queue, are required to avoid over expansion of vertices. They are estimated from previous collision checks and estimates of the sample density around vertices. Ordering the search on metrics other than the optimization objective can expand the search outside the future informed set and waste computational effort on states that are unnecessary to find the eventual solution.

Unlike these other A*-based methods, BIT* maintains anytime performance while ordering its search only on potential solution quality. This allows it to be run indefinitely until a suitable solution is found, return incrementally better solutions, and almost-surely asymptotically converge to the optimum. This also limits the search to the future informed set and avoids considering states that are unnecessary to find the eventual solution of each representation.

5.1.2 RRT-based Approaches

RRT and RRT* search continuous planning problems by using incremental sampling to build a tree through obstacle-free space. This avoids *a priori* discretizations of the problem domain and allows them to be run indefinitely until a (suitable) solution is found. This also makes the search dependent on the sequence of samples (i.e., makes it *unordered*) and significant work has sought ways to use heuristics to order the search by potential solution quality. Heuristics can also be used to focus the search, as in Anytime RRTs (Ferguson and Stentz, 2006) and Informed RRT*, but this does not change the order of the search.

Heuristically Guided RRT (hRRT; Urmson and Simmons, 2003) probabilistically includes heuristics in the RRT expansion step. Randomly sampled states are probabilistically added in proportion to their heuristic value relative to the tree. This balances the Voronoi bias of the RRT search with a preference towards expanding potentially high-quality paths. This improves performance, especially in problems with continuous cost functions (e.g., path length), but maintains a nonzero probability of searching outside the future informed set and wasting computational effort.

Kiesel et al. (2012) use a two-stage process to order sampling for RRT* in their *f-biasing* technique. First, a heuristic is calculated by solving a coarse discretization of the planning problem with Dijkstra's algorithm. This heuristic is then used to bias RRT* sampling to the regions of the problem domain where solutions were found. This increases the likelihood of finding solutions quickly but maintains a nonzero sampling probability over the entire problem domain for the entire search and allows search effort to be wasted outside the future informed set.

RRT$^{\#}$ (Arslan and Tsiotras, 2013, 2015) uses heuristics to order rewirings in RRT*. It uses LPA* techniques to propagate changes efficiently through the entire tree and assure that each vertex is optimally connected given the current samples. This allows it to avoid computational costs that cannot improve the solution once one is found but does not order the underlying RRT* search which may waste computational effort by searching outside the future informed set.

Fast Marching Tree (FMT*; Janson and Pavone, 2013; Janson et al., 2015b) uses a marching method to search a batch of samples in order of increasing cost-to-come, similar to Dijkstra's algorithm. This makes its search independent of the sampling order and decouples the approximation of the continuous planning problem from the search of the resulting representation but sacrifices anytime performance. Solutions are not returned until the search is finished and the search must be restarted if a (suitable) solution is not found, as with any other *a priori* discretization.

The Motion Planning Using Lower Bounds (MPLB) algorithm (Salzman and Halperin, 2015) extends FMT* with a heuristic and quasi-anytime resolution. Quasi-anytime performance is achieved by *independently* searching increasingly large batches of samples and returning the improved solutions when each individual search finishes. It is stated that this can be done efficiently by reusing information but no specific methods are presented.

Unlike these other RRT*-based methods, BIT* orders its search only on potential solution quality and still maintains anytime performance. This only searches inside the future informed sets of representations and reduces wasted computational effort. This also returns suboptimal solutions in an anytime manner while almost-surely converging asymptotically to the optimum.

5.2 Ordering Search with BIT*[1]

Any discrete set of states distributed in the state space of a planning problem, $X_{\text{samples}} \subset X$, can be viewed as a graph whose edges are given algorithmically by a transition function (an *edge-implicit* graph). When these states are sampled randomly, $X_{\text{samples}} = \{\mathbf{x} \sim \mathcal{U}(X)\}$, the properties of this graph can be described by a probabilistic model known as a RGG (Penrose, 2003).

The connections (edges) between states (vertices) in a RGG depend on their relative geometric position. Common RGGs have edges to a specific number of each state's nearest neighbours (a k-nearest graph; Eppstein et al., 1997) or to all neighbours within a specific distance (an r-disc graph; Gilbert, 1961). RGG theory provides probabilistic relationships between the number and distribution of samples, the k or r defining the graph, and specific properties such as connectivity or relative cost through the graph (Muthukrishnan and Pandurangan, 2005; Penrose, 2003).

Anytime sampling-based planners can be viewed as algorithms that construct a RGG in the problem domain and search it. Some algorithms perform this construction and search simultaneously (e.g., RRT*) and others separately (e.g., FMT*) but, as in graph-based search, their performance always depends on both the accuracy of their approximation and the quality of their search. RRT* uses RGG theory to limit graph complexity while maintaining probabilistic bounds on approximation accuracy but incompletely searches the RGG in the order its constructed (i.e., performs an *unordered* search). FMT* performs a complete ordered search but uses RGG theory to define an *a priori* approximation of the problem domain (i.e., it is not anytime).

BIT* uses RGG theory to limit graph complexity while simultaneously building the graph in an anytime manner and searching it in order of potential solution quality. This is made possible by using *batches* of random samples to build an increasingly dense *edge-implicit* RGG in the informed set and using incremental search techniques to update the search (Fig. 5.3). The anytime approximation allows BIT* to run indefinitely until a (suitable) solution is found. The incremental search technique allows it to search its changing representation efficiently by reusing previous information. The ordered search assures that it only considers states from the future informed set of a given approximation and avoids unnecessary computational cost.

The complete BIT* algorithm is presented in Algs. 5.1–5.3 and Sections 5.2.1–5.2.6 with a discussion on some practical considerations presented in Section 5.2.7. For simplicity, the discussion is limited to a search from a single start state to a finite set of goal states with a constant batch size but the formulation directly extends to searches from a start or goal region, and/or with variable batch sizes. This version is publicly available in OMPL.

[1] Pronounced *Bit-Star*.

Algorithm 5.1: BIT*($\mathbf{x}_{\text{start}} \in X_{\text{free}},\ X_{\text{goal}} \subset X_{\text{free}}$)

```
1  V ← {x_start}; E ← ∅; T = (V, E);                                   ┐
2  X_unconn ← X_goal;                                                  │
3  Q_V ← V; Q_E ← ∅;                                                   │ Initialize search
4  V_sol'n ← V ∩ X_goal; V_unexpnd ← V; X_new ← X_unconn;               │ (Section 5.2.1)
5  c_i ← min_{v_goal ∈ V_sol'n} {g_T(v_goal)} ;                        ┘
6  repeat
7      if Q_E ≡ ∅ and Q_V ≡ ∅ then                                     ┐
8          X_reuse ← Prune(T, X_unconn, c_i);                          │
9          X_sampling ← Sample(m, x_start, X_goal, c_i);               │ Add new batch
10         X_new ← X_reuse ∪ X_sampling;                               │ (Section 5.2.2)
11         X_unconn +← X_new;                                          │
12         Q_V ← V;                                                    ┘
13     while BestQueueValue(Q_V) ≤ BestQueueValue(Q_E) do              ┐ Select edge
14         ExpandNextVertex(Q_V, Q_E, c_i);                            │ (Section 5.2.3)
15     (v_min, x_min) ← PopBestInQueue(Q_E);                           ┘
16     if g_T(v_min) + ĉ(v_min, x_min) + ĥ(x_min) < c_i then           ┐
17         if g_T(v_min) + ĉ(v_min, x_min) < g_T(x_min) then           │
18             c_edge ← c(v_min, x_min);                               │
19             if g_T(v_min) + c_edge + ĥ(x_min) < c_i then            │
20                 if g_T(v_min) + c_edge < g_T(x_min) then            │
21                     if x_min ∈ V then                               │
22                         v_parent ← Parent(x_min);                   │
23                         E +← {(v_parent, x_min)};                   │
24                     else                                            │ Process edge
25                         X_unconn +← {x_min};                        │ (Section 5.2.4)
26                         V +← {x_min}; Q_V +← {x_min}; V_unexpnd +← {x_min};
27                         if x_min ∈ X_goal then                      │
28                             V_sol'n +← {x_min};                     │
29                     E +← {(v_min, x_min)};                          │
30                     c_i ← min_{v_goal ∈ V_sol'n} {g_T(v_goal)};     │
31     else                                                            │
32         Q_E ← ∅; Q_V ← ∅;                                           ┘
33 until STOP;
34 return T;
```

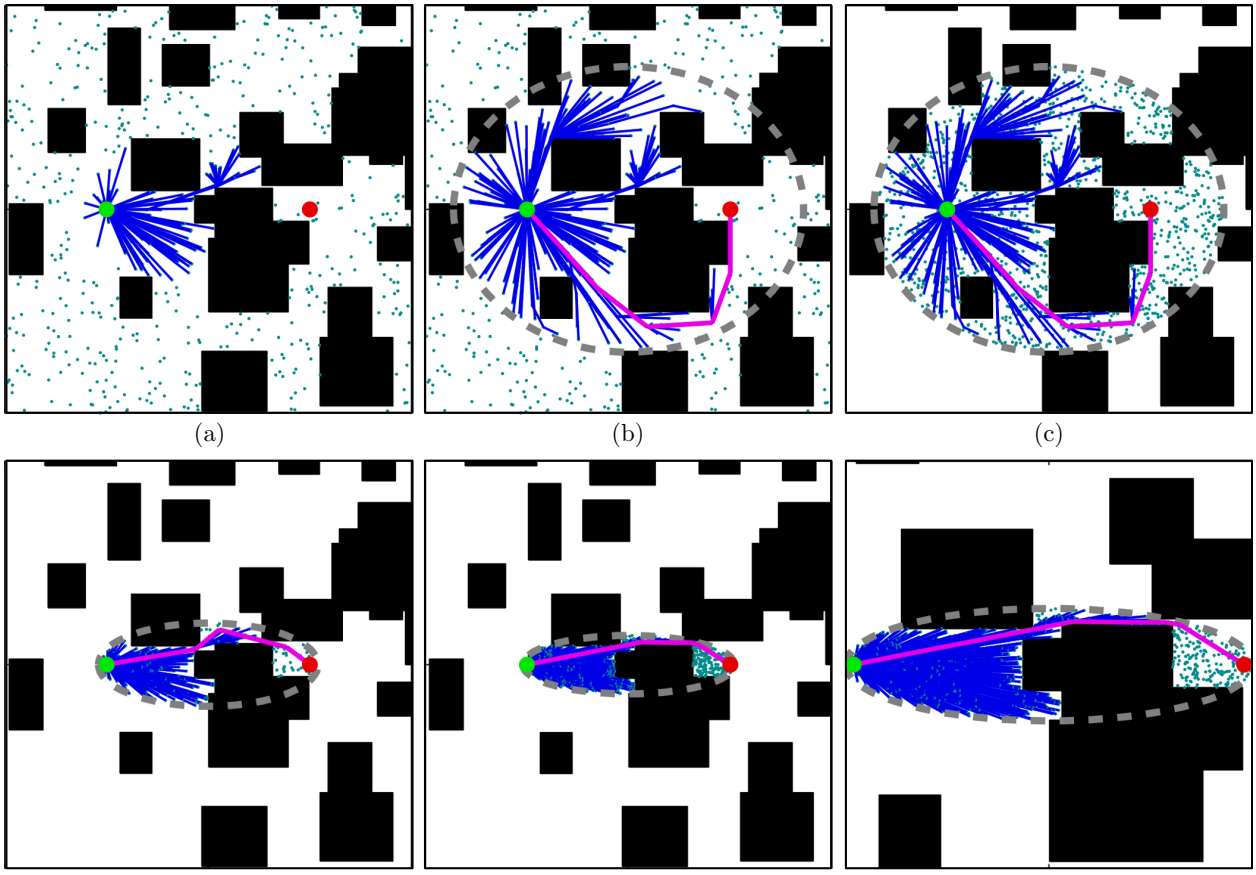


Figure 5.3: An example of how BIT* uses incremental techniques to efficiently search batches of samples in order of potential solution quality. The search starts with a batch of samples uniformly distributed in the planning domain. This batch is searched outward from the start in order of potential solution quality, (a). The search continues until the batch is exhausted or a solution is found that cannot be improved with the current samples, (b). New samples are then added to the informed set and incremental techniques are used to continue the search, (c)–(e). This results in an algorithm that performs an ordered anytime search that almost-surely asymptotically converges to the optimal solution, shown enlarged in (f). By ordering the search, BIT* finds solutions by searching the future informed set of a batch.

BIT* builds an explicit spanning tree of the implicit RGG defined by a batch of samples. The graph initially consists of only the start and goal (Alg. 5.1, Lines 1–5; Section 5.2.1) but is incrementally grown with batches of new samples during the search (Alg. 5.1, Lines 7–12; Section 5.2.2). The graph is searched in order of potential solution quality by selecting the best possible edge from a queue ordered by potential solution cost (Alg. 5.1, Lines 13–15; Section 5.2.3) and considering whether this edge improves the cost-to-come of its target vertex and could improve the current solution (Alg. 5.1, Lines 16–32; Section 5.2.4). The search continues until no edges in the implicit RGG could provide a better solution, at which point the accuracy of the RGG approximation is increased by adding a batch of new samples and the search is resumed. This process continues indefinitely until a suitable solution is found.

5.2.1 Initialization (Alg. 5.1, Lines 1–5)

BIT* begins searching a planning problem with the start, $\mathbf{x}_{\text{start}}$, in the spanning tree, $\mathcal{T} \coloneqq (V, E)$, and the goal states, X_{goal}, in the set of unconnected states, X_{unconn} (Alg. 5.1, Lines 1–2). This defines an implicit RGG whose vertices consist of all states (i.e., $V \cup X_{\text{unconn}}$) and whose edges are defined by a distance function and an appropriate connection limit. When the goal is a continuous region of the problem domain it will need to be discretized (e.g., sampled) before adding to the set of unconnected states.

The explicit spanning tree of this RGG is built using two queues, a vertex expansion queue, $\mathcal{Q}_V$, and an edge evaluation queue, $\mathcal{Q}_E$. These queues are sorted in order of potential solution quality through the current tree. Vertices in the vertex queue, $\mathbf{v} \in \mathcal{Q}_V$, are ordered by the sum of their current cost-to-come and an estimate of their cost-to-go, $g_{\mathcal{T}}(\mathbf{v}) + \widehat{h}(\mathbf{v})$. Edges in the edge queue, $(\mathbf{v}, \mathbf{x}) \in \mathcal{Q}_E$, are sorted by the sum of the current-cost-to-come of their source vertex, an estimate of the edge cost, and an estimate of the cost-to-go of their target vertex, $g_{\mathcal{T}}(\mathbf{v}) + \widehat{c}(\mathbf{v}, \mathbf{x}) + \widehat{h}(\mathbf{x})$. Ties are broken in the vertex queue in favour of entries with the lowest cost-to-come through the current tree, $g_{\mathcal{T}}(\mathbf{v})$, and in the edge queue in favour of the lowest cost-to-come through the current tree and estimated edge cost, $g_{\mathcal{T}}(\mathbf{v}) + \widehat{c}(\mathbf{v}, \mathbf{x})$, and then the cost-to-come through the current tree, $g_{\mathcal{T}}(\mathbf{v})$. These queues are initialized to contain all the vertices in the tree and an empty queue, respectively (Alg. 5.1, Line 3).

To improve search efficiency, BIT* tracks the vertices in the goal region, $V_{\text{sol'n}}$, the vertices that have never been expanded, V_{unexpnd}, the samples newly created during this batch, X_{new}, and the current best solution, c_i. These are initialized to any vertices already in the solution (empty in all but the most trivial planning problems), the existing vertices, the existing samples, and the current best solution, respectively (Alg. 5.1, Lines 4–5). As is customary, the minimum of an empty set is taken to be infinity.

Initialized, BIT* now alternately builds an increasingly accurate implicit RGG approximation of the continuous planning problem (Section 5.2.2) and searches these representations for explicit solutions in order of potential solution quality (Sections 5.2.3 and 5.2.4).

5.2.2 Batch Addition (Alg. 5.1, Lines 7–12)

BIT* alternates between building an increasingly dense approximation of the continuous planning problem and searching this representation for a solution. The approximation is updated whenever the search is finished (i.e., both queues are empty; Alg. 5.1, Line 7) by removing unnecessary states and adding a batch of new samples. This avoids the computational cost of representing regions of the problem domain that cannot provide a better solution while increasing the accuracy of approximating the regions that can (i.e., the informed set). This improving approximation allows BIT* to almost-surely converge asymptotically to the optimal solution.

The approximation is pruned to the informed set by removing any states or edges that cannot improve the *current* solution (Alg. 5.1, Line 8; Section 5.2.6). This reduces unnecessary complexity but may disconnect vertices in the informed set that cannot improve the solution solely because of their current connections. These vertices are recycled as additional 'new' samples in the batch so that they may be reconnected later if better connections are found.

The approximation is improved by adding m new randomly generated samples from the informed set (Alg. 5.1, Line 9). This can be accomplished with direct informed sampling or advanced rejection sampling (e.g., Kunz et al., 2016).

Each batch of states are labelled as X_{new} for the duration of that batch's search (Alg. 5.1, Line 10). This set is used to improve search efficiency and consists of both the newly generated samples and the recycled disconnected vertices. BIT* adds these new states to the set of unconnected states and initializes the vertex queue with all the vertices in the tree (Alg. 5.1, Lines 11–12) to restart the search (Sections 5.2.3 and 5.2.4).

5.2.3 Edge Selection (Alg. 5.1, Lines 13–15)

Traditional graph-based search techniques assume that finding and evaluating vertex connections is computationally inexpensive (e.g., given explicitly). This is not true in sampling-based planning as finding vertex connections (i.e., the edges in the edge-implicit RGG) requires performing a nearest-neighbour search and evaluating them requires checking for collisions and solving two-point BVPs (e.g., differential constraints). BIT* avoids these computational costs until required by using a lazy search procedure that delays both finding and evaluating connections in the RGG. Similar lazy techniques can be found in both advanced graph-based search and sampling-based planners (Bohlin and Kavraki, 2000; Branicky et al., 2001; Cohen et al., 2014b; Hauser, 2015; Helmert, 2006; Sánchez and Latombe, 2002).

Connections are found by using a vertex queue, $\mathcal{Q}_V$, ordered by potential solution quality. This queue delays processing a vertex (i.e., performing a nearest-neighbour search) until its outgoing connections *could* be part of the best solution to the current graph. These connections are evaluated by using an edge queue, $\mathcal{Q}_E$, also ordered by potential solution

quality. This queue delays evaluating an edge (i.e., performing collision checks and solving two-point BVPs) until it *could* be part of the best solution to the current graph.

A vertex could be part of the best solution when it could provide an outgoing edge better than the best edge in the edge queue. When the heuristic is consistent (e.g., the L^2 norm) the queue value of a vertex, $\mathbf{v} \in \mathcal{Q}_V$, is a lower-bounding estimate of the queue value of its outgoing edges,

$$\forall \mathbf{x} \in X,\ g_{\mathcal{T}}(\mathbf{v}) + \widehat{h}(\mathbf{v}) \leq g_{\mathcal{T}}(\mathbf{v}) + \widehat{c}(\mathbf{v}, \mathbf{x}) + \widehat{h}(\mathbf{x}).$$

The best edge at any iteration can therefore be found by processing the vertex queue until it is worse than the edge queue (Alg. 5.1, Line 13). This process of removing a vertex from the vertex queue and placing its outgoing edges in the edge queue is referred to as *expanding* a vertex (Alg. 5.1, Line 14; ; Section 5.2.5). Once all necessary vertices are expanded, the best edge in the queue, $(\mathbf{v}_{\min}, \mathbf{x}_{\min})$, is removed (Alg. 5.1, Line 15) and used for this iteration of the search (Section 5.2.4).

The functions `BestQueueValue` $(\cdot)$ and `PopBestInQueue` $(\cdot)$ return the value of the element at the front of a queue and pop the element off the front of a queue, respectively.

5.2.4 Edge Processing (Alg. 5.1, Lines 16–32)

BIT* also uses heuristics to avoid expensive calculations when evaluating the best edge, $(\mathbf{v}_{\min}, \mathbf{x}_{\min})$. An edge is added to the spanning tree if and only if

1. an *estimate* of its cost *could* provide a better *solution*, given the current tree,

$$g_{\mathcal{T}}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}_{\min}) + \widehat{h}(\mathbf{x}_{\min}) < c_i, \qquad \text{(Alg. 5.1, Line 16)}$$

2. an *estimate* of its cost *could* improve the *current tree*,

$$g_{\mathcal{T}}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}_{\min}) < g_{\mathcal{T}}(\mathbf{x}_{\min}), \qquad \text{(Alg. 5.1, Line 17)}$$

3. its *real* cost *could* provide a better *solution*, given the current tree,

$$g_{\mathcal{T}}(\mathbf{v}_{\min}) + c(\mathbf{v}_{\min}, \mathbf{x}_{\min}) + \widehat{h}(\mathbf{x}_{\min}) < c_i, \qquad \text{(Alg. 5.1, Line 19)}$$

4. and its *real* cost *will* improve the *current tree*,

$$g_{\mathcal{T}}(\mathbf{v}_{\min}) + c(\mathbf{v}_{\min}, \mathbf{x}_{\min}) < g_{\mathcal{T}}(\mathbf{x}_{\min}). \qquad \text{(Alg. 5.1, Line 20)}$$

Conditions 1 and 3 are always true in the absence of a solution, while Conditions 2 and 4 are always true when the target of the edge, $\mathbf{x}_{\min}$, is not in the spanning tree.

Checking if the edge could ever provide a better solution or improve the current tree (Conditions 1 and 2) allows BIT* to reject edges without calculating their true cost (Alg. 5.1, Lines 16–17). Condition 1 also provides a stopping condition for searching the current RGG. When an edge fails this condition so does the entire queue and both queues can be cleared

to start a new batch (Alg. 5.1, Line 32). If the edge fails Condition 2 it is discarded and the iteration finishes. If the edge passes both these conditions its true cost is calculated by performing a collision check and solving any two-point BVPs (Alg. 5.1, Line 18). Edges in collision are considered to have infinite cost.

Checking if the real edge could provide a better solution given the current tree (Condition 3), allows BIT* to reduce tree complexity by rejecting edges that could never improve the current solution (Alg. 5.1, Line 19). Checking if the real edge improves the current tree (Condition 4), assures the cost-to-come of the explicit tree decreases monotonically (Alg. 5.1, Line 20). If the edge fails either of these conditions it is discarded and the iteration finishes.

An edge passing these conditions is added to the spanning tree. If the target vertex is already connected (Alg. 5.1, Line 21), then the edge represents a *rewiring* and the current edge must be removed (Alg. 5.1, Lines 22–23). Otherwise, the edge represents an *expansion* of the tree and the target vertex must be moved from the set of unconnected states to the set of vertices, inserted into the vertex queue for future expansion, and marked as a never-expanded vertex (Alg. 5.1, Lines 25–26). The new vertex is also added to the set of vertices in the goal region if appropriate (Alg. 5.1, Lines 27–28).

The new edge is then finally added to the tree (Alg. 5.1, Line 29) and the current best solution is updated as necessary (Alg. 5.1, Line 30). The search then continues by selecting the next edge in the queue (Section 5.2.3) or increasing the approximation accuracy if the current RGG cannot provide a better solution (Section 5.2.2).

5.2.5 Vertex Expansion (Alg. 5.1, Line 14; Alg. 5.2)

The function `ExpandNextVertex` $(\mathcal{Q}_V, \mathcal{Q}_E, c_i)$ removes the front of the vertex queue (Alg. 5.2, Line 1) and adds its outgoing edges in the RGG to the edge queue. The RGG is defined using the results of Karaman and Frazzoli (2011), (3.4) and (3.5), to limit graph complexity while maintaining almost-sure asymptotic convergence to the optimum. Edges exist between a vertex and the k_{BIT^*}-closest states or all states within a distance of r_{BIT^*}, with

$$r_{\text{BIT}^*} > r^*_{\text{BIT}^*},$$
$$r^*_{\text{BIT}^*} \coloneqq \left(2\left(1+\frac{1}{n}\right)\left(\frac{\min\left\{\lambda\left(X\right), \lambda\left(X_{\widehat{f}}\right)\right\}}{\zeta_n}\right)\left(\frac{\log\left(|V|+|X_{\text{unconn}}|-m\right)}{|V|+|X_{\text{unconn}}|-m}\right)\right)^{\frac{1}{n}}, \quad (5.1)$$

and

$$k_{\text{BIT}^*} > k^*_{\text{BIT}^*},$$
$$k^*_{\text{BIT}^*} \coloneqq e\left(1+\frac{1}{n}\right)\log\left(|V|+|X_{\text{unconn}}|-m\right), \quad (5.2)$$

where m is the number of samples added in the last batch.

Algorithm 5.2: `ExpandNextVertex` $(\mathcal{Q}_V \subseteq V,\ \mathcal{Q}_E \subseteq V \times (V \cup X),\ c_i \in \mathbb{R}_{\geq 0})$

1 $\mathbf{v}_{\min} \leftarrow$ `PopBestInQueue` $(\mathcal{Q}_V)$;
2 **if** $\mathbf{v}_{\min} \in V_{\text{unexpnd}}$ **then**
3 $\quad X_{\text{near}} \leftarrow$ `Near` $(X_{\text{unconn}}, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
4 **else**
5 $\quad X_{\text{near}} \leftarrow$ `Near` $(X_{\text{new}} \cap X_{\text{unconn}}, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
6 $\mathcal{Q}_E \xleftarrow{+} \left\{(\mathbf{v}_{\min}, \mathbf{x}) \in V \times X_{\text{near}} \;\middle|\; \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}) + \widehat{h}(\mathbf{x}) < c_i\right\}$;
7 **if** $\mathbf{v}_{\min} \in V_{\text{unexpnd}}$ **then**
8 $\quad V_{\text{near}} \leftarrow$ `Near` $(V, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
9 $\quad \mathcal{Q}_E \xleftarrow{+} \left\{(\mathbf{v}_{\min}, \mathbf{w}) \in V \times V_{\text{near}} \;\middle|\; (\mathbf{v}_{\min}, \mathbf{w}) \notin E,\ \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{w}) + \widehat{h}(\mathbf{w}) < c_i,\ \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{w}) < g_{\mathcal{T}}(\mathbf{w})\right\}$;
10 $\quad V_{\text{unexpnd}} \xleftarrow{-} \{\mathbf{v}_{\min}\}$;

This connection limit is calculated from the cardinality of the graph *minus* the m new samples to simplify proving almost-sure asymptotic optimality (Section 5.3). This lower bound will be large for the initial sparse batches but it can be thresholded with a maximum edge length, as done by RRT* in (3.3). The function `Near` returns the states that meet the selected RGG connection criteria for a given vertex.

Every vertex in the tree is either expanded or pruned in every batch. Processing all the outgoing edges from vertices would result in BIT* repeatedly considering the same previously rejected edges. This can be avoided by using the sets of never-expanded vertices, V_{unexpnd}, and new samples, X_{new}, to add only *previously unconsidered* edges to the edge queue.

Whether edges to unconnected samples are new depends on whether the source vertex has previously been expanded (Alg. 5.2, Line 2). If it has not been expanded then no outgoing connections have been considered and all nearby unconnected samples are potential descendants (Alg. 5.2, Line 3). If it has been expanded then any connections to 'old' unconnected samples have already been considered and rejected and only the 'new' samples are considered as potential descendants (Alg. 5.2, Line 5). The subset of these potential edges that could improve the current solution are added to the queue in both situations (Alg. 5.2, Line 6).

Whether edges to connected samples (i.e., *rewirings*) are new also depends on whether the source vertex has been expanded (Alg. 5.2, Line 7). If it has not been expanded then all nearby connected vertices are considered as potential descendants (Alg. 5.2, Line 8). The subset of these potential edges that could improve the current solution *and* the current tree are added to the queue (Alg. 5.2, Line 9) and the vertex is then marked as expanded (Alg. 5.2, Line 10).

If a vertex has previously been expanded then no rewirings are considered. Improvements in the tree may now allow a previously considered edge to improve connected vertices but considering these connections would require repeatedly reconsidering infeasible edges. This lack of *propagated rewiring* has no effect on almost-sure asymptotic optimality, as in RRT*.

Algorithm 5.3: `Prune` $(\mathcal{T} = (V, E),\ X_{\text{unconn}} \subset X,\ c_i \in \mathbb{R}_{\geq 0})$

```
1  X_reuse ← ∅;
2  X_unconn ←⁻ { x ∈ X_unconn | f̂(x) ≥ c_i };
3  forall v ∈ V in order of increasing g_T(v) do
4      if f̂(v) > c_i or g_T(v) + ĥ(v) > c_i then
5          V ←⁻ {v}; V_sol'n ←⁻ {v}; V_unexpnd ←⁻ {v};
6          v_parent ← Parent(v);
7          E ←⁻ {(v_parent, v)};
8          if f̂(v) < c_i then
9              X_reuse ←⁺ {v};
10 return X_reuse;
```

5.2.6 Graph Pruning (Alg. 5.1, Line 8; Alg. 5.3)

The function, `Prune` $(\mathcal{T}, X_{\text{unconn}}, c_i)$, reduces the complexity of both the approximation of the continuous planning problem (i.e., the implicit RGG) and its search (i.e., the explicit spanning tree) by limiting them to the informed set. It removes any states that can *never* provide a better solution and disconnects any vertices that cannot provide a better solution *given the current tree*. Disconnected vertices that *could* improve the solution with a better connection are reused as new samples in the next batch to maintain uniform sample density in the informed set. This assures that in each batch every vertex is either expanded or pruned as assumed by `ExpandNextVertex` to avoid reconsidering edges (Section 5.2.5).

The set of recycled vertices is initialized as an empty set (Alg. 5.3, Line 1) and all unconnected states that cannot provide a better solution (i.e., are not members of the informed set) are removed (Alg. 5.3, Line 2). The connected vertices are then incrementally pruned in order of increasing cost-to-come (Alg. 5.3, Line 3) by identifying those that can never provide a better solution or improve the solution given the *current* tree (Alg. 5.3, Line 4). Vertices that fail either condition are removed from the tree by disconnecting their incoming edge and removing them from the vertex set and any labelling sets (Alg. 5.3, Lines 5–7).

Any disconnected vertex that could provide a better solution (i.e., is a member of the informed set) is reused as a sample in the next batch (Alg. 5.3, Lines 8–9). This maintains uniform sample density in the informed set and assures that vertices will be reconnected if future improvements allow them to provide a better solution. This set of disconnected vertices is returned to BIT* at the end of the pruning procedure (Alg. 5.3, Line 10).

5.2.7 Practical Considerations

Algs. 5.1–5.3 describe an *implementation-agnostic* version of BIT* and leave room for a number of practical improvements depending on the specific implementation.

Many of the sets (e.g., X_{new}, V_{unexpnd}) can be implemented more efficiently as labels. Searches (e.g., Alg. 5.2, Line 3) can be implemented efficiently with appropriate datastructures that do not require an exhaustive global search (e.g., k-d trees). Pruning (Alg. 5.1, Line 8; Alg. 5.3) is expensive and should only occur when a new solution has been found or limited to *significant* changes in solution cost.

Ordered containers provide efficient sorted queues but the order of elements in the vertex and edge queues change when vertices are rewired. In practice, there appears to be little difference between efficiently resorting the affected elements in these queues and only lazily resorting the queue to assure no elements have been missed when reaching the end.

Depending on the datastructure used for the edge queue, it may be beneficial to remove unnecessary entries when a new edge is added to the spanning tree, i.e., by adding

$$\mathcal{Q}_E \xleftarrow{-} \{(\mathbf{v}, \mathbf{x}_{\min}) \in \mathcal{Q}_E \mid \widehat{g}(\mathbf{v}) + \widehat{c}(\mathbf{v}, \mathbf{x}_{\min}) \geq g_{\mathcal{T}}(\mathbf{x}_{\min})\}$$

after Alg. 5.1, Line 29.

5.3 Analysis

The performance of BIT* is analyzed theoretically using the results of Karaman and Frazzoli (2011). It is shown to be probabilistically complete (Theorem 5.1) and almost-surely asymptotically optimal (Theorem 5.2). Its search order is also shown to be equivalent to a lazy version of LPA* (Lemma 5.3).

Theorem 5.1 (Probabilistic completeness of BIT*)**.** *The probability that BIT* finds a feasible solution to a given planning problem, if one exists, when given infinite samples is one,*

$$\liminf_{q \to \infty} P\left(\sigma_{q,\text{BIT}^*} \in \Sigma,\, \sigma_{q,\text{BIT}^*}(0) = \mathbf{x}_{\text{start}},\, \sigma_{q,\text{BIT}^*}(1) \in X_{\text{goal}}\right) = 1,$$

where q is the number of samples, σ_{q,BIT^*} is the path found by BIT from those samples, and Σ is the set of all feasible, collision-free paths.*

Proof. Proof of Theorem 5.1 follows from the proof of Theorem 5.2. □

Theorem 5.2 (Almost-sure asymptotic optimality of BIT*)**.** *The probability that BIT* converges asymptotically towards the optimal solution of a given planning problem, if one exists, when given infinite samples is one,*

$$P\left(\limsup_{q \to \infty} c\left(\sigma_{q,\text{BIT}^*}\right) = c\left(\sigma^*\right)\right) = 1,$$

where q is the number of samples, σ_{q,BIT^*} is the path found by BIT from those samples, and σ^* is optimal solution to the planning problem.*

Proof. Theorem 5.2 is proven by showing that BIT* considers at least the same edges as RRT* for a sequence of states, $X_{\text{samples}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q)$, and connection limit, $r_{\text{BIT}^*} \geq r_{\text{RRT}^*}$.

RRT* incrementally builds a tree from a sequence of samples. For each state in the sequence, $\mathbf{x}_k$, it considers the neighbourhood of earlier states that are within the connection limit,

$$X_{\text{near},k} \coloneqq \left\{ \mathbf{x}_j \in X_{\text{samples}} \;\middle|\; j < k,\ \|\mathbf{x}_k - \mathbf{x}_j\|_2 \leq r_{\text{RRT}^*} \right\}.$$

It selects the connection from this neighbourhood that minimizes the cost-to-come of the state and then evaluates the ability of connections from this state to reduce the cost-to-come of the other states in the neighbourhood.

Given the same sequence of states, BIT* groups them into batches of samples, $X_{\text{samples}} = (Y_1, Y_2, \ldots, Y_\ell)$, where each batch is a set of m samples, e.g., $Y_1 \coloneqq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. It incrementally builds a tree by processing this batched sequence of samples. For each state in the sequence, $\mathbf{y} \in Y_k$, it considers the neighbourhood of states from the same or earlier batches within the connection limit,

$$X_{\text{near},k} \coloneqq \{\mathbf{x} \in Y_j \mid j \leq k,\ \|\mathbf{y} - \mathbf{x}\|_2 \leq r_{\text{BIT}^*}\}.$$

The connection from this neighbourhood that minimizes the cost-to-come of the state is the one that will be added to the tree and all its outgoing edges will be considered in connecting its neighbours. This contains all the edges considered by RRT* for an equivalent connection limit, $r_{\text{BIT}^*} \geq r_{\text{RRT}^*}$. As (5.1) uses the same connection radius for a batch that RRT* would use for the first sample in the batch and the connection radius of both are monotonically decreasing, this shows that BIT* is almost-surely asymptotically optimal and proves Theorem 5.2. □

BIT* uses a search order that is equivalent to a version of LPA* that *lazily* estimates the edge cost between states in the queue (Lemma 5.3). As BIT* does not reconsider outgoing connections from rewired vertices (i.e., it does not propagate rewirings), it is not a complete LPA* search of the RGG but instead a search similar to TLPA* (Aine and Likhachev, 2016).

Lemma 5.3 (Equivalence of BIT* to a lazy LPA* search)**.** *An edge queue ordered first on the sum of a vertex's estimated cost-to-go, estimated incoming edge cost, and current cost-to-come of its parent,*

$$g_{\mathcal{T}}(\mathbf{u}) + \widehat{c}(\mathbf{u}, \mathbf{v}) + \widehat{h}(\mathbf{v}),$$

then the estimated cost-to-come of the vertex,

$$g_{\mathcal{T}}(\mathbf{u}) + \widehat{c}(\mathbf{u}, \mathbf{v}),$$

and then the cost-to-come of its parent,

$$g_{\mathcal{T}}(\mathbf{u}),$$

is an equivalent ordering to LPA (Koenig et al., 2004).*

Proof. LPA* uses a queue of vertices ordered lexicographically first on the solution cost constrained to go through the vertex and then the cost-to-come of the vertex. Both these terms are calculated for a vertex, $\mathbf{v} \in V$, considering all its possible incoming edges (referred to as the *rhs-value* in LPA*), i.e.,

$$\min_{(\mathbf{u},\mathbf{v})\in E} \{g_{\mathcal{T}}(\mathbf{u}) + c(\mathbf{u},\mathbf{v})\} + \widehat{h}(\mathbf{v})$$

and

$$\min_{(\mathbf{u},\mathbf{v})\in E} \{g_{\mathcal{T}}(\mathbf{u}) + c(\mathbf{u},\mathbf{v})\}.$$

This minimum requires calculating the true edge cost between a vertex and all of its possible parents. This calculation is expensive in sampling-based planning (e.g., collision checking, differential constraints, etc.), and reducing its calculation is desirable. This can be achieved by incrementally calculating the minimum in the order given by an admissible heuristic estimate of edge cost. Considering edges into the vertex in order of increasing *estimated* cost calculates a running minimum that finds the true minimum when the estimated cost through the next edge to consider is higher than the current value.

BIT* combines the minimum calculations for individual vertices into a single edge queue. It simultaneously calculates the minimum cost-to-come for each vertex in the current graph while expanding vertices in order of increasing estimated solution cost. □

5.4 Modifications and Extensions

The basic version of BIT* presented in Algs. 5.1–5.3 can be modified and extended to include more advanced features that may improve performance for some planning applications. Section 5.4.1 presents a method to delay rewiring the tree until an initial solution is found. This prioritizes exploring the RGG to find solutions and may be beneficial in time-constrained applications. Section 5.4.2 presents a method to delay sampling until necessary. This avoids approximating regions of the planning problem that are never searched, improves performance in large planning problems, and avoids the need to define *a priori* limits in unbounded problems.

Section 5.4.3 presents a method for BIT* to occasionally remove unconnected samples while maintaining almost-sure asymptotic optimality. This avoids repeated connection attempts to infeasible states and may be beneficial in problems when many regions of the free space are unreachable. Section 5.4.4 extends the idea of reducing the number of connections attempted per sample during an ordered search to develop Sorted RRT* (SORRT*). This version of RRT* uses batches of samples to order its search by potential solution quality, as in BIT*, but only makes one connection attempt per sample, as in RRT*.

Algorithm 5.4: `ExpandNextVertex` $(\mathcal{Q}_V \subseteq V,\ \mathcal{Q}_E \subseteq V \times (V \cup X),\ c_i \in \mathbb{R}_{\geq 0})$

1 $\mathbf{v}_{\min} \leftarrow$ `PopBestInQueue` $(\mathcal{Q}_V)$;
2 **if** $\mathbf{v}_{\min} \in V_{\text{unexpnd}}$ **then**
3 $\quad X_{\text{near}} \leftarrow$ `Near` $(X_{\text{unconn}}, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
4 $\quad V_{\text{unexpnd}} \xleftarrow{-} \{\mathbf{v}_{\min}\}$;
5 $\quad V_{\text{delayed}} \xleftarrow{+} \{\mathbf{v}_{\min}\}$;
6 **else**
7 $\quad X_{\text{near}} \leftarrow$ `Near` $(X_{\text{new}} \cap X_{\text{unconn}}, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
8 $\mathcal{Q}_E \xleftarrow{+} \left\{(\mathbf{v}_{\min}, \mathbf{x}) \in V \times X_{\text{near}} \;\middle|\; \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}) + \widehat{h}(\mathbf{x}) < c_i\right\}$;
9 **if** $\mathbf{v}_{\min} \in V_{\text{delayed}}$ **and** $c_i < \infty$ **then**
10 $\quad V_{\text{near}} \leftarrow$ `Near` $(V, \mathbf{v}_{\min}, r_{\text{BIT}^*})$;
11 $\quad \mathcal{Q}_E \xleftarrow{+} \left\{(\mathbf{v}_{\min}, \mathbf{w}) \in V \times V_{\text{near}} \;\middle|\; (\mathbf{v}_{\min}, \mathbf{w}) \notin E,\ \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{w}) + \widehat{h}(\mathbf{w}) < c_i,\ \widehat{g}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{w}) < g_{\mathcal{T}}(\mathbf{w})\right\}$;
12 $\quad V_{\text{delayed}} \xleftarrow{-} \{\mathbf{v}_{\min}\}$;

5.4.1 Delayed Rewiring

Many robotic systems have a finite amount of computational time available to solve planning problems. In these situations, improving the existing graph before an initial solution is found only reduces the likelihood of BIT* solving the given problem. A method to delay rewirings until a solution is found is presented in Alg. 5.4 as simple modifications to `ExpandNextVertex`, with changes highlighted in red (cf. Alg. 5.2). The rewirings are still performed once a solution is found and this method does not affect almost-sure asymptotic optimality.

Rewirings are delayed by separately tracking whether vertices are unexpanded to nearby unconnected samples, V_{unexpnd}, and unexpanded to nearby connected vertices, V_{delayed}. This allows BIT* to prioritize finding a solution by only considering edges to new samples and then once a solution is found improving the graph by considering rewirings from the existing vertices.

A vertex is moved from the never-expanded set to the delayed set when edges to its potential unconnected descendants are added to the edge queue (Alg. 5.4, Lines 4–5). This allows future expansions of the vertex to avoid old unconnected samples while tracking that the vertex's rewirings have not been considered. Vertices in the delayed set are expanded as potential rewirings of other connected vertices once a solution is found (Alg. 5.4, Line 9) and the delayed label is removed (Alg. 5.4, Line 12).

This extension requires initializing and resetting V_{delayed} along with the other labelling sets (e.g., Alg. 5.1, Line 4 and Alg. 5.3, Line 5). This extension is included in the publicly available OMPL implementation of BIT*.

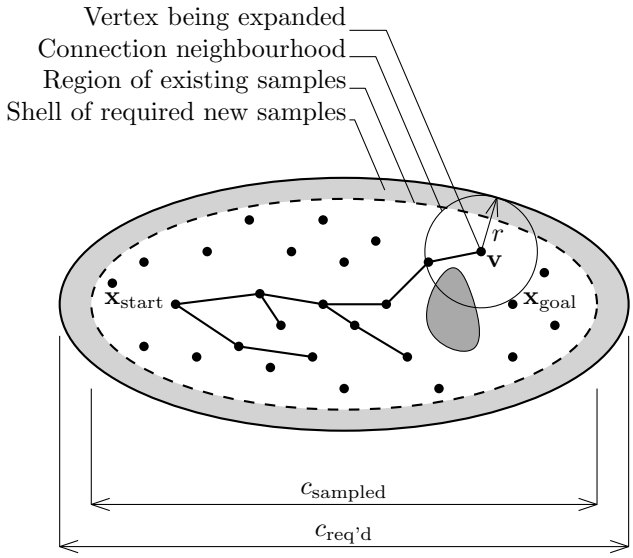


Figure 5.4: An illustration of just-in-time (JIT) sampling. Samples are generated only when they are necessary for the expansion of a vertex, $\mathbf{v}$, into the edge queue. This is accomplished while maintaining uniform sample density by an expanding informed set. The informed set is expanded by adding uniformly distributed samples in the prolate hyperspheroidal shell defined by the difference between the maximum heuristic value of the neighbourhood, $c_{\text{req'd}}$, and the currently sampled informed set, c_{sampled}.

5.4.2 Just-in-Time (JIT) Sampling

Many robotic systems operate in environments that are unbounded (e.g., the outdoors). These problems has previously required using *a priori* search limits to make the problem domain tractable. Selecting these limits can be difficult and may prevent finding a solution (e.g., defining a domain that does not contain a solution) or reduce performance (e.g., defining a domain too large to search sufficiently). A method to avoid these problems in BIT* by generating samples *just in time* (JIT) is presented in Alg. 5.5 and accompanying modifications to the main algorithm. This modification generates samples only when needed by BIT*'s search while still maintaining uniform sample density and almost-sure asymptotic convergence to the optimum. This avoids approximating regions of the problem that are not used to find a solution and allows BIT* to operate directly on large or unbounded planning problems.

BIT* searches a planning problem by constructing and searching an implicit RGG defined by a number of uniformly distributed samples (vertices) and their relative distances (edges). These samples are given explicitly in Algs. 5.1–5.3 but are not used until they could be a descendant of a vertex in the tree. This occurs when the samples are within the local neighbourhood of the vertex (Fig. 5.4). Samples therefore can be generated as needed by incrementally sampling an expanding informed set. The size of the informed set, $c_{\text{req'd}}$, necessary to contain the neighbourhood of a vertex, X_{near}, for planning problems seeking to

Algorithm 5.5: `UpdateSamples` $(\mathbf{v} \in V,\ c_{\text{sampled}} \leq c_i,\ c_i \in \mathbb{R}_{\geq 0})$

1 $c_{\text{req'd}} \leftarrow \min\left\{\widehat{f}(\mathbf{v}) + 2r_{\text{BIT}^*}, c_i\right\}$;
2 **if** $c_{\text{req'd}} > c_{\text{sampled}}$ **then**
3 $\quad \lambda_{\text{sample}} \leftarrow \lambda_{\text{PHS}}(c_{\text{req'd}}) - \lambda_{\text{PHS}}(c_{\text{sampled}})$;
4 $\quad m' \leftarrow \rho\lambda_{\text{sample}}$;
5 $\quad X_{\text{new}} \xleftarrow{+}$ `Sample` $(m', c_{\text{sampled}}, c_{\text{req'd}})$;
6 $\quad X_{\text{unconn}} \xleftarrow{+} X_{\text{new}}$;
7 $\quad c_{\text{sampled}} \leftarrow c_{\text{req'd}}$;

minimize path length,

$$c_{\text{req'd}} \coloneqq \max_{\mathbf{x} \in X_{\text{near}}} \left\{\widehat{f}(\mathbf{x})\right\},$$

is bounded from above as,

$$c_{\text{req'd}} \leq \widehat{f}(\mathbf{v}) + 2r_{\text{BIT}^*}.$$

JIT sampling only generates samples when necessary to expand vertices by incrementally sampling this growing informed set. It does this while maintaining uniform sample density by tracking the previously sampled size of the set, c_{sampled}, and only generating the new samples necessary to increase it. The function `UpdateSamples` $(\mathbf{v}, c_{\text{sampled}}, c_i)$ generates JIT samples for vertex expansion in geometric planning problems (Alg. 5.5). The required size of the sampled informed set, $c_{\text{req'd}}$, is a function of the neighbourhood and the maximum size of the informed set (Alg. 5.5, Line 1). If it is higher than the previously sampled cost, c_{sampled}, the local neighbourhood requires new samples (Alg. 5.5, Line 2). If it is less than the sampled cost the neighbourhood has already been sampled and no samples are generated.

The number of required new samples, m', can be calculated from the chosen batch sample density, ρ, and the volume of the shell being sampled (Alg. 5.5, Lines 3–4). These samples are added to the set of new states and the set of unconnected states (Alg. 5.5, Lines 5–6). Finally, the sampled cost is updated to reflect the new size of the sampled informed set (Alg. 5.5, Line 7).

The function $\lambda_{\text{PHS}}(\cdot)$ calculates the measure of the prolate hyperspheroid defined by the start and goal with the given cost using (3.8). The function `Sample` $(m', c_{\text{sampled}}, c_{\text{req'd}})$ generates samples within the cost interval $[c_{\text{sampled}}, c_{\text{req'd}})$ and may be implemented with rejection sampling.

Using Alg. 5.5 requires modifying Algs. 5.1 and 5.2. The `Sample` function of a batch (Alg. 5.1, Line 9) is replaced with an initialization of the sampled cost variable, $c_{\text{sampled}} \leftarrow 0$, and `UpdateSamples` $(\mathbf{v}, c_{\text{sampled}}, c_i)$ is added to the front of `ExpandNextVertex` (Alg. 5.2). This extension is included in the publicly available OMPL implementation of BIT*.

5.4.3 Sample Removal

BIT* approximates continuous planning problems by building an implicit RGG. It efficiently increases the accuracy of this approximation by focusing it to the informed set and alternately adds new samples to increase density and reduces the size of the informed set by searching the existing approximation for better solutions. This builds an explicit spanning tree that contains all states that could *currently* provide a better solution but may not use every state in the RGG.

States in an informed set may not be able to improve a solution for many reasons. The approximation may be insufficiently accurate (i.e., low sample density) to capture difficult features (e.g., narrow passages) or represent sufficiently optimal paths. The informed set may also include regions of the problem domain that cannot improve the solution due to unconsidered problem features (e.g., barriers separating free space) or because it is otherwise poorly chosen (i.e., low precision). Unconnected samples in the first situation may later be beneficial to the search but samples in the second represent unnecessary computational cost. Periodically removing these samples would reduce the complexity of the implicit RGG and avoid repeatedly attempting to connect them to new vertices in the tree.

Unconnected samples can be removed while maintaining the requirements for almost-sure asymptotic optimality by modifying the RGG connection limits to consider only uniformly distributed samples. This can be accomplished by using the number of uniformly distributed samples added since the last removal of unconnected states in (5.1) and (5.2). This simple extension is included in the publicly available OMPL implementation of BIT*.

5.4.4 Sorted RRT* (SORRT*)[2]

Approaching sampling-based planning as the search of an implicit RGG motivates BIT* to consider multiple connections to each sample. Section 5.4.3 presents a method to limit this number of attempts by periodically removing samples. The natural extension of this idea is to consider only a single connection attempt per sample, as in RRT*. This motivates the development of Sorted RRT* (SORRT*), a version of RRT* that orders its search by potential solution quality by sorting batches of samples.

SORRT* is presented in Alg. 5.6 as simple modifications of Informed RRT*, with changes highlighted in red (cf. Alg. 4.1). Instead of expanding the tree towards a randomly generated sample at each iteration, SORRT* extends the tree towards the best unconsidered sample in its current batch. It accomplishes this by using a queue samples, $\mathcal{Q}_{\text{Samples}}$, ordered by potential solution cost, $\widehat{f}(\cdot)$. This queue is filled with m samples (Alg. 5.6, Lines 5–6) and the search proceeds by expanding the tree towards the best sample in the queue (Alg. 5.6, Line 7). This orders the search for the m iterations required to use the samples in a batch, at which point a new batch of samples is generated and the search continues.

[2]Pronounced *Sort-Star*.

Algorithm 5.6: SORRT*($\mathbf{x}_{\text{start}} \in X_{\text{free}},\ X_{\text{goal}} \subset X$)

$V \leftarrow \{\mathbf{x}_{\text{start}}\};\ E \leftarrow \emptyset;\ \mathcal{T} = (V, E)$;
$V_{\text{sol'n}} \leftarrow \emptyset;\quad \mathcal{Q}_{\text{Samples}} \leftarrow \emptyset$;
for $i = 1 \dots q$ **do**
  $c_i \leftarrow \min_{\mathbf{v}_{\text{goal}} \in V_{\text{sol'n}}} \{g_{\mathcal{T}}(\mathbf{v}_{\text{goal}})\}$;
  if $\mathcal{Q}_{\text{Samples}} \equiv \emptyset$ **then**
    $\mathcal{Q}_{\text{Samples}} \leftarrow \texttt{Sample}(m, \mathbf{x}_{\text{start}}, X_{\text{goal}}, c_i)$;
  $\mathbf{x}_{\text{rand}} \leftarrow \texttt{PopBestInQueue}(\mathcal{Q}_{\text{Samples}})$;
  $\mathbf{v}_{\text{nearest}} \leftarrow \texttt{Nearest}(V, \mathbf{x}_{\text{rand}})$;
  $\mathbf{x}_{\text{new}} \leftarrow \texttt{Steer}(\mathbf{v}_{\text{nearest}}, \mathbf{x}_{\text{rand}})$;
  if $\texttt{CollisionFree}(\mathbf{v}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
    if $\mathbf{x}_{\text{new}} \in X_{\text{goal}}$ **then**
      $V_{\text{sol'n}} \xleftarrow{+} \{\mathbf{x}_{\text{new}}\}$;
    $V \xleftarrow{+} \{\mathbf{x}_{\text{new}}\}$;
    $V_{\text{near}} \leftarrow \texttt{Near}(V, \mathbf{x}_{\text{new}}, r_{\text{rewire}})$;
    $\mathbf{v}_{\min} \leftarrow \mathbf{v}_{\text{nearest}}$;
    forall $\mathbf{v}_{\text{near}} \in V_{\text{near}}$ **do**
      $c_{\text{new}} \leftarrow g_{\mathcal{T}}(\mathbf{v}_{\text{near}}) + c(\mathbf{v}_{\text{near}}, \mathbf{x}_{\text{new}})$;
      if $c_{\text{new}} < g_{\mathcal{T}}(\mathbf{v}_{\min}) + c(\mathbf{v}_{\min}, \mathbf{x}_{\text{new}})$ **then**
        if $\texttt{CollisionFree}(\mathbf{v}_{\text{near}}, \mathbf{x}_{\text{new}})$ **then**
          $\mathbf{v}_{\min} \leftarrow \mathbf{v}_{\text{near}}$;
    $E \xleftarrow{+} \{(\mathbf{v}_{\min}, \mathbf{x}_{\text{new}})\}$;
    forall $\mathbf{v}_{\text{near}} \in V_{\text{near}}$ **do**
      $c_{\text{near}} \leftarrow g_{\mathcal{T}}(\mathbf{x}_{\text{new}}) + c(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}})$;
      if $c_{\text{near}} < g_{\mathcal{T}}(\mathbf{v}_{\text{near}})$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}})$ **then**
          $\mathbf{v}_{\text{parent}} \leftarrow \texttt{Parent}(\mathbf{v}_{\text{near}})$;
          $E \xleftarrow{-} \{(\mathbf{v}_{\text{parent}}, \mathbf{v}_{\text{near}})\}$;
          $E \xleftarrow{+} \{(\mathbf{x}_{\text{new}}, \mathbf{v}_{\text{near}})\}$;
    $\texttt{Prune}(V, E, c_i)$;
return $\mathcal{T}$;

The function `PopBestInQueue`$(\cdot)$ pops the best element off the front of a queue given its ordering. A goal bias may be implemented in SORRT* by adding a small probability of sampling the goal instead of removing the best sample from the queue. This algorithm is publicly available in OMPL.

SORRT* can be viewed as a simplified version of BIT* that searches the RGG by considering only the best-possible edge to each vertex. Attempting to connect each sample once avoids the computational cost of repeated connection attempts to infeasible samples but makes the search dependent on the sampling order. High-utility samples (e.g., samples near the optimum) may be underutilized depending on the state of the tree when they are found. This can become problematic if these samples have a low sampling probability (e.g., samples in narrow passages). Making multiple connections attempts per sample and retaining samples for multiple batches allows BIT* to exploit these useful samples more than algorithms such as SORRT*. As seen in Section 5.5, this results in different performance especially in high state dimensions.

5.5 Experiments

The benefits of ordering the search of continuous planning problems are demonstrated on simulated problems in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ (Section 5.5.1) and one and two-armed problems for HERB (Section 5.5.2) using OMPL[3]. BIT* is compared to RRT, RRT-Connect, RRT*, FMT*, Informed RRT*, and SORRT*.

All planners used the same tuning parameters and configurations where possible. Planning time was limited to 3 seconds, 150 seconds, and 300 seconds in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ and 5 seconds and 600 seconds for HERB ($\mathbb{R}^7$ and $\mathbb{R}^{14}$), respectively. RRT-style planners used a goal-sampling bias of 5% and a maximum edge length of $\eta = 0.3$, 0.9, and 1.7 on the abstract problems ($\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$) and 0.7 and 1.3 on HERB, respectively. These values were selected experimentally to reduce the time required to find an initial solution on simple training problems.

The RRT* planners, FMT*, and BIT* all used a connection radius equal to 2 times their theoretical minimum (e.g., $r_{\text{RRT}^*} = 2r^*_{\text{RRT}^*}$) and approximated the Lebesgue measure of the free space with the measure of the entire planning problem. The RRT* planners also used the ordered rewiring technique presented in Perez et al. (2011). Informed RRT*, SORRT*, and BIT* used the L^2 norm as estimates of cost-to-come and cost-to-go, direct informed sampling, and delayed pruning the graph until solution cost changed by more than 5%. SORRT* and BIT* both used $m = 100$ samples per batch and BIT* used an approximately sorted queue. BIT* also thresholded its large initial connection radius by using the same radius for both the first and second batches.

[3]The experiments were run on a laptop with 16 GB of RAM and an Intel i7-4810MQ processor. The abstract experiments were run in Ubuntu 12.04 (64-bit) with Boost 1.58, while the HERB experiments were run in Ubuntu 14.04 (64-bit).

5.5.1 Simulated Planning Problems

The algorithms were tested on simulated problems in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ on worlds consisting of many different homotopy classes (Section 5.5.1.1) and randomly generated obstacles (Section 5.5.1.2). The planners were tested with 100 different pseudo-random seeds on each problem and state dimension. The solution cost of each planner was recorded every 1 millisecond by a separate thread and the median was calculated from the 100 trials by interpolating each trial at a period of 1 millisecond. The absence of a solution was considered an infinite cost for the purpose of calculating the median.

5.5.1.1 Worlds with Many Homotopy Classes

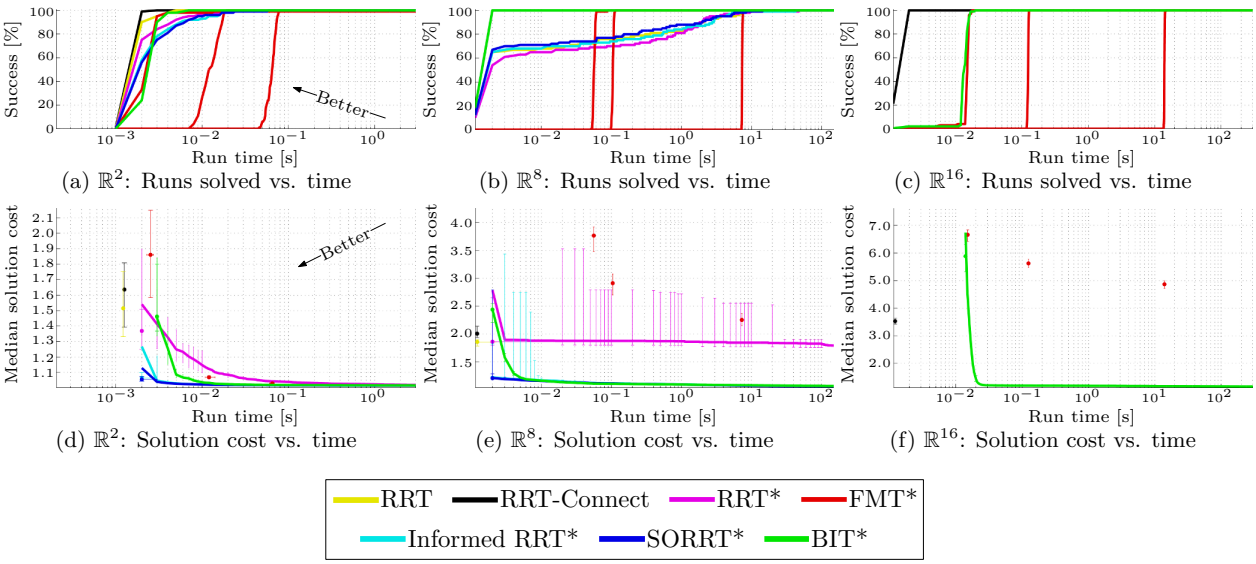
The algorithms were tested on problems with many homotopy classes in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ (Fig. 4.10b). The worlds consisted of a (hyper)cube of width $l = 4$ with the start and goal located at $[-0.5, 0, \ldots, 0]^T$ and $[0.5, 0, \ldots, 0]^T$, respectively. The problem domain was filled with a regular pattern of axis-aligned (hyper)rectangular obstacles with a width such that the start and goal were 5 'columns' apart.

The results are presented in Fig. 5.5 with the percent of trials solved and the median solution cost plotted versus run time. These results demonstrates the advantages and disadvantages of attempting multiple connections per sample. In low dimensions, the main challenge of this planning problem is avoiding obstacles and BIT* can be outperformed by planners that attempt only a single connection per sample (e.g., Informed RRT*). In high dimensions, the exponential increase in planning problem measure (i.e., the curse of dimensionality) makes navigating towards the goal an equal challenge to avoiding obstacles and BIT* outperforms other planners, even SORRT*.

This effect is visible in the relative performance of planners across state dimension. In $\mathbb{R}^2$, BIT* performs roughly the same as Informed RRT* and SORRT*. It is faster to solve 100% of the planning trials (Fig. 5.5a) but improves the median solution cost slower than the two RRT*-based algorithms (Fig. 5.5d). In $\mathbb{R}^8$, BIT* drastically outperforms Informed RRT* and SORRT* in solving all planning trials (Fig. 5.5b) but is slightly outperformed by those two planners in terms of median solution cost (Fig. 5.5e). In $\mathbb{R}^{16}$, the difference is stark and BIT* is the only anytime almost-surely asymptotically optimal planner to always solve the planning problem in the given 300 seconds.

5.5.1.2 Random Worlds

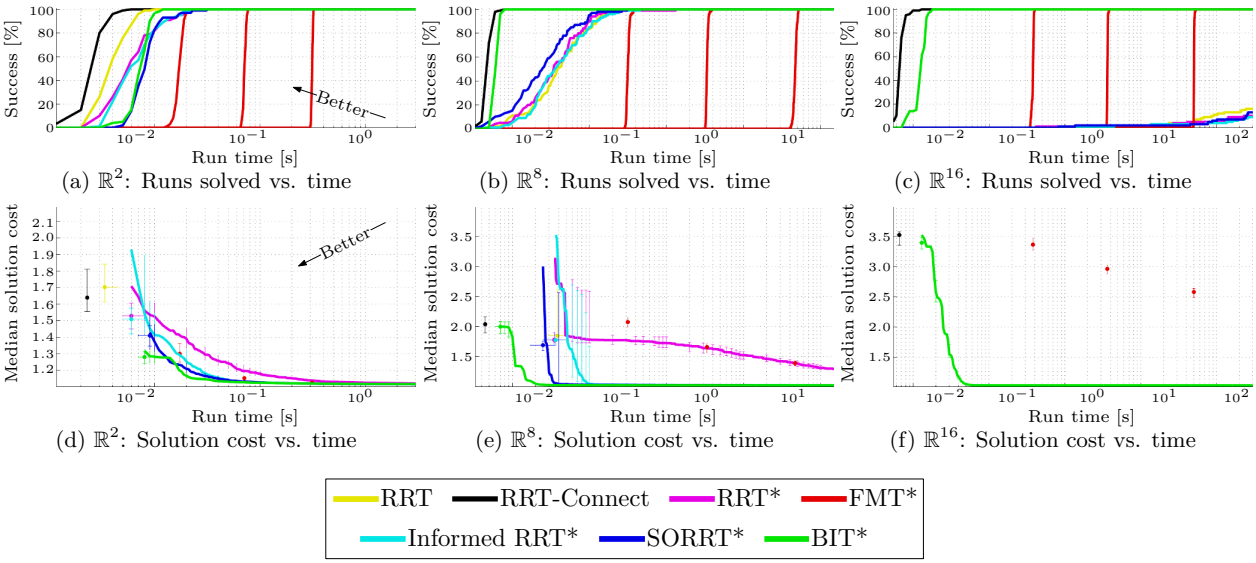
The planners were tested on randomly generated problems in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$. The worlds consisted of a (hyper)cube of width $l = 2$ populated with approximately 75 random axis-aligned (hyper)rectangular obstacles that obstruct at most one third of the environment.



(a) $\mathbb{R}^2$: Runs solved vs. time (b) $\mathbb{R}^8$: Runs solved vs. time (c) $\mathbb{R}^{16}$: Runs solved vs. time

(d) $\mathbb{R}^2$: Solution cost vs. time (e) $\mathbb{R}^8$: Solution cost vs. time (f) $\mathbb{R}^{16}$: Solution cost vs. time

Figure 5.5: Planner performance versus time for the problem illustrated in Fig. 4.10b. Each planner was run 100 different times in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ with $l = 4$ and run times limited to 3, 150, and 300 seconds, respectively. The percentage of trials solved is plotted versus run time for each planner and presented in (a)–(c). The median path length is plotted versus run time for each planner and presented in (d)–(f), with unsuccessful trials assigned infinite cost. The error bars denote a nonparamentric 99% confidence interval on the median. The results show that BIT* increasingly outperforms other almost-surely asymptotically optimal planners as state dimension increases. Note the low success rates of RRT* planners in high dimensions.



(a) $\mathbb{R}^2$: Runs solved vs. time (b) $\mathbb{R}^8$: Runs solved vs. time (c) $\mathbb{R}^{16}$: Runs solved vs. time

(d) $\mathbb{R}^2$: Solution cost vs. time (e) $\mathbb{R}^8$: Solution cost vs. time (f) $\mathbb{R}^{16}$: Solution cost vs. time

Figure 5.6: Planner performance versus time for a randomly generated problem. Each planner was run 100 different times in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ with $l = 2$ and run times limited to 3, 150, and 300 seconds, respectively. The percentage of trials solved is plotted versus run time for each planner and presented in (a)–(c). The median path length is plotted versus run time for each planner and presented in (d)–(f), with unsuccessful trials assigned infinite cost. The error bars denote a nonparamentric 99% confidence interval on the median. The results show that BIT* increasingly outperforms other almost-surely asymptotically optimal planners as state dimension increases. Note the low success rates of RRT* planners in high dimensions.

For each state dimension, 10 different random worlds were generated and the planners were tested on each with 100 different pseudo-random seeds. The true optima for these problems are different and unknown and there is no meaningful way to compare the results across problems. Results from a representative problem are instead presented in Fig 5.6 with the percent of trials solved and the median solution cost plotted versus computational time.

These experiments show that BIT* generally finds better solutions faster than other sampling-based optimal planners and RRT on these types of problems regardless of the state dimension. It has a higher likelihood of having found a solution at a given computational time (Fig. 5.6a–c), and converges faster towards the optimum (Fig. 5.6d–f), with the relative improvement increasing with state dimension. The only tested planner that found solutions faster than BIT* was RRT-Connect, a nonanytime planner that cannot converge to the optimum.

5.5.2 Path Planning for HERB

It is difficult to capture the challenges of actual high-dimensional planning in abstract worlds. Two planning problems inspired by manipulation scenarios were created for HERB, a 14-DOF mobile manipulation platform (Srinivasa et al., 2012).

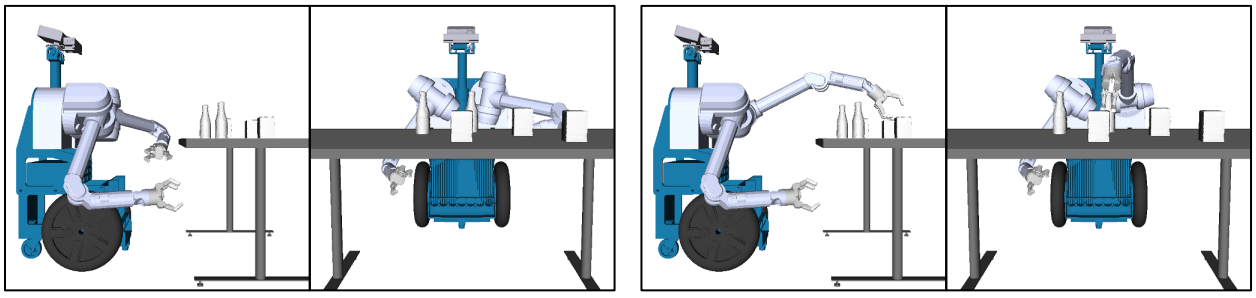
Start and goal poses were chosen for one arm (7 DOFs, Section 5.5.2.1) and two arms (14 DOFs, Section 5.5.2.2) to define planning problems with the objective of minimizing path length through configuration space. They were used to compare RRT, RRT-Connect, Informed RRT*, SORRT*, FMT*, and BIT*. The planners were run on each problem 50 times while recording success rate, initial solution time and cost, and final cost. Trials that did not find a solution were considered to have taken infinite time and have infinite path length, respectively, for the purpose of calculating medians. The number of FMT* samples for both problems was chosen to use the majority of the available computational time.

5.5.2.1 A One-Armed Planning Problem

A planning problem was defined for HERB's left arm around a cluttered table (Fig. 5.7). The arm starts folded at the elbow and held at approximately the level of the table (Figs. 5.7a and 5.7b) and must be moved into position to grasp a box (Figs. 5.7c and 5.7d). The planners were given 5 seconds of computational time to solve this 7-DOF problem with the objective of minimizing path length in configuration space. FMT* used $m = 30$ samples.

The percentage of trials that successfully found a solution (Fig. 5.8a), the median time and cost of the initial solution (Figs. 5.8b and 5.8c) and the final cost (Fig. 5.8d) were plotted for each planner. Infinite values are not plotted.

The results show BIT* finds solutions more often than other almost-sure asymptotically optimal planners on this problem (Fig. 5.8a) and also finds better solutions faster than all tested algorithms, including RRT-Connect (Figs. 5.8b–5.8d). Fig. 5.9 presents a composite photograph of HERB executing a path found by BIT* for a similar problem.



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

Figure 5.7: A one-armed motion planning problem for HERB in $\mathbb{R}^7$. Starting at a position level with the table, (a) and (b), HERB's left arm must be moved in preparation for grasping a box on the far side of the table, (c) and (d).

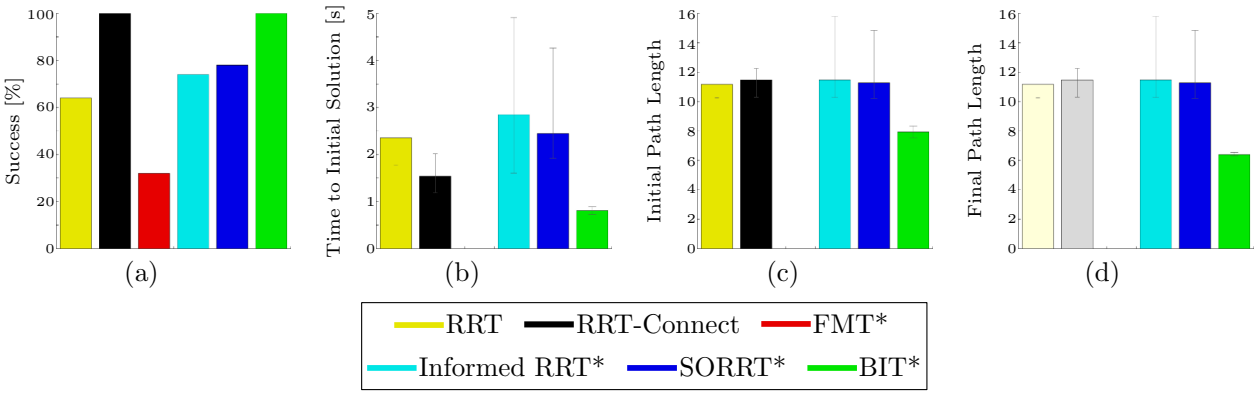


Figure 5.8: Results from 50, 5 second trials on the one-armed HERB planning problem shown in Fig. 5.7. The percent of solutions solved, (a), the median time to an initial solution, (b), the median initial path length, (c), and the median final path length, (d), are presented with 99% confidence intervals for each planner. Unsuccessful trials were assigned infinite time and cost. The inability of nonanytime planners (e.g., RRT, RRT-Connect, and FMT*) to use the remaining available time to improve their initial solution is denoted with diminished colour in (d), where present. BIT* is the only almost-surely asymptotically optimal planner to solve all 50 trials and does so in a time comparable to RRT-Connect. It also finds significantly lower cost paths then all the other planners.

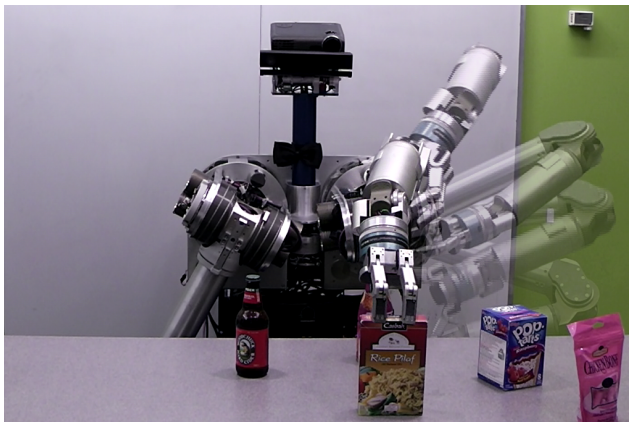
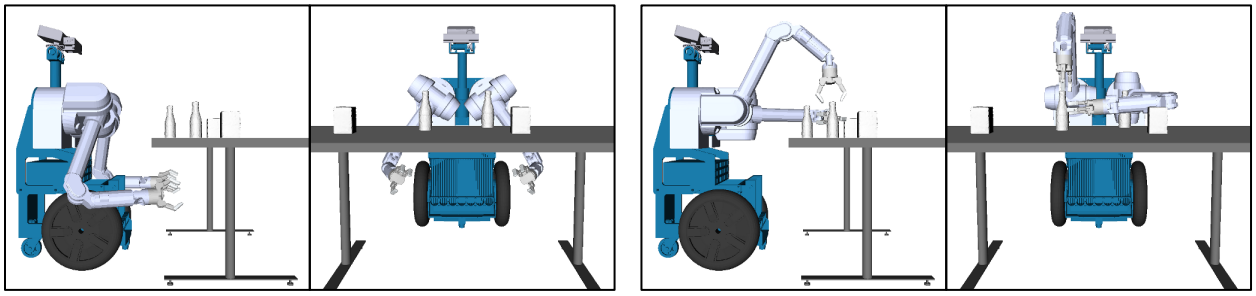


Figure 5.9: A composite figure of HERB executing a path found by BIT* on a one-armed planning problem similar to Fig. 5.7.



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

Figure 5.10: A two-armed motion planning problem for HERB in $\mathbb{R}^{14}$. Starting under the table, (a) and (b), HERB's arms must be moved in preparation for opening a bottle, (c) and (d).

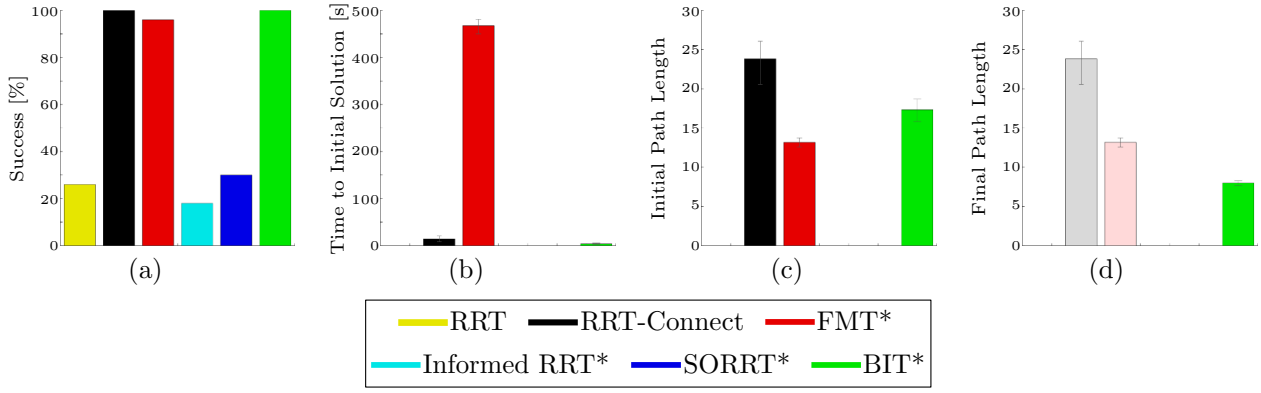


Figure 5.11: Results from 50, 600 second trials on the two-armed HERB planning problem shown in Fig. 5.10. The percent of solutions solved, (a), the median time to an initial solution, (b), the median initial path length, (c), and the median final path length, (d), are presented with 99% confidence intervals for each planner. Unsuccessful trials were assigned infinite time and cost. The inability of nonanytime planners (e.g., RRT, RRT-Connect, and FMT*) to use the remaining available time to improve their initial solution is denoted with diminished colour in (d), when present. BIT* is the only anytime planner to solve all 50 trials and does so in a time comparable to RRT-Connect. It focuses the search to the informed set once a solution is found and the resulting increasingly dense RGG allows it to find lower cost paths then all the other planners.

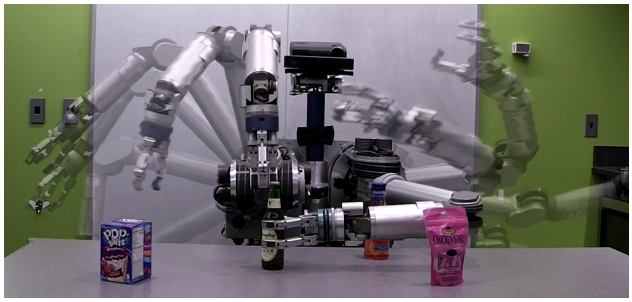


Figure 5.12: A composite figure of HERB executing a path found by BIT* on a two-armed planning problem similar to Fig. 5.10.

5.5.2.2 A Two-Armed Planning Problem

A second planning problem was defined for both of HERB's arms moving around a cluttered table (Fig. 5.10). The arms start at a neutral position with their forearms extended under the table (Figs. 5.10a and 5.10b) and must be moved into position to open a bottle (Figs. 5.10c and 5.10d). The planners were given 600 seconds of computational time to solve this 14-DOF problem with the objective of minimizing path length in configuration space. FMT* used $m = 3000$ samples.

The percentage of trials that successfully found a solution (Fig. 5.11a), the median time and cost of the initial solution (Figs. 5.11b and 5.11c) and the final cost (Fig. 5.11d) were plotted for each planner. Infinite values are not plotted.

The results show that even when more computational time is available, BIT* still finds solutions more often than other almost-surely asymptotically optimal planners (Fig. 5.11a) and also finds initial solutions faster than all tested algorithms, including RRT-Connect (Fig. 5.11b). As a nonanytime algorithm, FMT* is tuned to use the majority of the available time and finds a better initial solution (Fig. 5.11c) but as BIT* is able to improve its solution it still finds a better final path after approximately the same amount of time (Fig. 5.11d). Fig. 5.12 presents a composite photograph of HERB executing a path found by BIT* for a similar problem.

5.6 Discussion

Most anytime sampling-based planners, such as RRT*, solve planning problems by simultaneously searching the entire problem domain. This is inefficient and can become prohibitively expensive in large planning problems or high state dimensions. Informed graph-based search techniques, such as A*, avoid this unnecessary computational effort by ordering their search by potential solution quality. This only considers states when they could provide the best solution and avoids the computational costs of searching large regions of the problem domain. This solves planning problems by searching the informed set that will be defined by their eventual solution (the *future informed set*).

Existing methods to include heuristic ordering in sampling-based planning are insufficient (Section 5.1). They either partially apply heuristics (e.g., RRT#), sacrifice anytime resolution (e.g., RA* and FMT*), or waste computational effort on states that are unnecessary to find the solution by ordering the search on metrics other than solution cost and expanding outside the future informed set (e.g., SBA*).

BIT* is an informed anytime sampling-based planner that is ordered only by potential solution quality (Algs. 5.1–5.3; Section 5.2). It applies informed graph-based search techniques (i.e., LPA*) directly to continuous planning problems without requiring *a priori* approxi-

mations of the search domain. Instead, it alternately builds and searches a sampling-based approximation (i.e., a RGG) that increases in accuracy indefinitely with additional computational time. This avoids the computational costs associated with both choosing an insufficient approximation (graph-based search) and performing an unordered search (sampling-based planning). It is both probabilistically complete and almost-surely asymptotically optimal (Theorems 5.1 and 5.2) and can be stopped at any time if a suitable solution is found.

BIT* improves its approximation by adding batches of new samples into the existing RGG and uses incremental search techniques to update its search efficiently by reusing information. This is accomplished by using heuristics in all aspects of the search. Estimates of solution cost are used to order and focus the search, estimates of cost-to-come are used to account for future graph improvements, and estimates of edge costs are used to avoid unnecessary collision checks and boundary-value problems. Delayed edge cost calculations can also be found in *lazy* versions of both graph-based searches and sampling-based planners (Bohlin and Kavraki, 2000; Branicky et al., 2001; Cohen et al., 2014b; Hauser, 2015; Helmert, 2006).

A brief set of extensions to BIT* are presented Algs. 5.4–5.5 (Section 5.4). These include prioritizing an initial solution, avoiding the need to define *a priori* search limits in unbounded problems, and avoiding unreachable areas of the problem domain. These ideas also motivate the development of SORRT* as an extension of batch-ordered search to the algorithmic simplicity of RRT* (Alg. 5.6).

The benefits of BIT* are demonstrated experimentally on abstract planning problems and simulated experiments for HERB (Section 5.5). The results highlight the advantages of both using an ordered search and considering multiple connections per sample. BIT* is more likely to have found a solution at any given time on the tested problems and generally finds better solutions faster than the other almost-surely asymptotically optimal planners, especially in high state dimensions.

The experiments also highlight the relative sensitivity of existing anytime planners to tuning parameters. The performance of RRT-style planners depends heavily on the maximum edge length, η, and achieving the best performance requires tuning it for the problem size, dimension, and even obstacle characteristics. Alternatively, the performance of BIT* on the tested problems did not vary strongly with different choices for its main tuning parameter, batch size.

Avoiding unnecessary edge evaluations allows BIT* to spend more computational effort on edges. It is satisfying to note that this already being exploited in the literature. Xie et al. (2015) show that a two-point BVP solver can be used to calculate edges for BIT* for problems with differential constraints. They find that doing so is competitive to state-of-the-art optimal sampling-based techniques that are explicitly designed to avoid solving two-point BVPs. Choudhury et al. (2016) show that a path optimizer (i.e., CHOMP; Zucker et al., 2013) can be used on potential edges in BIT*. This provides a method to exploit local problem information (i.e., cost gradients) to propose higher-quality edges and improve performance.

BIT* has also been used in the literature as a part of multistage planning algorithms. Lan et al. (2016) use it in their two-stage online planning algorithm for micro UAVs. They convert the collision-free path found by BIT* into a dynamically feasible path by solving a series of two-point BVPs. This allows them to avoid the cost of considering differential constraints during the initial search.

This chapter shows the benefits of unifying informed graph-based search and sampling-based planning. Using incremental search techniques to efficiently search an increasingly dense RGG allows BIT* to outperform existing anytime almost-surely asymptotically optimal planners. These results will hopefully motivate further research into combining graph-based search and sampling-based planning. Of particular interest would be probabilistic statements about search efficiency analogous to the formal statements for A*. There is also a clear opportunity to consider different sampling-based approximations and to use more advanced graph-based-search techniques, such as ARA* and MHA*, to further accelerate the search performance of BIT*.

BIT* is described as using LPA* to efficiently search an incrementally built (i.e., changing) RGG embedded in a continuous planning problem. It is important to note a key difference in how these two algorithms reuse information. When LPA* updates the cost-to-come of a vertex it reconsiders the cost-to-come of all possibly descendent vertices. This can be prohibitively expensive in large graphs and the results of RRT* demonstrate that this propagation is unnecessary for a planner to almost-surely converge asymptotically to the optimum. By not propagating these changes, BIT* is more similar to TLPA*.

In summary, this chapter presents the following specific contributions:

- Reviews existing methods to combine informed search and sampling-based representations and shows that they either provide incomplete/inefficient ordering or sacrifice anytime performance (Section 5.1).
- Develops BIT* as a unification of informed graph-based search and sampling-based planning that is almost-surely asymptotically optimal (Algs. 5.1–5.3).
- Presents extensions to BIT* (Section 5.4), including JIT sampling to allow for the direct search of unbounded problems (Alg. 5.5) and SORRT* (Alg. 5.6) to apply ordered search concepts directly to RRT*.
- Demonstrates experimentally the benefits of using ordered search to combat the curse of dimensionality (Section 5.5).

Chapter 6

Conclusion

Autonomous robotic systems operating in dynamic and uncontrolled environments require path planning algorithms. These algorithms must be able to reliably find high-quality solutions to given problems in a reasonable amount of computational time. This is a nontrivial task as the problem domains are often continuously valued, high dimensional, and/or poorly bounded. The majority of existing techniques simplify this problem by approximating the search domain either with an *a priori* graph or through sampling. Both of these approaches are successful but each have important limitations.

A priori graphs approximate a problem before it is searched. Doing so 'correctly' is challenging since the relationship between resolution and search performance depends on the specific features of a planning problem (e.g., size and arrangement of obstacles). If the chosen approximation is insufficient (e.g., a sparse graph) it may preclude finding a (suitable) solution but if it is excessive (e.g., a dense graph) it may make finding a solution prohibitively expensive.

Graph-based searches are effective planning techniques despite these limitations. This is because informed algorithms, such as A*, use heuristics to order their search by potential solution quality. This not only finds the optimal solution to the given representation (i.e., it is *resolution optimal*) but does so by expanding the minimum number of vertices possible using the chosen heuristic (i.e., it is *optimally efficient*; Hart et al., 1968).

Sampling-based planners alternatively often build anytime approximations. This avoids the need to select a resolution *a priori* and allows them to be run indefinitely until a (suitable) solution is found. RRT* has a unity probability of finding a solution, if one exists, with an infinite number of samples (i.e., it is *probabilistically complete*) and finds continuously improving solutions (i.e., it is *almost-surely asymptotically optimal*). This has made it an effective planning technique despite the fact its search is inefficient. Incremental sampling-based planners tend to explore the entire problem domain equally (i.e., they are *space filling*). This wastes computational effort on regions of the problem domain that are not needed to find the solution and can be prohibitively expensive in large planning problems and/or high state dimensions.

This thesis demonstrates that better planning algorithms can be designed for continuous planning problems by unifying the informed graph-based search and sampling-based planning approaches. Chapter 2 discusses how previous work to do so has been incomplete and resulted in either anytime algorithms that are only partially ordered by solution quality or informed algorithms with reduced anytime performance. These tradeoffs are unnecessary and this thesis presents methods to include heuristics directly in anytime sampling-based planners, including to focus the search domain and order the search.

Chapter 3 analyzes the necessary conditions for incremental sampling-based single-query planners, such as RRT*, to improve a solution. These take the form of a subset of the planning problem that contains all better solutions (the omniscient set; Definition 3.1) as well as heuristic-based approximations (informed sets; Definition 3.2) quantified by their *precision* and *recall* (Definitions 3.3 and 3.4). It is shown that the probability of sampling these sets provides an upper bound on the probability of improving a solution (Lemmas 3.1, 3.2, 3.4, and 3.5 and Theorems 3.3 and 3.6). This motivates using heuristics to focus the search of incremental sampling-based planners to informed sets once a solution is found.

A common admissible heuristic for problems seeking to minimize path length is the L^2 norm (Ferguson and Stentz, 2006; Otte and Correll, 2013). The resulting informed set always contains the complete omniscient set and is exactly equal to it in the absence of obstacles (i.e., it is *sharp*). Analysis of its shape shows that common configurations of sampling-based planners suffer from a *minimum-path-length curse of dimensionality* that sees the likelihood of improving a solution decrease factorially (i.e., faster than exponentially) with state dimension (Theorem 3.7). This limitation is not addressed by existing methods to focus sampling-based search but can be avoided by directly sampling the L^2 informed set (Algs. 3.3–3.6).

Chapter 4 demonstrates how informed sets can be used to improve the rate at which incremental sampling-based planners improve solutions. Informed RRT* uses admissible graph pruning and informed sampling to restrict search to the set of states believed to be capable of providing improvements once a solution is found (i.e., the informed set). The shape of this informed set when seeking to minimize path length provides sharp analytically bounds on the expected cost of RRT* at any iteration (Lemma 4.1). It shows that while the original RRT* has provably sublinear convergence towards the optimum, informed techniques can have linear convergence (Theorems 4.2–4.4).

The practical effect of improved convergence is demonstrated on simulated problems in abstract worlds and for HERB, a 14-DOF mobile manipulation platform (Srinivasa et al., 2012). Informed RRT* is shown to outperform other methods to focus RRT*, especially in high dimensions and large planning problems.

The ideas of Chapters 3 and 4 were originally presented at the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) and are currently in preparation for submission as a journal paper. The associated code has been made publicly available in OMPL (Şucan et al., 2012).

In chronological order, the associated publications are

- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014b). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2997–3004, Chicago, Illinois, USA
- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014c). On recursive random prolate hyperspheroids. Technical Report TR-2014-JDG002, Autonomous Space Robotics Lab, University of Toronto. arXiv:1403.7664 [math.ST]
- Gammell, J. D. and Barfoot, T. D. (2014). The probability density function of a transformation-based hyperellipsoid sampling technique. Technical Report TR-2014-JDG004, Autonomous Space Robotics Lab, University of Toronto. arXiv:1404.1347 [math.ST]
- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2017b). Informed sampling for asymptotically optimal path planning. In preparation

The ideas in these publications have been extended independently. Specifically, Kunz et al. (2016) develop a hierarchical rejection sampling method for systems with differential constraints whose informed sets cannot be expressed in closed form. Kim and Song (2015) extend Informed RRT* by using a local optimizer to further accelerate convergence by rapidly improving solutions and shrinking the informed set. Lee et al. (2016) improve convergence towards the optimum of the current homotopy class for online planning scenarios by including path biasing in Informed RRT*. Burget et al. (2016) extend Informed RRT* to a bidirectional search in their BI^{2}RRT* algorithm. Primatesta et al. (2016) adapt Informed RRT* for efficient replanning in environments with dynamic obstacles. Oleynikova et al. (2016) and Bevilacqua et al. (2016) use Informed RRT* to find topologically feasible initial paths in their multistage planning algorithms for UAVs and service robots, respectively.

Chapter 5 extends the use of heuristics to all aspects of anytime sampling-based planning, including ordering the search by potential solution quality. This is accomplished in BIT* by alternately building and searching a sampling-based approximation of the continuous planning problem (i.e., a RGG). RGG theory is used to limit the complexity of this approximation while maintaining probabilistic bounds on its accuracy. Incremental graph-search techniques, such as LPA*, are used to search this changing approximation efficiently in order of potential solution quality while reusing information. This effectively separates the sampling-based approximation of a continuous problem domain from its search and challenges the traditional planning dichotomy of graph-based and sampling-based techniques (Fig. 2.2).

BIT* uses heuristics in all aspects of the search to reduce computational cost and prioritize searching for high-quality solutions (i.e., it performs an ordered lazy search). The result is an algorithm that demonstrates the efficient search nature of graph-based algorithms, such as A*, and the anytime almost-sure asymptotic optimality of sampling-based algorithms, such

as RRT*. Experiments on abstract worlds and problems for HERB show that BIT* scales more effectively to high state dimensions than other almost-surely asymptotically optimal planners while demonstrating less sensitivity to tuning parameters.

The ideas of Chapter 5 were originally presented at the Information-based Grasp and Manipulation Planning Workshop at the 2014 Robotics: Science and Systems Conference (RSS) and the 2015 IEEE International Conference on Robotics and Automation (ICRA). They are currently being prepared as a submission as a journal paper. The associated code has been made publicly available in OMPL.

In chronological order, the associated publications are

- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014a). BIT*: Sampling-based optimal planning via batch informed trees. In *The Information-based Grasp and Manipulation Planning Workshop, Robotics: Science and Systems (RSS)*, Berkeley, CA, USA
- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, Seattle, Washington, USA
- Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2017a). Informed asymptotically optimal anytime search. In preparation

The ideas in these publications have been extended both independently and in collaboration. Xie et al. (2015) use a two-point boundary-value problem (two-point BVP) solver to calculate edges in BIT* for systems with differential constraints. The results prove to be equivalent to state-of-the-art optimal techniques designed to avoid such calculations. Regionally Accelerated BIT* (RABIT*; Choudhury et al., 2016) uses a path optimizer to exploit local information (i.e., cost gradients) and propose high-quality edges that have been optimized around obstacles. Both of these works use BIT*'s ordered lazy search to support spending more computational effort on individual edges.

BIT* is also being used in the robotics literature. Lan et al. (2016) use BIT* in their multistage online planning algorithm for micro UAVs. They find an initial collision-free path with BIT* and then solve a series of two-point BVPs to make it dynamically feasible. It is has also been reported that the CMU AIRLab are successfully using the OMPL implementation of BIT* on their full-sized autonomous helicopter in a planner ensemble (Choudhury et al., 2014).

This thesis investigates unifying informed graph-based search and anytime sampling-based planning. The success of this approach will hopefully motivate further research into unifying these two approaches. There exists clear opportunities to both improve practical performance with more advanced approximations and graph-search techniques as well as make theoretical statements about the efficiency of the resulting search of the continuous planning problem.

In summary, this thesis makes the following specific contributions:

- Analyzes the necessary conditions for incremental sampling-based planners to improve a solution, formally defining the omniscient and informed sets and identifying that problems seeking to minimize path length suffer from a factorial curse of dimensionality (Chapter 3).
- Develops a direct informed sampling technique for problems seeking to minimize path length to a goal or a countable set of goals (Chapter 3).
- Derives bounds for the expected next-iteration cost of RRT* on problems seeking to minimize path length and shows that while it has sublinear convergence, focused variants can have linear convergence (Chapter 4).
- Develops Informed RRT* and shows that it outperforms other methods to focus RRT* (Chapter 4).
- Demonstrates that informed graph-based search and sampling-based planning can be unified to solve continuous planning problems efficiently while still maintaining anytime performance (Chapter 5).
- Develops BIT* and associated extensions including SORRT* and shows that BIT* outperforms other almost-sure asymptotically optimal sampling-based planners (Chapter 5).
- Provides public OMPL implementations of Informed RRT*, SORRT*, and BIT* (More information available at `http://asrl.utias.utoronto.ca/code`).

Appendix A

Proofs of Lemmas 3.1 and 3.2

This section restates and proves Lemmas 3.1 and 3.2.

A.1 Proof of Lemma 3.1

Lemma 3.1 (The necessity of adding states in the omniscient set)**.** *Adding a state from the omniscient set, $\mathbf{x}_{\text{new}} \in X_f$, is a necessary condition for RRT* to improve the current solution, c_i,*

$$c_{i+1} < c_i \implies \mathbf{x}_{\text{new}} \in X_f.$$

This condition is not sufficient *to improve the solution as the ability of states in X_f to provide better solutions at any iteration depends on the optimality of the tree.*

Proof. At the end of iteration $i+1$ the cost of the best solution found by RRT* will be the minimum of the previous best solution, c_i, and the best cost of any new or newly improved solutions, c_{new},

$$c_{i+1} = \min\left\{c_i, c_{\text{new}}\right\}. \tag{A.1}$$

Each iteration of RRT* only adds connections to or from the newly added state, $\mathbf{x}_{\text{new}}$, and therefore all new or modified paths pass through this new state. If any of these new paths extend to the goal region then their solution cost will be bounded by the optimal solution of a path through $\mathbf{x}_{\text{new}}$,

$$c_{\text{new}} \geq f\left(\mathbf{x}_{\text{new}}\right). \tag{A.2}$$

Lemma 3.1 is now proven by contradiction. Assume that after iteration i RRT* has a solution with cost c_i, which is improved at iteration $i+1$ by adding a state *not* in the omniscient set, $c_{i+1} < c_i$, $\mathbf{x}_{\text{new}} \notin X_f$. Then by (3.6), the cost of solutions through $\mathbf{x}_{\text{new}}$ is

bounded from below by the current solution,

$$f\left(\mathbf{x}_{\text{new}} \notin X_f\right) \geq c_i,$$

which by (A.2) is also a bound on the cost of any new or modified solutions,

$$c_{\text{new}} \geq f\left(\mathbf{x}_{\text{new}} \notin X_f\right) \geq c_i.$$

By (A.1), the cost of the best solution found by RRT* at the end of iteration $i+1$ must therefore be c_i. This contradicts the assumption that the solution was improved by a state not in the omniscient set and proves Lemma 3.1. □

A.2 Proof of Lemma 3.2

Lemma 3.2 (The necessity of sampling states in the omniscient set)**.** *Sampling the omniscient set, $\mathbf{x}_{\text{rand}} \in X_f$, is a necessary condition for RRT* to improve the current solution, c_i, after an initial κ iterations,*

$$\forall i \geq \kappa,\, c_{i+1} < c_i \implies \mathbf{x}_{\text{rand}} \in X_f,$$

for any sample distribution that maintains a nonzero probability over the entire omniscient set.

Proof. For simplicity Lemma 3.2 is proven for geometric planning in the absence of kinodynamic constraints but the proof can be extended to kinodynamic planning by adding appropriate assumptions. In RRT (and RRT*) the distribution of vertices in the graph approaches the sample distribution as the number of iterations approach infinity (LaValle, 1998). All regions of the problem domain with a nonzero sampling probability will therefore be sampled in the limit and the number of vertices in these regions will increase indefinitely with the number of iterations. This ever increasing number of vertices means that the worst-case distance between any state in a sampled subset and the nearest vertex in the graph will decrease indefinitely and monotonically.

Lemma 3.2 is now proven by contradiction. Assume that by iteration κ there are a sufficient number of vertices that all possible states in X_f are no further than η from a vertex,

$$\forall \mathbf{x} \in X_f,\, \min_{\mathbf{v} \in V} \left\{\|\mathbf{v} - \mathbf{x}\|_2\right\} < \eta, \tag{A.3}$$

and that after iteration $i \geq \kappa$, RRT* has a solution with cost c_i. Now assume that at iteration $i+1$ RRT* improves the solution *without sampling* the omniscient set, $c_{i+1} < c_i$, $\mathbf{x}_{\text{rand}} \notin X_f$.

As improving a solution requires *adding* a state that belongs to this subset, $\mathbf{x}_{\text{new}} \in X_f$, (Lemma 3.1) this implies that the state added to the graph is not the randomly sampled

state, $\mathbf{x}_{\text{new}} \neq \mathbf{x}_{\text{rand}}$. These two states are related through the `Steer` function, which finds a state as near to $\mathbf{x}_{\text{rand}}$ as possible that is no further than η from the nearest vertex in the tree,

$$\mathbf{x}_{\text{new}} \coloneqq \underset{\mathbf{x}\in X}{\arg\min} \left\{ \|\mathbf{x}_{\text{rand}} - \mathbf{x}\|_2 \;\middle|\; \|\mathbf{x} - \mathbf{v}_{\text{nearest}}\|_2 \leq \eta \right\}, \tag{3.2 redux}$$

where

$$\mathbf{v}_{\text{nearest}} \coloneqq \underset{\mathbf{v}\in V}{\arg\min} \left\{ \|\mathbf{v} - \mathbf{x}_{\text{rand}}\|_2 \right\}. \tag{3.1 redux}$$

In geometric planning, the triangle inequality implies that the nearest vertex to the sample is also the nearest vertex to the proposed new state,

$$\mathbf{v}_{\text{nearest}} \equiv \underset{\mathbf{v}\in V}{\arg\min} \left\{ \|\mathbf{v} - \mathbf{x}_{\text{new}}\|_2 \right\},$$

which from (A.3) is bounded in its distance from $\mathbf{x}_{\text{new}}$ by

$$\|\mathbf{v}_{\text{nearest}} - \mathbf{x}_{\text{new}}\|_2 < \eta. \tag{A.4}$$

The relationship in (A.4) is only satisfied in geometric planning when $\mathbf{x}_{\text{new}} \equiv \mathbf{x}_{\text{rand}}$ due to (3.2). As by assumption the random sample is not a member of the omniscient set, $\mathbf{x}_{\text{rand}} \notin X_f$, then therefore neither is the newly added state, $\mathbf{x}_{\text{new}} \notin X_f$, and by Lemma 3.1 the solution is not improved, $c_{i+1} = c_i$. This contradicts the assumption that the solution was improved by sampling a state not in the omniscient set and proves Lemma 3.2. □

Appendix B

Proof of Lemma 3.8

This section restates Lemma 3.8 as presented by Sun and Farooq (2002) along with a full proof as presented in Gammell and Barfoot (2014).

Lemma 3.8 (The uniform distribution of samples transformed into a hyperellipsoid from a unit n-ball. Originally Lemma 1 in Sun and Farooq, 2002)**.** *If the random points distributed in a hyperellipsoid are generated from the random points uniformly distributed in a hypersphere through a linear invertible nonorthogonal transformation, then the random points distributed in the hyperellipsoid are also uniformly distributed.*

Proof. Let the sets $X_{\text{ball}} \subset \mathbb{R}^n$ and $X_{\text{ellipse}} \subset \mathbb{R}^n$ be the unit n-dimensional ball and a n-dimensional hyperellipsoid with radii $\{r_j\}$, respectively, having measures of

$$\begin{aligned} \lambda\left(X_{\text{ball}}\right) &= \zeta_n, \\ \lambda\left(X_{\text{ellipse}}\right) &= \zeta_n \prod_{j=1}^{n} r_j. \end{aligned}$$

Let $p_{\text{ball}}\left(\cdot\right)$ be the probability density function of samples drawn uniformly from the unit n-ball such that,

$$p_{\text{ball}}\left(\mathbf{x}\right) \coloneqq \begin{cases} \dfrac{1}{\zeta_n}, & \forall \mathbf{x} \in X_{\text{ball}} \\ 0, & \text{otherwise.} \end{cases} \tag{B.1}$$

Let $\tau\left(\cdot\right)$ be an invertible transformation from the unit n-ball to a hyperellipsoid such that,

$$\begin{aligned} \tau &: \ X_{\text{ball}} \to X_{\text{ellipse}}, \\ \tau^{-1} &: \ X_{\text{ellipse}} \to X_{\text{ball}}. \end{aligned}$$

By definition, the probability density function in the hyperellipsoid, $p_{\text{ellipse}}(\cdot)$, resulting from applying this transformation to samples in the unit n-ball is then

$$p_{\text{ellipse}}(\mathbf{x}) \coloneqq p_{\text{ball}}\left(\tau^{-1}(\mathbf{x})\right)\left|\det\left(\left.\frac{d\tau^{-1}}{d\mathbf{x}_{\text{ellipse}}}\right|_{\mathbf{x}}\right)\right|. \tag{B.2}$$

The proposed transformation in (3.14) is,

$$\mathbf{x}_{\text{ellipse}} = \mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}, \tag{3.14 redux}$$

which has the inverse

$$\tau^{-1}(\mathbf{x}_{\text{ellipse}}) = \mathbf{L}^{-1}\left(\mathbf{x}_{\text{ellipse}} - \mathbf{x}_{\text{centre}}\right),$$

and the Jacobian

$$\frac{d\tau^{-1}}{d\mathbf{x}_{\text{ellipse}}} = \mathbf{L}^{-1}. \tag{B.3}$$

Substituting (B.3) and (B.1) into (B.2) gives,

$$p_{\text{ellipse}}(\mathbf{x}) \coloneqq \begin{cases} \dfrac{1}{\zeta_n}\left|\det\left(\mathbf{L}^{-1}\right)\right|, & \forall \mathbf{x} \in X_{\text{ellipse}} \\ 0, & \text{otherwise,} \end{cases} \tag{B.4}$$

using the fact that $\tau^{-1}(\mathbf{x}) \in X_{\text{ball}} \iff \mathbf{x} \in X_{\text{ellipse}}$. As $p_{\text{ellipse}}(\cdot)$ is constant for all $\mathbf{x}_{\text{ellipse}} \in X_{\text{ellipse}}$, this proves that when used to transform uniformly distributed samples in the unit n-ball (3.14) results in a uniform distribution over the hyperellipsoid and proves Lemma 3.8.

For hyperellipsoids whose axes are orthogonal (e.g., a prolate hyperspheroid), (B.4) can be expressed in a more familiar and intuitive form. Using the orthogonality of rotation matrices and (3.15),

$$\mathbf{x}_{\text{ellipse}} = \mathbf{C}_{\text{we}}\mathbf{L}'\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}, \tag{3.15 redux}$$

as $\tau(\cdot)$, makes (B.4)

$$p_{\text{ellipse}}(\mathbf{x}) \coloneqq \begin{cases} \dfrac{1}{\zeta_n}\left|\det\left(\mathbf{L}'^{-1}\mathbf{C}_{\text{we}}^{T}\right)\right|, & \forall \mathbf{x} \in X_{\text{ellipse}} \\ 0, & \text{otherwise.} \end{cases} \tag{B.5}$$

The diagonal hyperellipsoid matrix gives,

$$\mathbf{L}' = \operatorname{diag}(r_1, r_2, \ldots, r_n),$$

which makes (B.5)

$$p_{\text{ellipse}}(\mathbf{x}) \coloneqq \begin{cases} \dfrac{1}{\zeta_n \prod_{j=1}^{n} r_j}, & \forall \mathbf{x} \in X_{\text{ellipse}} \\ 0, & \text{otherwise,} \end{cases} \tag{B.6}$$

since the determinant is a linear operator, all rotation matrices have a unity determinant, $\det(\mathbf{C}_{\text{we}}) = 1$, and the determinant of a diagonal matrix is the product of its diagonal entries. As expected, (B.6) is the inverse of the volume of an n-dimensional hyperellipsoid with radii $\{r_j\}$. □

Appendix C

Proof of Lemma 4.1

This section restates and proves Lemma 4.1, which is used in support of Theorems 4.2–4.4. An earlier version of this proof appeared in Gammell et al. (2014c).

Lemma 4.1 (Expected next-iteration cost in geometric planning)**.** *The expected value of the next solution, $E[c_{i+1}]$, is bounded for geometric planning by*

$$p_f \frac{n c_i^2 + c_{\min}^2}{(n+1)\, c_i} + (1 - p_f)\, c_i \leq E[c_{i+1}] \leq c_i, \tag{4.3 redux}$$

where c_i is the current solution cost, $c_{\min}$ is the theoretical minimum solution cost, n is the state dimension of the planning problem, and $p_f = P(\mathbf{x}_{\text{new}} \in X_f)$ is the probability of adding a state that is a member of the omniscient set (i.e., that can belong to a better solution). While not explicitly shown, the subset, X_f, and the probability of improving the solution, p_f, generally are functions of the current solution cost.

The lower bound is sharp over the set of all possible planning problems and algorithm configurations and occurs for versions of RRT using an infinite rewiring radius (i.e., $\eta = \infty$, and $r_{\text{RRT}^*} = \infty$) in an obstacle-free environment.*

Proof. Proof of the upper bound is trivial. RRT* only accepts new solutions if they improve its existing solution, assuring that the cost monotonically decreases,

$$c_{i+1} \leq c_i. \tag{C.1}$$

Proof of the lower bound comes from finding an exact expression for the expected value of the solution cost found by using an infinite rewiring neighbourhood in the absence of obstacles.

The expected solution cost of RRT* depends on the probability of sampling the omniscient set,

$$E[c_{i+1}] = p_f E[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f] + (1 - p_f)\, E[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \notin X_f], \tag{C.2}$$

where $p_f = P(\mathbf{x}_{\text{new}} \in X_f)$. Adding a state from the omniscient set, X_f, is a necessary condition to improve the solution (Lemma 3.1) and any other state will not change the solution cost, $E[c_{i+1} \mid \mathbf{x}_{\text{new}} \notin X_f] = c_i$. This simplifies (C.2) to

$$E[c_{i+1}] = p_f E[c_{i+1} \mid \mathbf{x}_{\text{new}} \in X_f] + (1 - p_f)\, c_i. \tag{C.3}$$

Solutions found by adding states inside the omniscient are bounded from below by the optimal path through the state,

$$E[c_{i+1} \mid \mathbf{x}_{\text{new}} \in X_f] \geq E[f(\mathbf{x}_{\text{new}}) \mid \mathbf{x}_{\text{new}} \in X_f], \tag{C.4}$$

where $f(\mathbf{x})$ is the cost of the optimal path from the start to the goal constrained to pass through a state, $\mathbf{x}$. With a uniform sample distribution over X_f the right-hand side of (C.4) becomes

$$E[f(\mathbf{x}_{\text{new}}) \mid \mathbf{x}_{\text{new}} \in X_f] = \frac{1}{\lambda(X_f)} \int_{X_f} f(\mathbf{x}_{\text{new}})\, dV.$$

When RRT* uses an infinite rewiring radius it attempts connections between every new state and the start and goal. In the absence of obstacles these paths will be feasible and represent the optimal solutions using the state. This makes the expected value of this best-case configuration of RRT* equivalent to the expected optimal solution cost in the absence of obstacles,

$$E[c_{i+1} \mid \mathbf{x}_{\text{new}} \in X_f]^* \equiv E[f(\mathbf{x}_{\text{new}}) \mid \mathbf{x}_{\text{new}} \in X_f]. \tag{C.5}$$

The lower bound provided by (C.4) is therefore sharp over the set of all possible planning problems and algorithm configurations.

In this absence of obstacles, the optimal solution using any state is

$$f(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2, \tag{3.7 redux}$$

and the omniscient set is the prolate hyperspheroid, $X_f \equiv X_{\widehat{f}} \equiv X_{\text{PHS}}$. The measure of the omniscient set, $\lambda(X_f) \equiv \lambda_{\text{PHS}}$, is given by

$$\lambda_{\text{PHS}} \coloneqq \frac{c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}} \zeta_n}{2^n}, \tag{3.8 redux}$$

where ζ_n is the measure of a unit n-dimensional ball and $c_{\min}$ is the distance between the start and goal. This allows (C.5) to be written as

$$E[c_{i+1} \mid \mathbf{x}_{\text{new}} \in X_f]^* = \frac{1}{\lambda_{\text{PHS}}} \int_{X_{\text{PHS}}} \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2\, dV. \tag{C.6}$$

In prolate hyperspheroidal coordinates, $\mu, \nu, \psi_1, \ldots, \psi_{n-2}$, with the parameterization $a = 0.5c_{\min}$ (Appendix E), (3.7) can be expressed simply as

$$f(\mathbf{x}) = c_{\min} \cosh \mu. \tag{C.7}$$

Substituting (C.7) and the prolate hyperspheroidal differential volume,

$$dV = a^n \left(\sinh^2 \mu + \sin^2 \nu\right) \sinh^{n-2} \mu \sin^{n-2} \nu \sin^{n-3} \psi_1 \ldots \sin \psi_{n-3} \, d\mu \, d\nu \, d\psi_1 \, \ldots \, d\psi_{n-2}, \tag{E.9 redux}$$

into (C.6) results in

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{c_{\min}^{n+1}}{2^n \lambda_{\text{PHS}}} \int_{\mu=0}^{\mu_i} \int_{\nu=0}^{\pi} \int_{\psi_1=0}^{\pi} \cdots \int_{\psi_{n-3}=0}^{\pi} \int_{\psi_{n-2}=0}^{2\pi} \left(\sinh^2 \mu + \sin^2 \nu\right) \sinh^{n-2} \mu \cosh \mu \sin^{n-2} \nu \sin^{n-3} \psi_1 \ldots \sin \psi_{n-3} \, d\mu \, d\nu \, d\psi_1, \ldots d\psi_{n-2}, \tag{C.8}$$

where the integration limit for μ is derived from (C.7) as

$$\cosh \mu_i \coloneqq \frac{c_i}{c_{\min}}. \tag{C.9}$$

Integrating (C.8) requires applying a series of identities, first

$$(n-1)\, \zeta_{n-1} \equiv \int_{\psi_1=0}^{\pi} \cdots \int_{\psi_{n-3}=0}^{\pi} \int_{\psi_{n-2}=0}^{2\pi} \sin^{n-3} \psi_1 \ldots \sin \psi_{n-3} \, d\psi_1 \ldots \, d\psi_{n-2},$$

simplifies (C.8) to

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{(n-1)\, c_{\min}^{n+1} \zeta_{n-1}}{2^n \lambda_{\text{PHS}}} \int_{\mu=0}^{\mu_i} \int_{\nu=0}^{\pi} \left(\sinh^2 \mu + \sin^2 \nu\right) \sinh^{n-2} \mu \cosh \mu \sin^{n-2} \nu \, d\mu \, d\nu. \tag{C.10}$$

Next, the definite integral of the product of powers of sin and cos,

$$\int_0^{\pi} \sin^{2m-1} \theta \cos^{2n-1} \theta \, d\theta \equiv \mathrm{B}(m, n),$$

where $\mathrm{B}(\cdot, \cdot)$ is the beta function,

$$\mathrm{B}(m, n) \coloneqq \int_0^1 t^{m-1} (1-t)^{n-1} \, dt,$$

is used to evaluate the integral over ν in (C.10), giving

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{(n-1)\, c_{\min}^{n+1} \zeta_{n-1}}{2^n \lambda_{\text{PHS}}} \left(\mathrm{B}\left(\frac{n-1}{2}, \frac{1}{2}\right) \int_{\mu=0}^{\mu_i} \sinh^n \mu \cosh \mu \, d\mu \right. \\ \left. + \mathrm{B}\left(\frac{n+1}{2}, \frac{1}{2}\right) \int_{\mu=0}^{\mu_i} \sinh^{n-2} \mu \cosh \mu \, d\mu \right). \tag{C.11}$$

The identity,

$$\mathrm{B}\left(m+1, n\right) \equiv \frac{m}{m+n} \mathrm{B}\left(m, n\right),$$

and the recursive nature of the n-dimensional unit ball,

$$\zeta_n \equiv \mathrm{B}\left(\frac{n+1}{2}, \frac{1}{2}\right) \zeta_{n-1},$$

simplifies (C.11) to

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{c_{\min}^{n+1} \zeta_n}{2^n \lambda_{\text{PHS}}} \left(n \int_{\mu=0}^{\mu_i} \sinh^n \mu \cosh \mu \, d\mu \right. \\ \left. + (n-1) \int_{\mu=0}^{\mu_i} \sinh^{n-2} \mu \cosh \mu \, d\mu \right). \tag{C.12}$$

The indefinite integral,

$$\int \sinh^m \theta \cosh \theta \, d\theta \equiv \frac{\sinh^{m+1} \theta}{m+1},$$

is then used to evaluate (C.12), giving

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{c_{\min}^{n+1} \zeta_n}{2^n \lambda_{\text{PHS}}} \left(\frac{n}{n+1} \sinh^{n+1} \mu_i + \sinh^{n-1} \mu_i \right). \tag{C.13}$$

Using (3.8) to expand the measure λ_{PHS} in (C.13) cancels the measure of the unit n-ball, giving

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{c_{\min}^{n+1}}{c_i \left(c_i^2 - c_{\min}^2\right)^{\frac{n-1}{2}}} \left(\frac{n}{n+1} \sinh^{n+1} \mu_i + \sinh^{n-1} \mu_i \right). \tag{C.14}$$

Using the relationship

$$\cosh \mu = b \iff \sinh \mu = \sqrt{b^2 - 1},$$

with (C.9) and some algebraic manipulation finally simplifies (C.14) to

$$E\left[c_{i+1} \,|\, \mathbf{x}_{\text{new}} \in X_f\right]^* = \frac{nc_i^2 + c_{\min}^2}{(n+1)\, c_i},$$

an exact value for the best-case expected solution cost of RRT*. This result allows (C.3) to be written as the sharp bound,

$$E\left[c_{i+1}\right] \geq p_f \frac{nc_i^2 + c_{\min}^2}{(n+1)\, c_i} + (1 - p_f)\, c_i,$$

which when combined with (C.1) proves Lemma 4.1. □

Appendix D

Proofs of Theorems 4.2–4.4

This section restates and proves Theorems 4.2–4.4.

D.1 Proof of Theorem 4.2

Theorem 4.2 (Sublinear convergence of RRT* in geometric planning)**.** *RRT* converges* sublinearly *towards the optimum of geometric planning problems,*

$$E\left[\mu_{\mathrm{RRT}^*}\right] = 1. \tag{4.4 redux}$$

Proof. The expected rate of convergence of RRT* from Definition 4.1 is

$$E\left[\mu_{\mathrm{RRT}^*}\right] = E\left[\lim_{i\to\infty} \frac{c_i - c_{\min}}{c_{i-1} - c_{\min}}\right], \tag{D.1}$$

since, $\forall i$, $c_i \geq c_{\min}$. As RRT* almost-surely converges asymptotically to the optimum, this sequence also almost-surely converges,

$$P\left(\lim_{i\to\infty} \frac{c_i - c_{\min}}{c_{i-1} - c_{\min}} = \mu_{\mathrm{RRT}^*}\right) = 1,$$

to a finite value, $0 \leq \mu_{\mathrm{RRT}^*} \leq 1$. By Lebesgue's dominated convergence theorem this allows the expectation operator to be brought inside the limit of (D.1), giving

$$E\left[\mu_{\mathrm{RRT}^*}\right] = \lim_{i\to\infty} \frac{E\left[c_i\right] - c_{\min}}{c_{i-1} - c_{\min}}, \tag{D.2}$$

since c_i is the only random variable at iteration i.

Lemma 4.1 provides sharp bounds for the expected solution cost at any iteration, $E\left[c_i\right]$, with the lower-bound corresponding to an infinite rewiring radius in the absence of obstacles.

Substituting this lower bound into (D.2) and performing algebraic simplification gives an expression for the expected *best-case* convergence rate,

$$E\left[\mu^*_{\text{RRT}^*}\right] = 1 + \frac{1}{(n+1)} \lim_{i\to\infty} \frac{p_f \left(c^2_{\min} - c^2_{i-1}\right)}{\left(c^2_{i-1} - c_{\min} c_{i-1}\right)},$$

such that $E\left[\mu^*_{\text{RRT}^*}\right] \leq E\left[\mu_{\text{RRT}^*}\right]$ is a sharp bound over all possible planning problems and algorithm configurations. Applying l'Hôpital's rule (l'Hôpital, 1696) with respect to c_{i-1} gives

$$E\left[\mu^*_{\text{RRT}^*}\right] = 1 + \frac{1}{(n+1)} \lim_{i\to\infty} \frac{\frac{\partial p_f}{\partial c_{i-1}} \left(c^2_{\min} - c^2_{i-1}\right) - 2 p_f c_{i-1}}{\left(2c_{i-1} - c_{\min}\right)}. \tag{D.3}$$

As iterations go to infinity the probability of adding a sample in X_f becomes the probability of sampling it (Lemma 3.2). As the lower bound from Lemma 4.1 is for an obstacle-free problem, $X_f \equiv X_{\widehat{f}}$ and the probability of sampling it is given by (3.9),

$$p_f = \frac{\pi^{\frac{n}{2}} c_{i-1} \left(c^2_{i-1} - c^2_{\min}\right)^{\frac{n-1}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right) \lambda\left(X_{\text{sampling}}\right)}, \tag{3.9 redux}$$

with a partial derivative of

$$\frac{\partial p_f}{\partial c_{i-1}} = \frac{\pi^{\frac{n}{2}}}{2^n \Gamma\left(\frac{n}{2}+1\right) \lambda\left(X_{\text{sampling}}\right)} \left(c^2_{i-1} - c^2_{\min}\right)^{\frac{n-1}{2}} \left(1 + \frac{(n-1)\, c^2_{i-1}}{\left(c^2_{i-1} - c^2_{\min}\right)}\right).$$

Almost-sure convergence to $c_{\min}$ implies $\lim_{i\to\infty} c_{i-1} = c_{\min}$ and therefore $\lim_{i\to\infty} p_f = 0$ and $\lim_{i\to\infty} \frac{\partial p_f}{\partial c_{i-1}} = 0$, making (D.3),

$$E\left[\mu^*_{\text{RRT}^*}\right] = 1.$$

As by definition the expected rate of convergence of RRT* is bounded by,

$$E\left[\mu^*_{\text{RRT}^*}\right] \leq E\left[\mu_{\text{RRT}^*}\right] \leq 1,$$

this result proves Theorem 4.2. □

D.2 Proof of Theorem 4.3

Theorem 4.3 (Linear convergence of RRT* with adaptive rectangular rejection sampling in geometric planning)**.** *RRT* with adaptive rectangular rejection sampling converges at best* linearly *towards the optimum of geometric planning problems but factorially approaches sublinear convergence as state dimension increases,*

$$1 - \frac{\pi^{\frac{n}{2}}}{(n+1)\,2^{n-1}\Gamma\left(\frac{n}{2}+1\right)} \leq E\left[\mu_{\text{Reject}}\right] \leq 1. \tag{4.5 redux}$$

Proof. Proof of Theorem 4.3 follows that of Theorem 4.2 but with the probability of adding a new state from X_f calculated from (3.9) using (3.12), as

$$p_f \leq \frac{\pi^{\frac{n}{2}}}{2^n\Gamma\left(\frac{n}{2}+1\right)}. \tag{D.4}$$

As $\frac{\partial p_f}{\partial c_{i-1}} = 0$, (D.3) becomes

$$E\left[\mu^*_{\text{Reject}}\right] = 1 - \frac{1}{(n+1)} \lim_{i\to\infty} \frac{2p_f c_{i-1}}{(2c_{i-1} - c_{\min})}. \tag{D.5}$$

Noting that almost-sure convergence to $c_{\min}$ implies $\lim_{i\to\infty} c_{i-1} = c_{\min}$ and substituting (D.4) into (D.5) results in

$$\begin{aligned} E\left[\mu^*_{\text{Reject}}\right] &= 1 - \frac{2p_f}{(n+1)}, \\ &\geq 1 - \frac{\pi^{\frac{n}{2}}}{(n+1)\,2^{n-1}\Gamma\left(\frac{n}{2}+1\right)}. \end{aligned}$$

As by definition the expected rate of convergence of RRT* with rectangular rejection sampling is bounded by,

$$E\left[\mu^*_{\text{Reject}}\right] \leq E\left[\mu_{\text{Reject}}\right] \leq 1,$$

this result proves Theorem 4.3 with the bounds being sharp over all possible planning problems and algorithm configurations. □

D.3 Proof of Theorem 4.4

Theorem 4.4 (Linear convergence of Informed RRT* in geometric planning)**.** *Informed RRT* converges at best* linearly *towards the optimum of geometric planning problems,*

$$\frac{n-1}{n+1} \leq E\left[\mu_{\text{Informed}}\right] \leq 1, \tag{4.6 redux}$$

where the lower-bound occurs exactly with an infinite rewiring neighbourhood in the absence of obstacles.

Proof. Proof of Theorem 4.4 follows that of Theorem 4.2 but with unity probability of adding a new state from X_f. From (D.3), the convergence rate of Informed RRT* is then,

$$E\left[\mu^*_{\text{Informed}}\right] = 1 - \frac{1}{(n+1)} \lim_{i\to\infty} \frac{2c_{i-1}}{(2c_{i-1} - c_{\min})}.$$

As almost-sure convergence to $c_{\min}$ implies $\lim_{i\to\infty} c_{i-1} = c_{\min}$ this gives,

$$E\left[\mu^*_{\text{Informed}}\right] = \frac{n-1}{n+1}.$$

As by definition the expected rate of convergence of RRT* with rectangular rejection sampling is bounded by,

$$E\left[\mu^*_{\text{Informed}}\right] \leq E\left[\mu_{\text{Informed}}\right] \leq 1,$$

this result proves Theorem 4.4, with the bounds being sharp over all possible planning problems and algorithm configurations. □

Appendix E

The Prolate Hyperspheroid Coordinate System

Let $(x_1, x_2, \ldots, x_n)$ be the n-dimensional Cartesian coordinate system, then a prolate hyperspheroid coordinate system, $(\mu, \nu, \psi_1, \psi_2, \ldots, \psi_{n-2})$, can be defined as

$$\begin{aligned} x_1 &=: a\cosh\mu\cos\nu, \\ x_2 &=: a\sinh\mu\sin\nu\cos\psi_1, \\ x_3 &=: a\sinh\mu\sin\nu\sin\psi_1\cos\psi_2, \\ &\vdots \\ x_{n-1} &=: a\sinh\mu\sin\nu\sin\psi_1\sin\psi_2\ldots\sin\psi_{n-3}\cos\psi_{n-2}, \\ x_n &=: a\sinh\mu\sin\nu\sin\psi_1\sin\psi_2\ldots\sin\psi_{n-3}\sin\psi_{n-2}, \end{aligned} \tag{E.1}$$

where the foci of the prolate hyperspheroid occurs in Cartesian coorinates at $(\pm a, 0, \ldots, 0)$ and every point will lie on the surface of a prolate hyperspheroid with diameter, $d = 2a\cosh\mu$.

These coordinates can be viewed as a 2D elliptical coordinate system, μ, ν, rotated by spherical coordinates, $\psi_1, \psi_2, \ldots, \psi_{n-2}$ (Fig. E.1). The coordinates take the values $\mu \in [0, \infty)$, $\nu \in [0, \pi]$, $\psi_1, \psi_2, \ldots, \psi_{n-3} \in [0, \pi]$, and $\psi_{n-2} \in [0, 2\pi)$, except in the 2D case when $\nu \in [0, 2\pi)$.

The basis vectors of this curvilinear coordinate system, $\mathbf{e}_\mu, \mathbf{e}_\nu, \mathbf{e}_{\psi_1}, \ldots, \mathbf{e}_{\psi_{n-2}}$, are defined as the partial derivatives of (E.1) with respect to the respective prolate hyperspheroid coordinates, i.e.,

$$\mathbf{e}_\mu \coloneqq \begin{bmatrix} \frac{\partial x_1}{\partial \mu} & \frac{\partial x_2}{\partial \mu} & \ldots & \frac{\partial x_n}{\partial \mu} \end{bmatrix}^T, \text{ etc.},$$

from which the scale factors, $h_\mu, h_\nu, h_{\psi_1}, \ldots, h_{\psi_{n-2}}$, are calculated as

$$h_\mu \coloneqq \|\mathbf{e}_\mu\|_2, \text{ etc.} \tag{E.2}$$

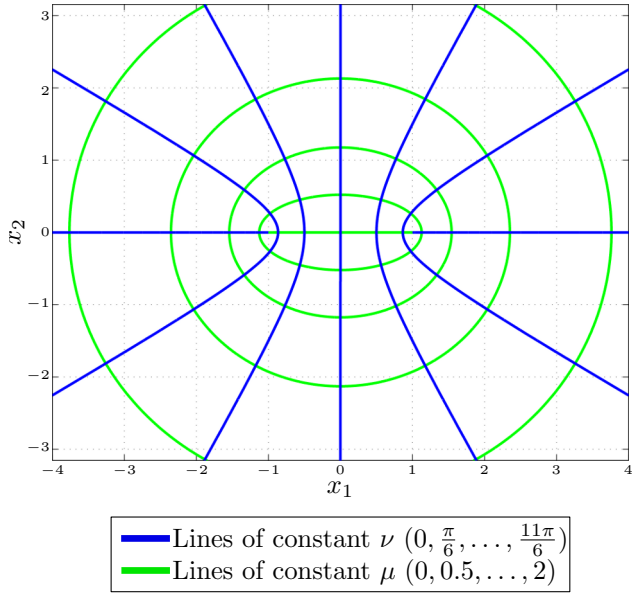


Figure E.1: The 2D prolate hyperspheroidal coordinate system for $a = 1$ (i.e., the elliptical coordinate system). It is extended into higher dimensions by rotation about the transverse axis (i.e., x_2) in a manner analogous to the extension of the polar coordinate system to the hyperspherical coordinate system. Starting from the line on the positive x_1 axis and proceeding counterclockwise, the blue lines of constant ν correspond to angles of $0, \frac{\pi}{6}, \frac{2\pi}{6}, \ldots \frac{11\pi}{6}$ radians, respectively. Proceeding outwards, the green lines of constant μ correspond to values of $0, 0.5, \ldots, 2$, respectively.

The differential unit of volume, dV, is then

$$dV \coloneqq h_\mu d\mu \, h_\nu d\nu \, h_{\psi_1} d\psi_1 \, \ldots h_{\psi_{n-2}} d\psi_{n-2} = h_\mu h_\nu d\mu \, d\nu \prod_{i=1}^{n-2} \left(h_{\psi_i} d\psi_i \right). \tag{E.3}$$

First Scale Factor

The partial derivatives of (E.1) with respect to μ are

$$\begin{aligned}
\frac{\partial x_1}{\partial \mu} &= -a \sinh \mu \cos \nu, \\
\frac{\partial x_2}{\partial \mu} &= a \cosh \mu \sin \nu \cos \psi_1, \\
\frac{\partial x_3}{\partial \mu} &= a \cosh \mu \sin \nu \sin \psi_1 \cos \psi_2, \\
&\vdots \\
\frac{\partial x_{n-1}}{\partial \mu} &= a \cosh \mu \sin \nu \sin \psi_1 \ldots \sin \psi_{n-3} \cos \psi_{n-2}, \\
\frac{\partial x_n}{\partial \mu} &= a \cosh \mu \sin \nu \sin \psi_1 \ldots \sin \psi_{n-3} \sin \psi_{n-2}.
\end{aligned}$$

From (E.2) the scale factor, h_μ, is then

$$\begin{aligned} h_\mu = a \big(&\sinh^2\mu\cos^2\nu + \cosh^2\mu\sin^2\nu\cos^2\psi_1 + \cosh^2\mu\sin^2\nu\sin^2\psi_1\cos^2\psi_2 + \dots \\ &+ \cosh^2\mu\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\cos^2\psi_{n-2} \\ &+ \cosh^2\mu\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\sin^2\psi_{n-2}\big)^{\frac{1}{2}} . \end{aligned}$$

Using the Pythagorean trigonometric identity and the hyperbolic analogue,

$$\begin{aligned} \sin^2 b + \cos^2 b &\equiv 1 \\ \cosh^2 a - \sinh^2 a &\equiv 1, \end{aligned} \tag{E.4}$$

for every instance of $\cos^2(\cdot)$ and $\cosh^2(\cdot)$ yields

$$\begin{aligned} h_\mu = a \big(&\overbrace{\sinh^2\mu - \sinh^2\mu\sin^2\nu}^{x_1 \text{ term}} + \overbrace{\sin^2\nu + \sinh^2\mu\sin^2\nu - \sin^2\nu\sin^2\psi_1 - \sinh^2\mu\sin^2\nu\sin^2\psi_1}^{x_2 \text{ term}} \\ &+ \overbrace{\sin^2\nu\sin^2\psi_1 + \sinh^2\mu\sin^2\nu\sin^2\psi_1 - \sin^2\nu\sin^2\psi_1\sin^2\psi_2 - \sinh^2\mu\sin^2\nu\sin^2\psi_1\sin^2\psi_2}^{x_3 \text{ term}} \\ &+ \dots + \overbrace{\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3} + \sinh^2\mu\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}}^{x_{n-1} \text{ term}} \\ &\overbrace{- \sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\sin^2\psi_{n-2} - \sinh^2\mu\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_{n-1} \text{ term}} \\ &+ \overbrace{\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\sin^2\psi_{n-2} + \sinh^2\mu\sin^2\nu\sin^2\psi_1\dots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_n \text{ term}} \big)^{\frac{1}{2}} , \end{aligned}$$

about which the following observations are made:

- The second term of the x_1 grouping cancels with the second term of the x_2 grouping.
- In the intermediate terms, $x_2, x_3, \dots x_{n-1}$, the third and fourth term of each x_j grouping cancels with the first and second terms of the subsequent x_{j+1} grouping.
- The final x_n grouping cancels with the third and fourth terms of the x_{n-1} grouping.

These observations simplify the first scale factor to

$$h_\mu = a \left(\sinh^2\mu + \sin^2\nu\right)^{\frac{1}{2}} . \tag{E.5}$$

Second Scale Factor

The partial derivatives of (E.1) with respect to ν are

$$\begin{aligned}
\frac{\partial x_1}{\partial \nu} &= -a\cosh\mu\sin\nu,\\
\frac{\partial x_2}{\partial \nu} &= a\sinh\mu\cos\nu\cos\psi_1,\\
\frac{\partial x_3}{\partial \nu} &= a\sinh\mu\cos\nu\sin\psi_1\cos\psi_2,\\
&\vdots\\
\frac{\partial x_{n-1}}{\partial \nu} &= a\sinh\mu\cos\nu\sin\psi_1\ldots\sin\psi_{n-3}\cos\psi_{n-2},\\
\frac{\partial x_n}{\partial \nu} &= a\sinh\mu\cos\nu\sin\psi_1\ldots\sin\psi_{n-3}\sin\psi_{n-2}.
\end{aligned}$$

From (E.2) the scale factor, h_ν, is then

$$\begin{aligned}
h_\nu = a\big(&\cosh^2\mu\sin^2\nu+\sinh^2\mu\cos^2\nu\cos^2\psi_1+\sinh^2\mu\cos^2\nu\sin^2\psi_1\cos^2\psi_2+\ldots\\
&+\sinh^2\mu\cos^2\nu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\cos^2\psi_{n-2}\\
&+\sinh^2\mu\cos^2\nu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}\big)^{\frac{1}{2}}.
\end{aligned}$$

Once again making use of the Pythagorean identities, (E.4), for every instance of $\cos^2(\cdot)$ and $\cosh^2(\cdot)$ gives

$$\begin{aligned}
h_\nu = a\big(&\overbrace{\sin^2\nu+\sinh^2\mu\sin^2\nu}^{x_1\text{ term}}+\overbrace{\sinh^2\mu-\sinh^2\mu\sin^2\nu-\sinh^2\mu\sin^2\psi_1+\sinh^2\mu\sin^2\nu\sin^2\psi_1}^{x_2\text{ term}}\\
&+\overbrace{\sinh^2\mu\sin^2\psi_1-\sinh^2\mu\sin^2\nu\sin^2\psi_1-\sinh^2\mu\sin^2\psi_1\sin^2\psi_2+\sinh^2\mu\sin^2\nu\sin^2\psi_1\sin^2\psi_2}^{x_3\text{term}}\\
&+\ldots+\overbrace{\sinh^2\mu\sin^2\psi_1\ldots\sin^2\psi_{n-3}-\sinh^2\mu\sin^2\nu\sin^2\psi_1\ldots\sin^2\psi_{n-3}}^{x_{n-1}\text{ term}}\\
&\overbrace{-\sinh^2\mu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}+\sinh^2\mu\sin^2\nu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_{n-1}\text{ term}}\\
&+\overbrace{\sinh^2\mu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}-\sinh^2\mu\sin^2\nu\sin^2\psi_1\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_n\text{ term}}\big)^{\frac{1}{2}},
\end{aligned}$$

about which the following observations are made:

- The second term of the x_1 grouping cancels with the second term of the x_2 grouping.
- In the intermediate terms, $x_2, x_3, \ldots x_{n-1}$, the third and fourth term of each x_j grouping cancels with the first and second terms of the subsequent x_{j+1} grouping.
- The final x_n grouping cancels with the third and fourth terms of the x_{n-1} grouping.

These observations simplify the second scale factor to

$$h_\nu = a\left(\sinh^2\mu + \sin^2\nu\right)^{\frac{1}{2}}. \tag{E.6}$$

Intermediate Scale Factors

The partial derivatives of (E.1) with respect to ψ_i, $1 \le i \le n-3$ are

$$\begin{aligned}
\frac{\partial x_1}{\partial \psi_i} &= \frac{\partial x_2}{\partial \psi_i} = \ldots = \frac{\partial x_i}{\partial \psi_i} = 0,\\
\frac{\partial x_{i+1}}{\partial \psi_i} &= -a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\sin\psi_i,\\
\frac{\partial x_{i+2}}{\partial \psi_i} &= a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\cos\psi_i\cos\psi_{i+1},\\
\frac{\partial x_{i+3}}{\partial \psi_i} &= a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\cos\psi_i\sin\psi_{i+1}\cos\psi_{i+2},\\
&\vdots\\
\frac{\partial x_{n-1}}{\partial \psi_i} &= a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\cos\psi_i\sin\psi_{i+1}\ldots\sin\psi_{n-3}\cos\psi_{n-2},\\
\frac{\partial x_n}{\partial \psi_i} &= a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\cos\psi_i\sin\psi_{i+1}\ldots\sin\psi_{n-3}\sin\psi_{n-2}.
\end{aligned}$$

From (E.2) the scale factors, h_{ψ_i}, $1 \le i \le n-3$, are then

$$\begin{aligned}
h_{\psi_i} = a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\big(&\sin^2\psi_i + \cos^2\psi_i\cos^2\psi_{i+1} + \cos^2\psi_i\sin^2\psi_{i+1}\cos^2\psi_{i+2}\\
&+\ldots+\cos^2\psi_i\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\cos^2\psi_{n-2}\\
&+\cos^2\psi_i\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}\big)^{\frac{1}{2}}.
\end{aligned}$$

Returning to the Pythagorean trigonometric identity, (E.4), for every instance of $\cos^2(\cdot)$ gives

$$\begin{aligned}
h_{\psi_i} = a\sinh\mu\sin\nu\sin\psi_1\ldots\sin\psi_{i-1}\big(&\overbrace{\sin^2\psi_i}^{x_{i+1}\text{ term}} + \overbrace{1-\sin^2\psi_i-\sin^2\psi_{i+1}+\sin^2\psi_i\sin^2\psi_{i+1}}^{x_{i+2}\text{ term}}\\
&+\overbrace{\sin^2\psi_{i+1}-\sin^2\psi_i\sin^2\psi_{i+1}-\sin^2\psi_{i+1}\sin^2\psi_{i+2}+\sin^2\psi_i\sin^2\psi_{i+1}\sin^2\psi_{i+2}}^{x_{i+3}\text{ term}}\\
&+\ldots+\overbrace{\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}-\sin^2\psi_i\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}}^{x_{n-1}\text{ term}}\\
&\overbrace{-\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}+\sin^2\psi_i\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_{n-1}\text{ term}}\\
&+\overbrace{\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}-\sin^2\psi_i\sin^2\psi_{i+1}\ldots\sin^2\psi_{n-3}\sin^2\psi_{n-2}}^{x_n\text{ term}}\big)^{\frac{1}{2}},
\end{aligned}$$

about which the following observations are made:

- The only term of the x_i grouping cancels with the second term of the x_{i+1} grouping.
- For the intermediate terms, $x_{i+2}, x_{i+3}, \ldots x_{n-1}$, the third and fourth term of each x_j grouping cancels with the first and second terms of the subsequent x_{j+1} grouping.
- The final x_n grouping cancels with the third and fourth terms of the x_{n-1} grouping.

This leaves unity inside the square root, giving

$$h_{\psi_i} = a \sinh\mu \sin\nu \sin\psi_1 \ldots \sin\psi_{i-1}, \quad 1 \leq i \leq n-3. \tag{E.7}$$

Final Scale Factor

The partial derivatives of (E.1) with respect to the last prolate hyperspheroid coordinate, ψ_{n-2}, are

$$\begin{aligned}
\frac{\partial x_1}{\partial \psi_{n-2}} &= \frac{\partial x_2}{\partial \psi_{n-2}} = \ldots = \frac{\partial x_{n-2}}{\partial \psi_{n-2}} = 0, \\
\frac{\partial x_{n-1}}{\partial \psi_{n-2}} &= -a \sinh\mu \sin\nu \sin\psi_1 \ldots\ldots \sin\psi_{n-3} \sin\psi_{n-2}, \\
\frac{\partial x_n}{\partial \psi_{n-2}} &= a \sinh\mu \sin\nu \sin\psi_1 \ldots\ldots \sin\psi_{n-3} \cos\psi_{n-2}.
\end{aligned}$$

From (E.2) the scale factor, $h_{\psi_{n-2}}$, is then

$$h_{\psi_{n-2}} = a \sinh\mu \sin\nu \sin\psi_1 \ldots \sin\psi_{n-3} \left(\sin^2\psi_{n-2} + \cos^2\psi_{n-2}\right)^{\frac{1}{2}},$$

to which one last application of the Pythagorean trigonometric identity, (E.4), gives

$$h_{\psi_{n-2}} = a \sinh\mu \sin\nu \sin\psi_1 \ldots \sin\psi_{n-3}.$$

This result expands (E.7) to a single expression for all ψ_i:

$$h_{\psi_i} = a \sinh\mu \sin\nu \sin\psi_1 \ldots \sin\psi_{i-1}, \quad 1 \leq i \leq n-2. \tag{E.8}$$

Differential Volume

Substituting (E.5), (E.6), and (E.8) into (E.3), gives a final expression for the differential volume

$$\begin{aligned}
dV = a^n \left(\sinh^2\mu + \sin^2\nu\right) \sinh^{n-2}\mu \sin^{n-2}\nu \sin^{n-3}\psi_1 \sin^{n-4}\psi_2 \ldots \\
\sin\psi_{n-3}\, d\mu\, d\nu\, d\psi_1\, d\psi_2 \ldots d\psi_{n-3}\, d\psi_{n-2}.
\end{aligned} \tag{E.9}$$

Bibliography

Aine, S. and Likhachev, M. (2016). Truncated incremental search. *Artificial Intelligence*, 234:49–77. [Cited pp. 16, 17, and 92]

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V., and Likhachev, M. (2014). Multi-heuristic A*. In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA. [Cited p. 14]

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V., and Likhachev, M. (2015). Multi-heuristic A*. *The International Journal of Robotics Research (IJRR)*, 35(1–3):224–243. [Cited p. 14]

Akgun, B. and Stilman, M. (2011). Sampling heuristics for optimal motion planning in high dimensions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2640–2645. [Cited pp. 23, 55, and 56]

Alterovitz, R., Patil, S., and Derbakova, A. (2011). Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3706–3712. [Cited p. 57]

Amato, N. M. and Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 113–120. [Cited p. 23]

Amazon (2015). Retrieved from `http://www.amazon.com/b?node=8037720011`. [Cited p. 3]

Arslan, O. and Tsiotras, P. (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2421–2428. [Cited pp. 25, 56, and 81]

Arslan, O. and Tsiotras, P. (2015). Dynamic programming guided exploration for sampling-based motion planning algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4819–4826. [Cited pp. 25, 56, and 81]

Baldwin, I. and Newman, P. (2010). Teaching a randomized planner to plan with semantic fields. In *Towards Autonomous Robotic Systems*. [Cited p. 23]

Barraquand, J., Kavraki, L., Motwani, R., Latombe, J.-C., Li, T.-Y., and Raghavan, P. (1996). A random sampling scheme for path planning. In Giralt, G. and Hirzinger, G., editors, *Robotics Research: The Seventh International Symposium (ISRR)*, pages 249–264. Springer London. [Cited p. 18]

Barraquand, J. and Latombe, J.-C. (1993). Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2–4):121–155. [Cited p. 14]

Bellman, R. E. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society (AMS)*, 60(6):503–516. [Cited pp. 3, 10, and 17]

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press. [Cited pp. 3, 10, and 17]

Bellman, R. E. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90. [Cited p. 10]

Berchtold, S. and Glavina, B. (1994). A scalable optimizer for automatically generated manipulator motions. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1796–1802. [Cited p. 21]

Berczi, L.-P., Gammell, J. D., Tong, C., Daly, M., and Barfoot, T. D. (2013). A proof-of-concept, rover-based system for autonomously locating methane gas sources on Mars. In *Proceedings of the International Conference on Computer and Robot Vision (CRV)*, pages 29–36. [Cited p. 3]

Bertsekas, D. P. (1975). Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control (TAC)*, 20(3):415–419. [Cited pp. 3 and 17]

Bevilacqua, P., Frego, M., Bertolazzi, E., Fontanelli, D., Palopoli, L., and Biral, F. (2016). Path planning maximising human comfort for assistive robots. In *IEEE Conference on Control Applications (CCA)*, pages 1421–1427. [Cited pp. 75 and 110]

Bohlin, R. (2001). Path planning in practice; lazy evaluation on a multi-resolution grid. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 49–54. [Cited pp. 17 and 78]

Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy PRM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 521–528. [Cited pp. 19, 78, 86, and 106]

Boor, V., Overmars, M. H., and van der Stappen, A. F. (1999). The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1018–1023. [Cited p. 23]

Bostock, M., Carter, S., Corum, J., and White, J. (2014). 28 months on Mars. *The New York Times*. Retrieved from `http://www.nytimes.com/interactive/2014/12/09/science/space/curiosity-rover-28-months-on-mars.html`. [Cited p. 1]

Branicky, M. S., LaValle, S. M., Olson, K., and Yang, L. (2001). Quasi-randomized path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1481–1487. [Cited pp. 18, 19, 29, 78, 86, and 106]

Burget, F., Bennewitz, M., and Burgard, W. (2016). BI^{2}RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3714–3721. [Cited pp. 75 and 110]

Burns, B. and Brock, O. (2005a). Sampling-based motion planning using predictive models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3120–3125. [Cited p. 23]

Burns, B. and Brock, O. (2005b). Toward optimal configuration space sampling. In *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, USA. [Cited p. 23]

Chen, P. C. and Hwang, Y. K. (1992). SANDROS: a motion planner with performance proportional to task difficulty. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2346–2353. [Cited p. 17]

Cherif, M. (1999). Kinodynamic motion planning for all-terrain wheeled vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 317–322. [Cited p. 14]

Choset, H. and Burdick, J. (1995). Sensor based planning, part I: The generalized Voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1649–1655. [Cited p. 55]

Choudhury, S., Arora, S., and Scherer, S. (2014). The planner ensemble and trajectory executive: A high performance motion planning system with guaranteed safety. In *Proceedings of the American Helicopter Society (AHS) International 70th Annual Forum*, Montréal, Québec, Canada. [Cited p. 111]

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S., and Scherer, S. (2016). Regionally Accelerated Batch Informed Trees (RABIT*): A framework to integrate local information into optimal path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4207–4214, Stockholm, Sweden. [Cited pp. 106 and 111]

Cohen, B., Chitta, S., and Likhachev, M. (2014a). Single- and dual-arm motion planning with heuristic search. *The International Journal of Robotics Research (IJRR)*, 33(2):305–320. [Cited p. 17]

Cohen, B., Phillips, M., and Likhachev, M. (2014b). Planning single-arm manipulations with n-arm robots. In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA. [Cited pp. 14, 29, 78, 86, and 106]

de Ruiter, A. H. J. and Forbes, J. R. (2013). On the solution of Wahba's problem on SO(n). *The Journal of the Astronautical Sciences*, 60(1):1–31. [Cited p. 44]

Devaurs, D., Siméon, T., and Cortés, J. (2015). Efficient sampling-based approaches to optimal path planning in complex cost spaces. In Akin, H. L., Amato, N. M., Isler, V., and van der Stappen, A. F., editors, *Algorithmic Foundations of Robotics XI*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 143–159. Springer International Publishing. [Cited p. 58]

Diankov, R. and Kuffner Jr., J. J. (2007). Randomized statistical path planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [Cited pp. 24 and 79]

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271. [Cited p. 12]

Dirichlet, J. P. G. L. (1850). Über die reduktion der positiven quadratischen formen mit drei unbestimmten ganzen zahlen. *Journal für die Reine und Angewandte Mathematik*, 40:209–227. [Cited p. 19]

Dobson, A. and Bekris, K. E. (2014). Sparse roadmap spanners for asymptotically near-optimal motion planning. *The International Journal of Robotics Research (IJRR)*, 33(1):18–47. [Cited p. 23]

Dobson, A., Moustakides, G. V., and Bekris, K. E. (2015). Geometric probability results for bounding path quality in sampling-based roadmaps after finite computation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4180–4186. [Cited p. 63]

Donald, B., Xavier, P., Canny, J., and Reif, J. (1993). Kinodynamic motion planning. *Journal of the Association for Computing Machinery (JACM)*, 40(5):1048–1066. [Cited p. 17]

Eppstein, D., Paterson, M. S., and Yao, F. F. (1997). On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282. [Cited p. 82]

Euler, L. (1738). De progressionibus transcendentibus seu quarum termini generales algebraice dari nequeunt. *Commentarii academiae scientiarum Petropolitanae*, 5:36–57. [Cited pp. xiv and 40]

Ferguson, D. and Stentz, A. (2005). The delayed D* algorithm for efficient path replanning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2045–2050. [Cited p. 15]

Ferguson, D. and Stentz, A. (2006). Anytime RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5369–5375. [Cited pp. 29, 30, 35, 40, 57, 80, and 109]

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the Association for Computing Machinery (CACM)*, 5(6):345. [Cited p. 10]

Ford Jr., L. R. (1956). Network flow theory. Technical Report Paper P-923, RAND Corporation, Santa Monica, California. [Cited p. 10]

Gammell, J. D. and Barfoot, T. D. (2014). The probability density function of a transformation-based hyperellipsoid sampling technique. Technical Report TR-2014-JDG004, Autonomous Space Robotics Lab, University of Toronto. arXiv:1404.1347 [math.ST]. [Cited pp. 6 and 116]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014a). BIT*: Sampling-based optimal planning via batch informed trees. In *The Information-based Grasp and Manipulation Planning Workshop, Robotics: Science and Systems (RSS)*, Berkeley, CA, USA. [Cited p. 7]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014b). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2997–3004, Chicago, Illinois, USA. [Cited p. 6]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014c). On recursive random prolate hyperspheroids. Technical Report TR-2014-JDG002, Autonomous Space Robotics Lab, University of Toronto. arXiv:1403.7664 [math.ST]. [Cited pp. 6 and 119]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, Seattle, Washington, USA. [Cited p. 7]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2017a). Informed asymptotically optimal anytime search. In preparation. [Cited p. 7]

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2017b). Informed sampling for asymptotically optimal path planning. In preparation. [Cited p. 6]

Gammell, J. D., Tong, C. H., and Barfoot, T. D. (2013a). Blinded by the light: Exploiting the deficiencies of a laser rangefinder for rover attitude estimation. In *Proceedings of the International Conference on Computer and Robot Vision (CRV)*, pages 144–150. [Cited p. 3]

Gammell, J. D., Tong, C. H., Berczi, P., Anderson, S., Barfoot, T. D., and Enright, J. (2013b). Rover odometry aided by a star tracker. In *Proceedings of the IEEE Aerospace Conference*, pages 1–10. [Cited p. 3]

Geraerts, R. and Overmars, M. H. (2004). Sampling techniques for probabilistic roadmap planners. In *Conference on Intelligent Autonomous Systems*, pages 600–609. [Cited p. 23]

Geraerts, R. and Overmars, M. H. (2007). Creating high-quality paths for motion planning. *The International Journal of Robotics Research (IJRR)*, 26(8):845–863. [Cited p. 21]

Gilbert, E. N. (1961). Random plane networks. *Journal of the Society for Industrial and Applied Mathematics (JSIAM)*, 9(4):533–543. [Cited p. 82]

Google (2015). Retrieved from `https://plus.google.com/+SelfDrivingCar/posts/9WBWP2E4GDu`. [Cited p. 3]

Hammill, R. and Hendricks, M. (2014). More dreaded chores outsourced to robots (they do windows). *The New York Times*. Retrieved from `http://www.nytimes.com/2014/03/27/technology/personaltech/they-do-windows-more-dreaded-chores-being-outsourced-to-robots.html`. [Cited p. 1]

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107. [Cited pp. 2, 8, 13, 14, 28, 29, 79, and 108]

Hauser, K. (2015). Lazy collision checking in asymptotically-optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2951–2957. [Cited pp. 78, 86, and 106]

Hauser, K. and Ng-Thow-Hing, V. (2010). Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2493–2498. [Cited p. 21]

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, 26:191–246. [Cited pp. 78, 86, and 106]

Holleman, C. and Kavraki, L. E. (2000). A framework for using the workspace medial axis in PRM planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1408–1413. [Cited p. 23]

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255. [Cited pp. 18 and 19]

Hsu, D., Latombe, J.-C., and Motwani, R. (1997). Path planning in expansive configuration spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2719–2726. [Cited pp. 18 and 80]

Hsu, D., Latombe, J.-C., and Motwani, R. (1999a). Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications (IJCGA)*, 9(4–5):495–512. [Cited pp. 18 and 80]

Hsu, D., Latombe, J.-C., and Sorkin, S. (1999b). Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP)*, pages 280–285. [Cited p. 21]

Ingerman, P. Z. (1962). Algorithm 141: Path matrix. *Communications of the Association for Computing Machinery (CACM)*, 5(11):556. [Cited p. 10]

iRobot (2015). Retrieved from `http://media.irobot.com/index.php?s=34135&cat=4`. [Cited p. 1]

Islam, F., Narayanan, V., and Likhachev, M. (2015). Dynamic multi-heuristic A*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2376–2382. [Cited p. 14]

Jaillet, L., Cortés, J., and Siméon, T. (2008). Transition-based RRT for path planning in continuous cost spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2145–2150. [Cited p. 23]

Jaillet, L., Cortés, J., and Siméon, T. (2010). Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics (TRO)*, 26(4):635–646. [Cited p. 23]

Janson, L., Ichter, B., and Pavone, M. (2015a). Deterministic sampling-based motion planning: Optimality, complexity, and performance. In *Proceedings of the the International Symposium on Robotics Research (ISRR)*. [Cited p. 18]

Janson, L. and Pavone, M. (2013). Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions. In *Proceedings of the International Symposium on Robotics Research (ISRR)*. [Cited pp. 25, 26, and 81]

Janson, L., Schmerling, E., Clark, A., and Pavone, M. (2015b). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research (IJRR)*, 34(7):883–921. [Cited pp. 6, 24, 25, 26, 29, 55, 63, and 81]

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574. [Cited p. 8]

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems (RSS)*. [Cited pp. 21, 22, and 31]

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894. [Cited pp. 3, 8, 19, 20, 21, 22, 23, 28, 29, 31, 34, 56, 62, 88, and 91]

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the RRT*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1478–1483. [Cited p. 57]

Kavraki, L. E., Kolountzakis, M. N., and Latombe, J.-C. (1998). Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171. [Cited pp. 19 and 23]

Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580. [Cited pp. 3 and 18]

Kelly, D. (2014). *Wired*. Retrieved from `http://www.wired.com/2014/12/amazon-reveals-robots-heart-epic-cyber-monday-operation#slide-8`. [Cited p. 1]

Kiesel, S., Burns, E., and Ruml, W. (2012). Abstraction-guided sampling for motion planning. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SoCS)*. [Cited pp. 23, 55, and 81]

Kim, D., Lee, J., and eui Yoon, S. (2014). Cloud RRT*: Sampling cloud based RRT*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2519–2526. [Cited pp. 23 and 55]

Kim, M.-C. and Song, J.-B. (2015). Informed RRT* towards optimality by reducing size of hyperellipsoid. In *Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 244–248. [Cited pp. 75 and 110]

Koenig, S. and Likhachev, M. (2005). Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics (TRO)*, 21(3):354–363. [Cited p. 15]

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence (AI)*, 155(1–2):93–146. [Cited pp. 6, 16, 29, and 92]

Kuffner Jr., J. J. and LaValle, S. M. (2000). RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001. [Cited pp. 6, 20, and 21]

Kunz, T., Thomaz, A., and Christensen, H. (2016). Hierarchical rejection sampling for informed kinodynamic planning in high-dimensional spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 89–96. [Cited pp. 51, 86, and 110]

LaFrance, A. (2015). The high-stakes race to rid the world of human drivers. *The Atlantic*. Retrieved from `http://www.theatlantic.com/technology/archive/2015/12/driverless-cars-are-this-centurys-space-race/417672/`. [Cited p. 1]

Lan, M., Lai, S., Bi, Y., Qin, H., Li, J., Lin, F., and Chen, B. M. (2016). BIT*-based path planning for micro aerial vehicles. In *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 6079–6084. [Cited pp. 107 and 111]

Larson, R. E. (1967). A survey of dynamic programming computational procedures. *IEEE Transactions on Automatic Control (TAC)*, 12(6):767–774. [Cited p. 10]

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Department, Iowa State University. [Cited pp. 18, 20, and 114]

LaValle, S. M. and Kuffner Jr., J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400. [Cited pp. 3, 18, and 20]

Lee, H. B., Kwak, H., Kim, J., Lee, C., and Kim, H. J. (2016). Improvement of online motion planning based on RRT* by modification of the sampling method. *Journal of Institute of Control, Robotics and Systems*, 22(3):192–198. [Cited pp. 75 and 110]

Leven, P. and Hutchinson, S. (2003). Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Transactions on Robotics and Automation (TRO)*, 19(6):1020–1026. [Cited p. 23]

l'Hôpital, G. (1696). *Analyse des Infiniment Petits pour l'Intelligence des Lignes Courbes*. [Cited p. 125]

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005). Anytime dynamic A*: An anytime, replanning algorithm. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS)*. [Cited p. 16]

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2008). Anytime search in dynamic graphs. *Artificial Intelligence (AI)*, 172(14):1613–1643. [Cited p. 16]

Likhachev, M., Gordon, G. J., and Thrun, S. (2003). ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, pages 767–774. [Cited p. 16]

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120. [Cited p. 9]

Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the Association for Computing Machinery (CACM)*, 22(10):560–570. [Cited p. 17]

Lynch, K. M. and Mason, M. T. (1996). Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research (IJRR)*, 15(6):533–556. [Cited p. 14]

Marble, J. D. and Bekris, K. E. (2013). Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics (TRO)*, 29(2):432–444. [Cited p. 23]

Marble, J. D. and Bekris, K. E. (2017). Asymptotically near-optimal is good enough for motion planning. In Christensen, H. I. and Khatib, O., editors, *Robotics Research: The 15th International Symposium (ISRR)*. Springer International Publishing. [Cited p. 23]

Moore, E. F. (1957). The shortest path through a maze. In *Proceedings of the International Symposium on Switching Theory*, pages 285–292. Harvard Univ. Press, Cambridge, Mass. [Cited p. 10]

Morali, J. and Willis, V. (1978). Y.M.C.A. *Cruisin'*. Casablanca Records NBLP 7118. [Cited pp. xii and 72]

Muthukrishnan, S. and Pandurangan, G. (2005). The bin-covering technique for thresholding random geometric graph properties. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 989–998. [Cited p. 82]

Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S. (2013). RRT*-Smart: A rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10:299. [Cited pp. 23 and 55]

National Aeronautics and Space Administration (NASA) Jet Propoulsion Laboratory (JPL) (2012). Retrieved from `http://mars.nasa.gov/mer/gallery/press/opportunity/20120117a.html`. [Cited p. 2]

Nilsson, N. J. (1969). A mobile automaton: an application of artificial intelligence techniques. In *Proceedings of the International Joint Conference on Artificial intelligence (IJCAI)*, pages 509–520. [Cited p. 14]

Nissoux, C., Siméon, T., and Laumond, J.-P. (1999). Visibility based probabilistic roadmaps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1316–1321. [Cited p. 23]

Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., and Galceran, E. (2016). Continuous-time trajectory optimization for online UAV replanning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339. [Cited pp. 75 and 110]

Otte, M. and Correll, N. (2013). C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics (TRO)*, 29(3):798–806. [Cited pp. 30, 40, 42, 46, 47, 56, 64, 73, and 109]

Otte, M. and Frazzoli, E. (2016). RRT$^{\text{X}}$: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research (IJRR)*, 35(7):797–822. [Cited p. 58]

Penrose, M. (2003). *Random Geometric Graphs*, volume 5 of *Oxford Studies in Probability*. Oxford University Press. [Cited pp. 5, 28, and 82]

Perez, A., Karaman, S., Shkolnik, A., Frazzoli, E., Teller, S., and Walter, M. R. (2011). Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4307–4313. [Cited pp. 68 and 99]

Persson, S. M. and Sharf, I. (2014). Sampling-based A* algorithm for robot path-planning. *The International Journal of Robotics Research (IJRR)*, 33(13):1683–1708. [Cited pp. 24 and 80]

Pivtoraiko, M. and Kelly, A. (2005). Efficient constrained path planning via search in state lattices. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS)*. [Cited p. 17]

Pivtoraiko, M., Knepper, R. A., and Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics (JFR)*, 26(3):308–333. [Cited p. 17]

Pohl, I. (1970a). First results on the effect of error in heuristic search. *Machine Intelligence*, 5:219–236. [Cited pp. 14 and 15]

Pohl, I. (1970b). Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204. [Cited pp. 14 and 15]

Pohl, I. (1973). The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 12–17. [Cited pp. 14 and 15]

Primatesta, S., Russo, L. O., and Bona, B. (2016). Dynamic trajectory planning for mobile robot navigation in crowded environments. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. [Cited pp. 75 and 110]

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494. [Cited p. 8]

Rickert, M., Brock, O., and Knoll, A. (2008). Balancing exploration and exploitation in motion planning. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (IROS)*, pages 2812–2817. [Cited p. 23]

Roy, B. (1959). Transitivité et connexité. *Comptes Rendus Hebdomadaires Des Seances De L'Academie Des Sciences*, 249(2):216–218. [Cited p. 10]

Sallaberger, C. S. and D'Eleuterio, G. M. (1995). Optimal robotic path planning using dynamic programming and randomization. *Acta Astronautica*, 35(2–3):143–156. [Cited pp. 17 and 79]

Salzman, O. and Halperin, D. (2014). Asymptotically near-optimal RRT for fast, high-quality, motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4680–4685. [Cited pp. 23 and 58]

Salzman, O. and Halperin, D. (2015). Asymptotically-optimal motion planning using lower bounds on cost. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4167–4172. [Cited pp. 26 and 81]

Sánchez, G. and Latombe, J.-C. (2002). On delaying collision checking in PRM planning: Application to multi-robot coordination. *The International Journal of Robotics Research (IJRR)*, 21(1):5–26. [Cited pp. 78 and 86]

Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., and Abbeel, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, Berlin, Germany. [Cited p. 8]

Sekhavat, S., Švestka, P., Laumond, J.-P., and Overmars, M. H. (1998). Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *The International Journal of Robotics Research (IJRR)*, 17(8):840–857. [Cited p. 21]

Sethian, J. A. (1996). Theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica*, 5:309–395. [Cited p. 25]

Shan, Y. X., Li, B. J., Zhou, J., and Zhang, Y. (2014). An approach to speed up RRT*. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 594–598. [Cited p. 58]

Siméon, T., Laumond, J.-P., and Nissoux, C. (2000). Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493. [Cited p. 23]

Srinivasa, S., Berenson, D., Cakmak, M., Collet Romea, A., Dogar, M., Dragan, A., Knepper, R. A., Niemueller, T. D., Strabala, K., Vandeweghe, J. M., and Ziegler, J. (2012). HERB 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of the IEEE*, 100(8):1–19. [Cited pp. 3, 5, 72, 102, and 109]

Starek, J. A., Gomez, J. V., Schmerling, E., Janson, L., Moreno, L., and Pavone, M. (2015). An asymptotically-optimal sampling-based algorithm for bi-directional motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2072–2078. [Cited p. 26]

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317. [Cited p. 15]

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659. [Cited p. 15]

Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. Library available from `http://ompl.kavrakilab.org/`. [Cited pp. 4 and 109]

Sun, H. and Farooq, M. (2002). Note on the generation of random points uniformly distributed in hyper-ellipsoids. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 489–496. [Cited pp. viii, 42, 45, and 116]

Sun, Z., Hsu, D., Jiang, T., Kurniawati, H., and Reif, J. H. (2005). Narrow passage sampling for probabilistic roadmap planning. *IEEE Transactions on Robotics (TRO)*, 21(6):1105–1115. [Cited p. 23]

Teniente, E. H. and Andrade-Cetto, J. (2013). HRA*: Hybrid randomized path planning for complex 3D environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1766–1771. [Cited pp. 24 and 80]

Thomas, S., Morales, M., Tang, X., and Amato, N. M. (2007). Biasing samplers to improve motion planning performance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1625–1630. [Cited p. 23]

Urmson, C. and Simmons, R. (2003). Approaches for heuristically biasing RRT growth. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1178–1183. [Cited pp. 20, 24, and 81]

van den Berg, J. P. and Overmars, M. H. (2004). Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 453–460. [Cited p. 23]

Voronoi, G. F. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 133:97–178. [Cited p. 19]

Wahba, G. (1965). A least squares estimate of satellite attitude. *Society for Industrial and Applied Mathematics (SIAM) Review*, 7:409. [Cited p. 44]

Wang, W., Balkcom, D., and Chakrabarti, A. (2015). A fast online spanner for roadmap construction. *The International Journal of Robotics Research (IJRR)*, 34(11):1418–1432. [Cited p. 23]

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the Association for Computing Machinery (JACM)*, 9(1):11–12. [Cited p. 10]

Wilmarth, S. A., Amato, N. M., and Stiller, P. F. (1999). MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1024–1031. [Cited p. 23]

Wingfield, N. (2015). Amazon wins approval to test delivery drones outdoors. *The New York Times*. Retrieved from `http://www.nytimes.com/2015/03/20/technology/amazon-wins-approval-to-test-delivery-drones-outdoors.html`. [Cited p. 1]

Wohlsen, M. (2014). Amazon reveals the robots at the heart of its epic cyber monday operation. *Wired*. Retrieved from `http://www.wired.com/2014/12/amazon-reveals-robots-heart-epic-cyber-monday-operation`. [Cited p. 1]

Xie, C., van den Berg, J., Patil, S., and Abbeel, P. (2015). Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4187–4194. [Cited pp. 106 and 111]

Yang, Y. and Brock, O. (2004). Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4405–4410. [Cited p. 23]

Yeh, H.-Y. C., Thomas, S., Eppstein, D., and Amato, N. M. (2012). UOBPRM: A uniformly distributed obstacle-based PRM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2655–2662. [Cited p. 23]

Yershova, A., Jaillet, L., Siméon, T., and LaValle, S. M. (2005). Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3856–3861. [Cited p. 23]

Zucker, M., Kuffner, J., and Bagnell, J. A. (2008). Adaptive workspace biasing for sampling-based planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3757–3762. [Cited p. 23]

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research (IJRR)*, 32(9–10):1164–1193. [Cited pp. 8 and 106]