Leveraging multiple sources of information to search continuous spaces

Marlin P. Strub

Lady Margaret Hall

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2021

Dedicated to

Josi, Bea, and Moris.

Acknowledgments

This thesis and the research it contains were made possible by the support of my colleagues, family, and friends. I take pleasure in this opportunity to publicly acknowledge their contributions and express my heartfelt gratitude for their help.

I want to start by thanking my supervisor, Dr. Jonathan D. Gammell. Your extensive expertise, inexhaustible patience, and impeccable academic integrity have been an immense source of inspiration throughout my studies. Other doctoral students sometimes seemed to have giants standing on their shoulders, but you have always let me stand on yours. For that I will always be grateful.

I want to thank my examiners, Prof. Nick Hawes and Prof. Wheeler Ruml. Thank you for considering my work, your feedback on my research is sincerely appreciated. In a similar vein, I want to thank Dr. Ioannis Havoutis. Thank you for your valuable comments on my research progress at Oxford.

I also want to thank the numerous other exceptional people at ORI that have influenced my thinking and enriched my D.Phil. experience. Special thanks go to Kevin and Rowan. Thank you for taking me under your wings. I also want to thank Irene Yang and Prof. Stephen J. Mellon of the Oxford Orthopaedic Engineering Centre. Your expertise on knee replacements has had great impact on my research.

I want to thank my collaborators at JPL, Mike, Issa, Travis, and Jacob. Thank you for the opportunity to work on such a cool project and for welcoming me to your incredible team. I often reminisce about the field trial in the Mojave Desert.

Next I want to thank my friends and housemates Costa and Ewa. I consider myself the luckiest person in Oxford to have lived with you and will never forget your generosity. I want you to know that I have truly cherished our time together.

Finally I want to extend my deepest gratitude to my family, Bea, Josi, and Moris, my partner, Rahel, and my closest friends, Alessandro, Laurenz, Luca, Diego, and Yves. To Bea, Josi, and Moris, I am forever grateful to be part of such a supporting, understanding, and loving family. You are my biggest sources of strength and I love you with all my heart. To Rahel, your support, understanding, and love have made my studies infinitely more enjoyable. I love you to Mars and back... To Alessandro, Laurenz, Luca, Diego, and Yves, your friendships mean more to me than you will ever know.

Abstract

Path planning algorithms can solve the problem of finding paths through continuous spaces. This problem appears in a wide range of applications, from navigating autonomous robots to automating assessments of surgical tolerances. The performance requirements on these algorithms tend to become more demanding as the problems they are applied to become more sophisticated. This simultaneous increase in performance requirements and application complexity calls for new approaches to the path planning problem and makes it an active area of research in robotics and beyond.

This thesis demonstrates how different types of information can be leveraged to solve the path planning problem more effectively. Optimization-specific information can guide the search towards high-quality solutions, environment-specific information can exploit incremental information about the surroundings, and intent-specific information can directly align the search of a problem with its priorities.

These three types of information are leveraged in this thesis by integrating advanced graph-search techniques in sampling-based path planning algorithms. The resulting planners, Advanced BIT* (ABIT*), Adaptively Informed Trees (AIT*), and Effort Informed Trees (EIT*), are theoretically shown to be almost-surely asymptotically optimal and experimentally demonstrated to outperform existing planners on diverse problems in abstract, robotic, and biomedical domains.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AA*	Anytime A*
AD*	Anytime D*
A-MHA*	Anytime Multi-Heuristic A*
ABIT*	Advanced BIT*
AEES	Anytime Explicit Estimation Search
AIT*	Adaptively Informed Trees
ANA*	Anytime Nonparametric A*
ARA*	Anytime Repairing A*
ARCRAS	Annual Review of Control, Robotics, and Autonomous Systems
ATD*	Anytime Truncated D*
BEAST	Bayesian Effort-Aided Search Trees
BFMT*	Bidirectional FMT*
BHFFA	Bidirectional Heuristic Front-to-Front Algorithm
BHPA	Bidirectional Heuristic Path Algorithm
BIT*	Batch Informed Trees
BIDA*	Bidirectional Iterative-Deepening A*
BVP	Boundary Value Problem
C-PRM	Customizable PRM
CAD	Computer-Aided Design
CDF	Cumulative Distribution Function
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
EES	Explicit Estimation Search
EIT*	Effort Informed Trees

EST	Expansive Space Trees
FCL	Flexible Collision Library
FMT*	Fast Marching Trees
GP	Generalized Pohl
HA*	Hierarchical A*
HCA*	Hierarchical Cooperative A*
ICRA	International Conference on Robotics and Automation
IDA*	Iterative-Deepening A*
IJRR	International Journal of Robotics Research
IROS	International Conference on Intelligent Robots and Systems
IDBHFFA	Iterative Deepening BHFFA
LWA*	Lazy Weighted A*
LPA*	Lifelong Planning A*
MHA*	Multi-Heuristic A*
MPLB	Motion Planning using Lower Bounds
NBS	Near-optimal Bidirectional Search
OBB	Oriented Bounding Box
OMPL	Open Motion Planning Library
OpenRAVE	Open Robotics Automation Virtual Environment
ORI	Oxford Robotics Institute
PRM	Probabilistic Roadmaps
PDF	Probability Density Function
PS	Perimeter Search
RABIT*	Regionally Accelerated BIT*
RGG	Random Geometric Graph
RRA*	Reverse Resumable A*
RRT	Rapidly-exploring Random Trees
RSS	Rectangle Swept Sphere
RWA*	Restarting Weighted A*

SA$^*_\varepsilon$	Simplified A$^*_\varepsilon$
SAT	Separating Axis Theorem
SEE	Surface Edge Explorer
SEES	Simplified EES
STOMP	Stochastic Trajectory Optimization for Motion Planning
TLPA*	Truncated Lifelong Planning A*
UKR	Unicompartmental Knee Replacement
WA*	Weighted A*
WAM	Whole-Arm Manipulator
WHCA*	Windowed HCA*

Notation

General

a	Lower-case variables in this font are scalars.
$\mathbf{a}$	Lower-case variables in this font are vectors.
A	Upper-case variables in this font are sets or spaces.
$\mathcal{A}$	Upper-case variables in this font are special data structures, e.g., graphs or queues.

Special symbols and scalars

$\emptyset$	The empty set.
∞	The symbol for infinity.
ε_i	An inflation factor greater or equal to one.
ε_t	A truncation factor greater or equal to one.
η	A scaling factor for connection parameters.
q	The number of samples generated by a planning algorithm.
n	The dimension of the search space.
m	The number of samples per batch in ABIT*, AIT*, and EIT*.
ρ	A (sparse) collision detection resolution.
c^*	The optimal solution cost of a planning problem.
c_current	The cost of the current best solution found by a planning algorithm.
t	A variable to denote the progress along a path.

Vectors

$\mathbf{x}$	A state in the search space.
$\mathbf{x}_\mathrm{start}$	The start state of a planning problem.
$\mathbf{x}_\mathrm{goal}$	A goal state of a planning problem.
$\mathbf{x}_\mathrm{s}$	The source state of an edge.
$\mathbf{x}_\mathrm{t}$	The target state of an edge.

Sets, spaces, data structures, and set operations

X	The search space of a planning problem.
X_{valid}	The set of valid states of a planning problem.
X_{invalid}	The set of invalid states of a planning problem.
X_{goal}	The goal states of a planning problem.
X_{sampled}	The sampled states of a planning algorithm.
Σ	A set of paths.
$X_{\hat{f}}$	The informed set.
$V_{\text{inconsistent}}$	The set of inconsistent vertices.
V	A set of vertices.
E	A set of edges.
$\mathcal{F}, \mathcal{R}, \mathcal{T}$	Graphs without cycles, i.e., trees.
$\mathcal{Q}$	A queue.
$A \xleftarrow{+} B$	Abbreviation for adding set B to set A, i.e., $A \leftarrow A \cup B$.
$A \xleftarrow{-} B$	Abbreviation for removing set B from set A, i.e., $A \leftarrow A \setminus B$.

Unary functions

$\sigma(\cdot)$	A path whose parameter is a variable between 0 and 1.
$H(\cdot)$	A homotopic map between two paths.
$\text{TV}(\cdot)$	The total variation of a path.
$\lambda(\cdot)$	The Lebesgue measure of a set.
$P(\cdot)$	The probability of an event.
$c(\cdot)$	The cost of a path.
$g_{\mathcal{T}}(\cdot)$	The cost to reach a state through the tree $\mathcal{T}$.
$\bar{d}(\cdot)$	An inadmissible estimate of the effort to reach a state from the start.
$\hat{g}(\cdot)$	An admissible estimate of the cost to reach a state from the start.
$\hat{h}(\cdot)$	An admissible estimate of the cost to reach the goal from a state.
$\hat{f}(\cdot)$	An admissible estimate of the solution cost when that solution is constrained to go through a state, often $\hat{f}(\cdot) \coloneqq \hat{g}(\cdot) + \hat{h}(\cdot)$.

Binary functions

$c(\cdot, \cdot)$	The path cost between two states.
$\hat{c}(\cdot, \cdot)$	An admissible estimate of the path cost between two states.
$\bar{c}(\cdot, \cdot)$	An inadmissible estimate of the path cost between two states.
$\bar{e}(\cdot, \cdot)$	An inadmissible estimate of the path effort between two states.

Chapter 1

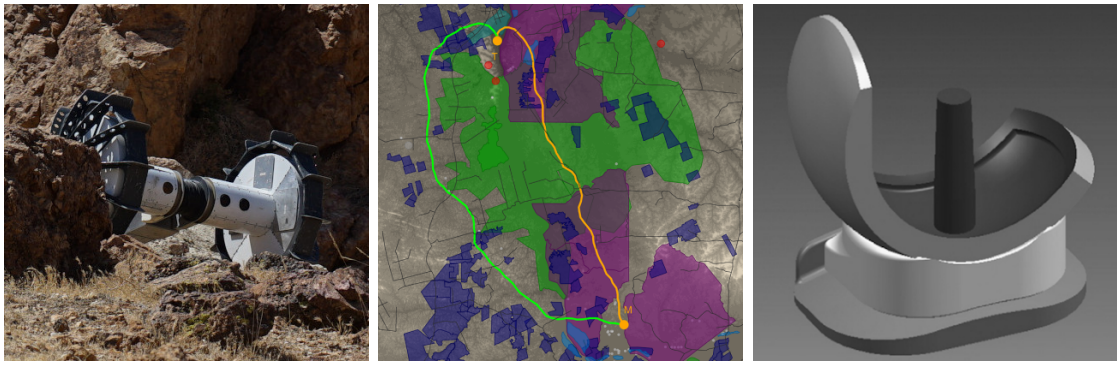
Introduction

Finding paths through continuous spaces

Path planning is about finding paths through continuous spaces that possibly contain obstacles. It is a fundamental task in a wide variety of applications, such as navigating next-generation rovers designed for Mars (Figure 1.1a; Paton et al., 2020), optimizing routes for power transmission lines (Figure 1.1b; Gonçalves et al., 2021), and assessing tolerances for knee replacements (Figure 1.1c; Yang et al., 2020). Many recent applications operate in increasingly complex environments and require reliably finding high-quality paths in short amounts of time.

This can be challenging for various reasons. Modern applications often have high-dimensional search spaces, are governed by complicated constraints, contain adverse obstacle configurations, or require computationally expensive cost evaluation or collision detection. These characteristics are frequently combined, and their difficulties compounded, such that many modern applications give rise to *path planning problems* that can currently not be solved in an exact manner.

Fortunately, approximate solutions often suffice. Approaches to approximately solving path planning problems include limiting the search to the surroundings of an initial solution-guess (i.e., optimization-based solvers), solving a discretized version of the problem at an *a priori* resolution (i.e., graph-based searches), and finding



(a) (b) (c)

Figure 1.1: Path planning appears in various applications, such as navigating next-generation rovers destined for Mars (a), planning optimal routes for power transmission lines (b), and assessing surgical tolerances for a knee replacement (c). Illustrations courtesy of NASA/JPL, Gonçalves et al. (2021), and Pandit et al. (2010), respectively.

paths by sampling and connecting individual states (i.e., sampling-based planners). These approaches have complementary strengths, but unifying them is challenging.

For example, popular graph-search algorithms, such as A* (Hart et al., 1968), offer principled and efficient searches but cannot directly be applied to continuous search spaces. Popular sampling-based planners, such as Rapidly-exploring Random Trees (RRT; LaValle and Kuffner Jr., 1999, 2001) and its asymptotically optimal extension, RRT* (Karaman and Frazzoli, 2010a, 2011), have random and inefficient searches but can directly be applied to continuous search spaces.

A recent effort that successfully incorporates some strengths of graph-based searches in sampling-based planning achieves this by viewing its sampled states as vertices of a graph that is embedded in the search space of a planning problem (Gammell, 2017). This perspective returns sampling-based planners to separating the approximation of the search space from the search of this approximation, which allows researching both subproblems independently. This thesis adopts this perspective and focuses on the search of sampling-based approximations. The presented algorithms can thereby outperform existing planners in many applications by leveraging insights and techniques from decades of research on graph-based searches and sampling-based approximations.

The fundamental insight this thesis builds on is that additional, problem-specific information can greatly improve search performance. For example, when searching for the shortest path through a road network, A* can leverage the fact that the shortest (partial) solution starting from any vertex in the network cannot be shorter than the straight-line distance between that vertex and the goal. Such seemingly trivial information enables *informed* algorithms (i.e., algorithms that can leverage problem-specific information) to solve problems that are unsolvable with *uninformed* algorithms (i.e., algorithms that cannot leverage problem-specific information).

This information about the (partial) solution cost is not the only type of problem-specific information that can be leveraged to improve search performance. Complementary information can provide additional benefits and this thesis demonstrates how such information can be leveraged in sampling-based path planning. The algorithms presented in this thesis leverage up to three different types of problem-specific information, which are introduced in the following three paragraphs:

Optimization-specific information Path planning problems that require high-quality solutions must provide a way to evaluate solution quality. This is often achieved by defining a *cost function*, i.e., a function that maps paths to scalars (e.g., path length). Many cost functions have properties that can be exploited when solving a specific path planning problem. This thesis calls information about such properties *optimization-specific information*. An example of such information is a *cost heuristic*, i.e., a function that approximates the true cost of a path (e.g., as the straight-line distance between the endpoints of the path when minimizing path length).

Environment-specific information Path planning problems are specific to the obstacles of the environments they operate in and must provide a way to determine whether individual states are in collision with these obstacles. This partitions the search space into the collectively exhaustive but mutually exclusive sets of *valid* and *invalid* states which correspond to the obstacles in the environment.

Many obstacle configurations have properties that can be exploited when solving a specific path planning problem. This thesis calls information about such properties *environment-specific information*. An example of such information is the distribution of valid and invalid states in the search space (e.g., as observed by incrementally checking samples for collision).

Intent-specific information Path planning problems are often solved with specific priorities. For example, mobile robots may prioritize solution time over solution quality due to hard time constraints. Algorithms designed for such problems can improve their performance by reflecting these priorities in their searches. This thesis calls information about such priorities *intent-specific information*. An example of such information for applications that prioritize solution times is an *effort heuristic*, i.e., a function that approximates the computational effort required to find a solution (e.g., by estimating the number of remaining collision checks).

These sources of information are leveraged in the algorithms presented in this thesis by building on Batch Informed Trees (BIT*; Gammell et al., 2015, 2020). BIT* is a single-query sampling-based path planning algorithm that solves problems by simultaneously approximating the search space with a sampling-based approximation and searching this approximation with an informed graph-based search.

BIT* approximates the search space by sampling multiple batches of states and viewing these samples as a series of increasingly dense, edge-implicit Random Geometric Graphs (RGGs; Penrose, 2003). This series of RGG approximations is focused to the relevant region of the search space by only sampling states that could potentially improve the current solution and pruning samples that no longer satisfy this requirement. BIT* achieves this by leveraging optimization-specific information in the form of a cost heuristic that provides a lower bound on the optimal solution cost between two states (Gammell et al., 2018).

BIT* also leverages such a cost heuristic to order its search on the total potential

solution costs of the edges in its RGG approximations. The total potential solution cost of an edge is computed as the sum of the current cost-to-come to the source state of the edge, a heuristic estimate of the edge cost, and a heuristic estimate of the cost-to-go to the goal from the target of the edge (i.e., similar to an edge-queue version of A*). This results in an efficient algorithm that almost-surely asymptotically finds optimal solutions to (continuous) path planning problems.

Optimization-specific information can be leveraged in sampling-based planning for more than focusing sampling-based approximations and estimating total potential solution costs of edges. Chapter 3 presents Advanced BIT* (ABIT*; Strub and Gammell, 2020b), which builds on BIT* by using advanced graph-search techniques, such as inflation and truncation, to further leverage the benefits of optimization-specific information in sampling-based planning.

ABIT* achieves this by inflating the estimates of the cost-to-go from the target of an edge when ordering its search. This biases the search towards the goal and often results in faster initial solution times. ABIT* additionally truncates the search when it is guaranteed that the current RGG approximation does not contain a solution that is significantly better than the best solution found so far. This gives ABIT* fine control over when it improves its RGG approximation, which allows it to balance the exploration of the search space with the exploitation of its current approximation.

BIT* and ABIT* use optimization-specific information but fail to guide their search with the environment-specific information that they gain from collision detection. Chapter 4 presents Adaptively Informed Trees (AIT*; Strub and Gammell, 2020a, 2021b), which uses a series of similar RGG approximations as BIT* but leverages both optimization- and environment-specific information to search it.

AIT* achieves this with a hierarchical bidirectional search that is asymmetric both in purpose and in computational cost and simultaneously calculates and exploits an approximation-specific cost heuristic. The inexpensive reverse search does not check edges for collision but uses the connectivity of the current RGG

Type of problem-specific information	RRT*	BIT*	ABIT*	AIT*	EIT*
Optimization-specific	✓	✓	✓✓	✓	✓
Environment-specific	✗	✗	✗	✓	✓
Intent-specific	✗	✗	✗	✗	✓

Table 1.1: An overview of the different types of problem-specific information leveraged by the searches of RRT*, BIT*, ABIT*, AIT*, and EIT*. A hollow check mark (✓) is used to indicate local or incomplete utilization of the specific type of information, a check mark (✓) for global or complete utilization, and a double check mark (✓✓) for additional utilization. A cross (✗) indicates no use of the information.

approximation to combine cost heuristics between multiple samples into a more accurate, approximation-specific cost heuristic between each sample and the goal. The expensive forward search checks edges for collision but is focused on promising paths by the calculated cost heuristic. When the forward search detects a collision on an edge that was used to calculate the heuristic in the reverse search, then the reverse search uses this information to update the calculated heuristic. In this way, the forward and reverse searches in AIT* continuously inform each other with complementary information.

AIT* leverages optimization- and environment-specific information but fails to directly reflect the priorities of a planning problem in the order of its search. Chapter 5 presents Effort Informed Trees (EIT*; Strub and Gammell, 2021b), which uses the same series of RGG approximations as AIT* but leverages optimization-, environment-, and intent-specific information to explicitly align its search with the priorities of the end-user that poses the planning problem.

EIT* achieves this with a similar hierarchical bidirectional search as AIT*, but simultaneously calculates and exploits cost and effort heuristics. As in AIT*, the inexpensive reverse search does not fully check edges for collision but uses the connectivity of the RGG approximation to combine cost and effort heuristics between multiple samples into more accurate cost and effort heuristics between each sample and the goal. The expensive forward search again checks edges for collision but is aligned

with the priorities of the problem by the calculated cost and effort heuristics. Similar to AIT*, if the forward search detects a collision on an edge that was used to calculate a heuristic with the reverse search, then it causes the reverse search to update the heuristic. The forward and reverse searches therefore also continuously inform each other with complementary information in EIT*. Table 1.1 shows an overview of the problem-specific information leveraged by RRT*, BIT*, ABIT*, AIT*, and EIT*.

The benefits of optimization-, environment- and intent-specific information are illustrated in Figure 1.2 and demonstrated on various problems in abstract, nonholonomic, robotic, and biomedical settings (Sections 3.4, 4.4, and 5.4). The results show that ABIT*, AIT*, and EIT* outperform other asymptotically optimal sampling-based planners on most of the tested problems.

The core contributions of this thesis are:

- A review of the relevant literature on graph-based searches and sampling-based planners that lists the advantages and drawbacks of each paradigm (Chapter 2).
- A detailed presentation of ABIT*, which extensively leverages optimization-specific information in sampling-based path planning with advanced graph-search techniques (Chapter 3).
- A detailed presentation of AIT*, which leverages optimization- and environment-specific information with an hierarchical bidirectional search in which both searches continuously inform each other with complementary information (Chapter 4).
- A detailed presentation of EIT*, which extends AIT* by additionally leveraging intent-specific information to directly align its search with the priorities of the problem (Chapter 5).
- Proofs of the almost-sure asymptotic optimality of all presented algorithms that combine established results from the literature on sampling-based planning and graph-based search (Sections 3.3, 4.3, and 5.3).

Versions of this work already resulted in or contributed to six scientific papers which were published in the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA; Strub and Gammell, 2020b,a), in the Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS; Paton et al., 2020), on arXiv (Strub and Gammell, 2021a), in the Annual Review of Control, Robotics, and Autonomous Systems (ARCRAS; Gammell and Strub, 2021), and in the International Journal of Robotics Research (IJRR; Strub and Gammell, 2021b). Reference implementations of ABIT*, AIT*, and EIT* are publicly available in the Open Motion Planning Library (OMPL; Şucan et al., 2012).

The work presented in this thesis has also been used by other researchers. ABIT* was used on two next-generation NASA/JPL-Caltech rovers (Paton et al., 2020; Reid et al., 2020), and AIT* is currently used at the Oxford Robotics Institute (ORI) to plan paths for a UR-10 manipulator arm. EIT* has not yet run on a real-world system, but is being investigated by colleagues at Cornell University and TU Berlin for task and motion planning and multiquery path planning, respectively.

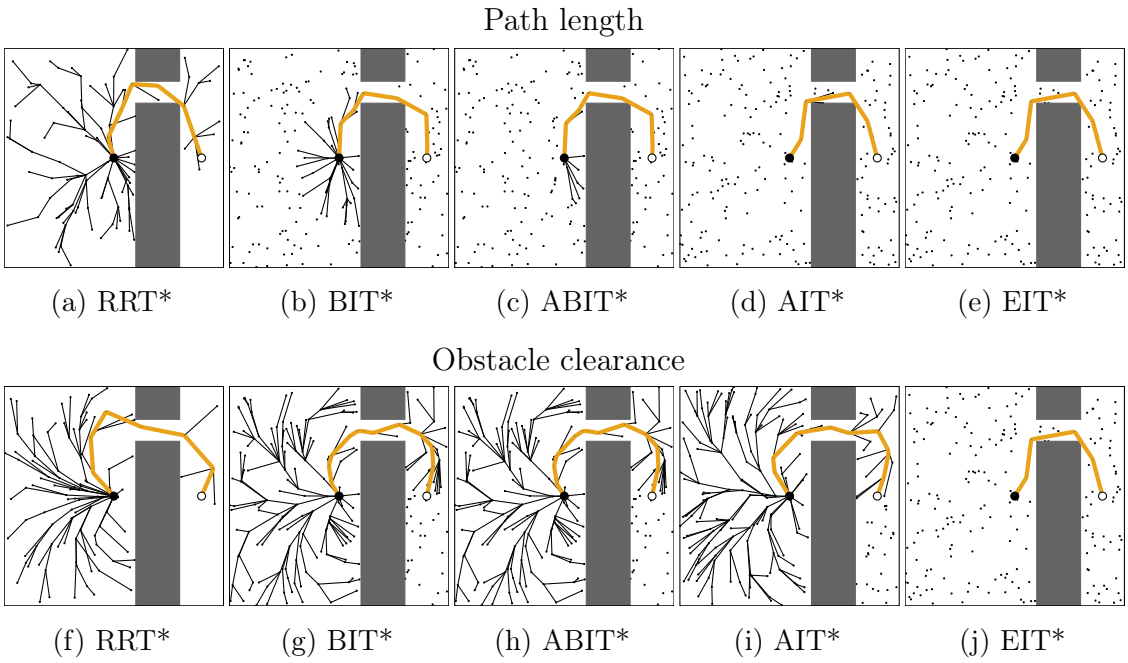


Figure 1.2: An illustration of the benefits of optimization-, environment-, and intent-specific information. The figure shows the search trees constructed by RRT*, BIT*, ABIT*, AIT*, and EIT* to find an initial solution when optimizing path length, where optimization-specific information is available (a–e), and obstacle clearance, where optimization-specific information might not be available (f–j). The start and goal are represented by a black dot (•) and circle (◦), respectively. Sampled states are represented by small black dots (·). Search-space obstacles are indicated in gray (▮). The initial solutions are shown in yellow (—) and the search trees constructed to find them are shown in black (—). Any edge in these search trees that is not part of the initial solution delayed finding it. RRT* does not leverage any problem-specific information to guide its search and fully evaluates many edges that are not part of the initial solution (a, f). BIT* leverages optimization-specific information and finds an initial solution after evaluating fewer edges than RRT*, but only if optimization-specific information is available (b, g). ABIT* leverages the available optimization-specific information more effectively and finds an initial solution after evaluating fewer edges than BIT*, but again only if optimization-specific information is available (c, h). AIT* leverages optimization- and environment-specific information and finds an initial solution after evaluating fewer edges than ABIT*, but yet again only if optimization-specific information is available (d, i). EIT* uses optimization-, environment- and intent-specific information and finds an initial solution after evaluating the fewest edges, regardless of whether optimization-specific information is available (e, j).

Chapter 2

Background

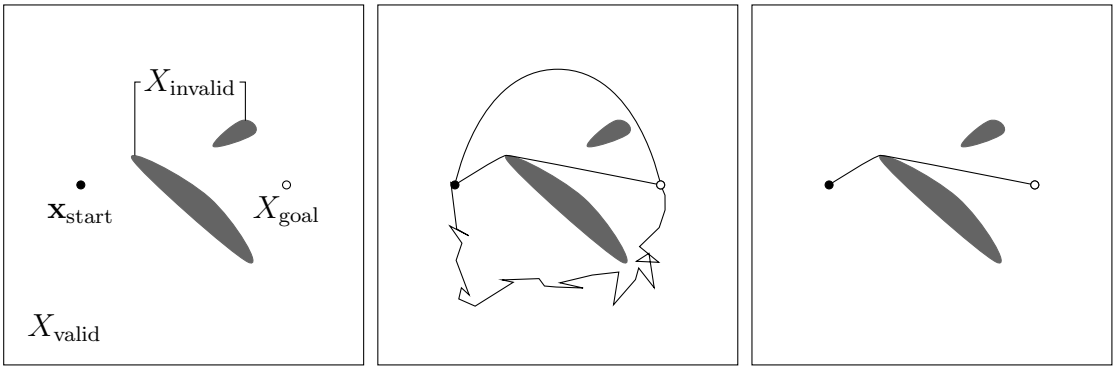
Definitions, related work, and assumptions

Contents

This chapter first introduces and formally defines two versions of the path planning problem (Section 2.1) and then reviews two popular techniques to solve them (Sections 2.2 and 2.3). It then formally defines the performance guarantees of sampling-based planners and states the assumptions required to prove them (Section 2.4).

The two widely studied problems of path planning are called *feasible* and *optimal* path planning. The *feasible* problem is the task of finding a path between a start and a goal that avoids obstacles and obeys system constraints. The *optimal* problem is the task of finding a feasible path that optimizes a given objective.

Both of these versions are often too difficult to solve exactly due to challenging search space properties, such as high dimensionality, complex constraints, adverse obstacle configurations, or computationally expensive collision detection or cost evaluation. One approach to address these challenges is to use optimization-based solvers,



(a) Path planning problem (b) Feasible paths (c) Optimal path

Figure 2.1: An illustration of the feasible and optimal path planning problems. The search space, X, of a path planning problem is partitioned into the collectively exhaustive but mutually exclusive sets of valid states, X_{valid}, and invalid states, X_{invalid}, and contains a start state, $\mathbf{x}_{\text{start}} \in X_{\text{valid}}$, and a goal state or region, $X_{\text{goal}} \subset X_{\text{valid}}$ (a). Any path that connects the start state to a goal state and only goes through valid states solves the feasible path planning problem (b). Only a path that optimizes a given objective, e.g., path length, solves the optimal path planning problem (c).

which limit their search to the neighborhood of an initial solution-guess. Popular 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 TrajOpt (Schulman et al., 2013, 2014). These algorithms can find solutions very quickly but only provide local optimality guarantees and are not discussed in detail in this thesis.

Other approaches to address the challenges of path planning are graph-based searches (Section 2.2) and sampling-based planners (Section 2.3), both of which search a simplified approximation of the search space. These approaches can provide global optimality guarantees, but they only hold either for a given discretization of the search space or asymptotically in the limit of infinite computation time.

This thesis builds on efforts to unify graph-based search and sampling-based planning. This allows the presented sampling-based algorithms to leverage different types of information with informed graph-search algorithms and results in algorithms that can solve the optimal path planning problem almost-surely asymptotically under mild technical assumptions (Section 2.4).

2.1 Path planning problems

There are two widely studied versions of the path planning problem (Figure 2.1). This section provides informal descriptions and formal definitions for both versions and presents intuitive reasons for why they are difficult to solve.

2.1.1 The feasible path planning problem

The *feasible* path planning problem requires finding a sequence of states, i.e., a path, that is collision-free, leads from the start to the goal, and satisfies all system constraints (Figure 2.1b). In robotic path planning the start is the initial state of the robot and the goal is either an individual state, e.g., the desired position of a mobile robot, or a set of multiple states, e.g., all joint angles that result in the same end-effector position of a manipulator arm. The feasible path planning problem is formally defined in Definition 1 (Karaman and Frazzoli, 2011).

Definition 1 (The feasible path planning problem)**.** *Let the search space of a path planning problem be denoted by X, the subset of invalid states by $X_{\text{invalid}} \subset X$, and the subset of valid states by $X_{\text{valid}} \coloneqq \text{closure}\,(X \setminus X_{\text{invalid}})$. Let the start state and the set of goal states be denoted by $\mathbf{x}_{\text{start}} \in X_{\text{valid}}$ and $X_{\text{goal}} \subset X_{\text{valid}}$, respectively. Let $\sigma\colon [0,1] \to X_{\text{valid}}$ be a continuous function with bounded total variation, i.e., a valid path, and let the set of all valid paths be denoted by Σ. The feasible path planning problem is the task of finding any valid path from the start to the goal,*

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

or reporting failure if no such path exists.

Many feasible problems have an infinite number of solutions, but finding any one of them in real-world applications can be challenging for various reasons. Some applications, such as robotic manipulator arms, might have many degrees of freedom

which results in high-dimensional search spaces. These spaces are difficult to search because the number of states required to approximate them with a given dispersion (i.e., sample-density) grows at least exponentially in dimension for any sampling scheme (Sukharev, 1971).

Other applications, such as autonomous cars, might have nonholonomic constraints (i.e., forced couplings between derivatives of state dimensions) or dynamic constraints (i.e., large momentum compared to the controllable forces), both of which can constrain local movement. These spaces are difficult to search because these local constraints increase computational cost and must be taken into account in addition to the global constraints imposed by environmental obstacles.

Still other applications, such as humanoid robots carrying glasses of water, might have constraints placed on their configurations, such as keeping the glasses level throughout their entire motion. These search spaces are difficult to search because the set of all states that satisfy such *manifold* constraints has a measure of zero, which makes it difficult to generate states in this set (Kingston et al., 2018, 2019).

Another category of applications, such as assessing surgical tolerances for medical implants, often require high-resolution collision detection with high-fidelity models of the implants which usually have complex geometries. These spaces are difficult to search because checking for collision between complex geometries is computationally expensive and planning algorithms typically check many states for collision.

These challenges are often combined in path planning problems that emerge from modern applications. For example, the NASA/JPL-Caltech rover mentioned in Chapter 1 (Figure 1.1a), combines complicated system constraints with complex collision detection (Section 3.5). The difficulty of the feasible planning problem is also confirmed in theory, as the problem is proven to be PSPACE-complete (Canny, 1988).

2.1.2 The optimal path planning problem

The feasible path planning problem is neutral about solution quality, but many applications require high-quality paths. The task of finding a feasible solution that optimizes a given objective is called the *optimal path planning problem* (Figure 2.1c).

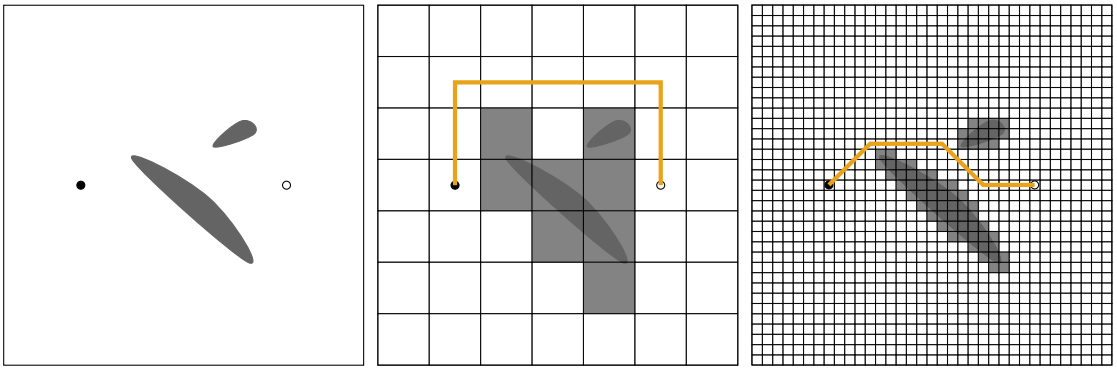
The optimal problem is at least as difficult as the feasible problem as any optimal solution is also a feasible solution. But while feasible problems often have an infinite number of solutions, there usually exist fewer optimal solutions, which makes the optimal problem often significantly more difficult than the feasible problem.

Optimal planning is further complicated simply because it requires evaluating path quality, which can be computationally expensive. For example, optimizing obstacle clearance (i.e., the distance to the nearest obstacle) requires computing the clearance of potential solutions, which is at least as expensive as collision detection, because a state with positive clearance must be collision free. The optimal path planning problem is formally defined in Definition 2 (Karaman and Frazzoli, 2011).

Definition 2 (The optimal path planning problem)**.** *Let the search space of a path planning problem be denoted by X, the subset of invalid states by $X_{\text{invalid}} \subset X$, and the subset of valid states by $X_{\text{valid}} \coloneqq \text{closure}\,(X \setminus X_{\text{invalid}})$. Let the start state and the set of goal states be denoted by $\mathbf{x}_{\text{start}} \in X_{\text{valid}}$ and $X_{\text{goal}} \subset X_{\text{valid}}$, respectively. Let $\sigma\colon [0,1] \to X_{\text{valid}}$ be a continuous function with bounded total variation, i.e., a valid path, and let the set of all valid paths be denoted by Σ. Let the optimization objective be defined by a cost function, $c\colon \Sigma \to [0,\infty)$, that maps each path to a nonnegative real number. The optimal path planning problem is the task of finding a path, $\sigma^* \in \Sigma$, from the start to the goal with minimum cost,*

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

or reporting failure if no such path exists.



(a) (b) (c)

Figure 2.2: An illustration of the difficulty of discretizing continuous path planning problems. Graph-search algorithms cannot directly solve a continuous problem (a). The problem must first be discretized, but selecting a good resolution is difficult *a priori*. A coarse resolution requires few computational resources to be searched but may not contain any solution or only solutions of low quality in the continuous sense of the problem (b). A fine resolution may contain higher-quality solutions, but may require too much computational effort to be searched, especially in high-dimensional search spaces (c).

2.2 Graph-based search

A popular approach to solve path planning problems are graph-based searches. These algorithms solve discretized versions of the path planning problem by searching those discretizations in a principled manner, potentially leveraging information about the optimization objective and the computational effort required for certain search operations. They can provide strong performance guarantees, but these guarantees only hold for the given discretization. For example, Dijkstra's algorithm (Dijkstra, 1959) is guaranteed to find the optimal solution when edge costs are nonnegative, but that solution is only optimal for the given discretization and can be arbitrarily bad in the continuous sense of the original problem.

Graph-based search algorithms require discrete structures and cannot directly be applied to continuous search spaces. Choosing a good discretization *a priori*, i.e, before solving a problem, is difficult (Figure 2.2). If the discretization is too coarse, then the algorithm might return a solution of insufficient quality in the continuous sense or fail to return any solution at all because the resolution is too

low to render small gaps between obstacles. If the discretization is too fine, then the algorithm might take unnecessarily long to return a solution or fail to return any solution at all because it exceeds the allocated computation time or memory, especially in high-dimensional search spaces.

This difficulty can be lessened by discretizing the search space adaptively (Moore and Atkenson, 1995; Yahja et al., 1998; García et al., 2014), searching multiple discretizations with different resolutions simultaneously (Du et al., 2020), or by applying graph-search techniques to anytime, sampling-based approximations. A detailed review of the first two approaches exceeds the scope of this thesis, but the last approach is implemented by the algorithms presented in this thesis and discussed in detail in Chapters 3–5.

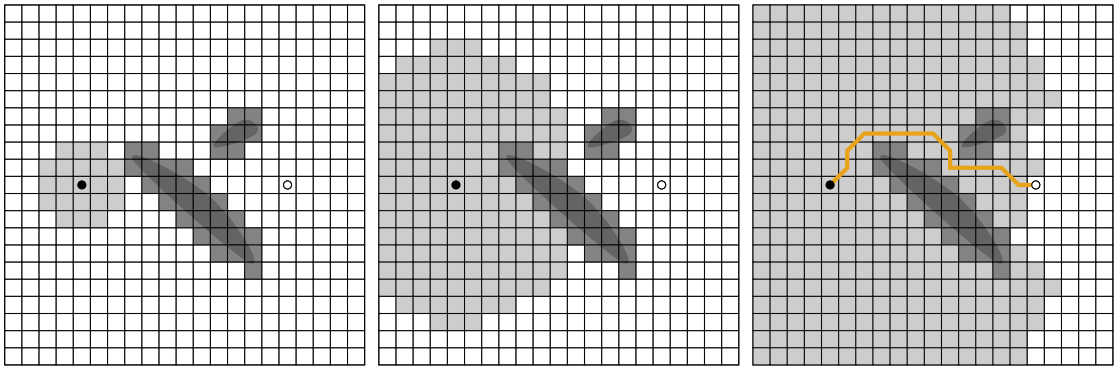
Graph-search algorithms are effective at solving numerous problems and many of these problems have specific characteristics that can be exploited with specialized algorithms. This thesis focuses on the literature that presents the ideas and graph-search algorithms that are most relevant to the work presented in Chapters 3–5.

These ideas and algorithms can be categorized in multiple ways. Algorithms can be categorized according to *what* they aim to achieve (e.g., finding a sufficiently good solution as fast as possible) and *how* they work (e.g., inflating remaining solution-cost estimates when ordering their searches). This thesis primarily categorizes algorithms in terms of what they aim to achieve and secondarily in terms of how they achieve it.

2.2.1 Optimal search

Graph-search algorithms can be used to find the optimal path between two vertices in a graph. For example, given the road network of the United Kingdom, a person in Cambridge might desire to find the shortest path to Oxford.

Dijkstra's algorithm is a common choice to solve such problems. It starts by evaluating the cost to reach each state that is connected to the start (e.g., the



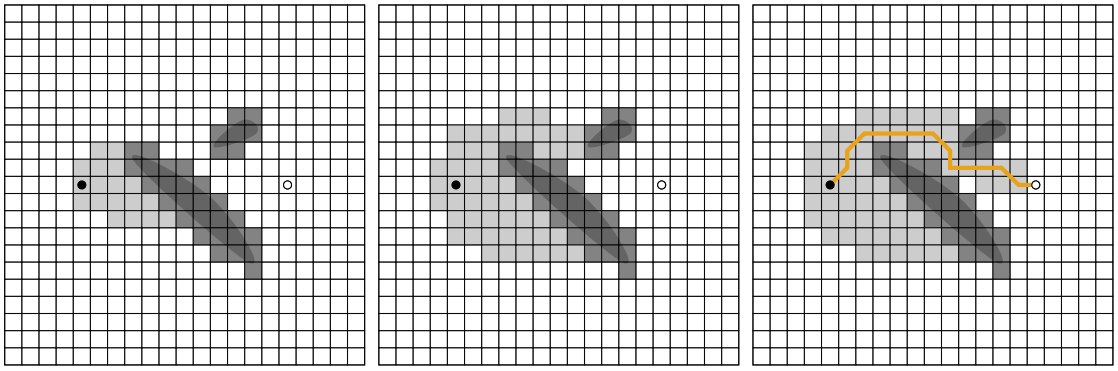
(a) (b) (c)

Figure 2.3: An illustration of how Dijkstra's algorithm searches a discretized planning problem. The search progresses outwards from the start in order of increasing cost-to-come (a, b). Dijkstra's algorithm has found the resolution-optimal solution when the next state to be expanded is the goal (c; Dijkstra, 1959). Note that the number of expanded states would be much larger if the search space had no boundaries.

road-distance from Cambridge). These states are then inserted into a queue and ordered by their cost from the start, called their cost-to-come or g-value. Dijkstra's algorithm then repeatedly selects the state in this queue with the lowest cost-to-come, removes it from the queue, updates the cost-to-come of its neighboring states, and inserts them into the queue (Figure 2.3). The optimal path is found when the vertex that is about to be selected is a goal state (Dijkstra, 1959).

Additional information can improve the performance of graph-search algorithms. For example, two vertices in the road network of the United Kingdom cannot be closer than their air-distance. Such problem-specific information can be leveraged by *informed* search algorithms, such as A* (Hart et al., 1968), to improve on the performance of *uninformed* algorithms, such as Dijkstra's.

A* can leverage such information if it is expressed as a *heuristic* function that provides an estimate of the remaining solution cost from any state, called the cost-to-go or h-value of that state (e.g., the air-distance to Oxford). Such a heuristic is called *admissible* if it never overestimates the true cost and *consistent* if it satisfies a specific triangle inequality. Given an admissible heuristic, A* finds an optimal solution, and given a consistent heuristic, A* does so optimally efficiently with



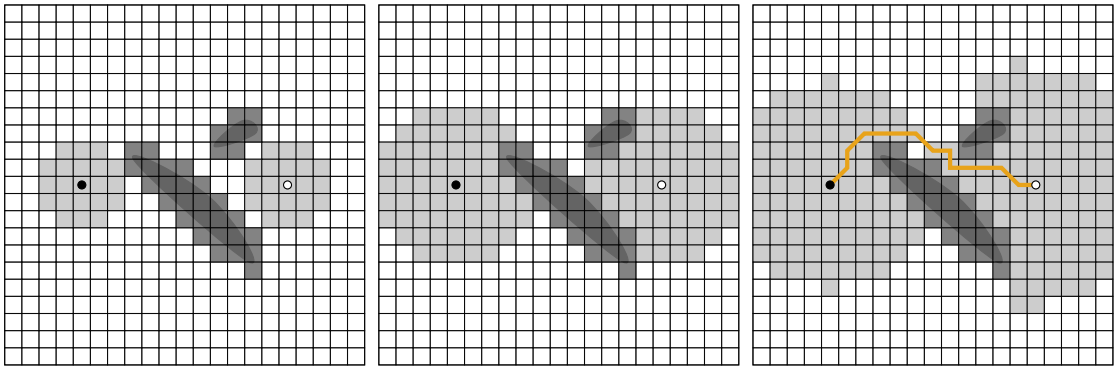
(a) (b) (c)

Figure 2.4: An illustration of how A* searches a discretized planning problem. The search progresses from the start toward the goal in order of increasing total potential solution cost (a, b). As in Dijkstra's algorithm, A* has found the resolution-optimal solution when the goal is the next state to be expanded (c; Hart et al., 1968). Note that A* expands far fewer states than Dijkstra's algorithm (*cf.* Figure 2.3).

respect to the number of expanded vertices (Dechter and Pearl, 1985).

A* is equivalent to Dijkstra's algorithm except that it processes vertices in order of their current cost-to-come plus their estimated cost-to-go, called their *total potential solution cost* or f-value (e.g., the road-distance from Cambridge plus the air-distance to Oxford). This small modification allows incorporating additional, problem-specific information in a principled manner and can significantly improve search performance (Figure 2.4). A* is used as the forward search in AIT* and the reverse search in EIT* and will be discussed in detail in Chapters 4 and 5.

The performance of uninformed search can also be improved by using a bidirectional search instead of additional problem-specific information (Helgason et al., 1993; Sturtevant and Felner, 2018). A bidirectional search algorithm simultaneously builds two search trees, one rooted at the start and one at the goal, and finds solutions by connecting these trees (Figure 2.5). Dantzig (Section 17.3; 1963) is attributed to have first published the idea for a bidirectional search algorithm but does not present a formal algorithm description. According to the interpretation of Pohl (Section 1.4; 1969), Dantzig's algorithm is a bidirectional version of Dijkstra's that alternately expands vertices in the forward and reverse search in each

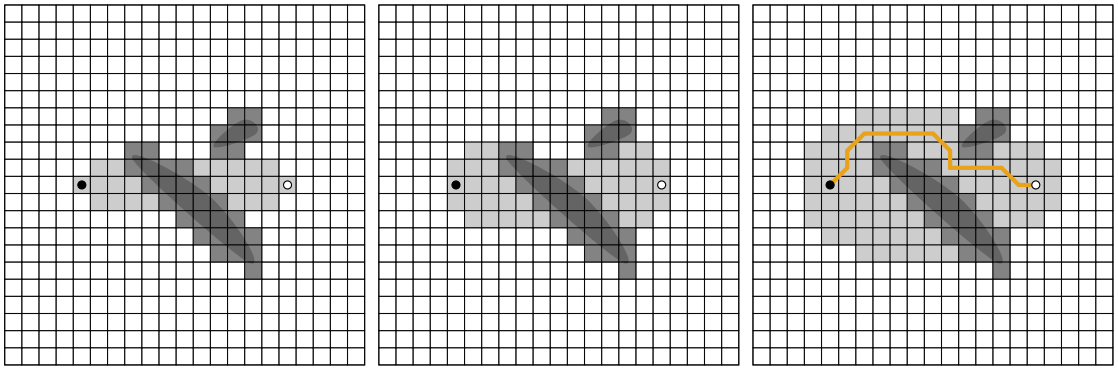


(a) (b) (c)

Figure 2.5: An illustration of how a bidirectional Dijkstra's algorithm searches a discretized planning problem. The search simultaneously progresses outward form the start and the goal in order of increasing cost-to-come and cost-to-go, respectively (a, b). Bidirectional Dijkstra's algorithm has found the resolution-optimal solution when the sum of the lowest cost-to-come values in each queue is larger or equal to the current solution cost (c; Sturtevant and Felner, 2018). Note that this bidirectional version of Dijkstra's algorithm expands fewer states than the unidirectional version (*cf.* Figure 2.3).

iteration. Other presented expansion schemes for a bidirectional Dijkstra's include expanding the vertex with the minimum cost-to-come of both queues (Nicholson, 1966) and expanding the vertex with the least cost-to-come in the queue with more vertices (Section 3.2; Pohl, 1969). Various termination conditions for bidirectional Dijkstra's are presented by Dantzig (Section 17.3; 1963), Nicholson (1966), Dreyfus (1969), Pohl (Section 1.6; 1969), and Holte et al. (2017).

Effectively combining the benefits of informed and bidirectional search is difficult (Figure 2.6). The Bidirectional Heuristic Path Algorithm (BHPA; Pohl, 1971) is a bidirectional version of A*, similar to bidirectional Dijkstra's algorithm. The heuristics in BHPA estimate the optimal cost-to-go to the root of the opposite search tree and are called *front-to-end* heuristics. A solution is found once the trees contain a common vertex but optimality cannot be guaranteed until the total potential solution cost of a vertex in one of the queues or the sum of the least cost-to-come in both queues are equal to or greater than the current solution cost. BS* (Kwa, 1989) modifies BHPA to avoid unnecessary expansions. MM (Holte



(a) (b) (c)

Figure 2.6: An illustration of how a bidirectional A* searches a discretized planning problem. The search simultaneously progresses from the start toward the goal and vice versa in order of total potential solution cost (a, b). Bidirectional A* has found the resolution-optimal solution when the minimum potential solution cost in either the forward or reverse queue is larger than the current cost (c; Sturtevant and Felner, 2018). Note that this bidirectional version of A* expands more states than the unidirectional version (*cf.* Figure 2.4).

et al., 2017) uses a search order that guarantees that its A* searches meet in the middle. Near-optimal Bidirectional Search (NBS; Chen et al., 2017) is optimal in the sense that no other front-to-end algorithm expands fewer vertices in the *worst case* but none of these algorithms consistently outperform bidirectional Dijkstra's algorithm and unidirectional A*.

Several explanations for why these traditional front-to-end algorithms underperform have been presented by Pohl (Section 10.2; 1969), Kwa (1989), Kaindl and Kainz (1997), Edelkamp and Schrödel (Section 9.6.2; 2011), and Barker and Korf (2015), but a recent analysis shows that if the *minimum vertex cover* of a bipartite *must-expand graph* is bidirectional, then there exists a front-to-end algorithm that outperforms A* (Sturtevant et al., 2020).

A different approach to front-to-end informed bidirectional search is presented by Kaindl and Kainz (1997). Instead of alternating between the forward and reverse searches, their approach first searches in reverse up to a user-specified depth and then uses the information gained in this reverse search to create a more accurate,

but still admissible, heuristic for the forward search.

Another strategy that incorporates the information revealed in one direction in the estimated cost-to-go of the other direction is to estimate the optimal cost-to-go to a vertex in the opposite search queue instead of the opposite root. Such estimates are called *front-to-front heuristics*. Examples of such algorithms include the Bidirectional Heuristic Front-to-Front Algorithm (BHFFA; de Champeaux and Sint, 1977; de Champeaux, 1983), Generalized Pohl (GP; Davis et al., 1984), D-Node retargeting (Politowski and Pohl, 1984), and Iterative Deepening BHFFA (IDBHFFA; Arefin and Saha, 2010). Front-to-front approaches avoid many of the problems of traditional front-to-end approaches but computing the front-to-front heuristic is often too computationally expensive to be beneficial in practice (Sturtevant and Felner, 2018).

Perimeter Search (PS; Dillenburg and Nelson, 1994) and Bidirectional Iterative-Deepening A* (BIDA*; Manzini, 1995) limit the computational complexity of front-to-front search in a manner similar to Kaindl and Kainz (1997). Instead of alternating between the forward and reverse searches, these algorithms first perform a reverse search up to a user-specified depth and then perform a forward search with front-to-front heuristics. This limits the computational complexity because the number of potential targets for the heuristic is limited by the depth of the reverse search.

Which of the resolution-optimal algorithms presented in this section performs best depends on various factors (e.g., whether problem-specific information is available and how much memory can be allocated). But sometimes it is unnecessary to find the resolution-optimum and it suffices to find a solution that is good enough, in which case a different family of algorithms often performs better.

2.2.2 Bounded suboptimal search

Bounded suboptimal search algorithms find a solution that may not be optimal but whose cost is guaranteed to be within a factor of the optimum (e.g., not more than

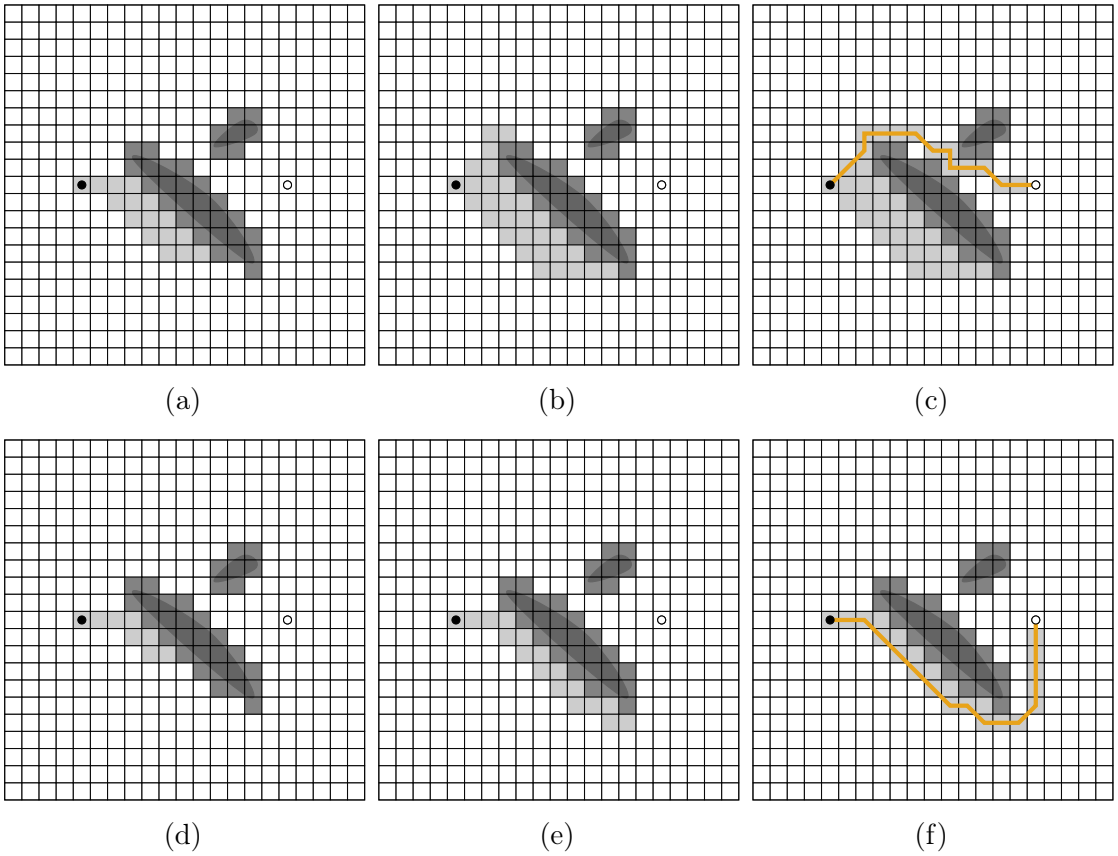


Figure 2.7: An illustration of how WA* with an inflation factor of two (a–c) and ten (d–f) searches a discretized planning problem. The search of WA* progresses from the start toward the goal, as in A*, but the inflated heuristic biases it towards states that are estimated to have a lower cost-to-go (a, b and d, e). In this example, WA* finds much better solutions than its suboptimality bound guarantees. When the inflation factor (and suboptimality bound) is two, WA* finds the resolution-optimum (c). When the inflation factor (and suboptimality bound) is ten, WA* finds a solution that is only about 27% longer than the resolution-optimum (f).

ten percent longer than the optimal road-distance from Cambridge to Oxford). These algorithms often find solutions faster than their resolution-optimal counterparts.

A popular approach to achieve faster solution times with bounded suboptimality is to multiply the admissible heuristic in A* with an *inflation* factor greater than one. This approach is called Weighted A* (WA*; Pohl, 1970) and biases the search towards vertices that have small cost-to-go values. The cost of the solution returned by WA* is guaranteed to be no worse than the inflation factor times the optimal solution cost (Pohl, 1970). A high inflation factor often results in fast solution

times, except when the cost heuristic is very inaccurate and in domains where the quality of a path does not correlate well with the computational effort required to find it (Wilt and Ruml, 2012). A high inflation factor also often leads to solutions that are much better than the guaranteed suboptimality bound (Figure 2.7), which can be exploited to improve performance (Thayer and Ruml, 2008).

WA* speeds up its search by inflating an admissible heuristic. Inadmissible heuristics can be an additional source of information that can be leveraged in bounded suboptimal search. For example, if the problem is to find the *fastest* path from Cambridge to Oxford instead of the shortest path, then traffic data can be an additional source of information. An admissible heuristic can be created by considering the speed limit and assuming no traffic, but this will not be very accurate. An inadmissible heuristic can instead incorporate historical traffic data specific for each road segment, even if this data may overestimate the traffic on a particular day. Multi-Heuristic A* (MHA*; Aine et al., 2014, 2016) is a version of A* that uses an admissible heuristic to guarantee bounded suboptimality but can use multiple arbitrarily inadmissible heuristics to guide its search. A*-Connect (Islam et al., 2016; Cheng et al., 2019) applies the ideas of MHA* to bidirectional search and calculates an inadmissible front-to-front heuristic by computing the heuristic between a state in one search and the last expanded state in the other search.

The heuristics used by A* and many of its variants estimate the remaining solution quality. But if the intent of an algorithm is to find an acceptable solution as fast as possible, then performance can be improved by also considering the required computational effort (Wilt and Ruml, 2015). A^*_ε (Pearl and Kim, 1982) and its bidirectional version (Köll and Kaindl, 1993) use an admissible heuristic to guarantee bounded suboptimality, similar to MHA*, but can guide its search with an effort heuristic that estimates the remaining number of vertex expansions on each path candidate. Explicit Estimation Search (EES; Thayer and Ruml, 2010, 2011) also uses an admissible heuristic to guarantee bounded suboptimality but can guide its search with both an inadmissible cost and an inadmissible effort heuristic.

These algorithms are preferred to their resolution-optimal counterparts when computational time (and/or memory) is not sufficient to find the optimum. But exactly how much time is available is often unknown *a priori*. If only a short amount of time ends up being available, then a suboptimal solution is better than no solution, but if a lot of time is available, then the resolution-optimum is likely preferred.

2.2.3 Anytime search

Anytime algorithms aim to find a solution as fast as possible and then use the remaining computational time to improve it. Anytime A* (AA*; Zhou and Hansen, 2002; Hansen and Zhou, 2007; Vadlamudi et al., 2011) is a version of WA* that simply continues its search with an inflated heuristic after the first solution is found until it converges to the optimal solution.

Restarting Weighted A* (RWA*; Richter et al., 2010) instead restarts each search once it is finished but with a lower inflation factor. The optimal solution is found once the inflation factor is one. This approach can duplicate search effort because each search tree is discarded after the search finishes. Anytime Repairing A* (ARA*; Likhachev et al., 2004) is similar to RWA* but avoids duplicated search effort by keeping track of suboptimally connected vertices and only repairing these connections and cascading changes as necessary. ARA* is the basis for the forward search algorithm used in ABIT* and will be described in detail in Chapter 3.

RWA* and ARA* require user-specified schedules for updating their inflation factors. Anytime Nonparametric A* (ANA*; van den Berg et al., 2011) is similar to ARA* but does not require such a schedule and instead adaptively reduces the inflation factor to expand the most promising vertex at each iteration.

Inadmissible cost and effort heuristics can also be leveraged in anytime search. To leverage such information, Anytime Multi-Heuristic A* (A-MHA*; Natarajan et al., 2019) extends MHA* and Anytime Explicit Estimation Search (AEES; Thayer

et al., 2012) extends EES to work in an anytime manner. AEES is the basis of the forward search algorithm used in EIT* and will be described in detail in Chapter 5.

All of the algorithms reviewed so far are designed to search a static graph but many applications require finding paths on a dynamic graph, i.e., a sequence of similar graphs. All of the reviewed algorithms could search each graph independently, but more efficient algorithms exist that reuse information from previous searches.

2.2.4 Incremental search

Algorithms that are designed to search dynamic graphs are called *incremental*. Lifelong Planning A* (LPA*; Koenig et al., 2004; Likhachev and Koenig, 2005) is an incremental graph-search algorithm that first searches a graph like A* but finds subsequent solutions in similar graphs more efficiently than rerunning A* from scratch. It achieves this by checking how the modification of the graph changed the cost-to-come of vertices with respect to when they were last expanded and cascading these cost changes through the search tree as necessary. A similar approach is used in D* (Stentz, 1995) and D* Lite (Koenig and Likhachev, 2002). LPA* is used as the reverse search in AIT* and will be discussed in detail in Chapter 4.

Truncated Lifelong Planning A* (TLPA*; Aine and Likhachev, 2016) is a version of LPA* that can speed up solution times on modified graphs when bounded suboptimal solutions are acceptable. It achieves this by only expanding vertices until it can guarantee that the current solution is within the suboptimality bound.

Other approaches to incremental A* searches exist, such as updating the heuristic between successive searches (Koenig and Likhachev, 2005, 2006; Sun et al., 2008; Matsuta et al., 2010) and rolling back the search until the modifications of the graph do not affect the search tree anymore (Sun and Koenig, 2007).

Incremental and anytime search are unified in Anytime D* (AD*; Likhachev et al., 2005, 2008) and Anytime Truncated D* (ATD*; Aine and Likhachev, 2016).

These algorithms search each graph in an anytime manner, similar to ARA*, but also find subsequent solutions to similar graphs efficiently, similar to D* and TLPA*.

Many of the algorithms that have been reviewed so far leverage cost and effort heuristics to improve performance. This performance gain is determined by the amount of information contained in the heuristics. The trivial zero-heuristic cannot improve search performance, but a perfectly accurate heuristic makes the search trivial. Instead of designing algorithms that better leverage existing problem-specific information, the performance of existing graph-search algorithms can be improved by increasing the accuracy of the existing information.

2.2.5 Improving graph-search heuristics

Increasing the accuracy of problem-specific information has improved performance in many problem domains, including the 15-Puzzle (Culberson and Schaeffer, 1996), Rubik's Cube (Korf, 1997), and robot vacuum on a grid (Thayer et al., 2011).

Pattern databases (Culberson and Schaeffer, 1996; Korf, 1997; Culberson and Schaeffer, 1998) are precomputed tables of exact solution costs to potentially simplified subproblems of a problem domain. The highest solution cost of any remaining subproblem in an ongoing search can be used as an accurate heuristic in an informed search. Additive pattern databases (Felner et al., 2004) are constructed such that the heuristic remains admissible when the solution costs of all remaining subproblems are combined, which can improve the accuracy of these heuristics.

Hierarchical A* (HA*; Holte et al., 1996) uses *homomorphic* transformations of the search space to create abstractions in which multiple states in the original space are mapped to a single state in the abstract space. These abstractions are then searched to create a heuristic for the original search space. This can result in fewer expanded states, but the presented technique is only shown to work for graphs with uniform edge costs. Hierarchical Cooperative A* (HCA*; Silver, 2005)

and Windowed HCA* (WHCA*; Silver, 2005) are multiagent versions of HA* that use a version of A*, called Reverse Resumable A* (RRA*; Silver, 2005), to search the abstraction from the goal to the start. The cost of the optimal paths to states from the goal in the abstract space is used as the heuristic for the corresponding states in the original space. If the forward search in the original space is about to process a state whose abstract representation has not been processed by RRA*, then RRA* is resumed until it finds the optimal path to the abstract state that corresponds to the state being processed by the forward search.

Thayer et al. (2011) present a technique to improve the accuracy of a heuristic during an ongoing search that is not specific to a particular search algorithm. Their method uses a relationship between the cost-to-go of a state and the cost-to-go of its best child to define a *single-step error* in the heuristic. The mean single-step error can then be calculated and used to improve the accuracy of the heuristic, but it is not guaranteed that the improved heuristic remains admissible.

There exists a lot more research on graph-search algorithms, but its presentation exceeds the scope of this thesis. But even the limited number of reviewed algorithms and techniques demonstrate that graph-search was and is an active area of research that has been and continues to be an incubator for many excellent ideas. Some of these ideas have been incorporated in sampling-based planning, but many have not, as will be evident by the review on sampling-based planners in the following section.

2.3 Sampling-based planning

The graph-search algorithms reviewed in Section 2.2 have strong formal guarantees on solution quality and algorithm efficiency but can only directly be applied to planning problems with discrete or discretized search spaces. The sampling-based planners reviewed in this section can instead directly solve planning problems with continuous search spaces, but have weaker formal guarantees that often only hold probabilistically and in the limit of infinite computation time (Section 2.4).

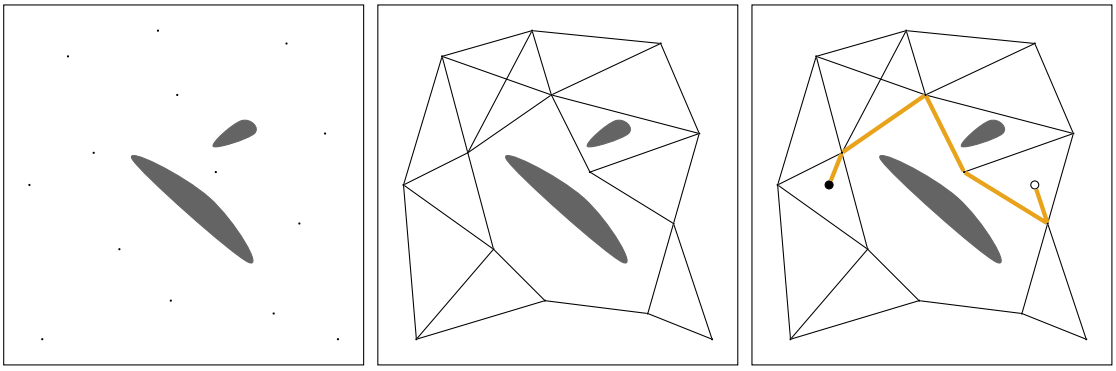
Most sampling-based algorithms are designed for one of two application categories. One category requires finding multiple paths between different starts and goals through the same environment. An example application of this category is a robot manipulator arm that works in a static environment. Algorithms designed for this category are called *multiquery planners* and reviewed in Section 2.3.1.

The other category requires finding a single path per environment. An example application of this category is an autonomous mobile robot that moves through an unstructured environment. Algorithms designed for this category are called *single-query planners* and reviewed in Section 2.3.2. The algorithms presented in this thesis are single-query planners but incorporate ideas from the multiquery literature.

2.3.1 Multiquery planning

Most multiquery algorithms separate the approximation and search of a problem into two distinct phases. Probabilistic Roadmaps (PRM; Overmars, 1992; Kavraki and Latombe, 1994; Overmars and Švestka, 1994; Kavraki et al., 1996) first approximate the entire search space by uniformly sampling a user-specified number of states and finding valid connections between nearby states with a local planner. This local planner often simply checks the straight-line connection. The resulting structure is an RGG embedded in the search space. Each sample is a vertex in the graph and each valid connection an edge between two vertices. When given a query, PRM finds valid connections from the start and goal to the graph with its local planner and then finds a valid path between the start and goal with a graph-search algorithm (Figure 2.8).

PRM has been extended and modified in many ways. Some extensions sample states near obstacles (Amato and Wu, 1996; Amato et al., 1998b), others far away from them (Wilmarth et al., 1999; Holleman and Kavraki, 2000; Pisula et al., 2000; Lien et al., 2003; Yang and Brock, 2004). Some aim to sample narrow passages (Hsu et al., 1998, 2003; Saha et al., 2005), others use Gaussian



(a) (b) (c)

Figure 2.8: An illustration of how PRM searches a continuous planning problem. The first phase of PRM samples states in the search space (a) and creates an embedded graph by connecting nearby states if these local connections are valid (b). The second phase of PRM connects the start and goal of a query to this embedded graph and uses a graph-search algorithm to find a solution (c).

sampling (Boor et al., 1999), deterministic sampling (Branicky et al., 2001; Geraerts and Overmars, 2004; LaValle et al., 2004; Yershova and LaValle, 2004; Janson et al., 2018; Tsao et al., 2020), sampling biased by information theory (Burns and Brock, 2003, 2004), or a combination of different sampling strategies (Hsu et al., 2005). Some extensions use different local planners (Amato et al., 1998a; Isto, 2002; Akinc et al., 2003; Bekris et al., 2003; Plaku et al., 2005; Cadmus To et al., 2019), or different connectivity models (Amato et al., 1998a; Solovey et al., 2018; Solovey and Kleinbort, 2020). Some extensions are specialized for flexible objects (Kavraki et al., 1998b; Anshelevich et al., 2000; Bayazit et al., 2002), others for ligand binding (Singh et al., 1999; Bayazit et al., 2001b; Apaydin et al., 2001), protein folding (Song and Amato, 2001; Apaydin et al., 2001; Song and Amato, 2004), or dynamic environments (Leven and Hutchinson, 2002). Some extensions are guided by a human operator (Bayazit et al., 2001a), and others predict invalid edges through matrix completion (Esposito and Wright, 2019).

The formal properties of a simplified version of PRM have been studied extensively (Barraquand et al., 1997; Kavraki et al., 1998a,c; Hsu et al., 2006; Karaman and Frazzoli, 2011). A famous result is that a version of PRM, called PRM*,

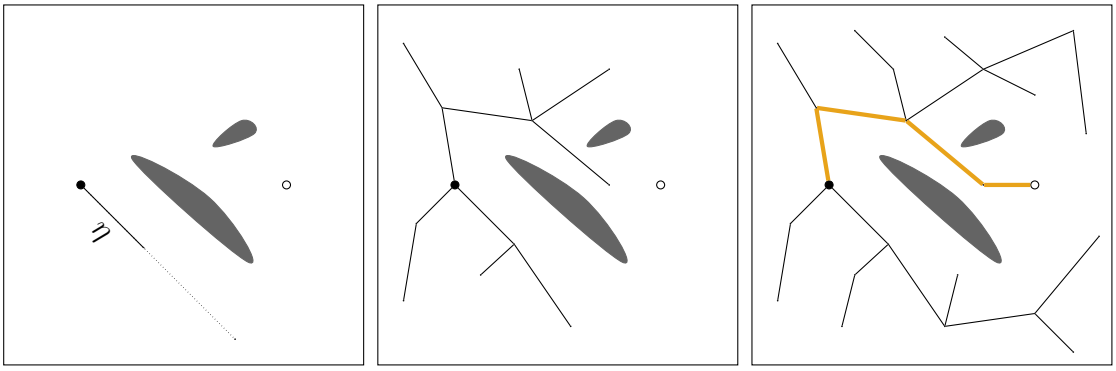
almost-surely asymptotically converges to the optimal solution in the limit of infinite samples when the (average) number of considered neighbors grows logarithmically with the number of samples (Karaman and Frazzoli, 2011). This result can be extended to deterministic asymptotic optimality when random sampling is replaced with a deterministic sampling strategy (Janson et al., 2018).

2.3.2 Single-query planning

Building an approximation of the entire search space is justified if solving multiple queries utilizes the whole approximation. A single query often only requires an approximation of a specific region of the search space and can in these cases be solved more efficiently with a query-specific approximation.

One approach to achieve this is to modify PRM to only check samples and connections for validity if they could be part of a solution. Lazy PRM (Bohlin and Kavraki, 2000) initially assumes that all samples and connections are valid. It then finds a path between the start and goal on this graph, and only checks the samples and connections on this path for collision. If collisions are detected, then the corresponding samples and connections are removed from the graph, a new path is found, and the process is repeated. A similar approach is used in Fuzzy PRM (Nielsen and Kavraki, 2000), Customizable PRM (C-PRM; Song et al., 2001), their generalizing framework (Song et al., 2003), and Lazy PRM* (Hauser, 2015).

Another approach to avoid the computational cost of approximating the entire search space is to simultaneously build and search a query-specific approximation. RRT (LaValle, 1998; LaValle and Kuffner Jr., 1999, 2001) builds a search tree by first randomly sampling a state in the search space and extending a path from the start toward this new sample. It then samples a new state and extends the closest state in its current search tree toward that sample. This process is repeated until the goal is reached (Figure 2.9). To bias the search toward the goal, RRT



(a) (b) (c)

Figure 2.9: An illustration of how RRT searches a continuous planning problem. RRT incrementally builds a search tree by first sampling a state and taking a step of length η from the start towards that state (a). It then continues to sample states and step towards them from the closest state in its search tree (b). RRT steps towards the goal instead of a sampled state with a user-specified probability. If the goal can be connected to the tree, then RRT has found a solution (c).

selects the goal state (or directly samples the goal region) with a user-specified probability instead of randomly sampling a state in the search space. This is especially important if the goal is a single state (or a set with zero measure), as otherwise the tree almost never connects to the goal.

RRT has been extended and modified in many ways. Some extensions build bidirectional search trees (Kuffner Jr. and LaValle, 2000; Klemm et al., 2015; Qureshi and Ayaz, 2015; Burget et al., 2016; Nayak and Otte, 2021), others ensure locally or globally optimal connections within a single tree (Karaman and Frazzoli, 2010a,b; Hwan et al., 2011; Karaman et al., 2011; Karaman and Frazzoli, 2011; Arslan and Tsiotras, 2013, 2015; Salzman and Halperin, 2014; Arslan et al., 2017). Some extensions use different nearest neighbor searches (Cheng and LaValle, 2001; Karaman and Frazzoli, 2013), or sampling strategies (Lindemann and LaValle, 2004; Jaillet et al., 2008; Dalibard and Laumond, 2008; Shkolnik and Tedrake, 2009; Jaillet et al., 2010; Berenson et al., 2011; Devaurs et al., 2013, 2014; Gammell et al., 2014; Arslan and Tsiotras, 2015; Devaurs et al., 2016; Gammell et al., 2018). Some extensions modify the tree growth (Urmson and Simmons, 2003; Bertram

et al., 2006; Ferguson and Stentz, 2006; Rodriguez et al., 2006; Ferguson and Stentz, 2007; Denny et al., 2013; Ko et al., 2014; Wang et al., 2017), others the distance metric (Glassman and Tedrake, 2010; Perez et al., 2012; Sivamurugan and Ravindran, 2014; Cadmus To et al., 2020). Some extensions sample states near obstacles (Yershova et al., 2005; Jaillet et al., 2005; Adiyatov et al., 2017), others near the current solution (Akgun and Stilman, 2011; Nasir et al., 2013). Some extensions are specialized for dynamic environments (Bruce and Veloso, 2002; Ferguson et al., 2006; Zucker et al., 2007; Otte and Frazzoli, 2015, 2016; Connell and Manh La, 2017), others for deformable systems (Lien and Amato, 2006), or molecular disassembly (Cortés et al., 2007).

The formal properties of different versions of RRT have been studied extensively (LaValle and Kuffner Jr., 2001; Karaman and Frazzoli, 2011; Caron et al., 2014; Kunz and Stilman, 2014; Kleinbort et al., 2019). A famous result is that a version of RRT, called RRT*, that ensures locally optimal connections is almost-surely asymptotically optimal when the (average) number of considered neighbors grows logarithmically with the number of samples (Karaman and Frazzoli, 2011).

Instead of sampling a state in the search space to grow the tree towards, a query-specific approximation can also be built by first selecting a vertex of the tree to grow from and then sampling the direction in which to grow the tree. Expansive Space Trees (EST; Hsu et al., 1997; Phillips et al., 2004) build two trees in this manner, one rooted at the start and one at the goal, and bias their vertex selection to vertices with few neighbors. The sampling of direction can be uniform, but can also be biased by heuristic costs computed from work-space decompositions (Plaku, 2013) or search space abstractions (Le and Plaku, 2014).

One drawback of these incremental approaches is that they tightly couple their random incremental approximations with their searches. This results in searches that process the randomly sampled states of the continuous search incrementally, which can be inefficient. Fast Marching Trees (FMT*; Janson and Pavone, 2013;

Janson et al., 2015) instead decouples the approximation from the search by first sampling a user-specified number of states and then searching these states in order of increasing cost-to-come with an efficient fast marching method (Sethian, 1996). If the resulting solution is not of sufficient quality, then FMT* has to be restarted from scratch with a larger number of samples. FMT* is reviewed in detail in Section 3.1.1.

BIT* (Gammell et al., 2015, 2020) also decouples the approximation from the search, but continuously improves the quality of its solution and leverages optimization-specific information with informed graph-search techniques. It samples batches of states and views these samples as a series of increasingly dense, edge-implicit RGGs. The edges in these RGGs are processed in order of their total potential solution cost, similar to A*. BIT* does so efficiently by reusing information from previous searches, similar to TLPA*. BIT* is reviewed in detail in Section 3.1.2.

The algorithms presented in this thesis are conceptual successors of BIT* in that they use similar series of RGG approximations and use graph-search techniques to leverage optimization-specific information. In contrast to BIT*, the presented algorithms either use the available optimization-specific information more effectively or leverage additional sources of information, such as the information implicit in the observed distribution of valid samples and information about the computational effort required to find solutions. The literature reviews for each presented algorithm present more detailed descriptions of the sampling-based algorithms that are directly related to them (Sections 3.1, 4.1, and 5.1).

2.4 Analysis of sampling-based planners

Sampling-based planners are often evaluated probabilistically over all possible realizations of a distribution as a function of the number of samples. Algorithms whose probability of solving the feasible path planning problem approaches one as the number of samples approaches infinity are called *probabilistically complete*. Probabilistic completeness is formally defined in Definition 3 (Karaman and Frazzoli, 2011).

Definition 3 (Probabilistic completeness)**.** *A sampling-based path planning algorithm is called* probabilistically complete *if the probability of it returning a feasible path goes to one as the number of samples goes to infinity (if a feasible path exists),*

$$\liminf_{q\to\infty} P\left(\Sigma_q \neq \emptyset\right) = 1,$$

where q is the number of samples and $\Sigma_q \subset \Sigma$ is the set of valid paths from the start to the goal found by the planner from those samples.

Algorithms that asymptotically solve the optimal planning problem as the number of samples approaches infinity with a probability of one are called *almost-surely asymptotically optimal*. Almost-sure asymptotic optimality implies probabilistic completeness and is formally defined in Definition 4 (Karaman and Frazzoli, 2011).

Definition 4 (Almost-sure asymptotic optimality)**.** *A sampling-based path planning algorithm is called* almost-surely asymptotically optimal *if it has a unity probability of asymptotically solving the optimal path planning problem as the number of samples approaches infinity (if an optimal solution exists),*

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

where q is the number of samples, $\Sigma_q \subset \Sigma$ is the set of valid paths from the start to the goal found by the planner from those samples, $c\colon \Sigma \to [0, \infty)$ is the cost function, and c^* is the optimal solution cost.

2.4.1 Assumptions

Formally analyzing sampling-based planners requires making assumptions about the path planning problem (e.g., Gammell and Strub, 2021). The work presented in this thesis builds on the results of Karaman and Frazzoli (2011) and makes the same assumptions (Sections 2.4.1.1–2.4.1.4).

2.4.1.1 Search space assumption

The search space of the planning problem is assumed to be an open, n-dimensional unit hypercube, $X \coloneqq (0,1)^n, n \geq 2$. The results of this thesis can be applied to other search spaces, such as $\mathrm{SE}(2)$ or $\mathrm{SE}(3)$, using the results of Kleinbort et al. (2020).

2.4.1.2 Cost function assumptions

Let $\sigma_1, \sigma_2 \in \Sigma$ be two paths such that $\sigma_1(1) = \sigma_2(0)$, and let $(\sigma_1|\sigma_2) \in \Sigma$ denote their concatenation,

$$(\sigma_1|\sigma_2)(t) \coloneqq \begin{cases} \sigma_1(2t) & \text{for } t \in [0, 1/2] \\ \sigma_2(2t-1) & \text{for } t \in (1/2, 1]. \end{cases}$$

The cost of any path, $\sigma \coloneqq (\sigma_1|\sigma_2)$, is assumed to be lower bounded by the cost of any of its segments,

$$\forall\, \sigma \coloneqq (\sigma_1|\sigma_2), \quad c(\sigma) \geq \max\{c(\sigma_1), c(\sigma_2)\},$$

and upper bounded by a multiple of its total variation,

$$\exists\, k \in [0, \infty), \quad c(\sigma) \leq k\mathrm{TV}(\sigma),$$

where $\mathrm{TV}(\cdot)$ denotes the total variation of a path (Karaman and Frazzoli, 2011).

It is also assumed that only trivial paths, which consist of a single state, have zero cost,

$$c(\sigma) = 0 \iff \forall\, t \in [0,1], \sigma(t) = \sigma(0).$$

2.4.1.3 Obstacle assumption

The obstacle configuration of the optimal path planning problem is assumed to allow for a valid path from the start to the goal that remains a fixed distance, $\delta > 0$, from its nearest obstacles for its entire length,

$$\exists\, \sigma \in \Sigma, \delta \in (0, \infty), \quad \text{such that} \quad \forall\, t \in [0,1], B_{\delta,n}\left(\sigma\left(t\right)\right) \subset X_{\text{valid}},$$

where $B_{\delta,n}\left(\sigma\left(t\right)\right)$ is an n-dimensional ball with radius δ centered at $\sigma\left(t\right)$,

$$B_{\delta,n}\left(\mathbf{x}\right) \coloneqq \{\mathbf{x}' \in X \mid \left\|\mathbf{x} - \mathbf{x}'\right\|_2 \leq \delta\}.$$

Such a path is said to have *strong δ-clearance*.

2.4.1.4 Optimal solution assumption

At least one solution of the optimal path planning problem, $\sigma^* \in \Sigma$, is assumed to be homotopic to a path, $\sigma_\delta \in \Sigma$, with strong δ-clearance,

$$\exists\, H \colon [0,1] \to \Sigma, \quad H\left(0\right) = \sigma^*, H\left(1\right) = \sigma_\delta,$$

where H is a continuous map whose codomain is the set of all valid paths from the start to the goal. Such a solution is said to have *weak δ-clearance*.

2.5 Discussion

Path planning is the problem of finding paths through continuous spaces that possibly contain obstacles. It is a fundamental task that appears in many real-world applications. Recent applications that give rise to path planning problems often require path planning algorithms to find high-quality paths in short amounts of time.

This is difficult for various reasons, including high-dimensional search spaces,

complicated system constraints, and expensive collision detection and cost evaluation. These difficulties are often compounded and many path planning problems that emerge from real-world applications currently cannot be solved in an exact manner. Two approaches that can solve these problems approximately and have complementary advantages are graph-based searches and sampling-based planners.

Graph-based searches offer a formal basis to leverage different types of information in a principled manner. Optimization-specific information can provide estimates of the remaining solution costs from each state in the search space. Such estimates can be used to guide the search toward high-quality solutions, truncate the search once a sufficiently high-quality solution is found, accelerate the search when suboptimal solutions are acceptable, and provide a lower bound on the resolution-optimal solution. Intent-specific information can additionally be used in graph-search algorithms to align their searches with the priorities of the problems they are designed for. For example, if fast solutions are required, then effort heuristics can be used to guide the search towards solutions that are computationally inexpensive to find.

Graph-based searches offer compelling advantages but cannot directly be applied to path planning problems with continuous search spaces. But continuous problems can directly be solved with sampling-based planners, and recent effort has successfully incorporated some ideas from graph-based search into sampling-based planning. This was achieved by viewing sampling-based approximations as RGGs embedded in (continuous) search spaces, which conceptually separates the approximation of the search space from the search of this approximation and allows to study each subproblem independently.

This thesis builds on this perspective and focuses on the search of sampling-based approximations. It leverages decades of research on graph-based search to design sampling-based planning algorithms that outperform existing techniques on many problems from different domains, including robotic manipulator arms and biomedical knee replacements. The main insight this thesis builds on is that additional, problem-specific information can greatly improve search performance.

ABIT* leverages optimization-specific information with advanced graph-search techniques, similar to ARA* and TLPA*. AIT* leverages optimization- and environment-specific information with an asymmetric bidirectional search in which both searches continuously inform each other. EIT* leverages optimization-, environment-, and intent-specific information with a similar asymmetric bidirectional search that is directly aligned with the priorities of the given planning problem.

Chapter 3

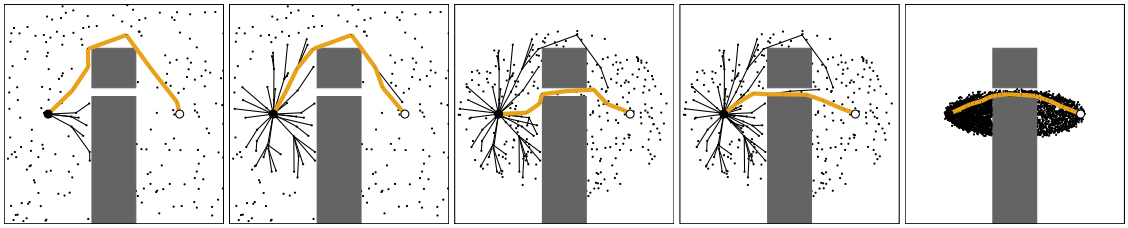
Advanced BIT* (ABIT*)

The extended benefits of optimization-specific information

Contents

Optimization-specific information is useful in search and planning. This chapter extends the benefits of optimization-specific information in sampling-based planning and shows how this information can be leveraged beyond focusing a sampling-based approximation and estimating total potential solution costs. It achieves this by extending BIT* with advanced graph-search techniques, similar to how ARA* and TLPA* extend A*.

BIT* is a single-query sampling-based path planning algorithm that unifies informed graph-based searches, such as A*, with incremental sampling-based planning, such as RRT*. It samples batches of states and views these samples as a series of increasingly dense, edge-implicit RGGs. This perspective allows BIT* to guide its search with optimization-specific information in the form of an admissible



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

Figure 3.1: Five snapshots of how ABIT* searches a planning problem. ABIT* first initializes the approximation and searches it with a highly inflated heuristic until it finds a solution (a). It then improves this solution until it can guarantee that its cost is within a user-specified bound of the optimal solution cost in the current approximation (b). ABIT* then improves the approximation by sampling more states and searches the problem again with a highly inflated heuristic to find better homotopy classes as fast as possible (c). It then improves this solution again until it can guarantee that it is within a user-specified bound of the resolution-optimum (d). ABIT* repeats these steps until it is stopped and almost-surely asymptotically converges towards the optimal solution in the limit of infinite computation time (e).

cost heuristic. BIT* leverages this information to order its search on the total potential solution quality of a path. This results in an efficient search for the resolution-optimal solution, but often does not return any solution until the first resolution-optimal solution is found.

ABIT* builds on BIT* by leveraging the available optimization-specific information more effectively using advanced graph-search techniques, such as inflation and truncation. Inflation biases the search towards the goal, which often leads to faster initial solution times. Truncation prevents the search from finding the resolution-optimal solution on an approximation that will change, which balances the exploitation of the approximation with the exploration of the search space (Figure 3.1).

The remainder of this chapter is organized as follows. It first presents a literature review of single-query planners that decouple the approximation of the search space from the search of this approximation (Section 3.1). It then presents detailed descriptions of the approximation and search used in ABIT* (Section 3.2) and combines established results from the literature on sampling-based planning and graph-search algorithms to prove its almost-sure asymptotic optimality (Section 3.3).

The performance of ABIT* is then evaluated by comparing it to other sampling-based planning algorithms on abstract and nonholonomic problems (Section 3.4). ABIT* is then adapted to work on a next-generation NASA/JPL-Caltech rover and demonstrated to work in the real world during a week-long field trial in the Mojave Desert, California, USA (Section 3.5). The chapter concludes by discussing the experimental results and the insights they provide (Section 3.6).

The core contributions of this chapter are:

- A review of the two popular paradigms single-query planners use to search sampling-based approximations in a principled manner (Section 3.1).
- A detailed presentation of ABIT*, which demonstrates how to incorporate advanced graph-search techniques in sampling-based planning to further leverage the benefits of optimization-specific information (Section 3.2).
- A proof that combines existing results from the literature to show that ABIT* is almost-surely asymptotically optimal (Section 3.3).
- Empirical demonstrations of the benefits of ABIT* on abstract and nonholonomic problems of up to 16 dimensions (Section 3.4).
- A modification of the sampling and rewiring strategies of ABIT* to support planning for a non-Markovian system (Section 3.5.1).
- Adaptations that enable ABIT* to work on a tethered NASA/JPL-Caltech rover and anecdotal results form a week-long JPL-led field trial in the Mojave Desert, California, USA (Section 3.5.2).

The works on ABIT* and its integration on a NASA/JPL-Caltech rover were previously published in the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA; Strub and Gammell, 2020b) and the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS; Paton et al., 2020), respectively.

3.1 Literature review

This section presents a literature review of single-query sampling-based planners that decouple approximating the search space from searching this approximation by sampling *batches* of states. Two popular paradigms that build on this idea are represented by Fast Marching Trees (FMT*; Janson and Pavone, 2013; Janson et al., 2015), which is based on a lazy dynamic programming approach, and Batch Informed Trees (BIT*; Gammell et al., 2015, 2020), which is based on an informed graph-search technique. These paradigms are reviewed in Sections 3.1.1 and 3.1.2, respectively.

3.1.1 Fast Marching Trees (FMT*)

FMT* is a sampling-based planning algorithm that samples a user-specified number of states and views these samples as the vertices of an RGG that is embedded in the search space. It performs a lazy programming recursion outward from the start which searches this edge-implicit RGG with a fast-marching method (Sethian, 1996). This search is ordered on increasing cost-to-come and does not leverage problem-specific information (similar to Dijkstra's algorithm; Dijkstra, 1959).

This results in an asymptotically optimal algorithm either *in probability* (Janson et al., 2015) or *deterministically* (Janson et al., 2018), depending on whether states are sampled uniformly at random or deterministically with a low-dispersion set or sequence. FMT* has proven bounds on its convergence and remains asymptotically optimal (in probability) even if kinodynamic constraints are considered (Schmerling et al., 2015a,b). But FMT* is not an anytime algorithm. It does not return a solution until it finishes and if no (suitable) solution is found from a given set of samples, then FMT* must be restarted from scratch with a different, often larger, set of samples.

An anytime version of FMT* is presented by Salzman and Halperin (2015), which, upon finishing a batch, simply restarts with twice as many samples but without reusing search effort from previous batches. Salzman and Halperin (2015)

also show how FMT* can improve the efficiency of its dynamic programming recursion by leveraging optimization-specific information when selecting which vertex to expand next. In this version, FMT* selects vertices in order of increasing total potential solution cost, which is estimated using an admissible cost heuristic, similar to A* (Hart et al., 1968).

Another way to improve the search efficiency of FMT* is with a bidirectional search similar to a bidirectional Dijkstra's algorithm (Pohl, 1969). Bidirectional FMT* (BFMT*; Starek et al., 2014, 2015) grows two search trees, the *forward tree* rooted at the start and the *reverse tree* rooted at the goal. The simplest version of BFMT* alternates between extending the forward and reverse trees and terminates as soon as the two trees meet (i.e., when the first solution is found). A more advanced version of BFMT* always extends the tree with the lowest-cost vertex that has not yet been expanded and continues until the same vertex has been selected for expansion in both directions (i.e., when the resolution-optimal solution is found). Both versions have improved solution times over FMT* but are not anytime and do not leverage problem-specific information.

Similar to FMT* and its extensions, ABIT* separates the approximation of the search space from the search of this approximation by sampling batches of states. Unlike FMT* and its extensions, ABIT* is an anytime algorithm that reuses previous search effort and extensively leverages optimization-specific information both to focus its sampling-based approximation to the relevant region of the search space and to order its search of this approximation in an effective manner.

3.1.2 Batch Informed Trees (BIT*)

Similar to FMT*, BIT* also samples multiple states at once and views these samples as vertices of an RGG that is embedded in the search space. Instead of sampling a single batch of states and viewing these samples as vertices of a *single and static*

RGG approximation of the search space, BIT* samples multiple batches of states and views them as vertices of *a series of increasingly dense* RGG approximations of the search space. This perspective allows BIT* to separate the approximation of the search space from the search of this approximation in an anytime manner.

BIT* searches its RGG approximations similar to an edge-queue version of A* by processing the implicit edges between nearby samples in order of their total potential solution cost. The total potential solution quality of an edge is computed as the sum of the cost-to-come to the source state of the edge, a heuristic estimate of the edge cost, and a heuristic estimate of the cost-to-go from the target state of the edge.

This results in an anytime, almost-surely asymptotically optimal algorithm that successfully leverages optimization-specific information and provides a framework for recent approaches to kinodynamic planning (Xie et al., 2015) and optimization-based solvers in sampling-based planning (Choudhury et al., 2016). But while BIT* is an anytime algorithm, it does not employ anytime graph-search techniques. The search of BIT* prioritizes efficiently finding the resolution-optimal solution on each RGG approximation over finding initial solutions as quickly as possible. It does so despite the fact that resolution-optimality is meaningless in the context of a single *random* approximation of the search space because the resolution-optimal solution may be arbitrarily bad in the continuous sense of the original problem and because the resolution of the approximation is about to be increased.

Fast-BIT* (Holston et al., 2017) is an extension of BIT* that speeds up initial solution times by first ordering its search greedily on the cost-to-go heuristic of the target states in its edge queue. If an initial solution is found, then Fast-BIT* orders its search on the total potential solution cost of the edges, updates its queue, and continues its search exactly as in BIT*. Fast-BIT* first prioritizes quickly finding an initial solution and only then optimizes this solution, but updating the queue without reconsidering expanded vertices that were suboptimally connected may lead to suboptimal connections that are never repaired as BIT* never adds previously considered edges to its edge queue.

Similar to BIT* and its extensions, ABIT* samples multiple batches of states and views these samples as vertices in a series of increasingly dense, edge-implicit RGGs. Unlike BIT* and its extensions, ABIT* uses anytime graph-search techniques to speed up initial solution times and avoids expending disproportionate computational effort to find the resolution-optimal solution on every RGG approximation.

3.2 Algorithm description

The perspective of searching a series of increasingly dense, edge-implicit RGGs provides a foundation for better path planning algorithms. ABIT* builds on this perspective by using more advanced graph-search techniques. It accelerates anytime performance without duplicating search effort, similar to anytime repairing graph-search algorithms, and avoids wasting effort to find the resolution-optimal solution in an approximation that will change, similar to truncated incremental graph-search algorithms.

This accelerated performance is achieved by inflating the cost-to-go estimate in the edge queue of ABIT*. This balances solution quality with faster initial solution times compared to the incremental search used by BIT*. The initial (potentially resolution-suboptimal) solution of ABIT* is subsequently repaired without duplicating search effort by tracking *inconsistent* states, i.e., states whose cost-to-come has decreased since they were last expanded (similar to ARA*). Tracking these states allows efficiently repairing the search tree by cascading the decreased cost-to-come of these inconsistent states through the tree as necessary.

Fully exploiting every RGG approximation by finding the resolution-optimal path is computationally expensive. ABIT* avoids this by truncating its search on a specific RGG as soon as it can guarantee that it has found a solution whose cost is within a factor of the resolution-optimum. This allows ABIT* to balance the exploitation of its approximation (i.e., repairing the search) with the exploration of the search space (i.e., increasing the resolution of the approximation).

3.2.1 Notation

The search space of a planning problem is denoted by X, the start state by $\mathbf{x}_{\text{start}} \in X$, and the goal states by $X_{\text{goal}} \subset X$. The current search is stored as a tree, $\mathcal{T} = (V, E)$, with vertices, V, and edges, $E \subset V \times V$. Vertices are associated with valid states and edges represent valid connections between states. An edge consists of a source state, $\mathbf{x}_{\text{s}}$, and a target state, $\mathbf{x}_{\text{t}}$, and is denoted as the pair $(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}})$. The set of inconsistent vertices is denoted by $V_{\text{inconsistent}}$ and the set of sampled states by X_{sampled}.

The function $\hat{g}\colon X \to [0, \infty)$ represents an admissible estimate (i.e., a lower bound) of the cost-to-come from the start to any state, $\mathbf{x} \in X$. The function $g_{\mathcal{T}}\colon X \to [0, \infty)$ represents the cost-to-come from the start to a state, $\mathbf{x} \in X$, through the current search tree, $\mathcal{T}$, as defined by the sum of the edge costs leading to the state. This cost is well-defined for states with associated vertices in the tree and taken to be infinite for any state with no associated vertex in the tree.

The function $\hat{h}\colon X \to [0, \infty)$ represents an admissible estimate of the cost-to-go from a state to a goal. The function $\hat{f}\colon X \to [0, \infty)$ represents an admissible estimate of the cost of a path from the start to a goal constrained to go through a state and is defined as $\hat{f}(\mathbf{x}) \coloneqq \hat{g}(\mathbf{x}) + \hat{h}(\mathbf{x})$ in this thesis. This estimate defines the informed set of states that could improve the current solution, $X_{\hat{f}} \coloneqq \left\{ \mathbf{x} \in X \;\middle|\; \hat{f}(\mathbf{x}) \leq c_{\text{current}} \right\}$, where c_{current} is the current solution cost (Gammell et al., 2018). The function $c\colon X \times X \to [0, \infty)$ denotes the true path cost between two states. The function $\hat{c}\colon X \times X \to [0, \infty)$ denotes an admissible estimate of this cost.

The compounding operations, $A \leftarrow A \cup B$ and $A \leftarrow A \setminus B$, are respectively abbreviated as $A \xleftarrow{+} B$ and $A \xleftarrow{-} B$, where A and B are subsets of a common set. The cardinality of a set is denoted by $|\cdot|$ and the minimum of an empty set is taken to be infinity. The number of states per batch is denoted by m and the function $\texttt{children}_{\mathcal{T}}(\cdot)$ returns the children of a vertex in the search tree.

Algorithm 1: Advanced BIT* (ABIT*)

$V \leftarrow \mathbf{x}_{\text{start}}; E \leftarrow \emptyset$
$X_{\text{sampled}} \leftarrow X_{\text{goal}} \cup \{\mathbf{x}_{\text{start}}\}$
$V_{\text{closed}} \leftarrow \emptyset; V_{\text{inconsistent}} \leftarrow \emptyset$
$\varepsilon_{\text{i}} \leftarrow \infty; \varepsilon_{\text{t}} \leftarrow \infty$
$\mathcal{Q} \leftarrow \texttt{expand}\,(\mathbf{x}_{\text{start}})$
} Initialization (Section 3.2.2)

repeat
  $(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \leftarrow \arg\min_{(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in \mathcal{Q}} \left\{ g_{\mathcal{T}}(\mathbf{x}_{\text{s}}) + \hat{c}(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) + \varepsilon_{\text{i}} \hat{h}(\mathbf{x}_{\text{t}}) \right\}$
  $\mathcal{Q} \xleftarrow{-} (\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}})$
  if $\varepsilon_{\text{t}} \left(g_{\mathcal{T}}(\mathbf{x}_{\text{s}}) + \hat{c}(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) + \hat{h}(\mathbf{x}_{\text{t}}) \right) \leq c_{\text{current}}$
    if $(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in E$
      $\texttt{expand\_or\_mark\_inconsistent}\,(\mathbf{x}_{\text{t}})$
    else if $g_{\mathcal{T}}(\mathbf{x}_{\text{s}}) + \hat{c}(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) < g_{\mathcal{T}}(\mathbf{x}_{\text{t}})$
      if $\texttt{collision\_free}\,(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}})$
        if $g_{\mathcal{T}}(\mathbf{x}_{\text{s}}) + c(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) + \hat{h}(\mathbf{x}_{\text{t}}) < c_{\text{current}}$
          if $g_{\mathcal{T}}(\mathbf{x}_{\text{s}}) + c(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) < g_{\mathcal{T}}(\mathbf{x}_{\text{t}})$
            if $\mathbf{x}_{\text{t}} \in V$
              $E \xleftarrow{-} (\texttt{parent}\,(\mathbf{x}_{\text{t}}), \mathbf{x}_{\text{t}})$
            else
              $V \xleftarrow{+} \mathbf{x}_{\text{t}}$
            $E \xleftarrow{+} (\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}})$
            $\texttt{expand\_or\_mark\_inconsistent}\,(\mathbf{x}_{\text{t}})$
            $c_{\text{current}} \leftarrow \min_{\mathbf{x}_{\text{goal}} \in X_{\text{goal}}} \{ g_{\mathcal{F}}(\mathbf{x}_{\text{goal}}) \}$
      else
        $E_{\text{invalid}} \xleftarrow{+} (\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}})$
  else
    if $\texttt{update\_approximation}\,(\varepsilon_{\text{i}}, \varepsilon_{\text{t}})$
      $\texttt{prune}\,(X_{\text{sampled}})$
      $X_{\text{sampled}} \xleftarrow{+} \texttt{sample}\,(m, c_{\text{current}})$
      $\mathcal{Q} \leftarrow \texttt{expand}\,(\mathbf{x}_{\text{start}})$
    else
      $\mathcal{Q} \xleftarrow{+} \texttt{expand}\,(V_{\text{inconsistent}})$
    $\varepsilon_{\text{i}} \leftarrow \texttt{update\_inflation\_factor}\,()$
    $\varepsilon_{\text{t}} \leftarrow \texttt{update\_truncation\_factor}\,()$
    $V_{\text{closed}} \leftarrow \emptyset; V_{\text{inconsistent}} \leftarrow \emptyset$
} Search (Section 3.2.3)

} Approximation (Section 3.2.4) and Update policies (Section 3.2.5)

until `stopped`

3.2.2 Initialization

ABIT* starts by initializing the search tree with the start state as its root, the set of sampled states with the start and goals, and the sets of closed and inconsistent vertices as empty (Alg. 1, lines 1–3). The inflation factor, $\varepsilon_\mathrm{i} \geq 1$, and the truncation factor, $\varepsilon_\mathrm{t} \geq 1$, are initialized to infinity (Alg. 1, line 4). ABIT* completes the initialization by expanding all outgoing edges of the start state into the queue (Alg. 1, line 5). This only inserts direct connections between start and goal states into the queue, since no random states have been sampled yet and the current "RGG" approximation therefore only consists of start and goal states.

3.2.3 Search

The search of ABIT* delays expensive computation of true edge cost by using a lazy edge queue similar to Lazy Weighted A* (LWA*; Cohen et al., 2014). This queue is ordered lexicographically first by the (inflated) total potential solution cost of the edges and then by the cost-to-come of the source vertices. A search iteration starts by removing the edge with the lowest key-value from the queue and checking whether the search can be truncated (Alg. 1, lines 7–9). If the search cannot be truncated and the edge is part of the search tree, then the target state is expanded if it has not already been expanded during the current search (Alg. 1, lines 10 and 11 and Algs. 2 and 3). ABIT* otherwise checks if the new edge can possibly contribute to a solution better than the current one (Alg. 1, lines 12–15).

An edge that passes these checks improves the cost-to-come of the target state and possibly the current solution. If the target state is already part of the tree, adding this edge constitutes a rewiring (Alg. 1, line 17). Otherwise, this state is added to the search tree (Alg. 1, line 19). The edge is added to the search tree in both cases (Alg. 1, line 20). The target state of the edge is then expanded unless it has already been expanded during the current search, in which case it is added to the set of inconsistent vertices (Alg. 1, line 21 and Algs. 2 and 3).

Algorithm 2: ABIT*: `expand_or_mark_inconsistent` $(\mathbf{x})$

```
1 if x ∈ V_closed
2 |  V_inconsistent ←+ x
3 else
4 |  Q ←+ expand (x)
5 |  V_closed ←+ x
```

Algorithm 3: ABIT*: `expand` $(\mathbf{x})$

```
1 E_out ← ∅
2 for all x_i ∈ neighbors (x) do
3 |  E_out ←+ (x, x_i)
4 return E_out
```

3.2.4 Approximation

ABIT* incrementally approximates the search space by sampling batches of m valid states (Alg. 1, line 28). States are sampled uniformly in the informed set, using informed sampling when possible (Gammell et al., 2018). These batches of samples are viewed as a series of increasingly dense, edge-implicit RGGs where edges are defined either by a connection radius, r, or by the k-nearest neighbors (Alg. 4, line 1). The connection parameters, r and k, scale as in PRM* (Karaman and Frazzoli, 2011), using the measure of the informed set as in BIT* (Gammell et al., 2020),

$$r(q) \coloneqq 2\eta\left(1+\frac{1}{n}\right)^{\frac{1}{n}}\left(\frac{\min\left\{\lambda\left(X\right),\lambda\left(X_{\hat{f}}\right)\right\}}{\lambda\left(B_{1,n}\right)}\right)^{\frac{1}{n}}\left(\frac{\log\left(q\right)}{q}\right)^{\frac{1}{n}}$$

$$k(q) \coloneqq \eta\,\mathrm{e}\left(1+\frac{1}{n}\right)\log\left(q\right),$$

where q is the number of samples in the informed set, $\eta > 1$ is a tuning parameter, $\lambda\left(\,\cdot\,\right)$ denotes the Lebesgue measure, and $B_{1,n}$ is an n-dimensional unit ball. ABIT* combines this RGG definition with the existing connections in the search tree (Alg. 4, lines 1 and 2) and ignores edges known to be invalid (Alg. 4, line 3). Improved sampling-based approximations with deterministic sampling sequences and faster-decreasing radii exist (Branicky et al., 2001; LaValle et al., 2004; Yershova and

Algorithm 4: ABIT*: `neighbors` $(\mathbf{x})$

$V_{\text{neighbors}} \leftarrow \texttt{nearest}\,(\mathbf{x}, r \text{ or } k)$
$V_{\text{neighbors}} \xleftarrow{+} \{\texttt{children}_{\mathcal{T}}\,(\mathbf{x}) \setminus V_{\text{neighbors}}\}$
$V_{\text{neighbors}} \xleftarrow{-} \{\mathbf{x}_i \in V_{\text{neighbors}} \mid (\mathbf{x}, \mathbf{x}_i) \in E_{\text{invalid}}\}$
return $V_{\text{neighbors}}$

Algorithm 5: ABIT*: `prune` $(V, E, X_{\text{sampled}}, c_{\text{current}})$

$X_{\text{sampled}} \leftarrow \left\{\mathbf{x} \in X_{\text{sampled}} \;\middle|\; \hat{f}\,(\mathbf{x}) \leq c_{\text{current}}\right\}$
$V \leftarrow \left\{\mathbf{x} \in V \;\middle|\; \hat{f}\,(\mathbf{x}) \leq c_{\text{current}}\right\}$
$E \leftarrow \left\{(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in E \;\middle|\; \hat{f}\,(\mathbf{x}_{\text{s}}) \leq c_{\text{current}} \text{ and } \hat{f}\,(\mathbf{x}_{\text{t}}) \leq c_{\text{current}}\right\}$

LaValle, 2004; Yershova et al., 2010; Janson et al., 2015, 2018; Solovey et al., 2018; Palmieri et al., 2020; Solovey and Kleinbort, 2020) but are not used in this thesis to isolate the reasons for the improved performance of the presented planners relative to existing algorithms in the literature. Graph complexity is reduced by pruning states that cannot possibly improve the current solution (Alg. 1, line 27 and Alg. 5).

3.2.5 Approximation, inflation, and truncation policies

ABIT* stops searching a specific RGG approximation once a solution is found that is guaranteed to be no worse than the product of the inflation and truncation factors, $\varepsilon_{\text{i}}\varepsilon_{\text{t}}$, times the resolution-optimum. ABIT* then updates the inflation and truncation factors and either updates the approximation and restarts the search or continues searching the current RGG approximation, depending on a user-specified policy denoted as `update_approximation` (Alg. 1, lines 26–33). This policy may depend on the inflation factor, ε_{i}, and truncation factor, ε_{t}, which are updated after each search according to other user-specified policies, denoted as `update_inflation_factor` and `update_truncation_factor`, respectively (Alg. 1, lines 32 and 33).

A high inflation factor biases the search towards the goal by prioritizing edges with low cost-to-go estimates. Searching each improved RGG first with a high inflation factor often speeds up initial solution times and, once a solution is found, can

accelerate the discovery of better solutions in new homotopy classes that are unlocked by improved RGG approximations. But searching with a high inflation factor also results in a loose suboptimality bound and often leads to highly suboptimal solutions. A low inflation factor instead results in a tight suboptimality bound and often leads to higher-quality solutions but results in a more conservative search with slower initial solution times. ABIT* was found to work best when each RGG approximation is first searched with a high inflation factor and then with a low inflation factor.

A high truncation factor stops the search as soon as a loose bound on the suboptimality of the current solution is guaranteed, which leads to faster improvements of the RGG approximation. This promotes the exploration of the search space with new samples. A low truncation factor instead allows the search to continue until a tight bound on the suboptimality of the current solution is guaranteed, which often leads to better solutions. This promotes the exploitation of the current RGG approximation with a more complete search. ABIT* can therefore balance the exploration of the search space with the exploitation of its current RGG approximation with the truncation factor update-policy.

Keeping these policies flexible allows ABIT* to be tuned to specific problems. Section 3.4 presents the policies used for the experimental results presented in this thesis.

3.3 Analysis

Any path planning algorithm that processes a sampling-based approximation with a graph-search algorithm is almost-surely asymptotically optimal if the approximation almost-surely contains an asymptotically optimal solution and the graph-search algorithm is resolution-optimal. This condition is sufficient but not necessary. The almost-sure asymptotic optimality of ABIT* follows from proven properties of its approximation and search algorithms.

3.3.1 Approximation

The approximation constructed by ABIT* almost-surely contains an asymptotically optimal solution because it contains all the edges in PRM* for any set of samples and PRM* is almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

3.3.2 Search

ATD* finds a solution no worse than the product of the inflation and truncation factors, $\varepsilon_{\mathrm{i}}\varepsilon_{\mathrm{t}}$, times the resolution-optimum (Aine and Likhachev, 2016). The search of ABIT* processes at least all of the edges processed by ATD* (Theorem 1) and updates the cost-to-come of any vertex under the same condition as ATD*. ABIT* therefore asymptotically finds a resolution-optimal solution when the product of the inflation and truncation factors, $\varepsilon_{\mathrm{i}}\varepsilon_{\mathrm{t}}$, approaches one as the number of samples goes to infinity,

$$\lim_{q\to\infty} \varepsilon_{\mathrm{i}}\varepsilon_{\mathrm{t}} = 1.$$

Theorem 1. *The search of ABIT* processes at least all of the edges that would be processed by ATD* for a given RGG approximation.*

Proof. ATD* can handle improved and worsened connections between states, but adding states and edges to a graph can only improve connections. Therefore only states with improved cost-to-come are considered in this proof (these states are called overconsistent by Aine and Likhachev, 2016). The vertex queue of ATD* is first converted to an edge queue and it is then shown that ATD* finishes sooner than the search in ABIT* due to a stricter termination condition. The search of ABIT* therefore processes at least all of the edges that would be processed by ATD* for a given RGG approximation, since both searches add elements to their queues under the same condition, namely when the cost-to-come of a state is improved.

Ordering ATD* computes a sort key for every state, $\mathbf{x}$, in its queue as the sum of its cost-to-come and the inflated heuristic estimate of its cost-to-go,

$$\texttt{key}_V^{\text{ATD}^*}(\mathbf{x}) \coloneqq g_{\text{p}}[\mathbf{x}] + \varepsilon_{\text{i}}\hat{h}(\mathbf{x}),$$

where $g_{\text{p}}[\mathbf{x}]$ is the cost-to-come label. This label is recursively calculated as the minimum sum of the cost-to-come to a neighbor and the edge cost between the state and that neighbor,

$$g_{\text{p}}[\mathbf{x}] \coloneqq \min_{\mathbf{x}_{\text{s}} \in X_{\text{n}}(\mathbf{x})} \{g_{\text{p}}[\mathbf{x}_{\text{s}}] + c(\mathbf{x}_{\text{s}}, \mathbf{x})\},$$

where $X_{\text{n}}(\mathbf{x})$ denotes the neighbors of the state, $\mathbf{x}$. The base case is a cost-to-come label of zero for the start state, $g_{\text{p}}[\mathbf{x}_{\text{start}}] = 0$. ATD* processes vertices in its queue in order of increasing key-values, $\texttt{key}_V^{\text{ATD}^*}$. If a better connection to a state in the queue is found, then the key and queue position of that state are updated.

Instead of updating the key and queue position, the vertex queue of ATD* could alternatively contain an instance of the same state for each parent and key-value. Selecting the minimum from the whole queue would ensure that the best discovered connection for each state is considered before alternative connections. This would be equivalent to an edge-queue version of ATD* sorted in order of increasing edge key-values,

$$\texttt{key}_E^{\text{ATD}^*}(\mathbf{x}_i, \mathbf{x}_j) \coloneqq g_{\text{p}}[\mathbf{x}_i] + c(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon_{\text{i}}\hat{h}(\mathbf{x}_j).$$

Truncation If the truncation factor times the smallest vertex key-value, $\texttt{key}_V^{\text{ATD}^*}$, of any vertex in the vertex queue of ATD* is greater than or equal to the cost-to-come label of the goal or the current solution cost, then ATD* terminates (Aine and Likhachev, 2016). This is equivalent to terminating when the smallest key-value, $\texttt{key}_E^{\text{ATD}^*}$, in its edge queue would be greater than or equal to the cost-to-come label

of the goal or the current solution cost,

$$\varepsilon_{\mathrm{t}} \min_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{Q}_E^{\mathrm{ATD}^*}} \left\{ g_{\mathrm{p}}\left[\mathbf{x}_i\right] + c\left(\mathbf{x}_i,\mathbf{x}_j\right) + \varepsilon_{\mathrm{i}}\hat{h}\left(\mathbf{x}_j\right) \right\} \geq \min\{ g_{\mathrm{p}}\left[\mathbf{x}_{\mathrm{goal}}\right], g_{\mathcal{T}}\left(\mathbf{x}_{\mathrm{goal}}\right)\}, \quad (3.1)$$

again because the minimum is taken over all edges in the queue. The search of ABIT* terminates if the truncation factor, ε_{t} times the smallest total potential solution cost of the best element in its queue is greater than or equal to the current solution cost,

$$\varepsilon_{\mathrm{t}} \min_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{Q}_{\mathrm{ABIT}^*}} \left\{ g_{\mathcal{T}}\left(\mathbf{x}_i\right) + \hat{c}\left(\mathbf{x}_i,\mathbf{x}_j\right) + \hat{h}\left(\mathbf{x}_j\right) \right\} \geq g_{\mathcal{T}}\left(\mathbf{x}_{\mathrm{goal}}\right). \quad (3.2)$$

This condition cannot be satisfied sooner than that of ATD*, as the inflation factor, ε_{i}, in Equation 3.1 is greater than or equal to one, the heuristic, $\hat{c}$, in Equation 3.2 is admissible, and rewirings can only improve the current cost-to-come, $g_{\mathcal{T}}$, which is therefore smaller than or equal to the cost-to-come label, $\forall\, \mathbf{x} \in X, g_{\mathcal{T}}\left(\mathbf{x}\right) \leq g_{\mathrm{p}}[\mathbf{x}]$. ABIT* therefore considers at least all edges that ATD* would consider. □

3.4 Evaluation

The performance of sampling-based planning algorithms depends on the specific states they happen to sample. They cannot be meaningfully evaluated by comparing a single run of each planner because sampling is often random and some sample configurations are better than others. This thesis therefore presents planner performance as statistical results, where each planner attempts to solve each problem 100 times. These results are shown in two plots per problem.

The first plot shows the planner success rates over time, i.e., the ratio of successful attempts after a given duration of planning. The second plot shows the median solution cost over time and the median initial solution costs and times, where unsuccessful attempts were taken as infinite cost. This plot includes nonparametric

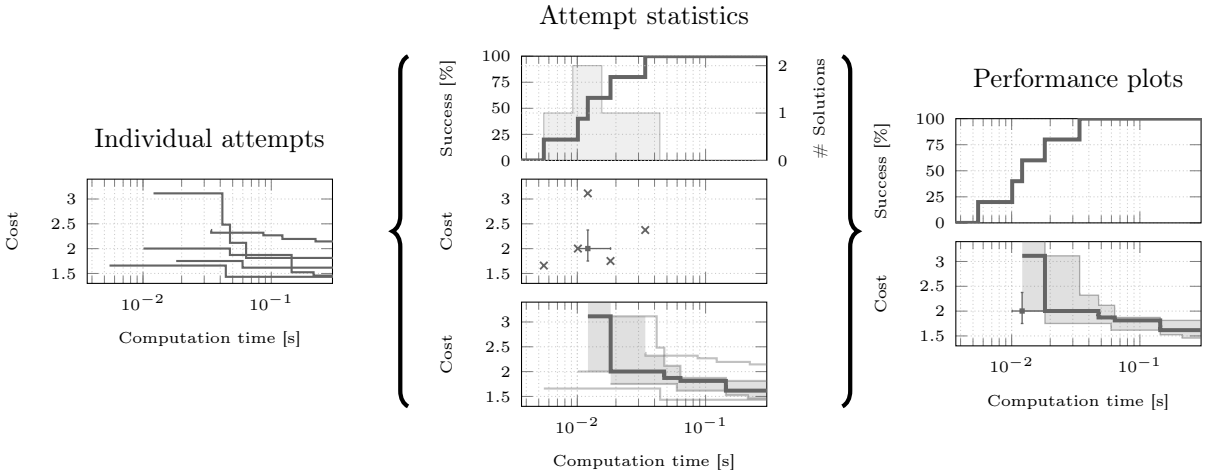


Figure 3.2: An illustration of how the performance plots are generated. Planners are evaluated in this thesis by letting each planner (attempt to) solve each problem multiple times. The solution cost over time is recorded for each attempt and different statistics are computed from this data. These statistics are then summarized in two *performance plots*. The left column shows five attempts of a planner. The middle column shows the empirical CDF and PDF of the initial solution times (top), a scatter plot of the initial solution times and costs and their median values with nonparametric confidence intervals (middle), and the median solution cost over time, also with a nonparametric confidence interval (bottom). The right column shows the performance plots that summarize these statistics with the empirical CDF of the initial solution times (top) and the median initial solution time and cost as well as the median solution cost over time with confidence intervals (bottom). This illustration only shows five attempts of a single planner and 62.5% confidence intervals for clarity. All other performance plots in this thesis summarize 100 attempts per planner and include 99% confidence intervals.

99% confidence intervals for the median costs over time and the initial solution costs and times (Figure 3.2).

ABIT* was evaluated in this manner against the OMPL implementations of RRT-Connect, Informed RRT*, FMT*, Lazy PRM*, and BIT* on abstract problems (Section 3.4.1) and on Reeds-Shepp car problems (Section 3.4.2)[1]. Asymptotically optimal planners optimized path length in search space and informed planners used the Euclidean distance in search space as the admissible cost heuristic.

RRT-based planners used a maximum edge length of 0.3, 0.4, 1.25, 2.4, and 3.0 in 2, 3, 8, 14, and 16 dimensions, respectively. FMT* is not an anytime

[1]The presented results were obtained with OMPL v1.5.0 on two dedicated cores of an Intel i7-4910MQ (2.9 GHz) processor in a laptop with 16 GB of RAM running Ubuntu 18.04.

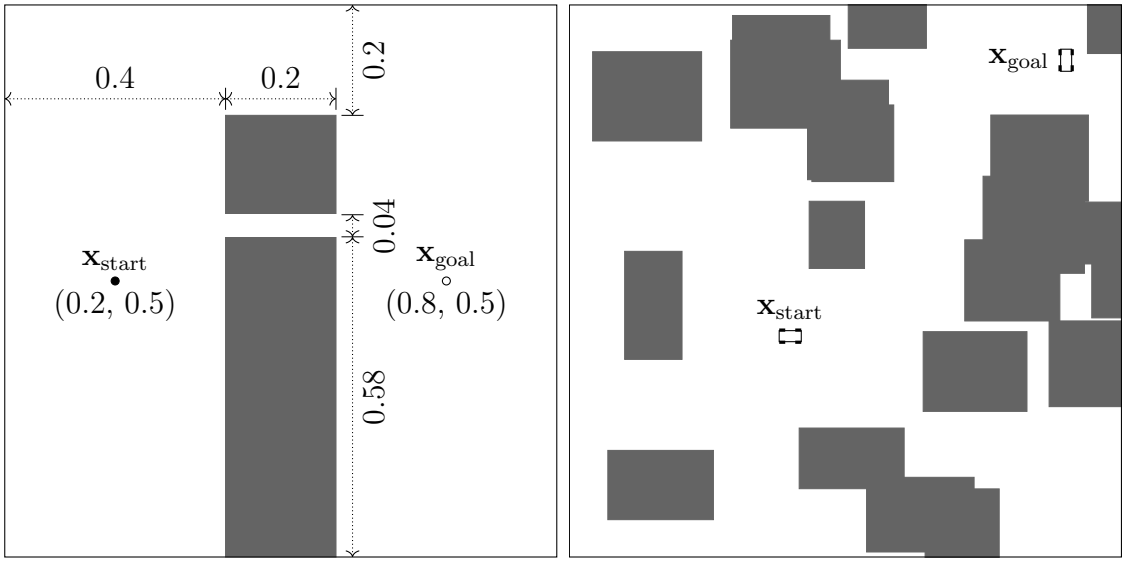


Figure 3.3: Illustrations of the wall gap problem in $\mathbb{R}^2$ (a; Section 3.4.1) and an example of a Reeds-Shepp car problem (b; Section 3.4.2). The start and goal states are represented by a black dot (•) and circle (∘), respectively, or by a cartoon car (⊏⊐). Search-space obstacles are indicated with gray rectangles (▮). Each Euclidean search space dimension was bounded to the interval $(0, 1)$. The results of the tested planners on the wall gap problems in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ are shown in Figures 3.4a, 3.4b, and 3.4c, and those for the best and worst relative performances of ABIT* on instances of the Reeds-Shepp problem in Figures 3.5a and 3.5b, respectively.

algorithm and requires the user to specify the number of samples in advance. All experiments presented in this thesis tested configurations of FMT* with 10, 50, 100, 500, 1000, and 5000 samples. There are multiple lines for FMT* in the presented plots because success rates and median solution times and costs were computed separately for each configuration.

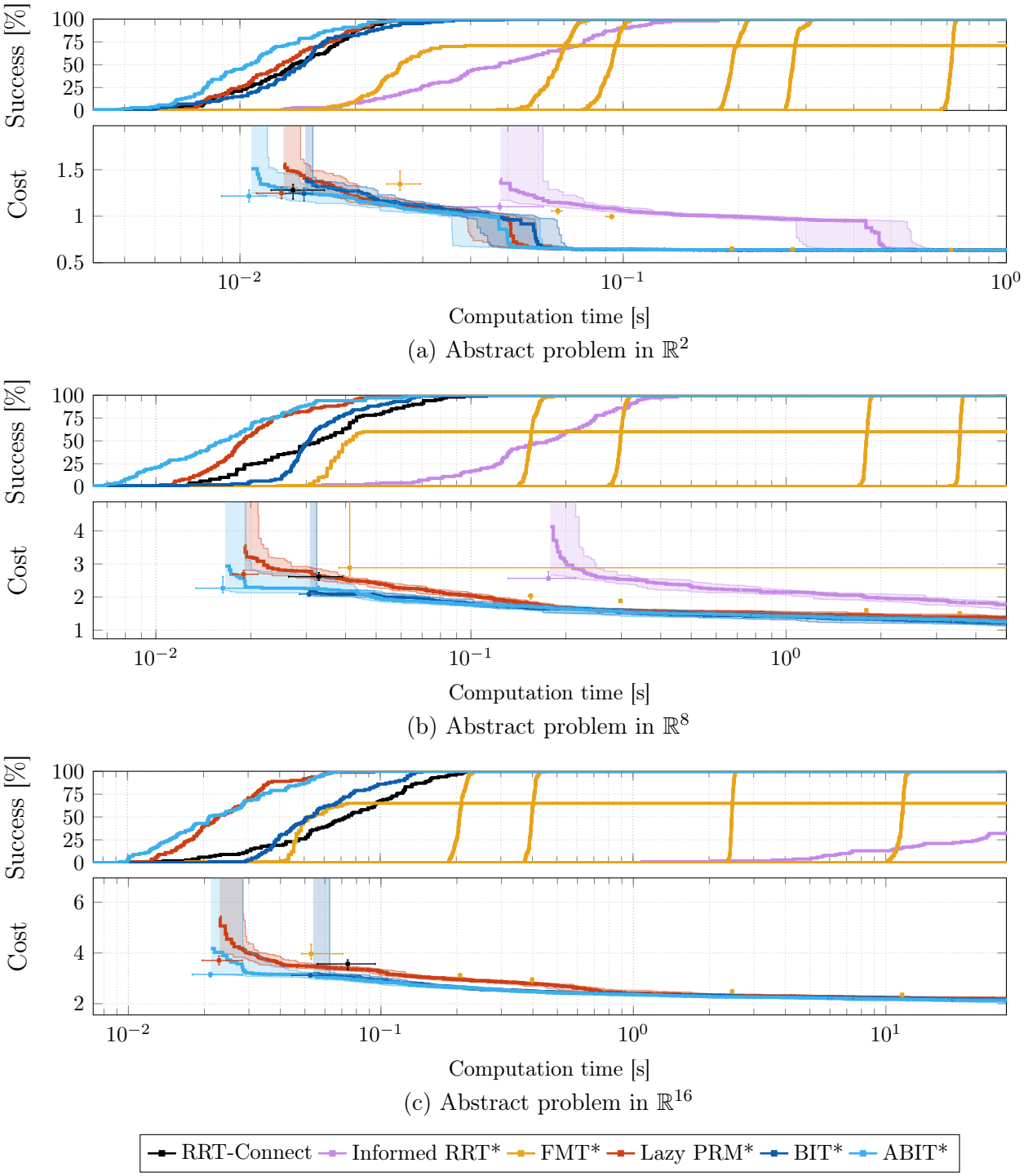
BIT* and ABIT* sampled 100 states per batch and used the k-nearest RGG connection strategy with a factor of $\eta = 1.001$ regardless of the problem dimension. ABIT* used inflation and truncation factor update policies that search each RGG approximation twice, first with a high inflation factor, $\varepsilon_{\text{i}} = 10^6$, and then with a lower factor that depends on the number of states in the informed set, $\varepsilon_{\text{i}} = 1 + {}^{10}/_{q}$. This ensures that new homotopy classes are found quickly with the first search on each improved RGG approximation and guarantees asymptotic optimality (Section 3.3).

The truncation factor was set to $\varepsilon_t = 1 + 5/q$ for all searches. ABIT* was not found to be sensitive to the exact form of these policies and achieved comparable results with other tested policies that searched the first batch of samples with a highly inflated heuristic and gradually let the inflation and truncation factors approach one as the number of samples approaches infinity. These policies could be tuned to each problem separately, but were kept constant in this thesis to show that ABIT* can perform well without tuning them.

3.4.1 Abstract problems

Search-space obstacles have complex geometries even for relatively simple problems (e.g., Figure 1; Das and Yip, 2020). This often makes it difficult to reason about the performance of a planner on a given problem. Directly designing abstract search-space obstacles from basic geometries can help develop intuition on the performance of a planner for a given obstacle configuration and helps the algorithm design process, but the basic geometries of these abstract obstacles make collision detection computationally much less expensive than in real-world problems.

Another factor that affects the computational cost of collision detection is the resolution at which edges are checked for collision. This resolution determines the false negative collision rate, i.e., the percentage of edges that are believed to be valid by the planner but in reality are not. What is considered an acceptable false negative collision rate is application-specific. To make the abstract problems in this thesis representative of real-world applications, the collision detection resolution that results in a 1% false negative collision rate on robotic manipulator arm problems was determined and the collision detection resolution on the abstract problems was set such the average time to evaluate a valid edge is equal in both settings (which on the tested hardware required a resolution of $5 \cdot 10^{-6}$ for abstract problems).



(a) Abstract problem in $\mathbb{R}^2$

(b) Abstract problem in $\mathbb{R}^8$

(c) Abstract problem in $\mathbb{R}^{16}$

Figure 3.4: The planner performances on the wall gap experiments in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ described in Section 3.4.1. The results show that ABIT* outperforms the other tested planners on these problems. In terms of success rates, ABIT* is as good as the most reliable tested planner (Lazy PRM*). In terms of initial solutions, ABIT* finds them as fast as the fastest other planners (Lazy PRM*) but with better solution quality. In terms of convergence rates, ABIT* finds the optimal homotopy class as fast as the fastest other tested planner (BIT*).

The abstract obstacle configuration on which the planners were tested consists of a wall with a narrow gap between the start and goal states (Figure 3.3a). This obstacle configuration illustrates how fast planners are able to find a hard-to-find optimal homotopy class. The wall gap obstacle configuration was tested in $2, 8$, and 16 dimensions with respective time limits of $1, 5$, and 30 seconds per attempt.

Figure 3.4 shows the performances of all algorithms on the $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ versions of this problem. They show that ABIT* outperforms the other tested algorithms on these problems by finding high-quality initial solutions faster while still converging to the optimal solution with a competitive convergence rate.

3.4.2 Reeds-Shepp car problems

ABIT* was also tested against the other planners when planning for a Reeds-Shepp car (Reeds and Shepp, 1990) in an environment with 50 random obstacles (Figure 3.3b). This problem shows the performance of informed planning with inflation and truncation in the presence of nonholonomic constraints. The Euclidean dimensions of the search space were normalized to have a side length of one. The obstacles were placed uniformly at random and had a rectangular shape with uniformly distributed side lengths between 0.1 and 0.2. The car had a rectangular shape with normalized width and length of 0.01 and 0.02, respectively. Collisions between the rectangular car model and the rectangular obstacles were detected at a resolution of 0.001 (i.e., a tenth of the car width) with an optimized implementation of the two-dimensional Separating Axis Theorem (SAT; Gottschalk et al., 1996; Huynh, 2008).

Random obstacle configurations have different effects on the *absolute* performance of each planner, i.e., the individually achieved solution times and costs. The obstacle configuration consequently also affects the *relative* performances of the tested planners, i.e., the ranking of all planners. Multiple obstacle configurations must therefore be tested for a comprehensive planner evaluation on problems with random obstacles.

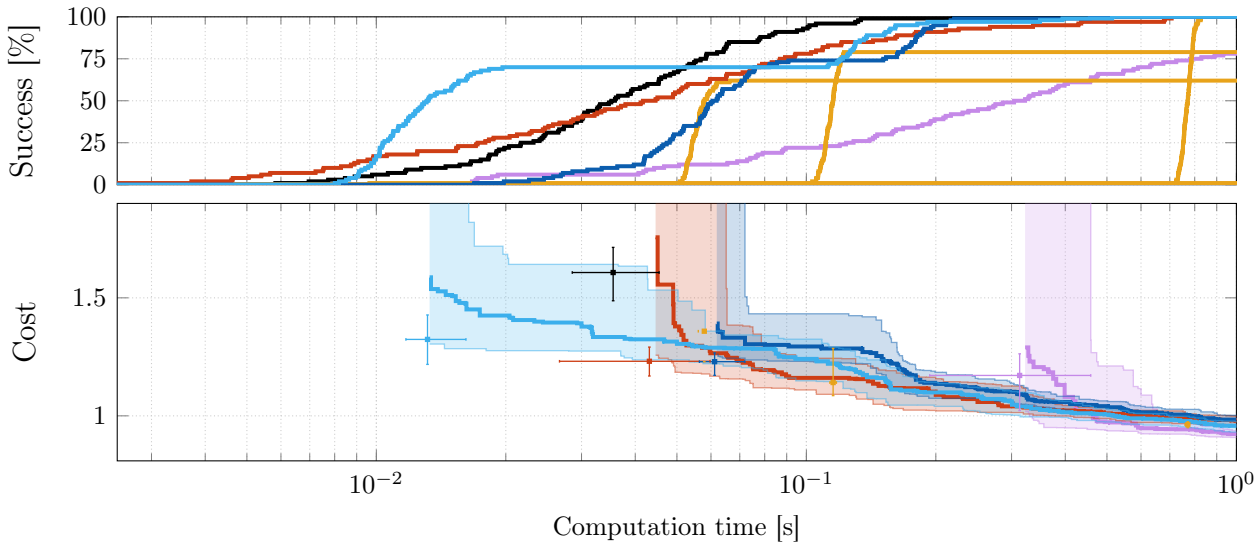
RRT-Connect	Informed RRT*	FMT*	Lazy PRM*	BIT*	ABIT*
198 (99%)	179 (89.5%)	186 (93%)	180 (90%)	188 (94%)	186 (93%)

Table 3.1: The numbers (and percentages) of the 200 Reeds-Shepp car problem instances on which the tested planners achieved a success rate of at least 50%. ABIT* is tied for the second most reliable asymptotically optimal planner. The only tested planner that solved significantly more instances is RRT-Connect, which is not an anytime algorithm.

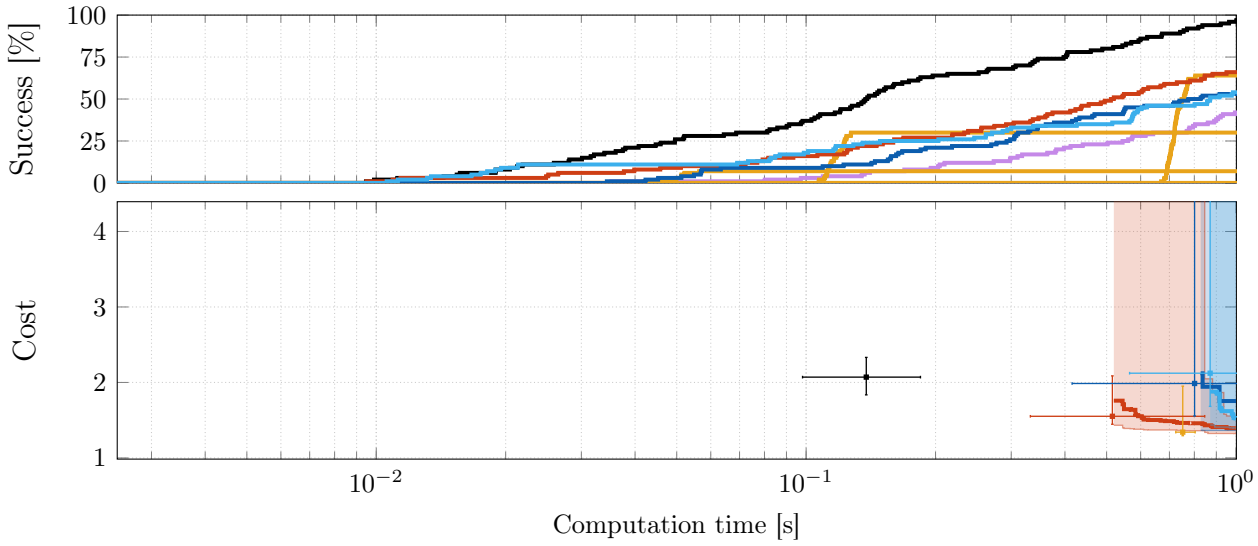
The planners were tested on 200 solvable Reeds-Shepp problem instances with the same start and goal states but different, random obstacle configurations. The time limit for these problems was 1 second per attempt. Testing 200 problems leads to comprehensive results, but complicates the evaluation of planner performance on the whole set of problems. Computing the success rates and median solution times and costs across all problem instances conflates the absolute and relative performance of a planner, which depend on individual problem instances. Selecting a representative problem instance is difficult, because the relative planner performance varies too much across this set of problems. This thesis instead shows isolated results for the instances with the best and worst performances of each presented planner.

The performances of ABIT* are ranked in terms of its median initial solution time relative to the fastest median initial solution time of the other tested planners, limited to problem instances where ABIT* had at least 50% success rate (Table 3.1).

These results show that no planner outperforms all other tested planners on all instances of the Reeds-Shepp problem. On the problem instance where ABIT* performs best relative to the other tested planners, ABIT* has the fastest median initial solution time and competitive success and convergence rates (Figure 3.5a). On the problem instance where ABIT* performs worst relative to the other test planners, many of the asymptotically optimal planners perform similarly to ABIT* and only RRT-Connect has a significantly faster median initial solution time (Figure 3.5b).



(a) Best relative performance of ABIT*



(b) Worst relative performance of ABIT*

Figure 3.5: The planner performances on the Reeds-Shepp car problem that resulted in the best (a) and worst (b) performances of ABIT* relative to the other tested planners (Section 3.4.2). The results show that ABIT* outperforms all other tested planners on some instances (a) but is outperformed by some of the same planners on others (b). On the best instance for ABIT*, it outperforms the other tested planners by finding initial solutions faster while having competitive convergence and success rates (a). On the worst instance for ABIT*, it is only outperformed by RRT-Connect, which finds initial solutions significantly faster (b). But RRT-Connect is not an anytime planner and does not improve its solution given more computation time.

(a) (b) (c)

Figure 3.6: NASA/JPL-Caltech's Axel rover system. Axel is a mechanically simple and robust system consisting of two wheels, a cylindrical body, and an integrated tether that is controlled by an actuated boom (a, b). Axel can take *in-situ* measurements of the soil with a sensor load that is enclosed in its wheel base (c). Photos courtesy of NASA/JPL.

3.5 Deploying ABIT* on a next-generation rover

ABIT* was deployed and demonstrated to work on NASA/JPL-Caltech's Axel rover system. Axel is a tethered rover designed to access science targets on steep slopes and other difficult terrain on the Moon and Mars (Figure 3.6; Nesnas et al., 2012). This section first summarizes the adaptations required for ABIT* to work on Axel (Section 3.5.1) and then presents empirical evidence from simulations and from a week-long field trial that shows that ABIT* can successfully plan paths for Axel with these adaptations (Section 3.5.2).

3.5.1 Adapting ABIT* to plan for Axel

Planning for Axel is challenging due to the complex interactions of its tether with the targeted terrain. Axel requires sequences of SE(3) states on the surface manifold of the terrain. The validity of each state is evaluated by checking it for collision and analyzing its static stability. Collisions are detected by projecting each sampled state onto the surface manifold such that the wheels are in contact and then checking whether the body intersects with the terrain. Static stability is analyzed by considering gravity, the tether force, and the ground reaction forces of the wheels and the boom. Computing the tether force requires predicting where the tether anchors to the terrain. The full details of the state projection, stability analysis, and anchor prediction are presented by Paton et al. (2020).

ABIT* is agnostic to the system it plans for, but assumes that the sampled states are Markovian in the sense that their validity does not depend on how the states are reached. The states of Axel are non-Markovian because the same state may be stable or unstable because its stability depends on the tether force, which in turn depends on its *anchor history*. The anchor history encodes all points where the tether anchors to the environment and is determined by how a state was reached.

Rewiring a state to a parent with an incompatible anchor history can make it unstable and is prevented in this adaptation of ABIT*. New samples initially have an *undetermined* anchor history to ensure that they can be connected to the search tree. The anchor histories of these samples are evaluated when the first connection is considered, using the anchor prediction algorithm described by Paton et al. (2020).

These modifications ensure that no previously valid state is invalidated by rewiring and allow new samples to be connected to the existing search tree, but may prevent ABIT* from finding paths through regions where states have lowest-cost connections with anchor histories that do not support further progress.

Planning though such regions is made possible by predetermining the anchor history of a user-specified percentage of new samples (the validation tests used values around 5%). The predetermined anchor history is copied form a neighboring sample, where the neighborhood is defined by user-specified radius (the validation tests used radii similar to the RGG connection radius described in Section 3.2.4). This maintains the exploration of the whole search space with all known anchor histories and allows ABIT* to plan paths with different anchor histories through regions where the lowest-cost paths have anchor histories that prevent further progress (Figure 3.7).

If ABIT* improves its solution, then a new goal state with undetermined anchor history is created to ensure that there is always a goal state the next improved solution can connect to. These adaptations allow ABIT* to plan paths for Axel (and potentially other tethered systems) in complex terrains such as on steep Martian slopes.

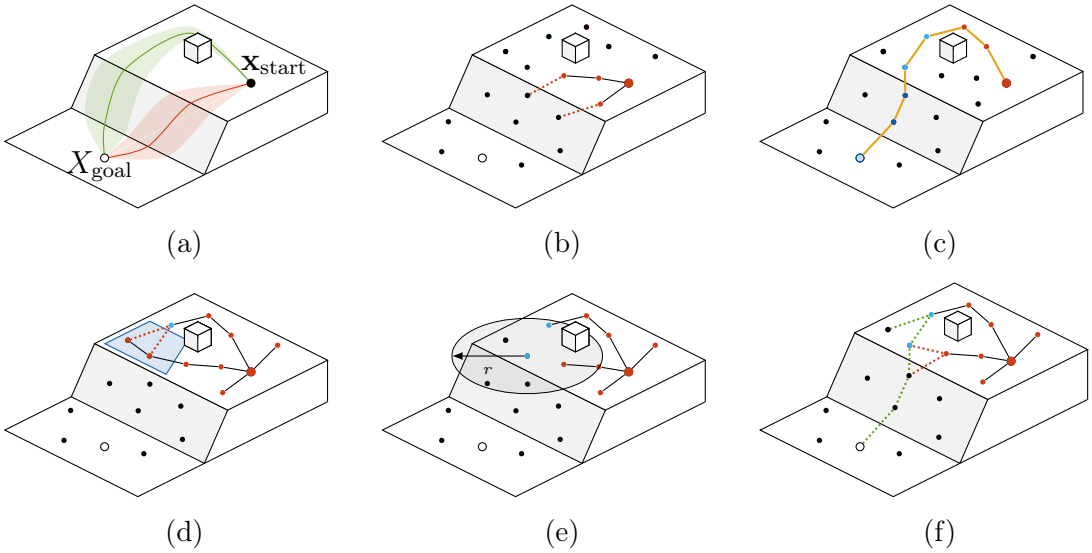


Figure 3.7: An illustration of an Axel planning problem that requires consideration of its non-Markovian tether constraints. The anchor history of each state is indicated in color. Axel must move from the start down to the goal, but the anchor history of the start does not support a direct descent. An obstacle induces two homotopy classes, indicated in red ( ) and green (; a). Paths in the red homotopy class are invalid due to the unstable anchor history of the start (b). Paths in the green homotopy class are valid as they anchor the tether at the obstacle before descending (c). If all new samples had undetermined anchor histories, then all lowest-cost paths to states in the blue area ( ) would connect to the start by going in front of the obstacle (red homotopy class), precluding the necessary connections to these states from behind the obstacle (green homotopy class) due to incompatible anchor histories (d). To prevent this, ABIT* samples some states with predetermined anchor histories by copying the anchor history of a sample within a user-specified radius (e). This inhibits connections to these states from neighbors with incompatible anchor histories and ensures that the blue region is explored by paths with anchor histories that correspond to the green homotopy class (f).

3.5.2 Verifying the adaptations of ABIT*

The non-Markovian adaptations of ABIT* were validated with three tests on simulated maps (Paton et al., 2020). The maps consisted of a $10 \times 5 \times 4$ meter ledge with a 55 degree slope on flat ground (Figure 3.8) and were generated with Voxblox (Oleynikova et al., 2017). The first map tests that ABIT* can find paths when the anchor history of the start state supports a direct descent down the sloped face of the ledge. The second map tests that ABIT* cannot erroneously find paths when the anchor history of the start state does not support a direct

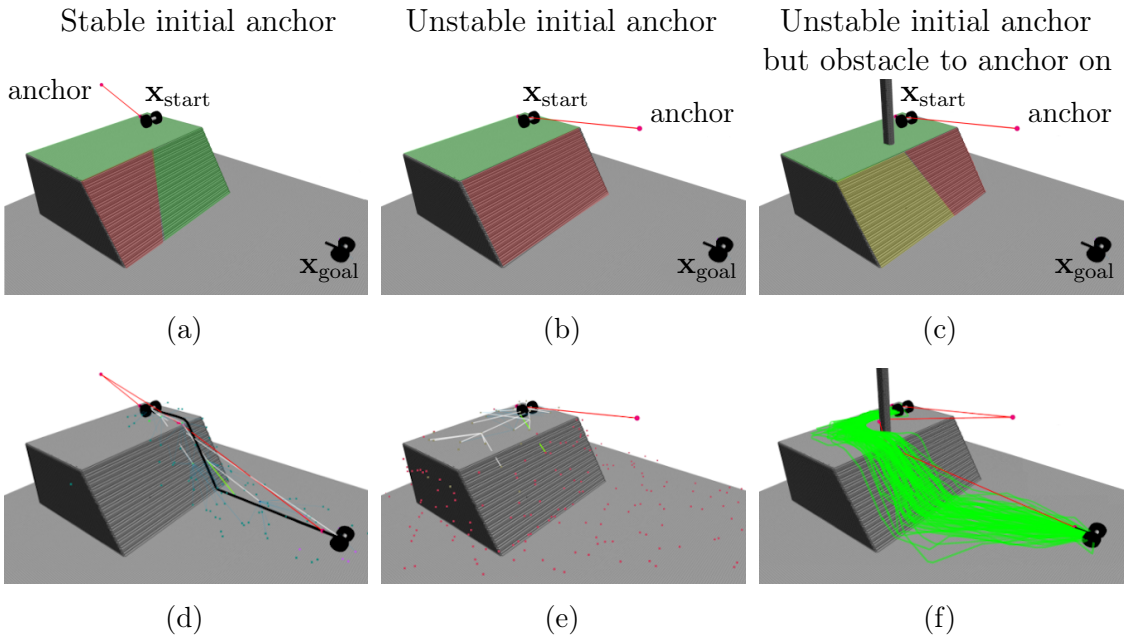
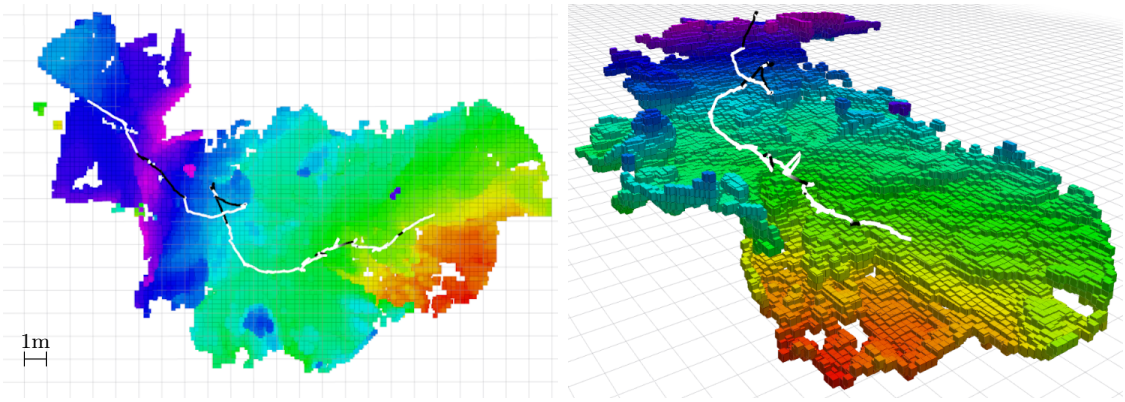


Figure 3.8: Experimental results (Paton et al., 2020) for variations of the problem in Figure 3.7 with a stable initial anchor and no obstacle (a, d), an unstable initial anchor and no obstacle (b, e), and an unstable initial anchor but an obstacle to anchor on (c, f). Statically stable areas are indicated in green ( ), unstable areas in red ( ), and conditionally stable areas in yellow (; a, b, c). If the initial anchor allows for a direct descent (a), then ABIT* found direct solutions 100 out of 100 times (d). If the initial anchor does not allow for a direct descent and there is no obstacle to anchor on (b), then ABIT* did not find any erroneous solutions in 100 tries (e). If the initial anchor does not allow for a direct descent but there is an obstacle to anchor on, which allows for stable intermediate anchors (c), then ABIT* found solutions 84 out of 100 times in the first 750 iterations (f).

descent down the sloped face of the ledge. The third map tests that ABIT* can find paths when the anchor history of the start state does not support a direct descent but there is an obstacle that allows for intermediate anchors that support a descent down the sloped face of the ledge (Figure 3.8).

ABIT* had a 100% success rate when the anchor history of the start state supported a direct descent and found no erroneous solutions when the start state did not support a direct descent and there was no obstacle to anchor to. When the start state did not support a direct descent but there was an obstacle that allows for intermediate anchors that support a descent down the slope, then ABIT* had a success rate of 84% in the first 750 iterations.



(a) (b)

Figure 3.9: An example trajectory from the week-long field trial of Axel in the Mojave Desert, California, USA. The map data is colored by elevation (high is purple, low is red). Axel drove 44 meters of the 46 meter traverse autonomously (i.e., 95% autonomy by distance), which is indicated in white on the maps (Paton et al., 2020). This field trial did not include tether stability checks, but demonstrated successful operation of Axel and ABIT* on noisy data collected from an onboard stereo camera.

An early version of the whole system that did not include anchor predictions was tested during a week-long field test at a Martian analogue site at the Golden Queen Mine in the Mojave Desert, California, USA. During the autonomy test, Axel was able to drive 44 meters of a 46 meter traverse autonomously (which is 95% autonomy by distance). This field test demonstrated the ability of ABIT* to robustly plan paths for Axel on real-world maps that were generated with the onboard stereo camera (Figure 3.9).

3.6 Discussion

ABIT* leverages optimization-specific information in the form of an admissible cost heuristic to improve the performance of sampling-based algorithms. It approximates the search space of a planning problem by sampling multiple batches of states, which it views as a series of increasingly dense, edge-implicit RGGs, as in BIT*. In contrast to BIT*, it uses advanced graph-search techniques, such as inflation and truncation, to speed up initial solution times and to avoid spending excessive computational effort to exploit an approximation that will change.

ABIT* relies on an admissible cost heuristic for edge-costs and remaining solution-cost estimates. If no informative admissible heuristic is available, then ABIT* can be run with the trivial heuristic, i.e., $\forall\, \mathbf{x}_i, \mathbf{x}_j \in X, \hat{h}\left(\mathbf{x}_i\right) \equiv \hat{c}\left(\mathbf{x}_i, \mathbf{x}_j\right) \equiv 0$. This turns the search of ABIT* into an edge-queue version of Dijkstra's algorithm.

If an admissible cost heuristic is available and ABIT* is run with unit inflation and truncation factors, then it can be viewed as a simplified but equally effective version of BIT* that cascades rewirings. ABIT* merges the dual vertex and edge queues of BIT* into a single edge queue. It avoids repeated collision checks by caching checked edges in an object-oriented manner instead of discerning between *new* and *old* states as in BIT*. This clarifies the conceptual ideas behind these algorithms and simplifies their implementation without impeding their practical performance.

If ABIT* is run with inflation and truncation factors greater than one, then ABIT* can improve on the performance of BIT* (Section 3.4). Inflating the heuristic biases the search towards the goal and can find initial solutions quickly. Truncating the search once a sufficient bound on the current solution quality is achieved avoids spending an excessive amount of computational effort to fully exploit an approximation that will change. Flexible update policies of the inflation and truncation factors ensure that ABIT* can leverage high and low inflation and truncation depending on the accuracy of its approximation. ABIT* was not found to be sensitive to the exact form of these policies. All examined policies that conduct the initial search of each approximation with a very high inflation factor and asymptotically let both factors approach one as the number of states goes to infinity lead to results comparable to those presented in Section 3.4 and in Strub and Gammell (2020b).

ATD* and ABIT* both use ideas from ARA* and TLPA* but in different ways because they are designed for different problems. ATD* is designed to search dynamic but discrete problems represented by graphs which change because of updates to the problem, e.g., because of new information about the environment of a robot. It cannot control when it receives these updates and must handle vertices

with both decreased and increased cost-to-come. ABIT* is instead designed to directly search static but continuous problems represented by a series of increasingly dense RGGs. ABIT* controls when it improves its sampling-based approximation and only has to handle vertices with decreased cost-to-come because adding more vertices and edges to its approximation cannot worsen the path to any state.

ABIT* was also adapted to a non-Markovian system by changing its sampling and rewiring strategies. These adaptations were demonstrated in collaboration with NASA/JPL-Caltech to work on Axel, a tethered rover specialized for navigating steep slopes, during a week-long field test in the Mojave Desert, California, USA (Paton et al., 2020). Independently of this collaboration, ABIT* also successfully planned paths for RoboSimian, another next-generation NASA/JPL-Caltech rover, through salt-evaporite fields in the Death Valley, California, USA (Reid et al., 2020). ABIT* was separately extended to consider *search momentum* by prioritizing the expansion of vertices that have recently resulted in search progress (Chen et al., 2021).

In summary, this chapter presented the following core contributions:

- A review of other sampling-based algorithms that decouple their approximation from their search by sampling batches of states (Section 3.1).
- A detailed description of ABIT*, which incorporates advanced graph-search techniques in sampling-based planning to leverage optimization-specific information more efficiently than BIT* (Section 3.2).
- A proof of the almost-sure asymptotic optimality of ABIT* that combines established results from the literature (Section 3.3).
- Empirical demonstrations of the benefits of ABIT* on a set of abstract and nonholonomic problems (Section 3.4).
- Adaptations of the sampling and rewiring strategies in ABIT* to work with NASA/JPL-Caltech's Axel rover system and anecdotal verification of these adaptations (Section 3.5).

Chapter 4

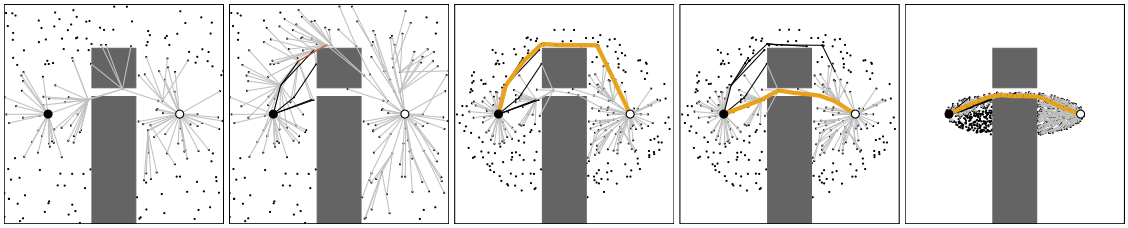
Adaptively Informed Trees (AIT*)

The benefits of environment-specific information

Contents

Path planning problems are in part defined by the obstacles in their environment (Definitions 1 and 2; Chapter 2). They must therefore distinguish between valid and invalid states of their search spaces, i.e., states that are collision-free and states that are not. This partitions the search space into two collectively exhaustive but mutually exclusive sets of states (i.e., all states of the search space are either valid or invalid). These sets are impossible or prohibitively complex to express analytically for many path planning problems, but determining algorithmically whether a single state is valid or invalid is often computationally feasible.

This property is exploited by sampling-based planning algorithms. Instead of computing complex obstacle representations, they simply sample individual states, check whether these states are valid, and try to find valid connections between them if they are. In this manner, sampling-based planning algorithms gain incremental



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

Figure 4.1: Five snapshots of how AIT* searches a continuous planning problem. AIT* starts by initializing the approximation and calculating an approximation-specific cost heuristic with a reverse search without collision detection (a). AIT* exploits the calculated heuristic with its forward search and repairs the reverse search tree whenever the forward search reveals that it contains an invalid edge (b). When the forward search finds the resolution-optimal solution, the approximation is improved by sampling and pruning states and the heuristic is updated on the improved approximation (c). This updated heuristic is again exploited with the next forward search and repaired when found to use invalid edges (d). AIT* repeats these steps until it is stopped and almost-surely asymptotically converges towards the optimal solution in the limit of infinite computation time (e).

information about the search-space obstacles (i.e., the sets of valid and invalid states) every time they check a state for collision. While this environment-specific information is used by all sampling-based planning algorithms when approximating the search space, it is ignored by most planners when searching their approximation.

This chapter presents AIT*, a sampling-based path planning algorithm that leverages optimization- and environment-specific information both when approximating the search space and when searching this approximation. It leverages this additional information when ordering its search without making any assumptions about the location, shape, or number of obstacles and without requiring additional user input.

AIT* achieves this by leveraging optimization- and environment-specific information with a search that is bidirectional, hierarchical, and asymmetric. This search is bidirectional as AIT* builds two search trees, one rooted at the start and one at the goal. This bidirectional search is hierarchical as the reverse search can be seen as working in an abstract search space because it does not perform collision detection on the edges. This hierarchical bidirectional search is asymmetric because the reverse and forward searches have different purposes and computational cost.

The inexpensive reverse search calculates an accurate cost heuristic by combining heuristic edge costs while considering the connectivity of the current approximation. The expensive forward search leverages this heuristic and informs the reverse search about invalid edges (Figure 4.1). For brevity, this type of search is sometimes simply called an *asymmetric bidirectional search* in this thesis (in the graph-search literature it would be called a hierarchical search).

The remainder of this chapter is organized as follows. It first presents a literature review of other sampling-based planning algorithms that leverage environment-specific information to guide their search (Section 4.1). It then presents a detailed description of AIT* (Section 4.2) and combines established results from the literature on sampling-based planning and graph-based search to prove its almost-sure asymptotic optimality (Section 4.3). The performance of AIT* is then evaluated by comparing it to other sampling-based planning algorithms on nonholonomic, manipulator, and biomedical problems (Section 4.4). The chapter concludes by discussing these results and potential improvements for AIT* (Section 4.5).

The core contributions of this chapter are:

- A review of other sampling-based algorithms that leverage environment-specific information to guide their search and a discussion of their conceptual differences to AIT* (Section 4.1).
- A detailed presentation of AIT*, which demonstrates how optimization- and environment-specific information can be leveraged in all aspects of sampling-based planning, including the search (Section 4.2).
- A proof that combines established results from the literature on sampling-based planning and graph-based search to show that AIT* is almost-surely asymptotically optimal (Section 4.3).
- Empirical demonstrations of the benefits of AIT* on Reeds-Shepp problems, a manipulator problem, and a problem from the biomedical domain (Section 4.4).

The work on AIT* was previously published in the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA; Strub and Gammell, 2020a) and has been accepted for publication in the International Journal of Robotics Research (IJRR; Strub and Gammell, 2021b).

4.1 Literature review

All sampling-based planning algorithms use environment-specific information in the form of collision detection results to build a discrete approximation of the continuous search space. This section reviews the few sampling-based planning algorithms that also leverage such information when searching their sampling-based approximations. Motion Planning using Lower Bounds (MPLB; Salzman and Halperin, 2015) is most closely related to AIT* as both planners leverage environment-specific information by computing accurate cost heuristics from the connectivity of their sampling-based approximations. MPLB is reviewed in detail in Section 4.1.1. Other sampling-based planners leverage environment-specific information less directly when searching by tightly coupling the search with the sampling and using environment-specific information to bias the sampling. These planners are reviewed in Section 4.1.2.

The forward search in AIT* often fully evaluates fewer edges than that of planners that directly use an *a priori* heuristic, because combining an *a priori* heuristic along the edges of a path as in the reverse search of AIT* results in a heuristic that *dominates* (i.e., is greater than or equal to) the *a priori* heuristic if this heuristic is consistent. Fully evaluating few edges is not the explicit goal of AIT* but it connects AIT* to *lazy* graph-search algorithms that explicitly aim to minimize the number of fully evaluated edges (e.g., Cohen et al., 2014; Dellin and Srinivasa, 2016; Mandalika et al., 2019; Lim et al., 2021). These methods can significantly improve search performance but a detailed review of them exceeds the scope of this thesis.

4.1.1 Motion Planning using Lower Bounds (MPLB)

MPLB is a sampling-based planning algorithm that builds on the quasi-anytime version of FMT* reviewed in Section 3.1.1. It starts by sampling a batch of states, which it views as an edge-implicit RGG. MPLB then determines all *promising* samples in its current batch (i.e., samples that could potentially improve its current solution). It does so by running two instances of Dijkstra's algorithm, one from the start and one from the goal, until they have respectively found all states with a cost-to-come or cost-to-go of less than or equal to half the current solution cost. MPLB then runs another instance of Dijkstra's algorithm from the goal, which calculates the cost-to-go of all promising samples up to the current solution cost. These three searches with Dijkstra's algorithm are computationally inexpensive because they do not perform collision detection on the edges.

The promising samples are then searched with the informed version of FMT* reviewed in Section 3.1.1 using the calculated, approximation-specific cost-to-go values as an accurate and admissible cost heuristic. Once this FMT* search finishes, MPLB restarts this process by replacing its approximation with twice as many new samples. This results in an almost-surely asymptotically optimal algorithm that leverages environment-specific information in its search, but cannot reuse previous search effort when searching improved RGG approximations, and does not update the accuracy of its heuristic given new collision detection results.

Similar to MPLB, AIT* leverages environment-specific information by calculating an accurate, approximation-specific cost heuristic with a reverse search that does not check edges for collision. Unlike MPLB, AIT* uses heuristic estimates of edge costs in the reverse search, reuses previous search effort when searching improved RGG approximations, and updates its calculated heuristic when the forward search detects that its calculation used invalid edges.

4.1.2 Indirectly leveraging environment-specific information

Environment-specific information can also be leveraged indirectly when searching sampling-based approximations by tightly coupling the search with the sampling and using environment-specific information to bias the sampling.

Kiesel et al. (2012) present a technique called f-biasing, which uses environment-specific information to bias the sampling of RRT-based algorithms. Their technique first discretizes the search space of the planning problem (e.g., with a uniform grid) and then calculates the cost-to-come and cost-to-go of each discretized region. It does so with two passes of Dijkstra's algorithm, one from the start and one from the goal. The calculated cost-to-come and cost-to-go are then combined to compute an f-value for each abstract state, similar to how the f-value is computed in A* (Section 2.2.1). An f-biased RRT or RRT* then proceeds as usual, except that its sampling is biased to discretized regions with low f-values.

A similar idea can also be applied to kinodynamic and nonholonomic sampling-based planning based on random control inputs (Plaku et al., 2010; Plaku, 2013, 2015). In a preprocessing phase, these approaches discretize the work space (e.g., with a triangulation) and identify promising discretized regions (e.g., by calculating their cost-to-go with a reverse Dijkstra's search starting from the goal). In the planning phase, these promising regions are then used to guide the search of incremental sampling-based planners by biasing them to extend states that map to promising work-space regions.

Instead of using a triangulation to discretize the work space, Le and Plaku (2014) use a PRM to discretize a simplified version of the search space that ignores nonholonomic and kinodynamic constraints. Their approach first calculates the cost-to-go of each state in this PRM with a reverse Dijkstra's algorithm starting from the goal. It then initializes a search tree rooted at the start state of the original search space. Each iteration then extends a randomly selected state in this search

tree with a random control input, but the random state selection is biased towards states that are close to PRM-states with low cost-to-go.

Instead of simplifying the search space by ignoring constraints, several approaches exist that create simplified search spaces with lower-dimensional projections (Maly and Kavraki, 2012; Orthey et al., 2018; Orthey and Toussaint, 2019; Orthey et al., 2020; Westbrook and Ruml, 2020). These techniques bias the sampling of RRT-based approaches or the selection of vertices in control-based planning by solving simplified, lower-dimensional versions of the original path planning problem and then mapping lower-dimensional states to the original search space.

Similar to the approaches of this section, AIT* leverages environment-specific information in all aspects of sampling-based planning. Unlike these approaches, AIT* directly leverages this information in its search by decoupling the approximation from the search and neither relies on user-defined discretizations of the work space nor on simplified abstractions or lower-dimensional projections of the search space.

4.2 Algorithm description

AIT* leverages optimization- and environment-specific information both when approximating the search space and when searching this approximation. It approximates the search space with a similar series of increasingly dense, edge-implicit RGGs as BIT* and ABIT*. These RGG approximations only contain valid states and are focused on the relevant region of the search space with informed sampling (Gammell et al., 2018). AIT* searches these approximations with an asymmetric bidirectional search in which both searches continuously inform each other with complementary information. This bidirectional search is asymmetric both in purpose and in computational cost (Alg. 6).

The reverse search of AIT* is an LPA* search that calculates an accurate cost heuristic by combining admissible edge cost heuristics between multiple samples

Algorithm 6: Conceptual AIT*

```
repeat
    improve RGG approximation (sampling)
    calculate heuristic (reverse search)
    while RGG approximation is useful do
        find solution (forward search)
        if found invalid edge or unprocessed state
            update heuristic (reverse search)
until stopped
```

into a more accurate, but still admissible, cost-to-go heuristic between each sample and the goal. This calculated cost-to-go heuristic considers the connectivity of the current RGG approximation and therefore contains the environment-specific information inherent in the observed distribution of valid samples. The reverse search is computationally inexpensive because it does not evaluate true edge costs and does not perform collision detection on the edges.

The forward search of AIT* is an edge-queue version of A* that efficiently finds solutions to the given planning problem by leveraging the environment-specific information in the calculated heuristic. If the forward search finds that the reverse search used an invalid edge to calculate the heuristic, then it causes the reverse search to update the heuristic with this information. The forward search is computationally expensive, as it evaluates true edge costs and performs collision detections on the edges, but focused on connections likely to yield a solution by the accurate heuristic that is calculated with the reverse search.

Both searches process a similar series of increasingly dense, edge-implicit RGG approximations as BIT* and ABIT*. Once the resolution-optimal solution is found on a specific RGG approximation, then AIT* increases the density of this approximation by sampling more states and restarts its bidirectional search. This process is repeated until stopped and almost-surely asymptotically converges towards the optimal solution.

Algorithm 7: Adaptively Informed Trees (AIT*)

```
V_F ← x_start ; E_F ← ∅                                                  } Initialization
V_R ← X_goal ; E_R ← ∅                                                   } (Section 4.2.2)
X_sampled ← {x_start} ∪ X_goal                                           }
Q_F ← expand(x_start) ; Q_R ← X_goal                                     }
repeat
    if continue_reverse_search()                                         } Reverse search
        x ← arg min_{x∈Q_R} {key_R^{AIT*}(x)}                            } (Section 4.2.3)
        Q_R ←⁻ x                                                         }
        if ĥ_con[x] < ĥ_exp[x]                                           }
            ĥ_exp[x] ← ĥ_con[x]                                          }
        else                                                             }
            ĥ_exp[x] ← ∞                                                 }
            update_state(x)                                              }
        for all x_i ∈ neighbors(x) do                                    }
            update_state(x_i)                                            }
    else if continue_forward_search()                                    } Forward search
        (x_s, x_t) ← arg min_{(x_s,x_t)∈Q_F} {key_F^{AIT*}(x_s, x_t)}    } (Section 4.2.4)
        Q_F ←⁻ (x_s, x_t)                                                }
        if (x_s, x_t) ∈ E_F                                              }
            Q_F ←⁺ expand(x_t)                                           }
        else if g_F(x_s) + ĉ(x_s, x_t) < g_F(x_t)                        }
            if collision_free(x_s, x_t)                                  }
                if g_F(x_s) + c(x_s, x_t) + ĥ_con[x_t] < c_current       }
                    if g_F(x_s) + c(x_s, x_t) < g_F(x_t)                 }
                        if x_t ∉ V_F                                     }
                            V_F ←⁺ x_t                                   }
                        else                                             }
                            E_F ←⁻ (parent_F(x_t), x_t)                  }
                        E_F ←⁺ (x_s, x_t)                                }
                        Q_F ←⁺ expand(x_t)                               }
                        c_current ← min_{x_goal∈X_goal} {g_F(x_goal)}    }
            else                                                         }
                E_invalid ←⁺ {(x_s, x_t), (x_t, x_s)}                    }
                if (x_s, x_t) ∈ E_R                                      }
                    invalidate_reverse_branch(x_s)                       }
    else                                                                 } Approximation
        prune(X_sampled)                                                 } (Section 4.2.5)
        X_sampled ←⁺ sample(m, c_current)                                }
        V_R ← X_goal ; E_R ← ∅                                           }
        Q_F ← expand(x_start) ; Q_R ← X_goal                             }
until stopped
```

4.2.1 Notation

AIT* is described using the same notation as ABIT* (Section 3.2.1) with minor changes. The forward and reverse search trees are denoted by $\mathcal{F} = (V_{\mathcal{F}}, E_{\mathcal{F}})$ and $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$, respectively. The vertices in these trees, denoted by $V_{\mathcal{F}}$ and $V_{\mathcal{R}}$, are associated with valid states. The edges in the forward tree, $E_{\mathcal{F}} \subseteq V_{\mathcal{F}} \times V_{\mathcal{F}}$, represent valid connection between states. The edges in the reverse tree, $E_{\mathcal{R}} \subseteq V_{\mathcal{R}} \times V_{\mathcal{R}}$, may lead through invalid states.

The functions $\texttt{parent}_{\mathcal{F}}(\,\cdot\,)$, $\texttt{parent}_{\mathcal{R}}(\,\cdot\,)$, $\texttt{children}_{\mathcal{F}}(\,\cdot\,)$ and $\texttt{children}_{\mathcal{R}}(\,\cdot\,)$ return the parent and children of a state in the forward and reverse trees, respectively. Symbols with square brackets denote labels, e.g., $l[\mathbf{x}] \in \mathbb{R}$ refers to a real number, l, associated with the state $\mathbf{x}$. Labels keep their values until they are updated, i.e., they are used in AIT* similar to how g-values are used in A*.

4.2.2 Initialization

AIT* starts by initializing the forward and reverse search trees with the start and goal states as their respective roots and the set of sampled states with the start and goals (Alg. 7, lines 1–3). It then expands all outgoing edges of the start state into the forward queue and inserts all goal states into the reverse queue (Alg. 7, line 4).

4.2.3 Reverse search

The reverse search of AIT* calculates an admissible cost-to-go heuristic for each RGG approximation. This cost-to-go heuristic is calculated by combining *a priori* admissible edge cost heuristics with an LPA* search that preserves the admissibility of the heuristic between each state and the goal. The calculated cost-to-go heuristic contains environment-specific information because the reverse search considers the connectivity of the RGG approximation, which is determined by the current distribution of *valid* samples. This reverse search is computationally inexpensive because edges are not checked for collision and only evaluated with a cost heuristic.

Algorithm 8: AIT*: `update_state` $(\mathbf{x})$

1 **if** $\mathbf{x} \neq \mathbf{x}_{\text{start}}$
2 $\quad \mathbf{x}_{\text{s}} \leftarrow \arg\min_{\mathbf{x}_i \in \texttt{neighbors}(\mathbf{x})} \left\{ \hat{h}_{\text{exp}}\left[\mathbf{x}_i\right] + \hat{c}\left(\mathbf{x}, \mathbf{x}_i\right) \right\}$
3 $\quad \hat{h}_{\text{con}}\left[\mathbf{x}\right] \leftarrow \hat{h}_{\text{exp}}\left[\mathbf{x}_{\text{s}}\right] + \hat{c}\left(\mathbf{x}, \mathbf{x}_{\text{s}}\right)$
4 $\quad$ **if** $\mathbf{x} \in V_{\mathcal{R}}$
5 $\quad\quad E_{\mathcal{R}} \xleftarrow{-} \left(\texttt{parent}_{\mathcal{R}}\left(\mathbf{x}\right), \mathbf{x}\right)$
6 $\quad$ **else**
7 $\quad\quad V_{\mathcal{R}} \xleftarrow{+} \mathbf{x}$
8 $\quad E_{\mathcal{R}} \xleftarrow{+} \left(\mathbf{x}_{\text{s}}, \mathbf{x}\right)$
9 $\quad$ **if** $\hat{h}_{\text{con}}\left[\mathbf{x}\right] \neq \hat{h}_{\text{exp}}\left[\mathbf{x}\right]$
10 $\quad\quad$ **if** $\mathbf{x} \notin \mathcal{Q}_{\mathcal{R}}$
11 $\quad\quad\quad \mathcal{Q}_{\mathcal{R}} \xleftarrow{+} \mathbf{x}$
12 $\quad$ **else if** $\mathbf{x} \in \mathcal{Q}_{\mathcal{R}}$
13 $\quad\quad \mathcal{Q}_{\mathcal{R}} \xleftarrow{-} \mathbf{x}$

The vertex queue of this reverse LPA* search is denoted by $\mathcal{Q}_{\mathcal{R}}$ and lexicographically ordered first by the heuristic total potential solution cost and then by the reverse cost-to-come,

$$\texttt{key}_{\mathcal{R}}^{\text{AIT*}}\left(\mathbf{x}\right) \coloneqq \Big(\min\Big\{ \hat{h}_{\text{con}}\left[\mathbf{x}\right], \hat{h}_{\text{exp}}\left[\mathbf{x}\right] \Big\} + \hat{g}\left(\mathbf{x}\right), \min\Big\{ \hat{h}_{\text{con}}\left[\mathbf{x}\right], \hat{h}_{\text{exp}}\left[\mathbf{x}\right] \Big\} \Big),$$

where $\hat{h}_{\text{con}}\left[\mathbf{x}\right]$ is the reverse cost-to-come to a state when it was first connected or last rewired, $\hat{h}_{\text{exp}}\left[\mathbf{x}\right]$ is the reverse cost-to-come to a state when it was last expanded, and $\hat{g}\left(\mathbf{x}\right)$ denotes an admissible *a priori* cost heuristic between a state and the start. The $\hat{h}_{\text{con}}$- and $\hat{h}_{\text{exp}}$-values are the g- and v-values in the forward LPA* search presented by Aine and Likhachev (2016). This key is used to select the next vertex in the reverse search (Alg. 7, lines 7 and 8).

An uninitialized LPA* search is used to calculate the cost-to-go heuristic on the first batch of samples and after each new batch is added. This is more efficient than incrementally updating the heuristic with LPA* for the large changes in the graph that result from increasing its resolution (Koenig et al., 2004; Likhachev

Algorithm 9: AIT*: `continue_reverse_search` ()

1 **if** $\mathcal{Q}_{\mathcal{F}} = \emptyset$ `or` $\mathcal{Q}_{\mathcal{R}} = \emptyset$
2 ⌊ **return** `false`
3 **if** $\forall (\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}, \hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}] = \hat{h}_{\mathrm{exp}}[\mathbf{x}_{\mathrm{t}}]$ `and` $\hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}] + \hat{g}(\mathbf{x}_{\mathrm{t}}) \leq \min_{\mathbf{x} \in \mathcal{Q}_{\mathcal{R}}} \left\{ \hat{h}_{\mathrm{con}}[\mathbf{x}] + \hat{g}(\mathbf{x}) \right\}$
4 ⌊ **return** `false`
5 $(\mathbf{x}_{\mathrm{s}}^*, \mathbf{x}_{\mathrm{t}}^*) \leftarrow$ `get_best_edge` $(\mathcal{Q}_{\mathcal{F}})$
6 $\mathbf{x}^* \leftarrow$ `get_best_vertex` $(\mathcal{Q}_{\mathcal{R}})$
7 **if** $\hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}^*] = \hat{h}_{\mathrm{exp}}[\mathbf{x}_{\mathrm{t}}^*]$ `and`
$g_{\mathcal{F}}(\mathbf{x}_{\mathrm{s}}^*) + \hat{c}(\mathbf{x}_{\mathrm{s}}^*, \mathbf{x}_{\mathrm{t}}^*) + \hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}^*] \leq \min\left\{ \hat{h}_{\mathrm{con}}[\mathbf{x}^*], \hat{h}_{\mathrm{exp}}[\mathbf{x}^*] \right\} + \hat{g}(\mathbf{x}^*)$
8 ⌊ **return** `false`
9 **return** `true`

and Koenig, 2005; Aine and Likhachev, 2016). An uninitialized LPA* search is started by clearing the reverse search tree (except for the goals), which resets the $\hat{h}_{\mathrm{con}}$- and $\hat{h}_{\mathrm{exp}}$-values of all cleared states to infinity, and inserting the goals into the reverse queue (Alg. 7, lines 2, 4, 39, and 40).

The heuristic is updated when an edge in the reverse search tree is found to be invalid. This is achieved efficiently by repairing the reverse search tree with LPA*. AIT* resets the calculated cost-to-go of all states in the affected branch of the reverse tree, inserts the necessary states into the reverse queue, and runs the reverse search to update the calculated cost-to-go as necessary (Alg. 10). The heuristics edge costs are evaluated in the reverse search from the target to the source to ensure admissible calculated cost heuristics for the forward search if the edge cost heuristic is not symmetric in its arguments (Alg. 8, lines 2 and 3).

4.2.3.1 Termination and suspension conditions

The reverse search calculates the admissible cost heuristic in a just-in-time manner and is terminated or suspended under multiple conditions. The reverse search is terminated if the reverse queue is empty (Alg. 9, lines 1 and 2). The reverse search is also terminated if the forward queue is empty (Alg. 9, lines 1 and 2), because the reverse search cannot fill the forward queue and an empty forward queue terminates the forward search (Alg. 7, line 16, and Alg. 11, lines 1 and 2).

The reverse search is suspended if all edges in the forward queue have consistent target states with a key-value less than or equal to the minimum key-value in the reverse queue (Alg. 9, line 3 and 4). The reverse search is also suspended if the heuristic total potential solution cost of the best state in the reverse queue is higher than or equal to the total potential solution cost of the best edge in the forward queue and the target state of that edge is consistent (Alg. 9, lines 5–8).

The last suspension condition allows the forward search to continue even if some edges in its queue have suboptimally connected targets. The forward *search* is still ordered according to an admissible cost heuristic because the suspension condition guarantees that the next best edge in the forward *queue* is known at each iteration (Section 4.3.1).

4.2.4 Forward search

AIT* finds solutions to a planning problem by building a search tree rooted at the start with an edge-queue version of A* that leverages the heuristic calculated by the reverse search. AIT* uses an edge-queue version of A* to delay expensive edge evaluations, similar to LWA*. This queue is denoted by $\mathcal{Q}_{\mathcal{F}}$ and lexicographically ordered first by the total potential solution cost of an edge, then by the potential cost-to-come to the target of the edge, and then by the cost-to-come to the source of the edge,

$$\begin{aligned}\texttt{key}_{\mathcal{F}}^{\mathrm{AIT}^*}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right) \coloneqq \Big(& g_{\mathcal{F}}\left(\mathbf{x}_{\mathrm{s}}\right) + \hat{c}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right) + \hat{h}_{\mathrm{con}}\left[\mathbf{x}_{\mathrm{t}}\right], \\ & g_{\mathcal{F}}\left(\mathbf{x}_{\mathrm{s}}\right) + \hat{c}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right), \\ & g_{\mathcal{F}}\left(\mathbf{x}_{\mathrm{s}}\right) \Big).\end{aligned}$$

A forward search iteration begins by extracting the edge with the lowest key-value, $\texttt{key}_{\mathcal{F}}^{\mathrm{AIT}^*}$, from the forward queue (Alg. 7, lines 17 and 18). If this edge is already part of the forward tree, then its target state is expanded and the iteration

Algorithm 10: AIT*: `invalidate_reverse_branch` $(\mathbf{x})$

```
1 if x ∉ X_goal
2     ĥ_con [x] ← ∞
3     E_R ←⁻ (parent_R (x), x)
4 ĥ_exp [x] ← ∞
5 Q_R ←⁻ x
6 for x_c ∈ children_R (x) do
7     invalidate_reverse_branch (x_c)
8 update_state (x)
```

is complete (Alg. 7, lines 19 and 20). If the edge is not in the forward tree but can possibly improve it, then it is checked for validity (Alg. 7, lines 21 and 22). If the edge is invalid, then it is added to the set of invalid edges and if it is also part of the reverse tree, then the affected branch of the reverse search is invalidated (Alg. 7, lines 33 and 35, and Alg. 10). If the edge is valid, then its true cost is evaluated and it is checked whether the edge can actually improve the current solution and forward tree (Alg. 7, lines 23 and 24).

If the edge can improve the current solution and forward tree, then its target state is added to this tree if it is not already in it (Alg. 7, lines 25 and 26). If it is already in the forward search tree, then the new edge constitutes a rewiring and the old edge is removed from the tree (Alg. 7, line 28). The new edge is added to the forward tree and its target state is expanded regardless of whether the target state was already in the tree or not (Alg. 7, lines 29 and 30, and Alg. 12). A forward search iteration finishes by updating the current solution cost (Alg. 7, line 31).

4.2.4.1 Termination conditions

The forward search is terminated under multiple conditions. The forward search is terminated if the forward queue is empty (Alg. 11, lines 1 and 2). The forward search is also terminated if no edge in the forward queue has a target in the reverse search tree (Alg. 11, lines 3 and 4), because then the start and goal are not in the same component of the current RGG approximation and it is impossible for

Algorithm 11: AIT*: `continue_forward_search` ()

1 **if** $\mathcal{Q}_{\mathcal{F}} = \emptyset$
2 ⌊ **return** `false`
3 **if** $\forall (\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}, \hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}] = \infty$
4 ⌊ **return** `false`
5 **if** $c_{\mathrm{current}} < \min_{(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}(\mathbf{x}_{\mathrm{s}}) + \hat{c}(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) + \hat{h}_{\mathrm{con}}[\mathbf{x}_{\mathrm{t}}] \right\}$
6 ⌊ **return** `false`
7 **return** `true`

AIT* to find a solution without improving its approximation. The forward search is also terminated if no edge in the forward queue can possibly improve the current solution with the current RGG approximation (Alg. 11, lines 5 and 6), because then AIT* has found the resolution-optimal solution and can only improve it by improving its approximation first.

If the forward search is terminated, then AIT* clears the reverse tree, improves the approximation, and resets the search queues (Alg. 7, lines 37–40). The next main loop iteration (Alg. 7, lines 5–41) will then restart the reverse search. This process is repeated for as long as computational time allows or until a suitable solution is found. This results in a progressively more accurate heuristic for increasingly efficient searches of ever more refined approximations and almost-surely asymptotically converges to the optimal solution in the limit of infinite samples (Section 4.3).

4.2.5 Approximation

AIT* uses a series of increasingly dense, edge-implicit RGGs to approximate the search space, similar to BIT* and ABIT*. It also uniformly samples batches of m valid states using informed sampling when possible, and implicitly defines connections between these states either by a connection radius, r, or by the k-nearest neighbors (Alg. 13, line 1).

The k-nearest connection model generally does not result in symmetric neighbors and special care must be taken to add each sample to the neighbors of its k-nearest

Algorithm 12: AIT*: $\texttt{expand}\,(X)$

$E_{\text{out}} \leftarrow \emptyset$
for all $\mathbf{x}_i \in X$ **do**
  for all $\mathbf{x}_j \in \texttt{neighbors}\,(\mathbf{x}_i)$ **do**
    $E_{\text{out}} \xleftarrow{+} (\mathbf{x}_i, \mathbf{x}_j)$
return E_{out}

Algorithm 13: AIT*: $\texttt{neighbors}\,(\mathbf{x})$

$V_{\text{neighbors}} \leftarrow \texttt{nearest}\,(\mathbf{x}, r \text{ or } k)$
$V_{\text{neighbors}} \xleftarrow{+} \{\texttt{parent}_{\mathcal{F}}\,(\mathbf{x}) \cup \texttt{children}_{\mathcal{F}}\,(\mathbf{x})\} \setminus V_{\text{neighbors}}$
$V_{\text{neighbors}} \xleftarrow{-} \{\mathbf{x}_i \in V_{\text{neighbors}} \mid (\mathbf{x}, \mathbf{x}_i) \in E_{\text{invalid}}\}$
return $V_{\text{neighbors}}$

Algorithm 14: AIT*: $\texttt{prune}\,(V, E, X_{\text{sampled}})$

$X_{\text{sampled}} \leftarrow \left\{\mathbf{x} \in X_{\text{sampled}} \;\middle|\; \hat{f}\,(\mathbf{x}) \leq c_{\text{current}}\right\}$
$V_{\mathcal{F}} \leftarrow \left\{\mathbf{x} \in V_{\mathcal{F}} \;\middle|\; \hat{f}\,(\mathbf{x}) \leq c_{\text{current}}\right\}$
$E_{\mathcal{F}} \leftarrow \left\{(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in E_{\mathcal{F}} \;\middle|\; \hat{f}\,(\mathbf{x}_{\text{s}}) \leq c_{\text{current}} \text{ and } \hat{f}\,(\mathbf{x}_{\text{t}}) \leq c_{\text{current}}\right\}$

neighbors. Otherwise the reverse and forward searches may process different graphs, which can result in inadmissible heuristics for the forward search. A consequence of this is that, in contrast to BIT* and ABIT*, AIT* considers the *mutual* k-nearest neighbors of each sample, i.e., each sample is connected to its k-nearest neighbors and all samples that it is a k-nearest neighbor of (as in FMT*; Janson et al., 2015).

The connection parameters, r and k, again scale as in PRM* (Karaman and Frazzoli, 2011), using the measure of the informed set as in BIT* (Gammell et al., 2020),

$$r(q) \coloneqq 2\eta\left(1+\frac{1}{n}\right)^{\frac{1}{n}}\left(\frac{\min\left\{\lambda\left(X\right), \lambda\left(X_{\hat{f}}\right)\right\}}{\lambda\left(B_{1,n}\right)}\right)^{\frac{1}{n}}\left(\frac{\log\left(q\right)}{q}\right)^{\frac{1}{n}}$$
$$k(q) \coloneqq \eta\,\mathrm{e}\left(1+\frac{1}{n}\right)\log\left(q\right),$$

where q is the number of samples in the informed set, $\eta > 1$ is a tuning parameter, $\lambda(\,\cdot\,)$ denotes the Lebesgue measure, and $B_{1,n}$ is an n-dimensional unit ball. These are the same connection parameters as in ABIT* and are presented again in

this chapter for completeness. AIT* considers the combination of both this RGG definition and any existing connections in the forward tree (Alg. 13, lines 1 and 2) and ignores edges known to be invalid (Alg. 13, line 3). Graph complexity is again reduced by pruning samples that are not in the informed set (Alg. 7, line 37, and Alg. 14).

4.3 Analysis

As in the analysis of ABIT*, this analysis uses the fact that a sampling-based planning algorithm is almost-surely asymptotically optimal if its approximation almost-surely contains an asymptotically optimal solution and the graph-search algorithm is resolution-optimal. The almost-sure asymptotic optimality of AIT* follows again from proven properties of its approximation and graph-search algorithms.

The approximation constructed by AIT* almost-surely contains an asymptotically optimal solution because it contains all the edges in PRM* for any set of samples and PRM* is almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

The forward search of AIT* is resolution-optimal because A* is a resolution-optimal algorithm if it is provided with an admissible cost heuristic (Hart et al., 1968). Section 4.3.1 shows that the forward search processes edges as if all edge targets in its queue had an admissible cost heuristic if the reverse search is suspended as in AIT*.

The reverse search of AIT* without collision detection results in an admissible cost heuristic because checking collisions cannot decrease path cost and LPA* is a resolution-optimal algorithm (Aine and Likhachev, 2016). AIT* is therefore almost-surely asymptotically optimal when provided with a consistent *a priori* heuristic.

4.3.1 Reverse search suspension condition

This section shows that it is unnecessary to continue the reverse search of AIT* if at least one of its two reverse search suspension conditions is satisfied. AIT* suspends the reverse search if all edges in the forward queue have consistent target states with

a key-value less than or equal to the minimum key-value in the reverse queue (Alg. 9, line 3 and 4), because this guarantees that all of these edge targets are optimally connected in the reverse tree and the forward queue is therefore ordered according to an admissible cost heuristic (Theorem 2; Koenig et al., 2004).

AIT* also suspends the reverse search if the heuristic total potential solution cost of the best state in the reverse queue is higher than or equal to the total potential solution cost of the best edge in the forward queue and the target state of that edge is consistent (Alg. 9, lines 5–8), because this guarantees that the target state of the best edge in the forward queue is optimally connected in the reverse tree and no edge in the forward queue with a suboptimally connected target state would be better than this edge if its target state was optimally connected (Theorem 2).

Theorem 2 (AIT* reverse search suspension condition)**.** *Let the cost-to-come heuristic be defined by the edge cost heuristic,* $\hat{g}(\mathbf{x}) \coloneqq \hat{c}(\mathbf{x}_{\text{start}}, \mathbf{x})$*, and let this heuristic be consistent. If the heuristic total potential solution cost of the best vertex in the reverse queue is greater than or equal to the total potential solution cost of the best edge in the forward queue,*

$$\min_{\mathbf{x}^{\mathcal{R}} \in \mathcal{Q}_{\mathcal{R}}} \left\{ \min\left\{ \hat{h}_{\text{con}}\left[\mathbf{x}^{\mathcal{R}}\right], \hat{h}_{\text{exp}}\left[\mathbf{x}^{\mathcal{R}}\right] \right\} + \hat{g}\left(\mathbf{x}^{\mathcal{R}}\right) \right\} \geq \min_{\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}, \mathbf{x}_{\text{t}}^{\mathcal{F}}\right) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}\right) + \hat{c}\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}, \mathbf{x}_{\text{t}}^{\mathcal{F}}\right) + \hat{h}_{\text{con}}\left[\mathbf{x}_{\text{t}}^{\mathcal{F}}\right] \right\},$$

and the best edge in the forward queue has a consistent target state, then this target has an admissible calculated cost heuristic and the best edge in the forward queue will not be changed by continuing the reverse search.

Proof. Proving the theorem requires showing that if the reverse search is suspended, then the target state of the best edge in the forward queue is optimally connected in the reverse tree and no edge in the forward queue with a suboptimally connected target would be better than the best edge if the target was optimally connected. It

is first shown that the target state of the best edge in the forward queue is optimally connected in the reverse tree if the suspension condition is satisfied.

Let $(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*)$ be the best edge in the forward queue,

$$(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*) \coloneqq \underset{(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}) \in \mathcal{Q}_\mathcal{F}}{\arg\min} \left\{ g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) + \hat{h}_\mathrm{con}\left[\mathbf{x}_\mathrm{t}^\mathcal{F}\right] \right\}. \tag{4.1}$$

By the definition and consistency of the cost-to-come heuristic, $\hat{g}(\,\cdot\,)$, it holds that

$$\hat{g}(\mathbf{x}_\mathrm{t}^*) = \hat{c}(\mathbf{x}_\mathrm{start}, \mathbf{x}_\mathrm{t}^*) \leq \hat{c}(\mathbf{x}_\mathrm{start}, \mathbf{x}_\mathrm{s}^*) + \hat{c}(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*) = \hat{g}(\mathbf{x}_\mathrm{s}^*) + \hat{c}(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*)$$

and since $\hat{g}(\mathbf{x}_\mathrm{s}^*) \leq g_\mathcal{F}(\mathbf{x}_\mathrm{s}^*)$ by the admissibility of $\hat{g}(\,\cdot\,)$, it holds that

$$\hat{g}(\mathbf{x}_\mathrm{t}^*) \leq g_\mathcal{F}(\mathbf{x}_\mathrm{s}^*) + \hat{c}(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*).$$

Adding $\hat{h}_\mathrm{con}[\mathbf{x}_\mathrm{t}^*]$ to both sides of this inequality yields

$$\hat{h}_\mathrm{con}[\mathbf{x}_\mathrm{t}^*] + \hat{g}(\mathbf{x}_\mathrm{t}^*) \leq g_\mathcal{F}(\mathbf{x}_\mathrm{s}^*) + \hat{c}(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*) + \hat{h}_\mathrm{con}[\mathbf{x}_\mathrm{t}^*]. \tag{4.2}$$

As the edge $(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*)$ is the best in the forward queue (Equation 4.1) and Theorem 2 suspends the reverse search if the minimum key-value in the reverse queue is greater than or equal to the cost of this edge, the right hand side of Equation 4.2 must be less than or equal to the minimum key-value in the reverse queue,

$$\begin{aligned} g_\mathcal{F}(\mathbf{x}_\mathrm{s}^*) + \hat{c}(\mathbf{x}_\mathrm{s}^*, \mathbf{x}_\mathrm{t}^*) + \hat{h}_\mathrm{con}[\mathbf{x}_\mathrm{t}^*] & \\ \leq \min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}\left(\mathbf{x}^\mathcal{R}\right) \right\}. & \end{aligned} \tag{4.3}$$

Combining Equations 4.2 and 4.3 yields,

$$\hat{h}_\mathrm{con}[\mathbf{x}_\mathrm{t}^*] + \hat{g}(\mathbf{x}_\mathrm{t}^*) \leq \min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}\left(\mathbf{x}^\mathcal{R}\right) \right\}. \tag{4.4}$$

Theorem 2 requires the target state of the best edge, $\mathbf{x}_{\mathrm{t}}^{*}$, to be consistent. The connected and expanded versions of its cost-to-come are therefore equal,

$$\hat{h}_{\mathrm{con}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right] = \hat{h}_{\mathrm{exp}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right] \qquad \left(\implies \hat{h}_{\mathrm{con}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right] = \min\left\{\hat{h}_{\mathrm{con}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right], \hat{h}_{\mathrm{exp}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right]\right\}\right)$$

such that Equation 4.4 can be rewritten as

$$\begin{aligned}\min\left\{\hat{h}_{\mathrm{con}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right], \hat{h}_{\mathrm{exp}}\left[\mathbf{x}_{\mathrm{t}}^{*}\right]\right\} + \hat{g}\left(\mathbf{x}_{\mathrm{t}}^{*}\right) \\ \leq \min_{\mathbf{x}^{\mathcal{R}} \in \mathcal{Q}_{\mathcal{R}}} \left\{\min\left\{\hat{h}_{\mathrm{con}}\left[\mathbf{x}^{\mathcal{R}}\right], \hat{h}_{\mathrm{exp}}\left[\mathbf{x}^{\mathcal{R}}\right]\right\} + \hat{g}\left(\mathbf{x}^{\mathcal{R}}\right)\right\}. \quad (4.5)\end{aligned}$$

The left hand side of this inequality is the reverse key-value of the target state of the best edge in the forward queue, $\mathbf{x}_{\mathrm{t}}^{*}$, which is therefore shown to be upper bounded by the minimum key-value in the reverse queue.

Since the suspension condition of Theorem 2 requires the target state of the best edge in the forward queue, $\mathbf{x}_{\mathrm{t}}^{*}$, to be consistent and all consistent states with a key-value less than or equal to the minimum key-value in the queue are optimally connected by an LPA* search (Theorem 2; Koenig et al., 2004), the state $\mathbf{x}_{\mathrm{t}}^{*}$ is optimally connected and its calculated cost-to-go is an admissible cost heuristic for the forward search on the current RGG approximation.

It is now shown that no edge in the forward queue with a suboptimally connected target in the reverse tree would be better than the best edge in the forward queue if its target were optimally connected. First, a lower bound is derived on the optimal heuristic cost-to-go of the target state of any currently suboptimally connected edge in the forward queue. This lower bound is then used to prove that the suspension condition of Theorem 2 ensures that no edge in the forward queue with suboptimally connected target state would have a lower total potential solution cost than the best edge in the queue if its target state were optimally connected.

Let $(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}})$ be any edge in the forward queue with a suboptimally connected

target, $\mathbf{x}_\mathrm{t}$. Let its unknown optimal cost-to-go through the reverse tree be $\hat{h}^*_\mathrm{con}[\mathbf{x}_\mathrm{t}]$.

The total potential solution cost through the state $\mathbf{x}_\mathrm{t}$ is lower bounded by the minimum value in the reverse queue,

$$\hat{h}^*_\mathrm{con}[\mathbf{x}_\mathrm{t}] + \hat{g}(\mathbf{x}_\mathrm{t}) \geq \min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}\left(\mathbf{x}^\mathcal{R}\right) \right\},$$

because the key-values in an LPA* search are monotonically nondecreasing (Theorem 1; Koenig et al., 2004). Rearranging this inequality yields a lower bound on the value of the optimal connection cost of the target state, $\hat{h}^*_\mathrm{con}[\mathbf{x}_\mathrm{t}]$,

$$\hat{h}^*_\mathrm{con}[\mathbf{x}_\mathrm{t}] \geq \min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}(\mathbf{x}) \right\} - \hat{g}(\mathbf{x}_\mathrm{t}). \tag{4.6}$$

This lower bound is now used to prove that the suspension condition of the theorem ensures that the edge $(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t})$ would not have the lowest total potential solution cost in the forward queue even if its target were optimally connected.

Theorem 2 states that the search can be suspended when the heuristic total potential solution cost of the best vertex in the reverse queue is greater than or equal to that of the best edge in the forward queue,

$$\min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}\left(\mathbf{x}^\mathcal{R}\right) \right\} \geq \min_{\left(\mathbf{x}^\mathcal{F}_\mathrm{s}, \mathbf{x}^\mathcal{F}_\mathrm{t}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}^\mathcal{F}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}^\mathcal{F}_\mathrm{s}, \mathbf{x}^\mathcal{F}_\mathrm{t}\right) + \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{F}_\mathrm{t}\right] \right\}.$$

If this condition is satisfied, then it must also be true that

$$\underbrace{g_\mathcal{F}(\mathbf{x}_\mathrm{s}) + \hat{c}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}) - \hat{g}(\mathbf{x}_\mathrm{t})}_{\text{nonnegative}} + \min_{\mathbf{x}^\mathcal{R} \in \mathcal{Q}_\mathcal{R}} \left\{ \min\left\{ \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{R}\right], \hat{h}_\mathrm{exp}\left[\mathbf{x}^\mathcal{R}\right] \right\} + \hat{g}\left(\mathbf{x}^\mathcal{R}\right) \right\} \geq \min_{\left(\mathbf{x}^\mathcal{F}_\mathrm{s}, \mathbf{x}^\mathcal{F}_\mathrm{t}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}^\mathcal{F}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}^\mathcal{F}_\mathrm{s}, \mathbf{x}^\mathcal{F}_\mathrm{t}\right) + \hat{h}_\mathrm{con}\left[\mathbf{x}^\mathcal{F}_\mathrm{t}\right] \right\},$$

where the underbraced term is nonnegative due to the consistency of the heuristics

$\hat{g}(\cdot)$ and $\hat{c}(\cdot, \cdot)$. Rearranging this equation yields

$$g_{\mathcal{F}}(\mathbf{x}_\mathrm{s}) + \hat{c}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}) + \underbrace{\min_{\mathbf{x}^{\mathcal{R}} \in \mathcal{Q}_{\mathcal{R}}} \left\{ \min\left\{ \hat{h}_{\mathrm{con}}\left[\mathbf{x}^{\mathcal{R}}\right], \hat{h}_{\mathrm{exp}}\left[\mathbf{x}^{\mathcal{R}}\right] \right\} + \hat{g}\left(\mathbf{x}^{\mathcal{R}}\right) \right\} - \hat{g}(\mathbf{x}_\mathrm{t})}_{\text{lower bound on } \hat{h}^*_{\mathrm{con}}[\mathbf{x}_\mathrm{t}]} \geq \min_{\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}, \mathbf{x}_\mathrm{t}^{\mathcal{F}}\right) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}, \mathbf{x}_\mathrm{t}^{\mathcal{F}}\right) + \hat{h}_{\mathrm{con}}\left[\mathbf{x}_\mathrm{t}^{\mathcal{F}}\right] \right\},$$

where the underbraced term is the lower bound on the optimally connected cost of the target state, $\hat{h}^*_{\mathrm{con}}[\mathbf{x}_\mathrm{t}]$, from Equation 4.6. Substituting this lower bound yields

$$g_{\mathcal{F}}(\mathbf{x}_\mathrm{s}) + \hat{c}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}) + \hat{h}^*_{\mathrm{con}}[\mathbf{x}_\mathrm{t}] \geq \min_{\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}, \mathbf{x}_\mathrm{t}^{\mathcal{F}}\right) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^{\mathcal{F}}, \mathbf{x}_\mathrm{t}^{\mathcal{F}}\right) + \hat{h}_{\mathrm{con}}\left[\mathbf{x}_\mathrm{t}^{\mathcal{F}}\right] \right\},$$

which proves that the total potential solution cost of the edge $(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t})$ would be greater than or equal to the total potential solution cost of the best edge in the forward queue even if its target state, $\mathbf{x}_\mathrm{t}$, were optimally connected. This proves that continuing the reverse search if the suspension condition of Theorem 2 is satisfied will not change the best edge in the forward queue and is therefore unnecessary. □

4.4 Evaluation

AIT* was evaluated against the OMPL implementations of RRT-Connect, FMT*, Lazy PRM*, Informed RRT*, BIT*, and ABIT* on Reeds-Shepp car problems (Section 4.4.1), a robotic manipulator arm problem (Section 4.4.2), and a knee replacement dislocation problem (Section 4.4.3). Asymptotically optimal planners again optimized path length in search space and informed planners again used the Euclidean distance as admissible cost heuristic. The performances of all planners were again evaluated statistically by letting all tested planners attempt each problem 100 times. The settings of all planners were kept as in Section 3.4. AIT* used the same approximation parameters as BIT* and ABIT*, i.e., sampled 100 states per batch and used the k-nearest RGG connection strategy with a factor of $\eta = 1.001$ regardless of the problem dimension.

RRT-C.	Inf. RRT*	FMT*	Lazy PRM*	BIT*	ABIT*	AIT*
198 (99%)	179 (89.5%)	186 (93%)	180 (90%)	188 (94%)	186 (93%)	164 (82%)

Table 4.1: The numbers (and percentages) of the 200 Reeds-Shepp car problem instances on which the tested planners achieved a success rate of at least 50%. AIT* is the least reliable tested planner on this problem. Section 4.5 discusses potential reasons for this poor performance and ideas on how to mitigate them.

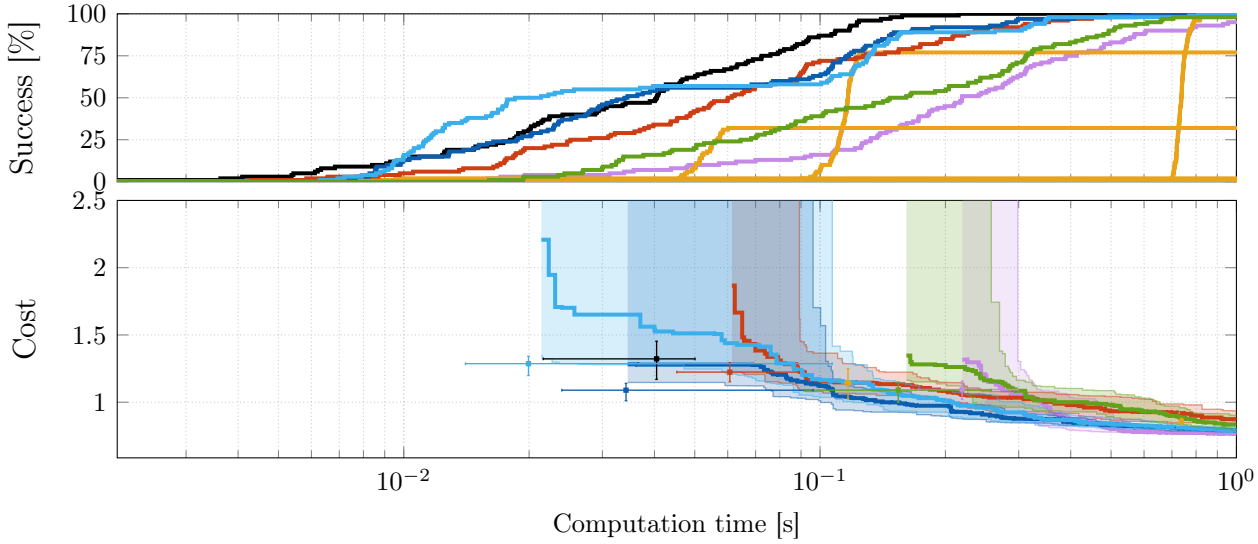
4.4.1 Reeds-Shepp car problems

AIT* was tested against the other planners on the same 200 instances of the Reeds-Shepp problem as ABIT* (Section 3.4.2). Figure 4.2 shows the best and worst performances of AIT* again in terms of its median initial solution time relative to the fastest median initial solution time of the other tested planners, limited to the problem instances where AIT* achieved at least 50% success rate (Table 4.1).

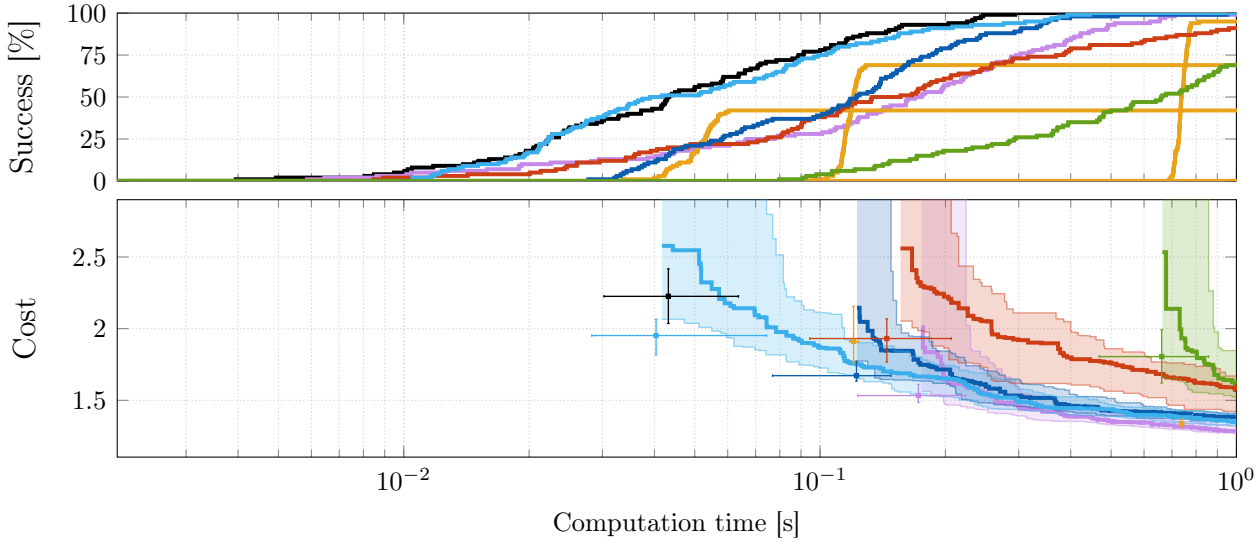
The results show that AIT* never outperforms all other tested planners on any instance of the Reeds-Shepp car problem. One reason for the poor relative performance of AIT* on these problems is that the obstacles are small relative to the extent of the search space (Figure 3.3). Small obstacles do not sufficiently impact the connectivity of the RGG approximations because they only induce small irregularities in the observed distribution of valid samples.

Another reason for the poor relative performance of AIT* on these problems is that the collision detection is relatively inexpensive due to the low collision detection resolution and simple, rectangular shapes of the car and obstacles. This decreases the computational cost of the forward search, which benefits the absolute performance of the other tested planners more than that of AIT*.

The combination of these effects leads to a calculated heuristic with too little improved accuracy to offset the computational cost of calculating it. Techniques to improve the performance of asymmetric bidirectional searches on such problems are discussed in Section 4.5 and implemented in Chapter 5.



(a) Best relative performance of AIT*



(b) Worst relative performance of AIT*

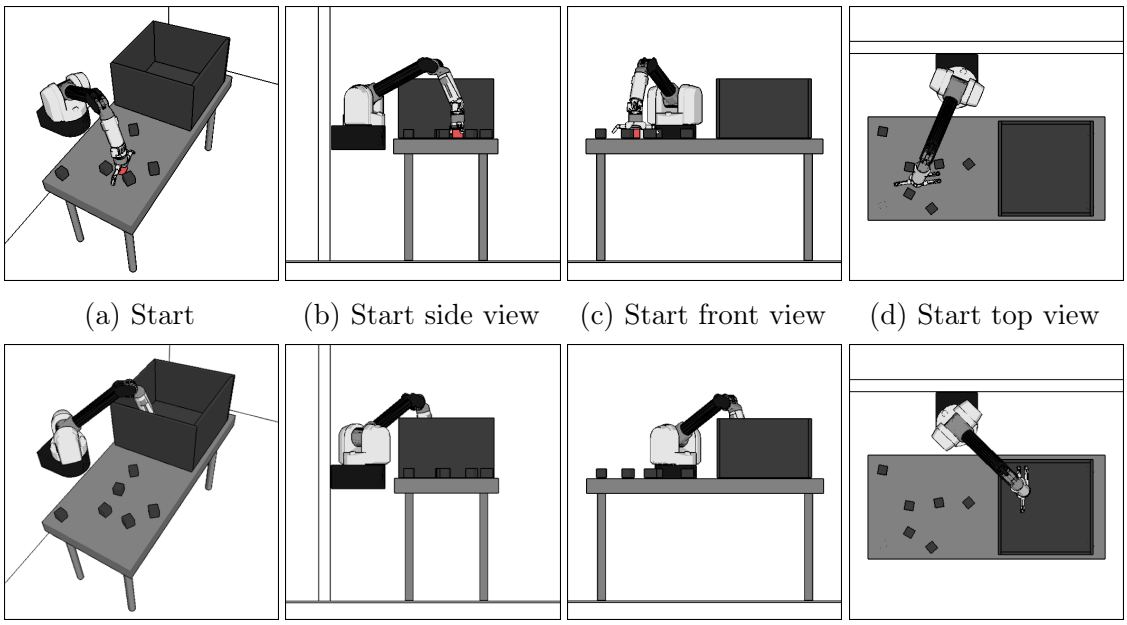
Figure 4.2: The planner performances on the Reeds-Shepp car problem that resulted in the best (a) and worst (b) performances of AIT* relative to the other tested planners (Section 4.4.1). The results show that AIT* never outperforms all other tested planners on any instance. On the best instance for AIT*, it performs similarly to Informed RRT* but is outperformed by the other tested planners in terms of median initial solution times and success rates (a). On the worst instance for AIT*, it is outperformed by all other tested planners in terms of median initial solution times and success and convergence rates (b). Section 4.5 discusses potential reasons for this poor performance and ideas on how to mitigate them.

4.4.2 Manipulator arm problem

The relative performance of AIT* improves when evaluated on problems where collision detection is more expensive, e.g., due to more complex geometries of search-space obstacles. This was the case when the planners were tested on a path planning problem for a single Barret Whole-Arm Manipulator (WAM) with seven degrees of freedom. This problem simulates the movement of a pick-and-place scenario, where the arm is required to pick up a small cube from a table and place it into a large box (Figure 4.3).

The problem was set up in the Open Robotics Automation Virtual Environment (OpenRAVE; Diankov, 2010), which was configured to use the Flexible Collision Library (FCL; Pan et al., 2012) with Oriented Bounding Box (OBB; Gottschalk et al., 1996) tree representations for collision detection. The time limit was 200 seconds per attempt and the collision detection resolution was $3.6 \cdot 10^{-3}$. This resolution resulted in a 1% false-negative detection rate for invalid edges on a representative problem, i.e., 1% of invalid edges are thought to be valid at this resolution.

Figure 4.4 shows the performances of all tested planners on this problem. The results show that on this problem AIT* performs better than Informed RRT* and FMT* and equally well as BIT* and ABIT*. AIT* is among the fastest tested planners to reach 100% success rate. The only two tested planners with faster median initial solutions times are RRT-Connect and Lazy PRM*. But RRT-Connect is not an anytime algorithm and does not improve the quality of its solution given more computation time. Lazy PRM* is an anytime algorithm but has the slowest convergence rate of all tested anytime algorithms on this problem and finds the lowest-quality solution in the available time.



(a) Start (b) Start side view (c) Start front view (d) Start top view

(e) Goal (f) Goal side view (g) Goal front view (h) Goal top view

Figure 4.3: Illustrations of the single-arm manipulator problem (Section 4.4.2). The top row shows the start configuration of the arm in position to pick up the red cube (a–d). The bottom row shows the goal configuration of the arm in position to place it in the box (e–h).

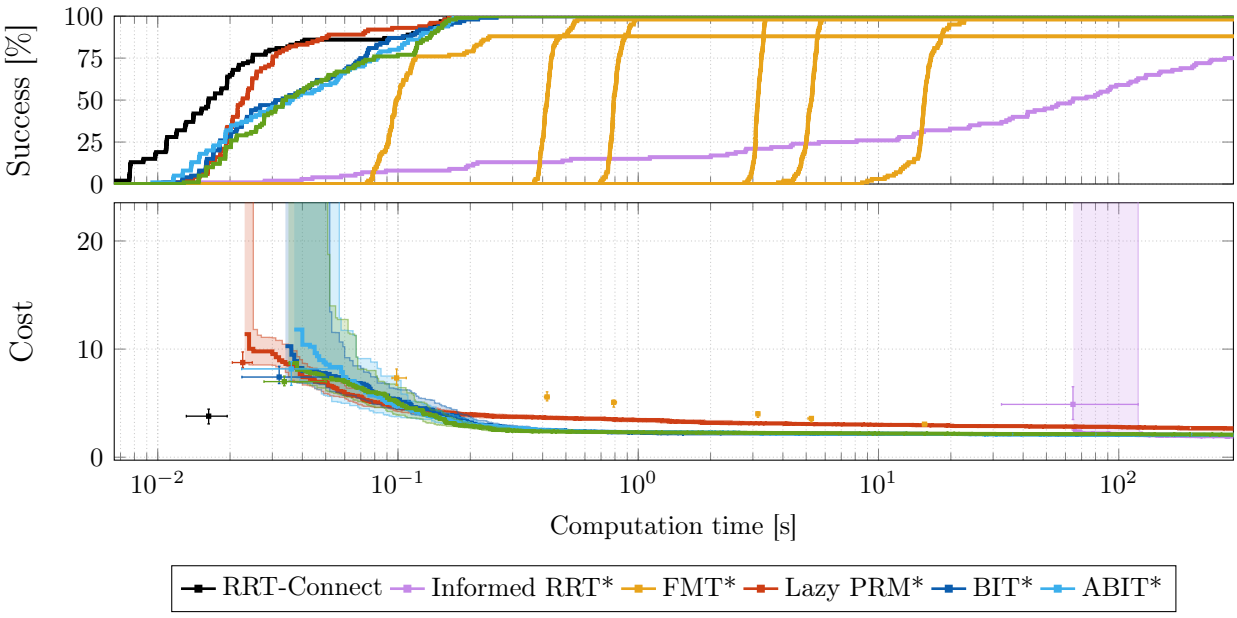


Figure 4.4: The planner performances on the manipulator arm problem described in Section 4.4.2 (Figure 4.3). The results show that AIT* performs as well as BIT* and ABIT*. In terms of success rates, AIT* can solve this problem as reliably as the most reliable planners (RRT-Connect, Lazy PRM*, BIT*, and ABIT*). In terms of initial solutions, AIT* is as fast as BIT* and ABIT* but is outperformed by the fastest planners (RRT-Connect and Lazy PRM*). In terms of convergence rate, AIT* is again as good as Informed RRT*, BIT* and ABIT*, which all find better solutions than Lazy PRM* in the available time.

4.4.3 Knee replacement dislocation problem

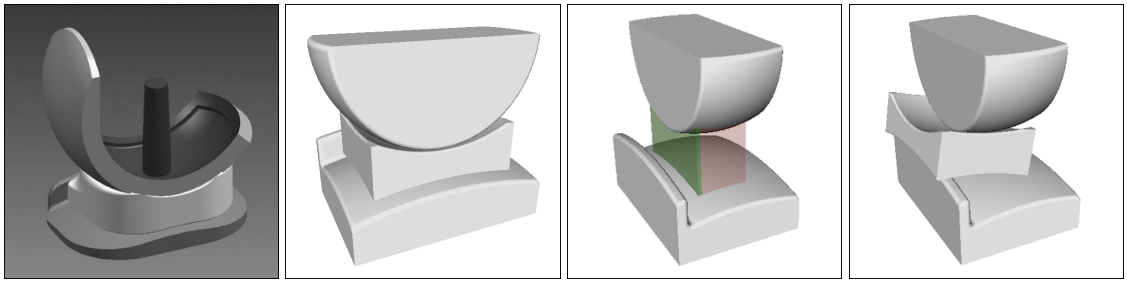
Environment-specific information can also improve performance on the feasible planning problem by guiding the search towards the goal. The knee replacement dislocation problem evaluates the potential of medial dislocation for the Oxford Domed Lateral Unicompartmental Knee Replacement (UKR; Pandit et al., 2010, Figure 4.5) by searching for a path to free the mobile bearing.[1]

The Oxford Domed Lateral UKR consists of metal femoral and tibial components which are fixed to the bone and a mobile polythylene bearing which separates the metal components (Gunther et al., 1996). Medial dislocation can occur when there is enough space between the femoral and tibial components for the mobile bearing to move onto the tibial-component wall where it may be trapped by the femoral component. The dislocation risk for different relative poses of the femoral and tibial components has been analyzed by using planning algorithms to search for paths that allow the bearing to reach a region representative of dislocation (Yang et al., 2020, 2021a,b). Figure 4.5 illustrates the search space, the fixed poses of the tibial and femoral components, and the start state and goal region of the mobile bearing used for this problem. The time limit was 200 seconds per attempt and the collision detection resolution was set to 0.0001 (as by Yang et al., 2020).

This problem requires a fine collision detection resolution and detailed Computer-Aided Design (CAD) models of all UKR components to ensure a valid assessment of the surgical tolerances. The combination of these two characteristics greatly increases the computational cost of collision detection. Computationally expensive collision detection improves the *relative* performance of AIT*, because it increases the benefits of using accurate heuristics to guide the (forward) search.

Figure 4.6 shows the performance of all tested planners on this problem excluding RRT-Connect, whose OMPL implementation does not support planning

[1]This problem used an approximation of the Oxford Domed Lateral UKR due to copyright restrictions.



(a) Original from Pandit et al. (2010) (b) Approximation with start state (c) Search space (red) and goal region (green) (d) Example goal state

Figure 4.5: A multiview illustration of the knee replacement dislocation problem (Section 4.4.3). The original 3D model of the Oxford Domed Lateral UKR (a) is reproduced from Figure 1 in Pandit et al. (2010). The experiments presented in this thesis used a simplified approximation of the Oxford Domed Lateral UKR (b). A medial dislocation occurs when the center of the mobile bearing moves from the red to the green region (c). The goal region is the set of bearing positions in medial dislocation between the fixed femoral and tibial components (d).

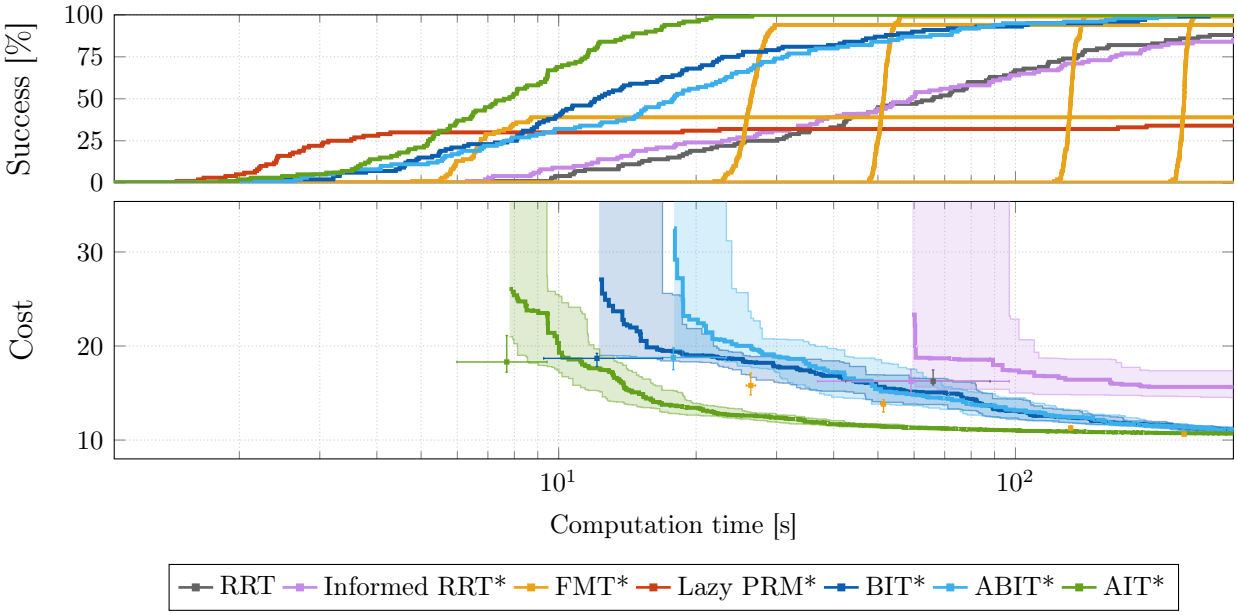


Figure 4.6: The planner performances on the knee replacement dislocation problem described in Section 4.4.3 (Figure 4.5). The results show that AIT* outperforms all other tested planners on this problem. In terms of success rates, AIT* finds solutions more reliably than any other tested planner. In terms of initial solutions, AIT* finds them faster than any other tested planner. In terms of convergence rate, AIT* finds the best solutions of any tested planner.

with goal regions. These results demonstrate the benefits of environment-specific information in sampling-based planning with computationally expensive collision detection. AIT* is the first planner to reach 100% success rate, returns the best-quality median solution at any time during the experiment, and has the fastest median initial solution time.

4.5 Discussion

It is often impossible or prohibitively complex to analytically represent the search-space obstacles of a planning problem, but algorithmically checking whether a single state is collision-free is often computationally feasible. This property is exploited by sampling-based algorithms. They sample individual states, check them for collision, and search for valid connections between samples that are collision-free.

Checking samples for collision reveals information about the environment of a search space. This environment-specific information is used by all sampling-based planners to build an approximation of the search space, but most planners ignore this information when searching this approximation. For example, BIT* and ABIT* use this information to build a series of increasingly dense, edge-implicit RGG approximations whose vertices are *valid* samples. But when BIT* and ABIT* search this approximation, they treat it as an abstract graph and ignore the information contained in the distribution of its vertices.

This chapter presents AIT*, a sampling-based planning algorithm that leverages optimization- and environment-specific information both when approximating the search space and when searching this approximation. It achieves this with a hierarchical bidirectional search that is asymmetric in both purpose and computational cost (anecdotally, processing a vertex in the reverse search on average only took 7% as long as processing an edge in the forward search on a representative example with abstract obstacles). The inexpensive reverse search calculates an accurate,

admissible cost heuristic for the current approximation by combining admissible edge cost heuristics between multiple samples while considering the potential connections of the current RGG approximation. The expensive forward search leverages this accurate heuristic to find a solution to the given planning problem and informs the reverse search about invalid states in the RGG connections it considered.

The performance improvement of AIT* depends on the proportion of the computational cost incurred by the reverse search when calculating the heuristic as compared to the computational savings gained by the forward search when leveraging this heuristic. AIT* therefore performs best relative to other planners when the calculated heuristic results in large computational savings, as is the case when full edge evaluation is computationally expensive. The knee replacement dislocation problem (Section 4.4.3) is an example of expensive edge evaluation, because it requires high-quality CAD-models of the knee replacement which makes collision detection computationally expensive.

The performance improvement of AIT* also depends on the size of the obstacles relative to the search space, because the additional information of its calculated heuristic is due to obstacle-induced irregularities in the observed distribution of valid samples. If the search-space obstacles are too small to cause large irregularities, then there is not much information to be discovered with the reverse search. The Reeds-Shepp car problems (Section 4.4.1) are examples of small obstacles that do not affect the distribution of valid states enough to offset the computational cost of the reverse search, which is why AIT* is outperformed by other planners on these problems.

One way to improve performance on such problems is to perform sparse collision detection on the edges during the reverse search. This can discover smaller obstacles without incurring large computational cost, such that an asymmetric bidirectional search can be competitive even when search-space obstacles are small. Another way to improve the performance of AIT* is to combine it with an anytime forward search, similar to how ABIT* improves the performance of BIT*. Both of these

approaches are explored in Chapter 5. AIT* could potentially also be improved by using truncation similar to ABIT* but this was not implemented in AIT* to avoid increasing its algorithmic complexity.

In summary, this chapter presented the following core contributions:

- A review of sampling-based algorithms that leverage environment-specific information to guide their search and a discussion of their main conceptual differences to AIT* (Section 4.1).
- A detailed description of AIT*, which is a sampling-based planning algorithm with a new asymmetric bidirectional search paradigm that leverages environment-specific information in all aspects of sampling-based planning (Section 4.2).
- A proof of the almost-sure asymptotic optimality of AIT* that combines established results with a new result on its reverse search (Section 4.3).
- Empirical demonstrations of the benefits (and shortcomings) of AIT* on nonholonomic, manipulator, and biomedical problems (Section 4.4).
- A discussion of the empirical results and potential ways to improve asymmetric bidirectional searches in sampling-based planning (Section 4.5).

Chapter 5

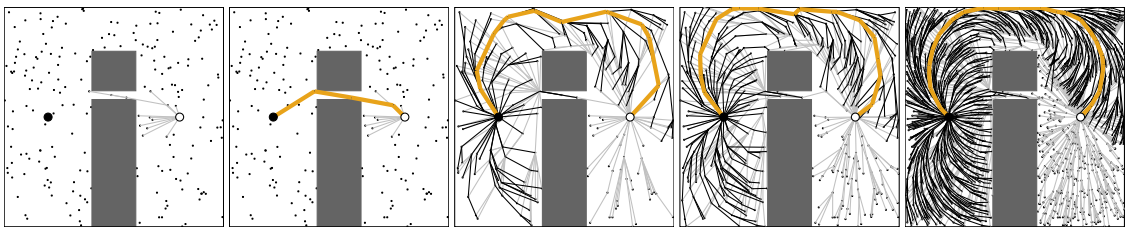
Effort Informed Trees (EIT*)

The benefits of intent-specific information

Contents

Path planning problems often have specific priorities. For examble, mobile robots in dynamic environments must potentially react to unexpected changes in their surroundings. These systems prioritize solution *time* over solution *quality* because not knowing how to react at all may be catastrophic.

Algorithms designed for such problems should reflect these priorities in their searches, but few planners directly align their searches with the priorities of the problems they are intended for. Many planners instead address these priorities by modifying their searches with indirect mechanisms that rely on implicit assumptions. For example, ABIT* seeks to accelerate initial solution times by inflating a cost heuristic. This can work, but implicitly assumes that the cost of a path correlates with the computational effort required to find it. This assumption is not always valid and can be removed by directly considering computational effort during the search.



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

Figure 5.1: Five snapshots of how EIT* searches a planning problem when optimizing obstacle clearance. EIT* starts by initializing the approximation and calculating cost and effort heuristics with a reverse search without full edge evaluation (a). EIT* then leverages these heuristics with its forward search to find initial solutions quickly. This forward search is greedily ordered on the effort heuristic until the initial solution is found (b). Once the initial solution is found, the forward search uses the calculated cost heuristics to find a resolution-optimal solution on the current approximation (c). Having found the resolution-optimal solution, EIT* improves the approximation, updates the heuristic, and aims to find the next best resolution-optimal solution with minimal computational effort (d). EIT* repeats these steps until it is stopped and almost-surely asymptotically converges towards the optimal solution given infinite computation time (e).

Computational effort is platform-specific and difficult to quantify, but collision detection and nearest-neighbor search are known to be the two computational bottlenecks in sampling-based planning (Sánchez and Latombe, 2003; Hauser, 2015; Kleinbort et al., 2016, 2020). Collision detection dominates the share of computational time early during the planning process while nearest-neighbor searches take over as the number of samples increases (Hauser, 2015; Kleinbort et al., 2016, 2020).

The number of collision checks required to find a solution is therefore a good proxy for computational effort when initial solution times are a priority. This number is equal to the length of a path divided by the collision detection resolution. The exact length of a solution is unknown *a priori* but can be estimated with the Euclidean distance between its start and end states. This provides a method to express intent-specific information about the computational effort required to find a solution as an effort heuristic that estimates the number of collision checks.

This chapter presents EIT*, a sampling-based planning algorithm that searches problems according to their priorities by leveraging intent-, optimization-, and

environment-specific information. EIT* uses these three sources of complementary information to simultaneously calculate and leverage two cost heuristics and one effort heuristic with a hierarchical bidirectional search that is asymmetric in purpose and computational cost, similar to AIT* (Figure 5.1).

The remainder of this chapter is organized as follows. It first presents a literature review of sampling-based planners and graph-search algorithms that leverage effort heuristics to guide their search (Section 5.1). It then presents a detailed description of EIT* (Section 5.2) and combines established results from the literature on sampling-based planning and graph-based search to prove its almost-sure asymptotic optimality (Section 5.3). The performance of EIT* is then evaluated by comparing it to other planners on many of the problems that were previously presented in this thesis, but with an additional optimization objective (Section 5.4). The chapter concludes by discussing these results and the insights they provide (Section 5.5).

The core contributions of this chapter are:

- A review of sampling-based planners and graph-search algorithms that use effort heuristics to guide their search (Section 5.1).
- A detailed presentation of EIT*, which demonstrates how to align the search of a continuous problem with its ultimate purpose by leveraging optimization-, environment-, and intent-specific information (Section 5.2).
- A proof that combines established results from the literature on sampling-based planning and graph-based search to show that EIT* is almost-surely asymptotically optimal (Section 5.3).
- Empirical demonstrations of the benefits of EIT* on multiple abstract, Reeds-Shepp, and manipulator problems, and one biomedical problem (Section 5.4).

The work on EIT* was has been accepted for publication in the International Journal of Robotics Research (IJRR; Strub and Gammell, 2021b).

5.1 Literature review

EIT* leverages optimization-, environment-, and intent-specific information in the form of admissible and inadmissible cost and effort heuristics. The literature reviews in Chapters 3 and 4 review the literature on optimization- and environment-specific information in sampling-based planning. This section focuses on the literature that calculates and exploits intent-specific information in the form of effort heuristics. Bayesian Effort-Aided Search Trees (BEAST; Kiesel et al., 2017) is the only other sampling-based algorithm that explicitly considers computational effort when ordering its search. BEAST is reviewed in detail in Section 5.1.1. Guiding the search with effort heuristics is more common in graph-based search. Graph-search algorithms that leverage effort heuristics are reviewed in Section 5.1.2.

5.1.1 Bayesian Effort-Aided Search Trees (BEAST)

BEAST is a sampling-based planning algorithm specialized for kinodynamic systems that aims to find solutions quickly by prioritizing low-effort paths. It first discretizes an abstraction of the search space into disjoint regions. It does so by building a PRM (Kavraki et al., 1996) in a version of the search space without kinodynamic constraints and defining regions by their nearest PRM-samples.

BEAST maintains estimates about the computational effort required to find a kinodynamically feasible path between adjacent regions and updates these estimates in a Bayesian manner each time it attempts to find a path between two regions. These effort estimates are associated with directed edges in the PRM so that BEAST can find low-effort paths from all regions to the goal by searching this PRM outward from the goal with D* Lite (Koenig and Likhachev, 2002). This assigns each region an effort-to-go value that reflects the current best estimate about how much computational effort is required to reach the goal from this region. These values are used to bias the sampling in BEAST.

BEAST initializes its search queue with all PRM-edges that have the region of the start state as the source. This queue is ordered on the total estimated effort-to-go, i.e., the estimated effort of the PRM-edge plus the estimated effort-to-go of the target region of the edge. Each iteration of BEAST then selects the edge from this queue with the lowest total estimated effort-to-go and samples a state from its target region. It then attempts to find a kinodynamically feasible path from the start state toward that sample. If it succeeds, then the effort estimate of the PRM-edge that represents the connection between the regions is lowered, and the outgoing edges of that region are added to the search queue. If the attempt was not successful, the estimated effort of the associated PRM-edge is increased. The total estimated effort-to-go values of all regions affected by this change are updated with D* Lite.

In this manner, BEAST effectively considers effort estimates in sampling-based planning and is shown to find solutions faster than other kinodynamic planners (Kiesel, 2016; Kiesel et al., 2017) but does not consider solution cost. BEAST probabilistically complete if it samples a fixed percentage of states uniformly at random (Kiesel et al., 2017).

Similar to BEAST, EIT* directly aligns its search with the priorities of the planning problem by calculating and leveraging a problem-specific effort heuristic. Unlike BEAST, EIT* requires an exact two-point Boundary Value Problem (BVP) solver for kinodynamic problems but does not require a user-specified abstraction of the search space and is almost-surely asymptotically optimal.

5.1.2 Graph-search algorithms with effort heuristics

Graph-based searches can accelerate solution times by incorporating estimates of computational effort to guide their searches towards low-effort solutions. Instead of approximating computational effort with the number of collision checks, graph-based searches often approximate it with *search distance*, i.e., the number of states that must be expanded to reach the goal.

A^*_ε (Pearl and Kim, 1982) is a version of A* with bounded suboptimality that aims to expand states as close to the goal as possible. It maintains two queues, one ordered on remaining solution effort and one on total potential solution cost, and always expands the state with the lowest remaining solution effort that is within the suboptimality bound. This results in few expanded states, especially when a loose suboptimality bound is acceptable, but introduces significant computational overhead and implementation complexity because the two queues have to be synchronized. Simplified A^*_ε (SA^*_ε; Hatem and Ruml, 2014) reduces this overhead by replacing the A* queue with iterative deepening, as in Iterative-Deepening A* (IDA*; Korf, 1985).

EES (Thayer and Ruml, 2010, 2011) builds on A^*_ε by additionally considering an inadmissible cost heuristic to guide the search. This can result in even fewer expanded states, but increases the computational overhead and complexity because EES has to maintain three synchronized queues. Simplified EES (SEES; Hatem and Ruml, 2014) simplifies EES by replacing two of these queues with iterative deepening, similarly to how SA^*_ε simplifies A^*_ε, but may expand more states than EES.

AEES (Thayer et al., 2012) is an anytime version of EES that aims to minimize the time between solution improvements. It orders its search based on admissible and inadmissible cost heuristics and an inadmissible effort heuristic. AEES starts by initializing its search queue with the start state. At each iteration, it then aims to select the state from this queue that leads to the fastest solution-cost improvement. It does so by prioritizing the state with the lowest effort estimate if it is estimated to lead to a sufficiently high-quality solution, and otherwise selecting the state that is estimated to be on the resolution-optimal solution or the state the provides a lower bound on the suboptimality of the current solution. An edge-queue version of AEES is used as the forward search in EIT* and described in detail in Section 5.2.4.

Instead of separately maintaining queues ordered on total potential solution cost and remaining effort estimates, BUGSY (Ruml and Do, 2007; Burns et al., 2013) is a best-first search algorithm that considers a weighted sum of total potential

solution cost and remaining effort estimates when ordering its search. The user can specify the ratio of the weights of this sum to reflect the amount of time they are willing to wait for an improvement of one cost unit, which provides an intuitive way to directly align the search with the priorities of the user.

Similar to these graph-based searches, EIT* uses estimates of computational effort to align its search with the priorities of the problem. Unlike these graph-based searches, EIT* can directly be applied to continuous path planning problems and includes a general way to calculate problem-specific effort heuristics.

5.2 Algorithm description

EIT* leverages optimization-, environment-, and intent-specific information to align its search with the priorities of the problem. It approximates the search space with the same series of increasingly dense, edge-implicit RGGs as AIT*, which are focused on the relevant region of the search space with informed sampling (Gammell et al., 2018). Similarly to AIT*, EIT* searches these approximations with an asymmetric bidirectional search in which both searches continuously inform each other. This bidirectional search is asymmetric both in purpose and in computational cost.

The reverse search of EIT* is an edge-queue version of A*. Similar to AIT*, it calculates two cost heuristics and one effort heuristic by combining *a priori* edge heuristics between multiple samples into more accurate approximation-specific heuristics between each sample and the goal. This considers the connectivity of the current RGG approximation and therefore leverages environment-specific information. Unlike AIT*, the reverse search detects large obstacles on potential connections by performing sparse collision detection on the edges. The reverse search is computationally inexpensive relative to the forward search because it does not evaluate true edge costs and because sparse collision detection is inexpensive relative to full collision detection (Sánchez and Latombe, 2003).

The forward search of EIT* is an edge-queue version of AEES that efficiently finds solutions to the given planning problem by leveraging the accurate, problem-specific heuristics calculated by the reverse search. If the forward search finds that the reverse search used an invalid edge to calculate the heuristics, then it causes the reverse search to update them with this information. The forward search is computationally expensive, as it evaluates true edge costs and performs full collision detections on the edges, but aligned with the priorities of the problem by the accurate heuristics calculated with the reverse search.

Both searches process the exact same increasingly dense, edge-implicit RGG as AIT*, which is presented in Section 4.2.5. The EIT* subroutines `expand`, `neighbors`, and `prune` are equivalent to their versions in AIT* and not repeated in this chapter. Once the resolution-optimal solution is found, EIT* samples more states and calculates new heuristics for the updated approximation. This process is repeated and will almost-surely asymptotically find the optimal solution.

5.2.1 Notation

EIT* is described using the same notation as ABIT* and AIT* (Sections 3.2.1 and 4.2.1) with minor extensions. A potentially inadmissible effort heuristic between two states is denoted by the function $\bar{e}\colon X \times X \to [0, \infty)$. This heuristic estimates the computational effort required to find and validate a path between the two states, e.g., the number of necessary collision detections to validate the path. A potentially inadmissible effort heuristic between the start and each state is denoted by the function $\bar{d}\colon X \to [0, \infty)$ and often defined as $\bar{d}(\mathbf{x}) \coloneqq \bar{e}(\mathbf{x}_{\text{start}}, \mathbf{x})$. A potentially inadmissible cost heuristic between two states is denoted by the function $\bar{c}\colon X \times X \to [0, \infty)$. It is assumed that the inadmissible cost heuristic is lower bounded by the admissible version, i.e.,

$$\forall\, \mathbf{x}_i, \mathbf{x}_j \in X, \quad \hat{c}(\mathbf{x}_i, \mathbf{x}_j) \leq \bar{c}(\mathbf{x}_i, \mathbf{x}_j)\,.$$

Algorithm 15: Effort Informed Trees (EIT*)

```
1  V_F ← x_start ; E_F ← ∅ ; V_R ← X_goal ; E_R ← ∅ ; V_R,closed ← ∅                    } Initialization
2  X_sampled ← {x_start} ∪ X_goal                                                      } (Section 5.2.2)
3  ρ ← update_sparse_collision_resolution () ; ε_i ← ∞                                 }
4  Q_F ← expand (x_start) ; Q_R ← expand (X_goal)                                      }
5  repeat
6      if continue_reverse_search ()                                                   } Reverse search
7          (x_s, x_t) ← arg min_{(x_s,x_t)∈Q_R} {key_R^EIT* (x_s, x_t)}                } (Section 5.2.3)
8          Q_R ←⁻ (x_s, x_t)                                                           }
9          V_R,closed ←⁺ x_s                                                           }
10         if no_sparse_collisions_detected ((x_s, x_t) , ρ)                           }
11             b̄ [x_t] ← min{b̄ [x_t] , b̄ [x_s] + c̄ (x_t, x_s)}                         }
12             b̄ [x_t] ← min{b̄ [x_t] , b̄ [x_s] + ē (x_t, x_s)}                         }
13             if ĥ [x_t] > ĥ [x_s] + ĉ (x_t, x_s)                                     }
14                 ĥ [x_t] ← ĥ [x_s] + ĉ (x_t, x_s)                                    }
15                 if x_t ∈ V_R                                                        }
16                     E_R ←⁻ (parent_R (x_t) , x_t)                                   }
17                 else                                                                }
18                     V_R ←⁺ x_t                                                      }
19                 E_R ←⁺ (x_s, x_t)                                                   }
20                 Q_R ←⁺ expand (x_t)                                                 }
21         else                                                                        }
22             E_invalid ←⁺ (x_s, x_t)                                                 }
23     else if continue_forward_search ()                                              } Forward search
24         (x_s, x_t) ← get_best_forward_edge (Q_F)                                    } (Section 5.2.4)
25         Q_F ←⁻ (x_s, x_t)                                                           }
26         if (x_s, x_t) ∈ E_F                                                         }
27             Q_F ←⁺ expand (x_t)                                                     }
28         else if g_F (x_s) + ĉ (x_s, x_t) < g_F (x_t)                                }
29             if collision_free (x_s, x_t)                                            }
30                 if g_F (x_s) + c (x_s, x_t) + ĥ_con [x_t] < c_current               }
31                     if g_F (x_s) + c (x_s, x_t) < g_F (x_t)                         }
32                         if x_t ∉ V_F                                                }
33                             V_F ←⁺ x_t                                              }
34                         else                                                        }
35                             E_F ←⁻ (parent_F (x_t) , x_t)                           }
36                         E_F ←⁺ (x_s, x_t)                                           }
37                         Q_F ←⁺ expand (x_t)                                         }
38                         if min_{x_goal∈X_goal} {g_F (x_goal)} < c_current           }
39                             c_current ← min_{x_goal∈X_goal} {g_F (x_goal)}          }
40                             ε_i ← update_inflation_factor ()                        }
41             else                                                                    }
42                 E_invalid ←⁺ {(x_s, x_t) , (x_t, x_s)}                              }
43                 if (x_s, x_t) ∈ E_R                                                 }
44                     ρ ← update_sparse_collision_resolution ()                       }
45                     V_R ← X_goal ; E_R ← ∅ ; Q_R ← expand (X_goal)                  }
46     else                                                                            } Approximation
47         V_R ← X_goal ; V_R,closed ← ∅ ; E_R ← ∅                                     } (Section 4.2.5)
48         prune (X_sampled) ; X_sampled ←⁺ sample (m, c_current)                      }
49         Q_F ← expand (x_start) ; Q_R ← expand (X_goal)                              }
50 until stopped
```

5.2.2 Initialization

EIT* starts by initializing the forward and reverse search trees with the start and goal states as their respective roots and the set of sampled states also with the start and goal states (Alg. 15, lines 1 and 2). It then initializes the sparse collision detection resolution, sets the inflation factor to infinity, and expands all outgoing edges of the start and goal states into the forward and reverse queues, respectively (Alg. 15, lines 3 and 4).

5.2.3 Reverse search

The reverse search of EIT* is an edge-queue version of A* with adaptive sparse collision detection. It calculates an admissible cost heuristic, an inadmissible cost heuristic, and an inadmissible effort heuristic for each RGG approximation. The calculated admissible cost heuristic is a lower bound on the optimal cost of a path from a state to the goal and is denoted by the label $\hat{h}[\,\cdot\,]$. The calculated inadmissible cost heuristic approximates the cost of an optimal path from a state to the goal and is denoted by the label $\bar{h}[\,\cdot\,]$. This inadmissible cost heuristic is often more accurate than the admissible version because it can contain additional information that may overestimate the true cost. The calculated inadmissible effort heuristic approximates the computational effort required to find and validate a path from a state to the goal and is denoted by the label $\bar{b}[\,\cdot\,]$. An example of such a heuristic is the number of collision checks required to validate a path, which is available and informative for all planning problems as it does not depend on the optimization objective but instead only on path length and collision detection resolution.

These heuristics are computed as in AIT* with a reverse search that combines *a priori* heuristics between multiple samples into more accurate heuristics between each sample and the goal. The calculated admissible cost heuristic, $\hat{h}[\,\cdot\,]$, is computed by combining *a priori* admissible cost heuristics, $\hat{c}(\,\cdot\,,\,\cdot\,)$, with a reverse

search that preserves the admissibility of the heuristic between each state and the goal. The calculated inadmissible cost and effort heuristics, $\bar{h}[\,\cdot\,]$ and $\bar{b}[\,\cdot\,]$, are similarly computed with the inadmissible *a priori* cost and effort heuristics, $\bar{c}(\,\cdot\,,\,\cdot\,)$ and $\bar{e}(\,\cdot\,,\,\cdot\,)$, respectively. All three heuristics always have a value of zero for all goal states. When calculating the heuristics with the reverse search, the edge heuristics are evaluated from the target to the source of the edge to ensure that the calculated heuristics are valid for the forward search even when the these heuristics are asymmetric in their arguments, as in AIT*.

The adaptive sparse collision detection is a computationally inexpensive way to find invalid edges in the heuristic calculation during the reverse search. Collision detection is considered a computationally expensive operation in sampling-based planning (Hauser, 2015; Kleinbort et al., 2016, 2020) but this is due to the computational cost of evaluating valid edges (Sánchez and Latombe, 2003). Detecting invalid edges with sparse collision detection is less computationally expensive and was found to be of similar computational cost to other operations in the reverse search when solving the problems presented in this thesis.

The edge queue of the reverse search in EIT* is denoted by $\mathcal{Q}_{\mathcal{R}}$ and lexicographically ordered first by the total potential solution cost of a path through an edge and then by the total potential computational effort required to validate this path,

$$\mathtt{key}_{\mathcal{R}}^{\mathrm{EIT}^*}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}) \coloneqq \Big(\hat{h}[\mathbf{x}_\mathrm{s}] + \hat{c}(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{s}) + \hat{g}(\mathbf{x}_\mathrm{t}),\; \bar{b}[\mathbf{x}_\mathrm{s}] + \bar{e}(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{s}) + \bar{d}(\mathbf{x}_\mathrm{t})\Big),$$

where $\hat{g}(\mathbf{x}_\mathrm{t})$ and $\bar{d}(\mathbf{x}_\mathrm{t})$ denote admissible *a priori* cost and inadmissible *a priori* effort heuristics for a path from the target state, $\mathbf{x}_\mathrm{t}$, to the start. The first part of the key ensures the admissibility of the calculated cost heuristic and the second part of the key breaks ties in favor of lower calculated effort, which is important if informative admissible cost heuristics are not available, e.g., when the trivial cost heuristic $\forall\, \mathbf{x}_1, \mathbf{x}_2 \in X, \hat{c}(\mathbf{x}_1, \mathbf{x}_2) \equiv \hat{g}(\mathbf{x}_1) \equiv 0$ is the only admissible option.

Algorithm 16: EIT*: `continue_reverse_search` ()

```
1  if Q_F = ∅ or Q_R = ∅
2      return false
3  if ∀ (x_s, x_t) ∈ Q_F, x_t ∈ V_R,closed
4      return false
5  if ε_i = ∞ and ∃ (x_s, x_t) ∈ Q_F, x_t ∈ V_R
6      return false
7  (x_s,F, x_t,F) ← get_best_forward_edge (Q_F)
8  (x_s,R, x_t,R) ← get_best_reverse_edge (Q_R)
9  if g_F (x_s,F) + ĉ (x_s,F, x_t,F) + ĥ [x_t,F] ≤ ĥ [x_s,R] + ĉ (x_t,R, x_s,R) + ĝ (x_t,R)
10     and x_t,F ∈ V_R,closed
11     return false
12 return true
```

Each iteration of the reverse search first removes the edge with the lowest key-value, $\texttt{key}_{\mathcal{R}}^{\text{EIT}^*}$, from the reverse queue, adds its source state to the set of closed vertices, and checks a number of evenly distributed states along this edge for collision (Alg. 15, lines 7–10). The number of collision checks, ρ, is adapted during the search with a user-specified policy. If a collision is found, then the edge is added to the set of invalid edges (Alg. 15, line 22), otherwise it is used to improve the inadmissible cost- and effort heuristics, if possible (Alg. 15, lines 11 and 12).

It is then checked whether the edge can improve the calculated admissible cost heuristic of the target state (Alg. 15, line 13). If it can, then the heuristic is updated and the target state is either rewired or added to the reverse search tree (Alg. 15, lines 14–19). The reverse search iteration is completed by expanding the outgoing edges of the target state into the reverse queue (Alg. 15, line 20 and Alg. 12).

5.2.3.1 Termination and suspension conditions

The reverse search calculates the admissible cost and effort heuristics in a just-in-time manner and is terminated or suspended under multiple conditions. The reverse search is terminated if the reverse queue is empty (Alg. 16, lines 1 and 2). The reverse search is also terminated if the forward queue is empty (Alg. 16, lines 1 and 2),

because the reverse search cannot fill the forward queue and an empty forward queue terminates the forward search (Alg. 15, lines 23 and 47, and Alg. 18, lines 1 and 2).

The reverse search is suspended if all edges in the forward queue have closed target states in the reverse search (Alg. 16, lines 3 and 4). The reverse search is also suspended if the forward search has an infinite inflation factor and any target state in the forward queue is in the reverse search tree (Alg. 16, lines 5 and 6). Finally, the reverse search is suspended if the total potential solution cost of the best edge in the reverse queue is greater than or equal to that of the best edge in the forward queue and the best edge in the forward queue has a closed target state (Alg. 16, lines 7–11).

The last suspension condition allows the reverse search to be suspended even if the inflation factor is not infinite and some edges in the forward queue have target states whose calculated heuristics are not guaranteed to be admissible. The forward *search* is still ordered according to an admissible cost heuristic because the suspension condition guarantees that the next best edge in the forward *queue* is known at each iteration (Section 5.3.1).

5.2.4 Forward search

The forward search of EIT* is an edge-queue version of AEES which leverages the cost and effort heuristics calculated by the reverse search. It leverages these heuristics in an anytime manner, which results in effective searches with fast initial solution times regardless of the optimization objective.

The forward search of EIT* searches the same RGG approximation multiple times with successively tighter suboptimality bounds. It initially prioritizes quickly finding any solution over efficiently finding the resolution-optimum, which improves anytime performance. Once an initial solution is found, EIT* uses both the calculated admissible and inadmissible cost heuristics to improve the tree until it can guarantee that it has found the resolution-optimal solution.

The forward search of EIT* delays expensive edge cost evaluations by using an edge queue similar to LWA*. This queue is ordered by separately considering a lower bound on the optimal solution cost (Section 5.2.4.1), an estimate of the optimal solution cost (Section 5.2.4.2), and an estimate of the minimum remaining effort to validate a solution within the current suboptimality bound (Section 5.2.4.3). Each of these values is respectively informed by the admissible cost heuristic, the inadmissible cost heuristic, and the inadmissible effort heuristic calculated by the reverse search.

5.2.4.1 Optimal cost bound

A lower bound on the optimal solution cost in the current RGG is computed as

$$\min_{(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})\in\mathcal{Q}_\mathcal{F}} \hat{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right),$$

where the function $\hat{s}\colon X_\mathrm{sampled}\times X_\mathrm{sampled}\to[0,\infty)$ estimates the cost of a solution that is constrained to go through a specific edge, $(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})$. This estimate is defined as

$$\hat{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right) \coloneqq g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right) + \hat{h}\left[\mathbf{x}_\mathrm{t}\right],$$

where $g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right)$ is the cost-to-come of the edge source, $\hat{c}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right)$ is the admissible cost heuristic of the edge, and $\hat{h}\left[\mathbf{x}_\mathrm{t}\right]$ is the admissible cost heuristic of the edge target, as calculated by the reverse search. At least one edge in the forward queue has an optimally connected source state (Lemma 1; Hart et al., 1968). The $\hat{s}$-value of that edge therefore provides a lower bound on the optimal solution cost in the current RGG approximation, if the heuristics $\hat{c}$ and $\hat{h}$ are admissible. The minimum $\hat{s}$-value in the forward queue is therefore also a lower bound on the optimal solution cost. The edge that corresponds to this lower bound is denoted as

$$\left(\mathbf{x}_\mathrm{s}^{\hat{s}},\mathbf{x}_\mathrm{t}^{\hat{s}}\right) \coloneqq \operatorname*{arg\,min}_{(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})\in\mathcal{Q}_\mathcal{F}} \hat{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right).$$

5.2.4.2 Optimal cost estimate

An inadmissible, but potentially more accurate, estimate of the optimal solution cost in the current RGG approximation is computed as

$$\min_{(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})\in\mathcal{Q}_\mathcal{F}} \bar{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right),$$

where $\bar{s}\colon X_\mathrm{sampled}\times X_\mathrm{sampled}\to[0,\infty)$ is also an estimate of the cost of a solution that is constrained to go through an edge but calculated with inadmissible cost heuristics. This estimate is defined as

$$\bar{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right) \coloneqq g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right) + \bar{c}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right) + \bar{h}\left[\mathbf{x}_\mathrm{t}\right],$$

where $g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right)$ is the cost to come to the edge source, $\bar{c}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right)$ is the inadmissible cost heuristic of the edge, and $\bar{h}\left[\mathbf{x}_\mathrm{t}\right]$ is the inadmissible cost heuristic of the edge target, as calculated by the reverse search. This estimate is often more accurate than the admissible lower bound because it can include information that may overestimate the true cost. The edge with the best inadmissible solution cost is denoted as

$$\left(\mathbf{x}_\mathrm{s}^{\bar{s}},\mathbf{x}_\mathrm{t}^{\bar{s}}\right) \coloneqq \operatorname*{arg\,min}_{(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})\in\mathcal{Q}_\mathcal{F}} \bar{s}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right).$$

5.2.4.3 Minimum effort estimate

An estimate of the minimum remaining effort to validate a solution within the suboptimality bound is computed as

$$\min_{(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t})\in\mathcal{Q}_\mathcal{F}^{\varepsilon_\mathrm{i}\bar{s}}} \bar{r}\left(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{t}\right),$$

where $\bar{r}\colon X_{\text{sampled}} \times X_{\text{sampled}} \to [0, \infty)$ estimates the remaining effort to validate a solution that goes through an edge. This estimate is defined as

$$\bar{r}\left(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}\right) \coloneqq \bar{e}\left(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}\right) + \bar{b}\left[\mathbf{x}_{\text{t}}\right],$$

where $\bar{e}\left(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}\right)$ is the heuristic effort to validate the edge and $\bar{b}\left[\mathbf{x}_{\text{t}}\right]$ is the heuristic effort to validate a path from the target of the edge to the goal, as calculated by the reverse search. The minimum is taken only over the edges in the queue that are estimated to lead to a solution within the current suboptimality factor, ε_{i},

$$\mathcal{Q}_{\mathcal{F}}^{\varepsilon_{\text{i}}\bar{s}} \coloneqq \{(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in \mathcal{Q}_{\mathcal{F}} \;\mid\; \bar{s}\left(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}\right) \leq \varepsilon_{\text{i}}\bar{s}\left(\mathbf{x}_{\text{s}}^{\bar{s}}, \mathbf{x}_{\text{t}}^{\bar{s}}\right)\}.$$

The edge that results in this estimate of the minimum remaining effort required to find a solution within the current suboptimality bound is denoted as

$$\left(\mathbf{x}_{\text{s}}^{\bar{r}}, \mathbf{x}_{\text{t}}^{\bar{r}}\right) \coloneqq \underset{(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}) \in \mathcal{Q}_{\mathcal{F}}^{\varepsilon_{\text{i}}\bar{s}}}{\arg\min}\; \bar{r}\left(\mathbf{x}_{\text{s}}, \mathbf{x}_{\text{t}}\right).$$

5.2.4.4 Edge processing

Each iteration of the forward search aims to process the edge in the queue that leads to the fastest solution-cost improvement within the current suboptimality bound. This edge is determined by prioritizing the edge with the lowest effort estimate, $(\mathbf{x}_{\text{s}}^{\bar{r}}, \mathbf{x}_{\text{t}}^{\bar{r}})$, if it satisfies the suboptimality bound, and otherwise selecting the edge with the lowest estimated cost, $(\mathbf{x}_{\text{s}}^{\bar{s}}, \mathbf{x}_{\text{t}}^{\bar{s}})$, if it satisfies the suboptimality bound. If neither of these edges satisfies the bound, then the edge that provides the current suboptimality bound, $\left(\mathbf{x}_{\text{s}}^{\hat{s}}, \mathbf{x}_{\text{t}}^{\hat{s}}\right)$, is selected.

The detailed selection steps are:

1. If the edge with the minimum remaining effort required to validate a solution, $(\mathbf{x}_{\text{s}}^{\bar{r}}, \mathbf{x}_{\text{t}}^{\bar{r}})$, is estimated to lead to a solution within the current suboptimality

Algorithm 17: EIT*: `get_best_forward_edge` $(\mathcal{Q}_{\mathcal{F}})$

1 $(\mathbf{x}_{\mathrm{s}}^{\bar{r}}, \mathbf{x}_{\mathrm{t}}^{\bar{r}}) \leftarrow \arg\min_{(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}^{\varepsilon_{\mathrm{i}} \bar{s}}} \left\{ \bar{e}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right) + \bar{b}\left[\mathbf{x}_{\mathrm{t}}\right] \right\}$
2 $(\mathbf{x}_{\mathrm{s}}^{\bar{s}}, \mathbf{x}_{\mathrm{t}}^{\bar{s}}) \leftarrow \arg\min_{(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_{\mathrm{s}}\right) + \bar{c}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right) + \bar{h}\left[\mathbf{x}_{\mathrm{t}}\right] \right\}$
3 $\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right) \leftarrow \arg\min_{(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_{\mathrm{s}}\right) + \hat{c}\left(\mathbf{x}_{\mathrm{s}}, \mathbf{x}_{\mathrm{t}}\right) + \hat{h}\left[\mathbf{x}_{\mathrm{t}}\right] \right\}$
4 **if** $\bar{s}\left(\mathbf{x}_{\mathrm{s}}^{\bar{r}}, \mathbf{x}_{\mathrm{t}}^{\bar{r}}\right) \leq \varepsilon_{\mathrm{i}} \hat{s}\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right)$
5 | **return** $\left(\mathbf{x}_{\mathrm{s}}^{\bar{r}}, \mathbf{x}_{\mathrm{t}}^{\bar{r}}\right)$
6 **else if** $\bar{s}\left(\mathbf{x}_{\mathrm{s}}^{\bar{s}}, \mathbf{x}_{\mathrm{t}}^{\bar{s}}\right) \leq \varepsilon_{\mathrm{i}} \hat{s}\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right)$
7 | **return** $\left(\mathbf{x}_{\mathrm{s}}^{\bar{s}}, \mathbf{x}_{\mathrm{t}}^{\bar{s}}\right)$
8 **else**
9 | **return** $\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right)$

bound,

$$\bar{s}\left(\mathbf{x}_{\mathrm{s}}^{\bar{r}}, \mathbf{x}_{\mathrm{t}}^{\bar{r}}\right) \leq \varepsilon_{\mathrm{i}} \hat{s}\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right),$$

then it is selected (Alg. 17, lines 4 and 5). This edge is estimated to improve the current solution with the least amount of computational effort.

2. If the edge that the inadmissible heuristic estimates to be on a resolution-optimal path, $(\mathbf{x}_{\mathrm{s}}^{\bar{s}}, \mathbf{x}_{\mathrm{t}}^{\bar{s}})$, is estimated to lead to a solution within the current suboptimality bound,

$$\bar{s}\left(\mathbf{x}_{\mathrm{s}}^{\bar{s}}, \mathbf{x}_{\mathrm{t}}^{\bar{s}}\right) \leq \varepsilon_{\mathrm{i}} \hat{s}\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right),$$

then it is selected (Alg. 17, lines 6 and 7). This edge is preferred over the edge that the admissible cost heuristic estimates to be on a resolution-optimal path, $\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right)$, because the inadmissible cost heuristic may contain more optimization-specific information than its admissible counterpart, including information that may overestimate the solution cost.

3. Otherwise the edge that provides the lower bound on the optimal solution cost in the current approximation, $\left(\mathbf{x}_{\mathrm{s}}^{\hat{s}}, \mathbf{x}_{\mathrm{t}}^{\hat{s}}\right)$, is selected. This raises the lower bound on the optimal solution cost in the current RGG approximation and

increases the number of candidates available to steps 1 and 2 in the next iteration (Alg. 17, lines 8 and 9).

The forward search of EIT* then proceeds similarly to that of AIT*, but accounts for the adaptive collision detection in the reverse search. If the selected edge is in the forward search tree, then its target state is expanded and the forward search iteration is complete (Alg. 15, lines 26 and 27). If the selected edge is not part of the forward search tree but can possibly improve it, then it is checked for collisions (Alg. 15, lines 28 and 29).

If collisions are detected, then the edge is added to the set of invalid edges (Alg. 15, line 42) and if it is in the reverse search tree, then the heuristics are updated by restarting the reverse search with an updated sparse collision detection resolution (Alg. 15, lines 43–45). The update policy of the sparse collision detection resolution is a user-specified parameter. The update policy used in the evaluation of EIT* is presented in Section 5.4.

If no collisions are detected, then the true cost of the edge is evaluated to check whether it actually improves the current solution and forward search tree (Alg. 15, lines 30 and 31). If it does, then its target state is added to the tree if it is not already in it (Alg. 15, lines 30–33). If the target state is already in the tree, then the edge causes a rewiring and the edge from the old parent is removed from the tree (Alg. 15, lines 34 and 35). The new edge is then added to the tree and its target state is expanded into the forward queue (Alg. 15, lines 36 and 37). If the edge improves the current solution, then the solution cost and the inflation factor are updated (Alg. 15, lines 39 and 40). The inflation factor update policy is a user-tuned parameter. The update policy used in the evaluation of EIT* is presented in Section 5.4.

Algorithm 18: EIT*: `continue_forward_search` ()

```
1 if Q_F = ∅
2     return false
3 if ∀ (x_s, x_t) ∈ Q_F, ĥ[x_t] = ∞
4     return false
5 if c_current < min_(x_s,x_t) { g_F(x_s) + ĉ(x_s, x_t) + ĥ[x_t] }
6     return false
7 return true
```

5.2.4.5 Termination conditions

The forward search can be terminated under the same conditions as the forward search of AIT*. The forward search is terminated if the forward queue is empty (Alg. 18, lines 1 and 2). The forward search is also terminated if no edge in the forward queue has a target in the reverse search tree (Alg. 18, lines 3 and 4) because then the start and goal are not in the same component of the current RGG approximation and it is impossible for EIT* to find a solution without improving its approximation. The forward search is also terminated if no edge in the forward queue can possibly improve the current solution in the current RGG approximation (Alg. 18, lines 5 and 6) because then EIT* has found the resolution-optimal solution and can only improve it by improving its approximation first.

If the forward search is terminated, then EIT* clears the set of closed vertices and the reverse search tree, improves the approximation, and resets the search queues (Alg. 15, lines 47–49). The next iteration of the main loop in EIT* (Alg. 15, lines 5–50) will then restart the reverse search. This process is repeated for as long as computational time allows or until a suitable solution is found. This results in progressively more accurate heuristics for increasingly effective searches of ever more refined approximations and will almost-surely asymptotically converge to the optimal solution in the limit of infinite computation time (Section 5.3).

5.3 Analysis

As in the analysis of ABIT* and AIT*, this analysis builds on the fact that a sampling-based path planning algorithm is almost-surely asymptotically optimal if its approximation almost-surely contains an asymptotically optimal solution and the graph-search algorithm is resolution-optimal. The almost-sure asymptotic optimality of EIT* follows again from proven properties of its approximation and graph-search algorithms.

The approximation constructed by EIT* almost-surely contains an asymptotically optimal solution because it contains all the edges in PRM* for any set of samples and PRM* is almost-surely asymptotically optimal (Karaman and Frazzoli, 2011).

The forward search of EIT* is resolution-optimal because AEES is a resolution-optimal algorithm if provided with an admissible cost heuristic and an inflation factor of one (Thayer and Ruml, 2011). Section 5.3.1 shows that the forward search processes edges as if all edge targets in its queue had an admissible cost heuristic if the reverse search is suspended as in EIT* and the provided admissible cost heuristic is consistent and symmetric in its arguments.

The reverse search of EIT* with sparse collision detection results in an admissible cost heuristic because full collision detection cannot decrease path cost and A* is a resolution-optimal graph-search algorithm when provided with an admissible cost heuristic (Hart et al., 1968). EIT* is therefore almost-surely asymptotically optimal if the provided admissible cost heuristic is consistent and symmetric in its arguments and the user-specified inflation factor update policy results in an inflation factor of one in the limit of infinite samples.

5.3.1 Reverse search suspension condition

This section shows that it is unnecessary to continue the reverse search of EIT* if at least one of its three reverse search suspension conditions is satisfied. EIT*

suspends the reverse search if all edges in the forward queue have closed target states (Alg. 16, lines 3 and 4), because then all of these states have admissible cost heuristics (Lemma 2; Hart et al., 1968).

EIT* also suspends the reverse search if the forward search has an infinite inflation factor and any target state in the forward queue is in the reverse search tree (Alg. 16, lines 5 and 6), because an infinite inflation factor results in an infinite suboptimality bound and therefore no guarantees about the solution quality can be made regardless of whether the calculated cost heuristic is admissible or not.

Finally, EIT* also suspends the reverse search if the total potential solution cost of the best edge in the reverse queue is greater than or equal to that of the best edge in the forward queue and the best edge in the forward queue has a closed target state (Alg. 16, lines 7–11), because this guarantees that the target state of the best edge in the forward queue is optimally connected in the reverse tree and no edge in the forward queue with a suboptimally connected target state would be better than this edge if its target state were optimally connected (Theorem 3).

Theorem 3 (EIT* reverse search stopping condition)**.** *Let the cost-to-come heuristic be defined by the edge cost heuristic,* $\hat{g}(\mathbf{x}) \coloneqq \hat{c}(\mathbf{x}_{\text{start}}, \mathbf{x})$*, and let this heuristic be consistent and symmetric in its arguments. If the heuristic total potential solution cost of the best edge in the reverse queue is greater than or equal to the total potential solution cost of the best edge in the forward queue,*

$$\min_{\left(\mathbf{x}_{\text{s}}^{\mathcal{R}}, \mathbf{x}_{\text{t}}^{\mathcal{R}}\right) \in \mathcal{Q}_{\mathcal{R}}} \left\{ \hat{h}\left[\mathbf{x}_{\text{s}}^{\mathcal{R}}\right] + \hat{c}\left(\mathbf{x}_{\text{t}}^{\mathcal{R}}, \mathbf{x}_{\text{s}}^{\mathcal{R}}\right) + \hat{g}\left(\mathbf{x}_{\text{t}}^{\mathcal{R}}\right) \right\} \geq \min_{\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}, \mathbf{x}_{\text{t}}^{\mathcal{F}}\right) \in \mathcal{Q}_{\mathcal{F}}} \left\{ g_{\mathcal{F}}\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}\right) + \hat{c}\left(\mathbf{x}_{\text{s}}^{\mathcal{F}}, \mathbf{x}_{\text{t}}^{\mathcal{F}}\right) + \hat{h}\left[\mathbf{x}_{\text{t}}^{\mathcal{F}}\right] \right\},$$

and the target state of the best edge in the forward queue is closed, then this target has an admissible calculated cost heuristic and the best edge in the forward queue will not be changed by continuing the reverse search.

Proof. Proving the theorem proceeds similarly to the proof of Theorem 2 for AIT*. It requires showing that if the reverse search is suspended, then the target state of the best edge in the forward queue is optimally connected in the reverse tree and no edge in the forward queue with a suboptimally connected target would be better than the best edge if the target were optimally connected.

The suspension condition results in an optimally connected target state of the best edge in the forward queue because of a proven property of A* when provided with a consistent heuristic. The suspension condition explicitly requires the target of this edge to be closed in the reverse search and a closed state is optimally connected in an A* search with a consistent heuristic (Lemma 2; Hart et al., 1968).

It is now shown that no edge in the forward queue with a suboptimally connected target would be better than the best edge in the forward queue if its target were optimally connected. First, a lower bound is derived on the optimal calculated cost heuristic for any edge target in the forward queue that is suboptimally connected. This lower bound is then used to prove that the suspension condition of the theorem ensures that no edge in the forward queue with suboptimally connected target would have the lowest total potential solution cost if its target were optimally connected.

Let $(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t})$ be any edge in the forward queue with a suboptimally connected target, $\mathbf{x}_\mathrm{t}$. Let its unknown optimally connected cost in the reverse tree be $\hat{h}^* [\mathbf{x}_\mathrm{t}]$. The state $\mathbf{x}_\mathrm{t}$ cannot be closed in the reverse search because it has an inadmissible calculated cost heuristic (Lemma 2; Hart et al., 1968). Every outgoing edge, $(\mathbf{x}_\mathrm{t}, \mathbf{x}_n)$, from that state to a neighboring state, $\mathbf{x}_n$, must therefore have a potential reverse solution cost greater than or equal to the minimum potential reverse solution cost in the reverse queue,

$$\min_{\mathbf{x}_n \in \texttt{neighbors}(\mathbf{x}_\mathrm{t})} \left\{ \hat{h}^* [\mathbf{x}_\mathrm{t}] + \hat{c} (\mathbf{x}_n, \mathbf{x}_\mathrm{t}) + \hat{g} (\mathbf{x}_n) \right\} \geq \min_{(\mathbf{x}_\mathrm{s}^\mathcal{R}, \mathbf{x}_\mathrm{t}^\mathcal{R}) \in \mathcal{Q}_\mathcal{R}} \left\{ \hat{h} \left[\mathbf{x}_\mathrm{s}^\mathcal{R} \right] + \hat{c} \left(\mathbf{x}_\mathrm{t}^\mathcal{R}, \mathbf{x}_\mathrm{s}^\mathcal{R} \right) + \hat{g} \left(\mathbf{x}_\mathrm{t}^\mathcal{R} \right) \right\}, \quad (5.1)$$

because otherwise the state $\mathbf{x}_\mathrm{t}$ would be closed in the reverse search.

Let the state $\mathbf{x}_n^*$ be the neighbor of the state $\mathbf{x}_\mathrm{t}$, such that the edge $(\mathbf{x}_\mathrm{t}, \mathbf{x}_n^*)$ has the lowest potential reverse solution cost of all neighbors of $\mathbf{x}_\mathrm{t}$,

$$\mathbf{x}_n^* \coloneqq \underset{\mathbf{x}_n \in \texttt{neighbors}(\mathbf{x}_\mathrm{t})}{\arg\min} \left\{ \hat{h}^* \left[\mathbf{x}_\mathrm{t} \right] + \hat{c} \left(\mathbf{x}_n, \mathbf{x}_\mathrm{t} \right) + \hat{g} \left(\mathbf{x}_n \right) \right\}. \tag{5.2}$$

Equation 5.1 can then be rewritten as a lower bound on the optimal cost of the target state,

$$\hat{h}^* \left[\mathbf{x}_\mathrm{t} \right] \geq \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{R}, \mathbf{x}_\mathrm{t}^\mathcal{R} \right) \in \mathcal{Q}_\mathcal{R}} \left\{ \hat{h} \left[\mathbf{x}_\mathrm{s}^\mathcal{R} \right] + \hat{c} \left(\mathbf{x}_\mathrm{t}^\mathcal{R}, \mathbf{x}_\mathrm{s}^\mathcal{R} \right) + \hat{g} \left(\mathbf{x}_\mathrm{t}^\mathcal{R} \right) \right\} - \hat{c} \left(\mathbf{x}_n^*, \mathbf{x}_\mathrm{t} \right) - \hat{g} \left(\mathbf{x}_n^* \right). \tag{5.3}$$

This lower bound is now used to prove that the suspension condition of the theorem ensures that the edge $(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t})$ would not have the lowest total potential solution cost in the forward queue if its target were optimally connected.

The edge source, $\mathbf{x}_\mathrm{s}$, is a neighbor of the edge target, $\mathbf{x}_\mathrm{t}$, and therefore its cost is greater than or equal to the cost through the best neighbor, $\mathbf{x}_n^*$,

$$\hat{g} \left(\mathbf{x}_\mathrm{s} \right) + \hat{c} \left(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{s} \right) \geq \hat{g} \left(\mathbf{x}_n^* \right) + \hat{c} \left(\mathbf{x}_\mathrm{t}, \mathbf{x}_n^* \right).$$

The current cost-to-come to the source is greater than or equal to the heuristic cost-to-come to the source, $g_\mathcal{F} \left(\mathbf{x}_\mathrm{s} \right) \geq \hat{g} \left(\mathbf{x}_\mathrm{s} \right)$, because the heuristic, $\hat{g} \left(\cdot \right)$, is admissible, and therefore it is also true that

$$g_\mathcal{F} \left(\mathbf{x}_\mathrm{s} \right) + \hat{c} \left(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{s} \right) \geq \hat{g} \left(\mathbf{x}_n^* \right) + \hat{c} \left(\mathbf{x}_\mathrm{t}, \mathbf{x}_n^* \right). \tag{5.4}$$

Theorem 3 states that the reverse search is suspended if the total potential solution cost of the best edge in the reverse queue is greater than or equal to that

of the best edge in the forward queue,

$$\min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{R}, \mathbf{x}_\mathrm{t}^\mathcal{R}\right) \in \mathcal{Q}_\mathcal{R}} \left\{ \hat{h}\left[\mathbf{x}_\mathrm{s}^\mathcal{R}\right] + \hat{c}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}, \mathbf{x}_\mathrm{s}^\mathcal{R}\right) + \hat{g}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}\right) \right\} \geq \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) + \hat{h}\left[\mathbf{x}_\mathrm{t}^\mathcal{F}\right] \right\}. \quad (5.5)$$

If this condition is satisfied, then it must also be true that

$$\underbrace{g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{s}\right) - \left(\hat{g}\left(\mathbf{x}_n^*\right) + \hat{c}\left(\mathbf{x}_\mathrm{t}, \mathbf{x}_n^*\right)\right)}_{\text{nonnegative}} + \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{R}, \mathbf{x}_\mathrm{t}^\mathcal{R}\right) \in \mathcal{Q}_\mathcal{R}} \left\{ \hat{h}\left[\mathbf{x}_\mathrm{s}^\mathcal{R}\right] + \hat{c}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}, \mathbf{x}_\mathrm{s}^\mathcal{R}\right) + \hat{g}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}\right) \right\} \geq \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) + \hat{h}\left[\mathbf{x}_\mathrm{t}^\mathcal{F}\right] \right\}, \quad (5.6)$$

where the underbraced sum is nonnegative because of Equation 5.4. Using the symmetry of the edge cost heuristic, $\hat{c}\left(\mathbf{x}_\mathrm{t}, \mathbf{x}_n^*\right) \equiv \hat{c}\left(\mathbf{x}_n^*, \mathbf{x}_\mathrm{t}\right)$, and rearranging Equation 5.6 yields

$$g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}\right) + \underbrace{\min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{R}, \mathbf{x}_\mathrm{t}^\mathcal{R}\right) \in \mathcal{Q}_\mathcal{R}} \left\{ \hat{h}\left[\mathbf{x}_\mathrm{s}^\mathcal{R}\right] + \hat{c}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}, \mathbf{x}_\mathrm{s}^\mathcal{R}\right) + \hat{g}\left(\mathbf{x}_\mathrm{t}^\mathcal{R}\right) \right\} - \hat{c}\left(\mathbf{x}_n^*, \mathbf{x}_\mathrm{t}\right) - \hat{g}\left(\mathbf{x}_n^*\right)}_{\text{lower bound on } \hat{h}^*[\mathbf{x}_\mathrm{t}]} \geq \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) + \hat{h}\left[\mathbf{x}_\mathrm{t}^\mathcal{F}\right] \right\}, \quad (5.7)$$

where the underbraced term is the lower bound on the optimal connection cost, $\hat{h}^*\left[\mathbf{x}_\mathrm{t}\right]$, given in Equation 5.3. Substituting for the underbraced term yields

$$g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}\right) + \hat{h}^*\left[\mathbf{x}_\mathrm{t}\right] \geq \min_{\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) \in \mathcal{Q}_\mathcal{F}} \left\{ g_\mathcal{F}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}\right) + \hat{c}\left(\mathbf{x}_\mathrm{s}^\mathcal{F}, \mathbf{x}_\mathrm{t}^\mathcal{F}\right) + \hat{h}\left[\mathbf{x}_\mathrm{t}^\mathcal{F}\right] \right\},$$

which proves that the total potential solution cost of the edge $\left(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}\right)$ would be

greater than or equal to the minimum total potential solution cost of any edge in the forward queue even if the target state, $\mathbf{x}_t$ of the edge were optimally connected. This proves that continuing the reverse search if the suspension condition of Theorem 3 is satisfied will not change the best edge in the forward queue and is therefore unnecessary. □

5.4 Evaluation

Path planning algorithms are often only evaluated when minimizing path length. This is a common optimization objective for real-world applications, but it is also a special case that has a computationally inexpensive admissible cost heuristic (the Euclidean distance) and high correlation between the cost of a path and the computational effort required to validate it. These properties are exploited by many algorithms, including ABIT* and AIT*, and such algorithms often perform less well when optimizing objectives that lack these properties.

An example of such an optimization objective is obstacle clearance. This is an important objective for path planning because it often results in safer paths than minimizing path length and is commonly studied (Chen and Hwang, 1998; Geraerts and Overmars, 2007; Bekris and Kavraki, 2008; Wein et al., 2008; Chen et al., 2013; Agarwal et al., 2018; Kim et al., 2018; Heiden et al., 2021; Sakcak and LaValle, 2021).

The benefits of directly aligning the search of a problem with its priorities as in EIT* is shown on most of the problems presented in Sections 3.4 and 4.4 for the path length and obstacle clearance objectives. Path length was optimized by minimizing path length in the search space. Obstacle clearance was optimized by minimizing the reciprocal of clearance in work space integrated over the length of the path, l,

$$c(\sigma) \coloneqq \int_0^l \frac{1}{\delta\left(\sigma\left(s/l\right)\right)} \mathrm{d}s,$$

where $\delta\colon X \to [10^{-6}, \infty)$ is the clearance of a state (i.e., the distance to its nearest obstacle) limited to be no smaller than 10^{-6},

$$\delta(\mathbf{x}) \coloneqq \max\{\text{clearance}(\mathbf{x}), 10^{-6}\}.$$

The lower limit on the clearance ensures numerical stability and that the cost of a path is bounded by a multiple of its total variation as assumed in Section 2.4.1.2.

The admissible cost heuristics used by the informed planners were the Euclidean distance for path length and the trivial zero-heuristic for obstacle clearance. The possibly inadmissible edge cost heuristic used by EIT* was again the Euclidean distance for path length and the reciprocal of the average clearance of the two end states times their distance in search space, d, for obstacle clearance,

$$\bar{c}(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{t}) \coloneqq \frac{2d}{\delta(\mathbf{x}_\mathrm{s}) + \delta(\mathbf{x}_\mathrm{t})}.$$

The effort heuristic used by EIT* was the number of collision checks required to validate a straight-line path for both objectives. It was computed by dividing the Euclidean distance between two states by the collision detection resolution.

The inflation factor update policy in EIT* was configured to have an infinite inflation factor until the initial solution is found and then switch to a unity inflation factor. This results in fast initial solutions and efficient subsequent searches to improve them.

The sparse collision detection resolution update policy in EIT* was configured to initially search each batch with a single collision check per edge. Whenever the forward search detected a collision on an edge used in the reverse search tree, then the collision detection resolution was doubled, unless doubling it would make it denser than the full collision detection resolution specified by the problem, in which case it was set to that resolution. These policies could be tuned to each problem separately, but were kept constant in this thesis to show that EIT* can perform well without tuning them.

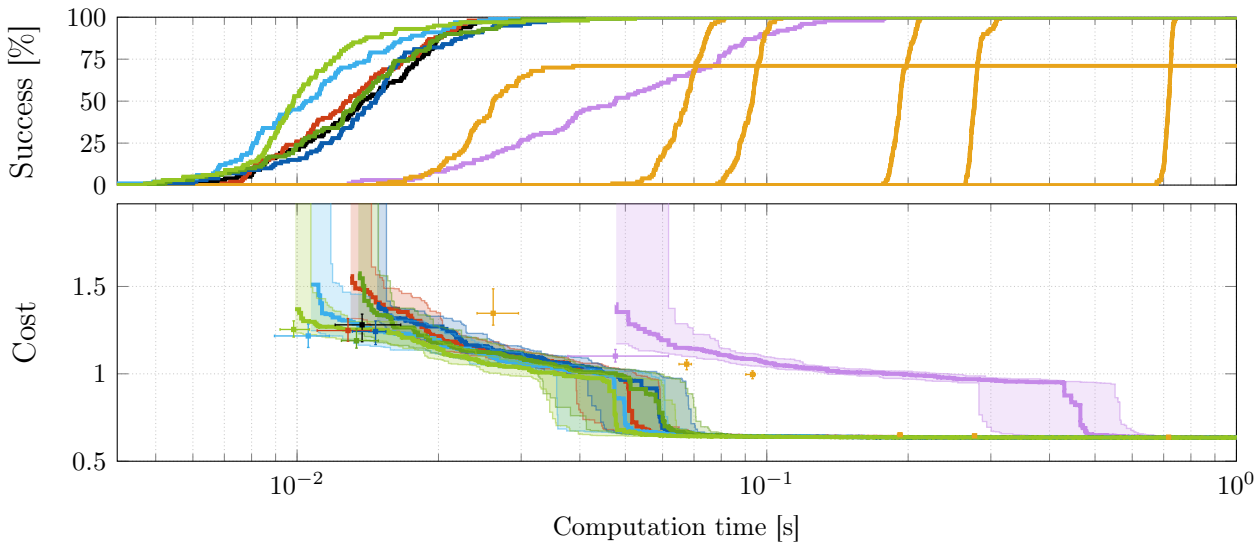
5.4.1 Abstract problems

EIT* was tested against the other planners on the same abstract problems in $\mathbb{R}^2$, $\mathbb{R}^8$, and $\mathbb{R}^{16}$ as ABIT* (Section 3.4.1, Figure 3.3), but this time when minimizing path length and optimizing obstacle clearance. While admissible cost heuristics exist for these abstract problems with clearance in search space (Strub and Gammell, 2021a), such heuristics often do not exist in real-world problems optimizing clearance in work space. The trivial admissible cost heuristic was therefore also used for the clearance objective in these abstract problems.

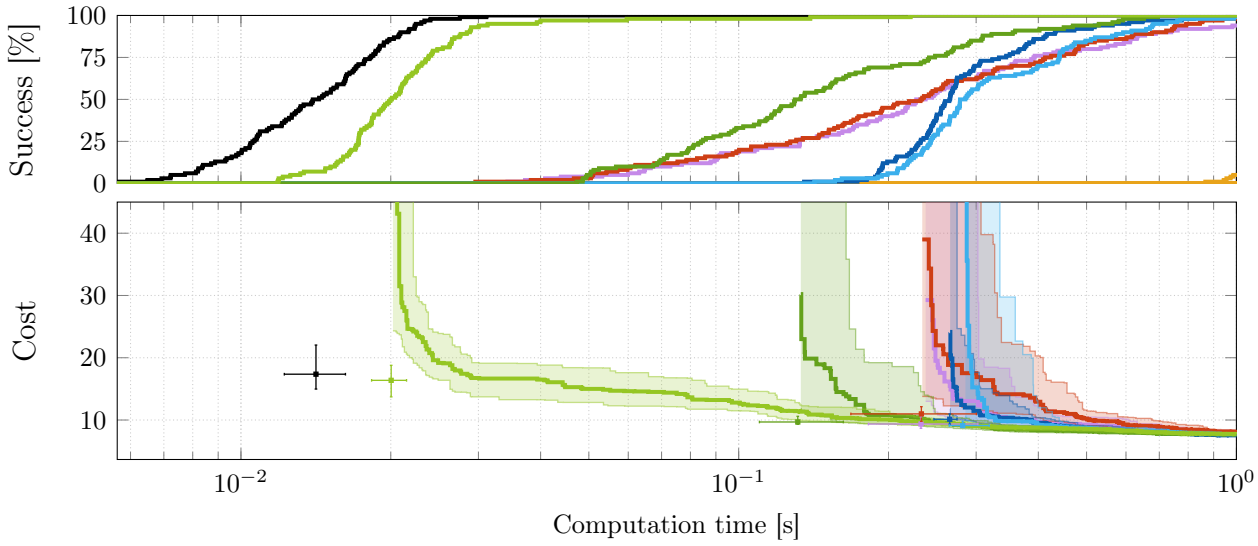
When minimizing path length, this obstacle configuration shows how fast EIT* can find the hard-to-find optimal homotopy class of the problem. When optimizing obstacle clearance, it shows the benefits of considering computational effort in addition to solution cost by decreasing initial solution times by up to an order of magnitude.

Figures 5.2, 5.3, and 5.4 show the performances of all planners on all versions of the problem for both objectives. When minimizing path length, EIT* is the only tested planner that is among the best-performing planners on all versions in terms of median initial solution times as well as convergence and success rates. When optimizing obstacle clearance, EIT* finds initial solutions and reaches 100% success up to an order of magnitude faster than the second best asymptotically optimal planner and is among the planners with the best convergence rates.

The only planner that finds initial solutions as fast as EIT* when optimizing obstacle clearance is RRT-Connect, which is not an anytime planner and has an inherent computational advantage because it does not evaluate solution cost. This advantage is amplified for this optimization objective, because evaluating obstacle clearance is at least as computationally expensive as collision detection.

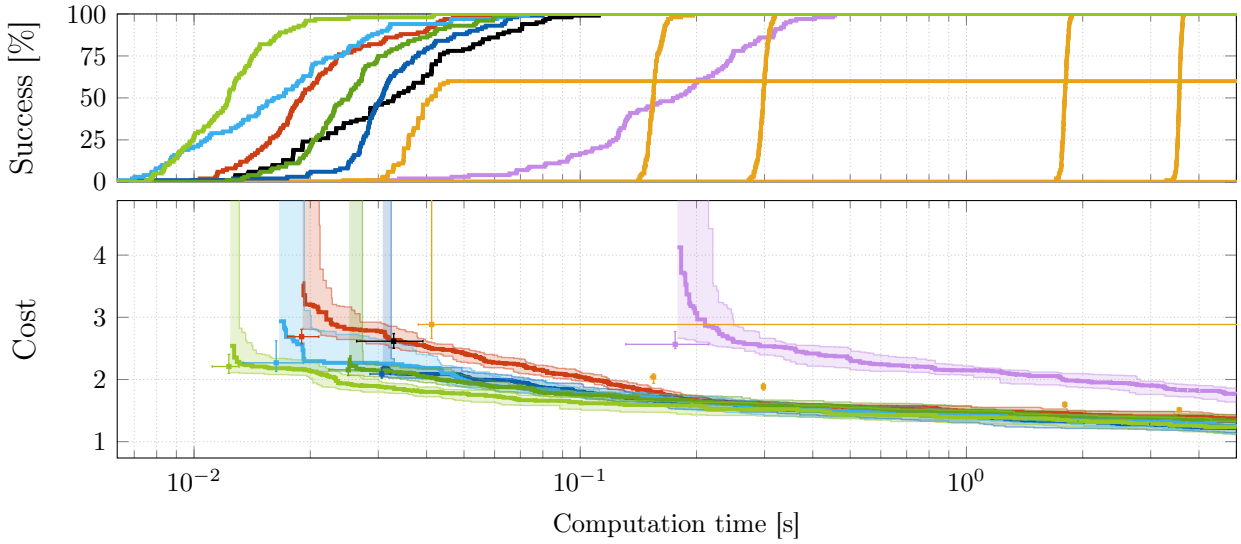


(a) Minimizing path length

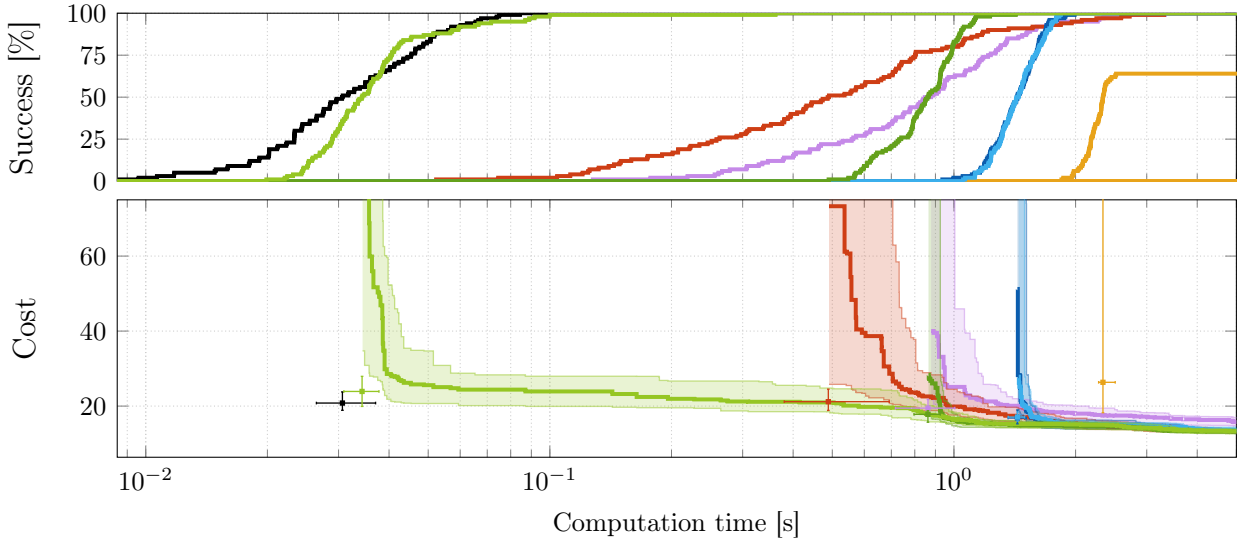


(b) Optimizing obstacle clearance

Figure 5.2: The planner performances on the wall gap experiment in $\mathbb{R}^2$ described in Section 5.4.1 (Figure 3.3). The results show that EIT* performs as well as the best performing other planner (ABIT*) in terms of success and convergence rates and median initial solution time and cost when minimizing path length (a). EIT* outperforms all other tested asymptotically optimal planners in terms of success and convergence rates and median initial solution times when optimizing obstacle clearance (b). The only tested planner that finds initial solutions faster than EIT* is RRT-Connect, which is not an asymptotically optimal planner and has an inherent computational advantage because it does not evaluate solution cost.

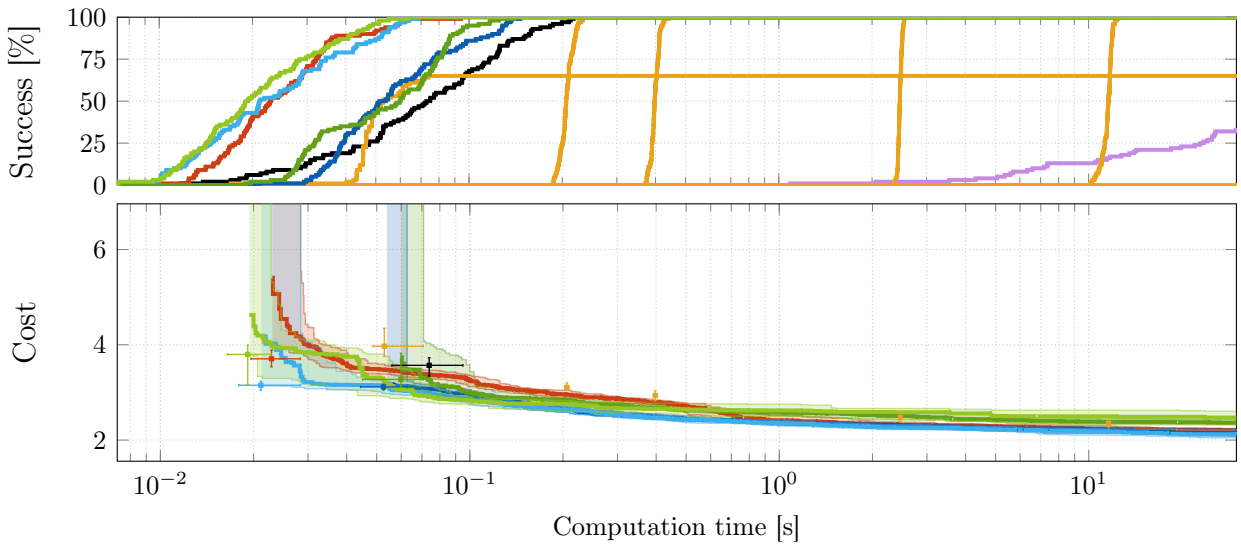


(a) Minimizing path length

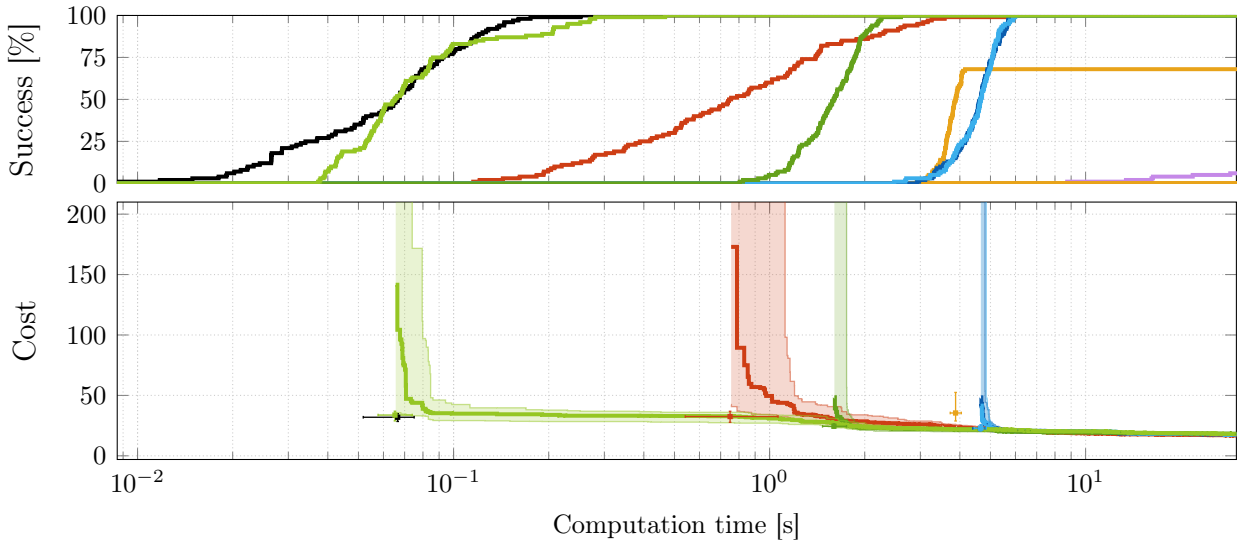


(b) Optimizing obstacle clearance

Figure 5.3: The planner performances on the wall gap experiment in $\mathbb{R}^8$ described in Section 5.4.1. The results show that EIT* outperforms all other tested asymptotically optimal planners for both objectives in terms of success rates, median initial solution times, and median solution-quality over time (a, b). The only tested planner that finds initial solutions as fast as EIT* when optimizing obstacle clearance is RRT-Connect, which is not an asymptotically optimal planner and has an inherent computational advantage because it does not evaluate solution cost.



(a) Minimizing path length



(b) Optimizing obstacle clearance

Figure 5.4: The planner performances on the wall gap experiment in $\mathbb{R}^{16}$ described in Section 5.4.1. The results show that EIT* is among the best-performing planners in terms of initial solution times and success rates when minimizing path length (a). EIT* outperforms all other tested asymptotically optimal planners in terms of success rates, median initial solution times, and median solution quality over time when optimizing obstacle clearance (b). The only planner that finds initial solutions as fast as EIT* when optimizing obstacle clearance is RRT-Connect, which is not an asymptotically optimal planner and has an inherent computational advantage because it does not evaluate solution cost.

RRT-C.	Inf. RRT*	FMT*	Lazy PRM*	BIT*	ABIT*	AIT*	EIT*
198 (99%)	179 (89.5%)	186 (93%)	180 (90%)	188 (94%)	186 (93%)	164 (82%)	190 (95%)

Table 5.1: The numbers (and percentages) of the 200 Reeds-Shepp car problem instances on which the tested planners achieved a success rate of at least 50%. EIT* is the most reliable asymptotically optimal planner. The only tested planner that solved significantly more instances is RRT-Connect, which is not an anytime algorithm.

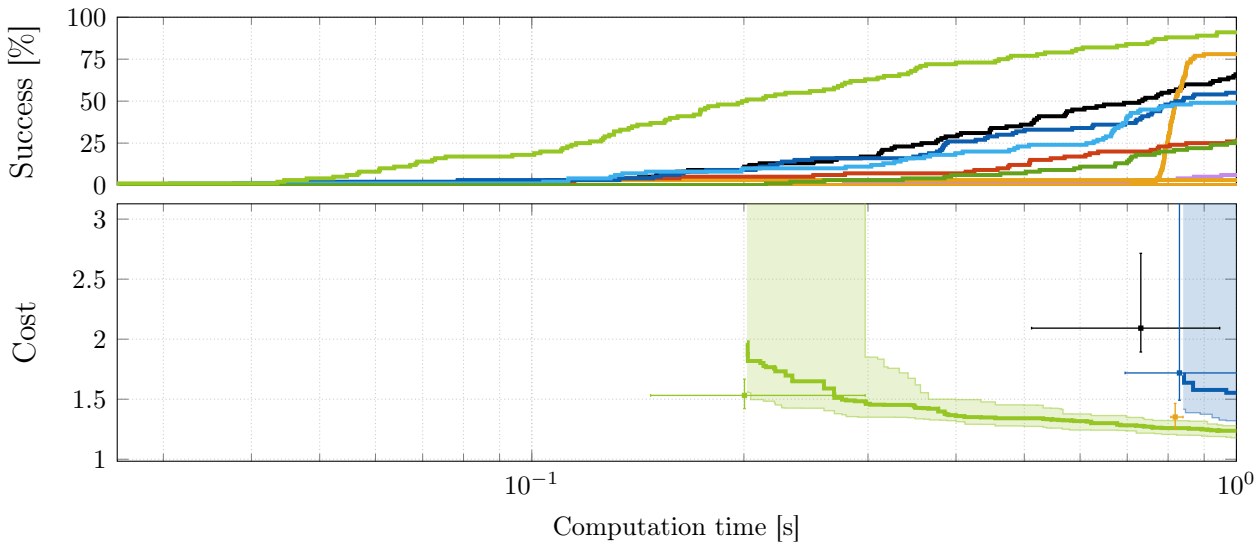
5.4.2 Reeds-Shepp car problems

EIT* was tested on the same 200 Reeds-Shepp car problem instances as ABIT* and AIT* when minimizing path length. These problems show again that EIT* can outperform all other tested planners on some problem instances, but is outperformed by some of the same planners on other problem instances.

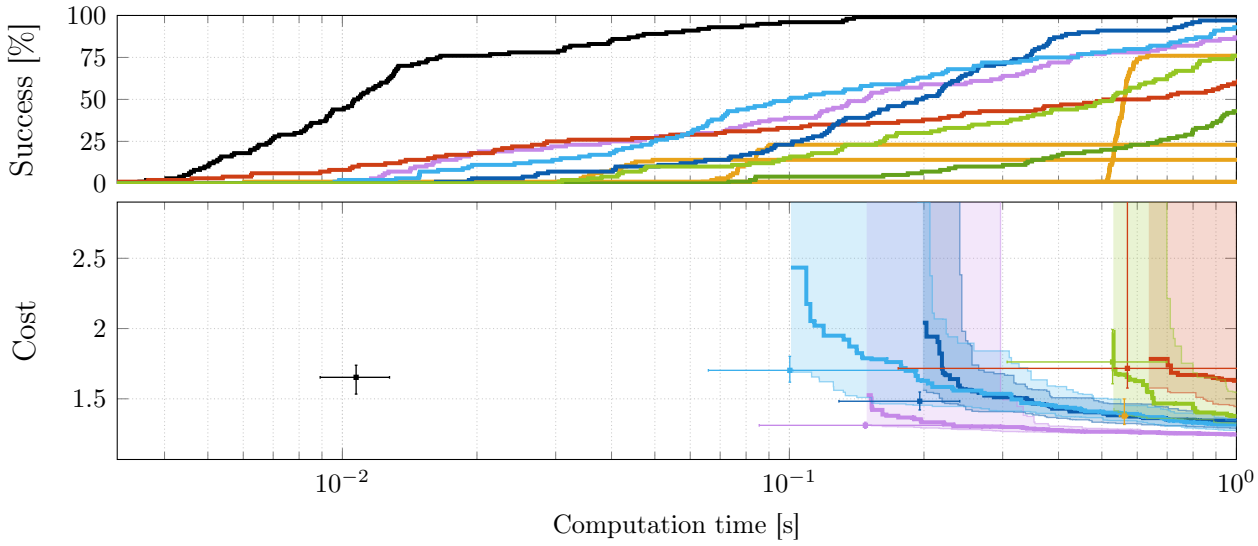
The best and worst performances of EIT* were again selected in terms of its median initial solution time relative to the fastest median initial solution time of the other tested planners, limited to the problem instances where EIT* achieved at least 50% success rate (Table 5.1). Figure 5.5 shows the results on the problem instances that resulted in the best and worst performances of EIT*.

In the best case, EIT* outperforms all other tested planners in terms of median initial solution times and success and convergence rates. In this instance, EIT* finds initial solutions significantly faster than even RRT-Connect, which is not an anytime algorithm and has an inherent computational advantage because it does not evaluate solution cost (Figure 5.5a).

In the worst case, EIT* has a slower median initial solution time than all other tested planners, except Lazy PRM* and AIT*. RRT-Connect finds initial solutions almost two orders of magnitude faster than EIT* in this instance, but is still not an anytime algorithm and cannot improve its solution quality in the available computation time (Figure 5.5b).

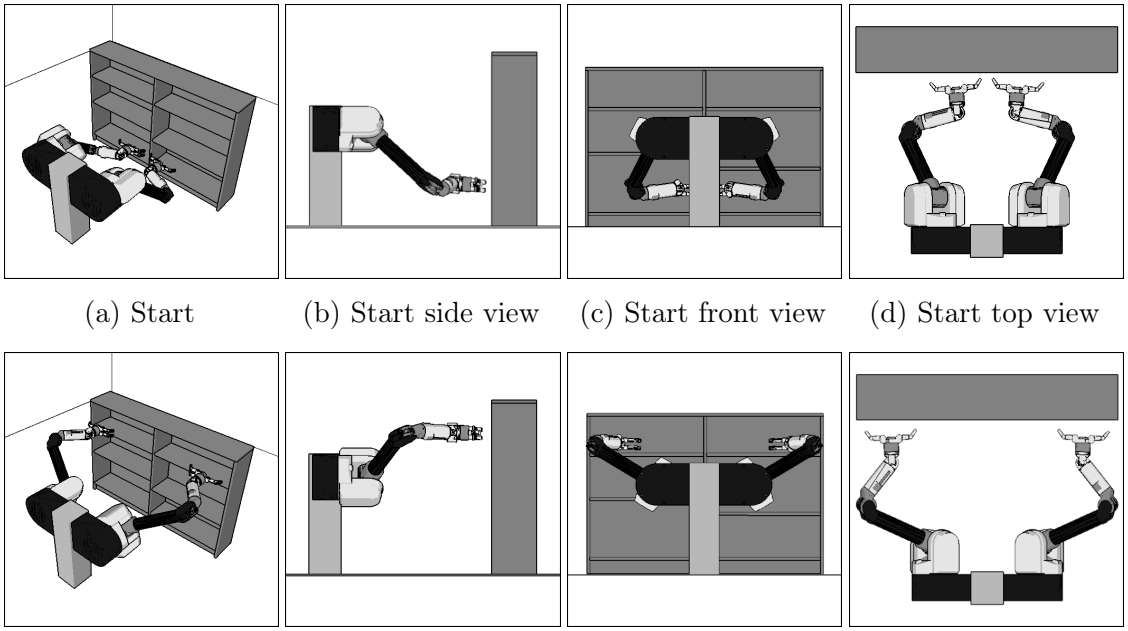


(a) Best relative performance of EIT*



(b) Worst relative performance of EIT*

Figure 5.5: The planner performances on the Reeds-Shepp car problem that resulted in the best (a) and worst (b) performances of EIT* relative to the other tested planners (Section 5.4.2). The results show that EIT* outperforms all other tested planners on some instances (a) but is outperformed by some of the same planners on others (b). On the best instance for EIT*, it outperforms the other tested planners in terms of median initial solution times, as well as success and convergence rate (a). On the worst instance for EIT*, it performs as well as Lazy PRM* and better than AIT* but is outperformed by all other tested planners in terms of median initial solution times and success and convergence rates (b).



(a) Start (b) Start side view (c) Start front view (d) Start top view

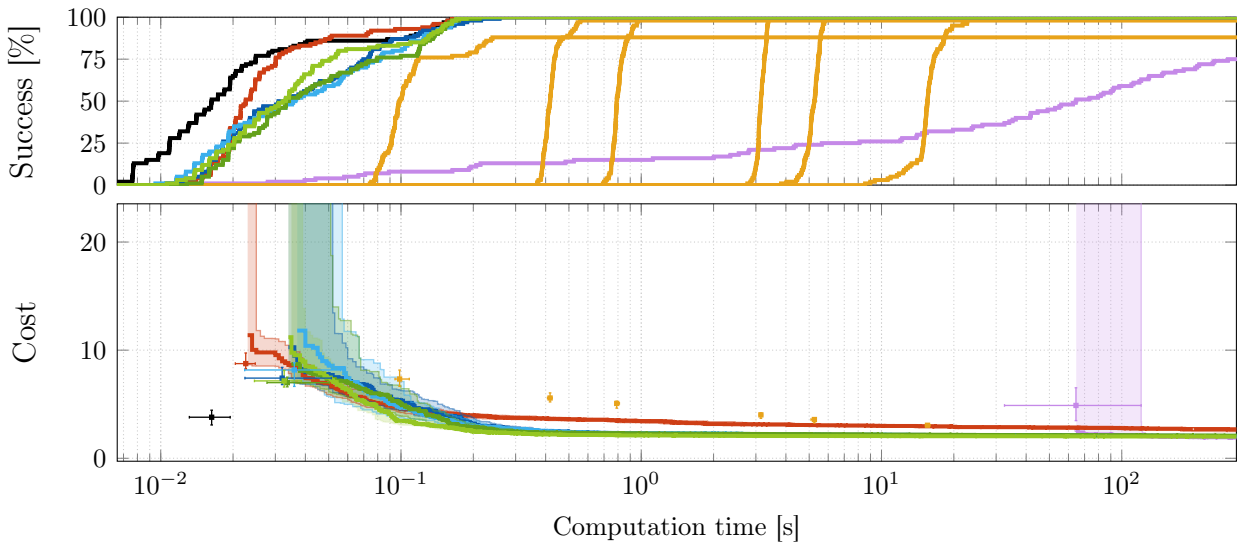
(e) Goal (f) Goal side view (g) Goal front view (h) Goal top view

Figure 5.6: Illustrations of the manipulator arm problem (Section 5.4.3). The top row shows the start configuration of the arms in position to pick up two objects on the bottom shelf (a–d). The bottom row shows the goal configuration of the arms in position to place these objects on the top shelf (e–h).

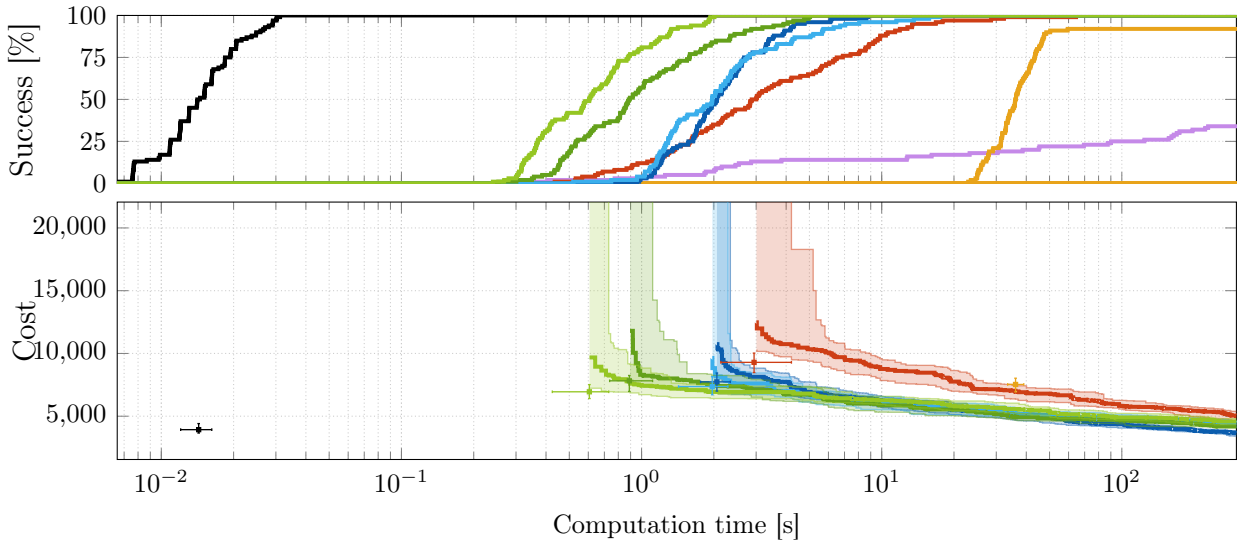
5.4.3 Manipulator arm problems

EIT* was also tested on the single-arm manipulator problem of Section 4.4.2 (Figure 4.3), but this time when minimizing path length and obstacle clearance. EIT* was additionally tested on a dual-arm manipulator problem with the same objectives. The dual-arm manipulator problem required the planners to find a path for two Barrett WAMs with a total of 14 degrees of freedom and had a time limit of 300 seconds per attempt. This problem simulates the movement of the two arms to pick up two objects on the bottom shelf and place them on the top shelf (Figure 5.6). The problems were simulated in OpenRAVE, which was configured to use OBB trees for collision detection as in Section 4.4.2 and additionally used Rectangle Swept Sphere (RSS; Larsen et al., 2000) volumes for efficient clearance computation.

Obstacle clearance is a common optimization objective for manipulator problems, as it often result in safer paths than path length. These problems demonstrate

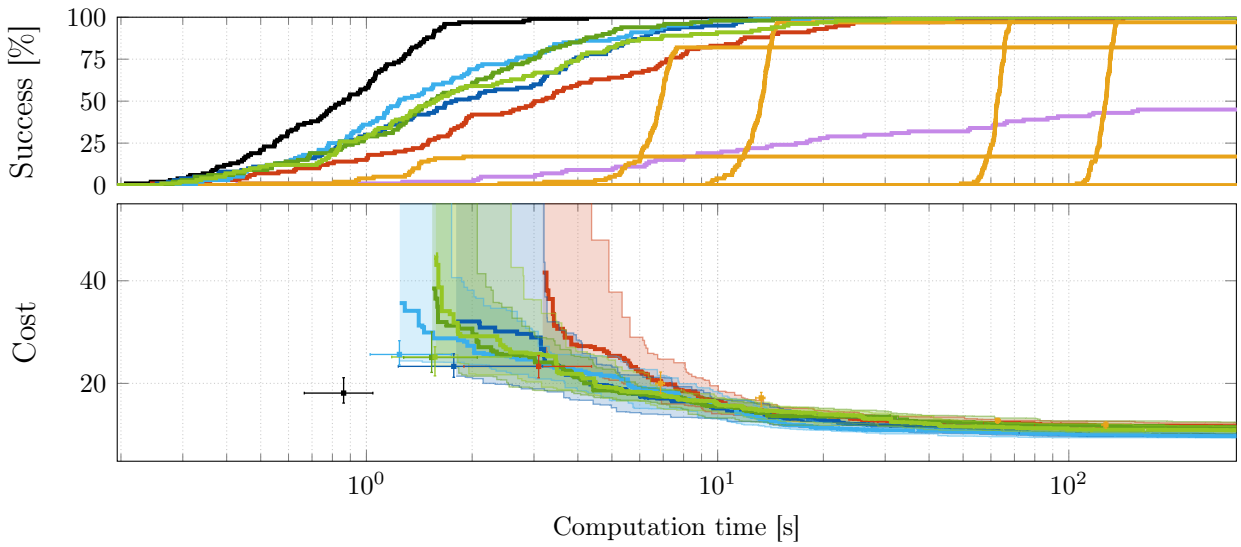


(a) Minimizing path length

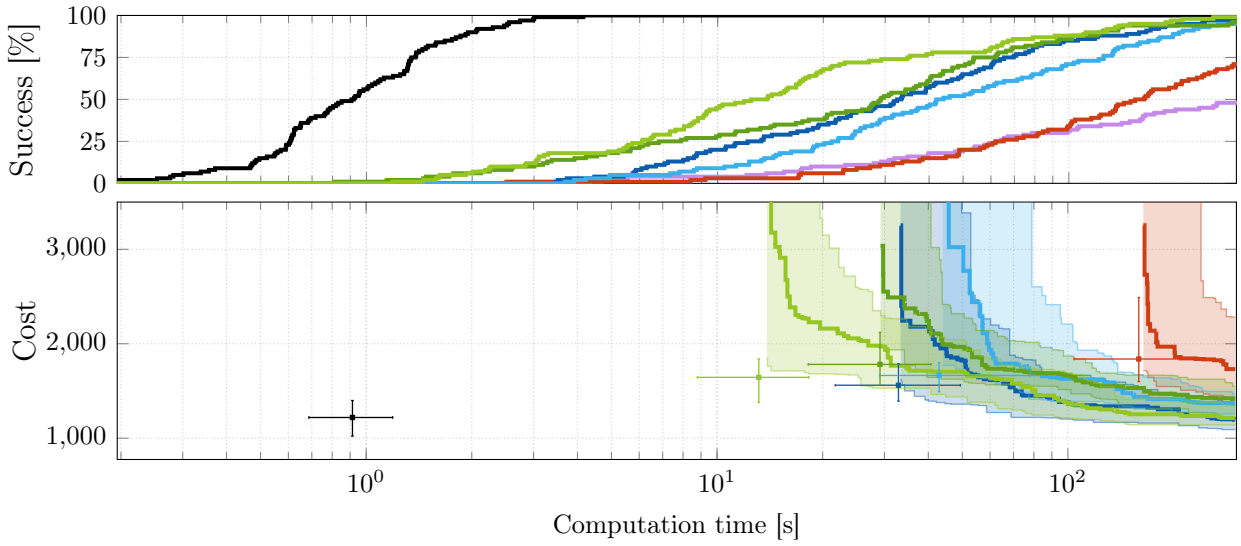


(b) Optimizing obstacle clearance

Figure 5.7: The planner performances on the single-arm manipulator problem described in Section 4.4.2 (Figure 4.3). The results show that when minimizing path length, EIT* is competitive in terms of success and convergence rates but does not find initial solutions as fast as Lazy PRM*, which converges more slowly than EIT* and other planners (a). When optimizing obstacle clearance, EIT* outperforms all other tested asymptotically optimal planners in terms of success rate and initial solutions, but converges slower than BIT* and ABIT* (b). The only planner that finds initial solutions faster than EIT* when optimizing obstacle clearance is again RRT-Connect, which is not an asymptotically optimal planner and has an inherent computational advantage because it does not evaluate solution cost.



(a) Minimizing path length



(b) Optimizing obstacle clearance

Figure 5.8: The planner performances on the dual-arm manipulator problem described in Section 5.4.3 (Figure 5.6). The results show that when minimizing path length, EIT* performs as well as the best performing other asymptotically optimal planners in terms of success and convergence rates and median initial solution time and cost (a). When optimizing obstacle clearance, EIT* outperforms all other tested asymptotically optimal planners in terms of success and convergence rates and median initial solution times (b). The only planner that finds initial solutions faster than EIT* is again RRT-Connect, which is not an asymptotically optimal planner and has an inherent computational advantage because it does not evaluate solution cost.

the benefits of EIT* on high-dimensional manipulator arm problems with realistic collision detection and optimization objectives.

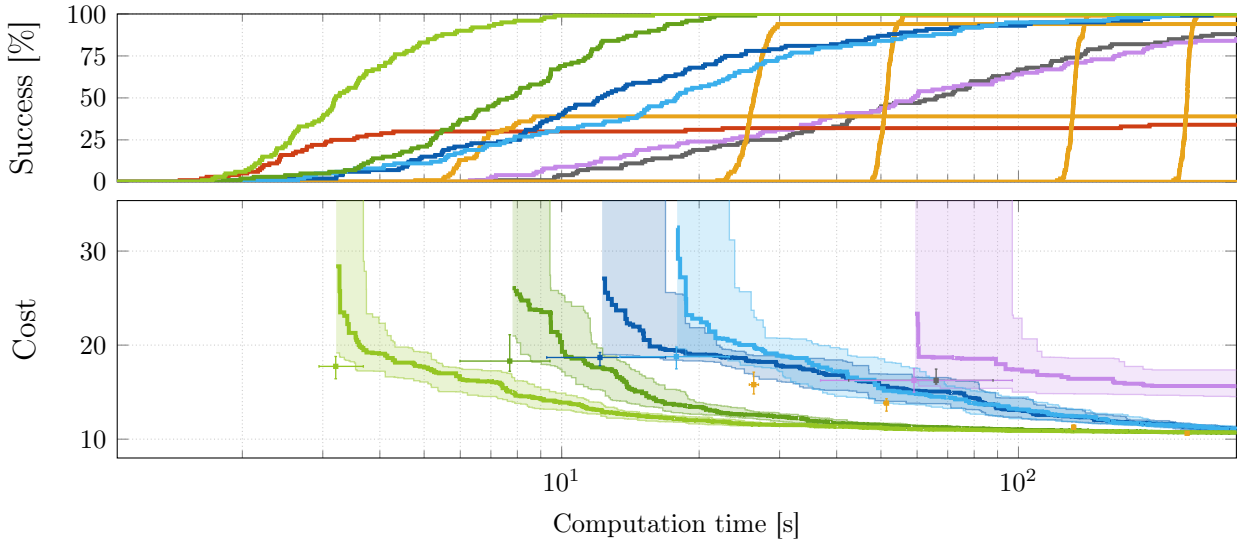
When minimizing path length on the single-arm manipulator problem, EIT* performs equally well as BIT*, ABIT*, and AIT*. When optimizing obstacle clearance on the single-arm problem, EIT* outperforms all other tested asymptotically optimal planners by having the fastest median initial solution time, the best success rate, and a competitive convergence rate (Figure 5.7).

EIT* performs similarly on the dual-arm manipulator problem. When minimizing path length, EIT* is among the best performing asymptotically optimal planners and only RRT-Connect has a faster median initial solution time. When optimizing obstacle clearance, EIT* outperforms all other tested asymptotically optimal planners with the fastest median initial solution time and competitive success and convergence rates (Figure 5.8).

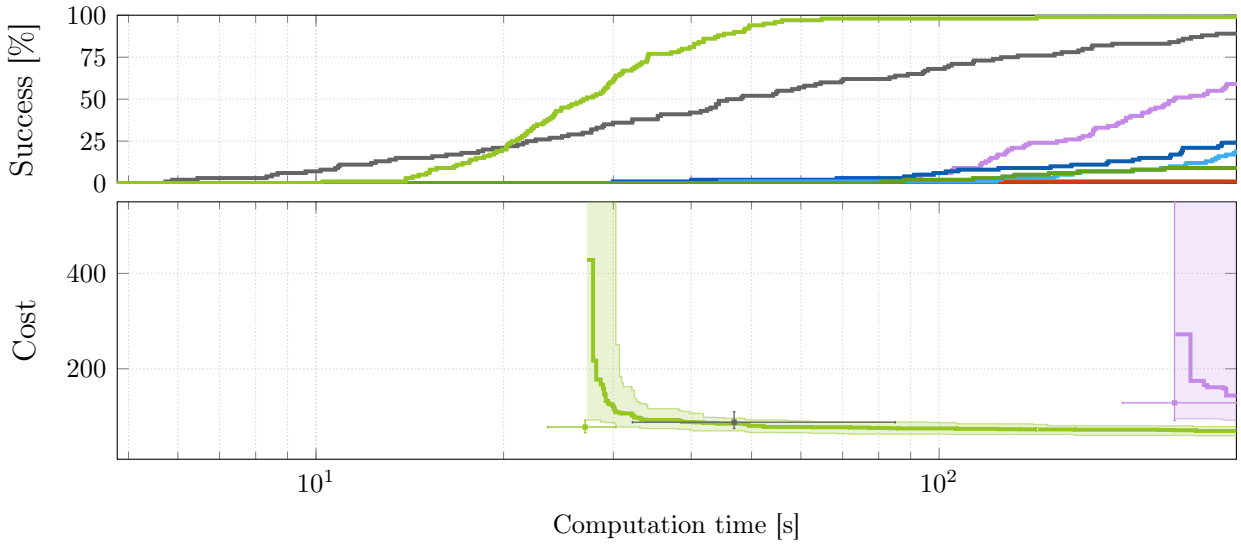
The only tested planner that consistently finds initial solutions faster than EIT* on these manipulator arm problems is RRT-Connect. But RRT-Connect is not an anytime planner and has an inherent computational advantage because it does not evaluate solution cost.

5.4.4 Knee replacement dislocation problem

EIT* was also tested on the knee replacement dislocation problem described in Section 4.4.3 (Figure 4.5), but this time again when minimizing path length and optimizing obstacle clearance. The problem is viewed as a feasible planning problem by Yang et al. (2020), which makes the median initial solution time the most important performance statistic. Nevertheless, the solution quality when optimizing obstacle clearance could potentially be used to assess the probability of dislocation, as paths with greater obstacle clearance indicate a larger gap for the mobile bearing.



(a) Minimizing path length



(b) Optimizing obstacle clearance

Figure 5.9: The planner performances on the knee replacement dislocation problem described in Section 5.4.4 (Figure 4.5). The results show that on this problem EIT* outperforms the other tested planners when optimizing both objectives by finding initial solutions and reaching 100% success rate faster and returning the best-quality median solution at any time during the experiment. When optimizing obstacle clearance, only EIT* reaches a 100% success rate within the time limit. Similarly to RRT-Connect, RRT has a computational advantage because it does not evaluate solution cost.

Figure 5.9 shows the performances of all planners on the knee replacement dislocation problem when minimizing path length and optimizing obstacle clearance. The results show that EIT* outperforms all other tested planners on this problem for both objectives. When minimizing path length, EIT* finds initial solutions and reaches a 100% success faster than any other tested planner. EIT* also returns the best-quality solution at any time during the experiment. When optimizing obstacle clearance, EIT* also outperforms all other tested planners in terms of median initial solution times and success and convergence rates. EIT* is the only tested planner that reaches a 100% success rate and outperforms the other tested planners in terms of success rates, median initial solution times, and median solution cost over time.

5.5 Discussion

Planning problems often have specific priorities, e.g., preferring fast low-quality solutions over slow high-quality solutions. Many planners address such priorities by modifying their searches with indirect mechanisms that rely on implicit assumptions. For example, ABIT* seeks to accelerate initial solution times by inflating a cost heuristic, which assumes that the quality of a solution correlates with the computational effort required to find it.

Computational effort is platform-specific and difficult to quantify, but the number of collision checks is a good proxy when fast initial solutions are prioritized (Hauser, 2015; Kleinbort et al., 2016, 2020). The number of collision checks required to find a solution depends on its length, which is unknown *a priori* but can be estimated with the Euclidean distance between its start and goal states. This provides a method to express intent-specific information about the computational effort required to find a solution as an effort heuristic that estimates the number of collision checks.

EIT* leverages such intent-specific information in addition to optimization- and environment-specific information to directly align its search with the priorities of

the given problem. It achieves this with a hierarchical bidirectional search that is asymmetric in both purpose and computational cost (anecdotally, processing an edge in the reverse search on average only took 1% as long as processing an edge in the forward search on a representative example with abstract obstacles). The inexpensive reverse search with sparse collision detection calculates accurate cost and effort heuristics for the current approximation by combining cost and effort heuristics between multiple samples while considering the potential connections of the current RGG approximation. The expensive forward search directly aligns itself with the priorities of the given problem by leveraging these accurate heuristics and informs the reverse search if it considered invalid RGG edges when calculating them.

EIT* used an effort heuristic based on the number of collision checks required to validate a potential solution in Section 5.4. This is justified because solution time is initially prioritized over solution cost and collision detection is one of the computational bottlenecks for the problems presented in this thesis. For problems that require solving a BVP for each edge evaluation, it may be worth considering the number of edges in a potential solution instead of (or in addition to) the number of collision checks because solutions that consist of a few long edges require solving fewer BVPs than solutions that consist of many short edges.

For applications that optimize the total energy required to solve a planning problem and execute the resulting path (e.g., Sudhakar et al., 2020), it may be worth considering a heuristic that estimates the required number of energy intensive search operations instead of the number of computationally expensive search operations. Such a heuristic could be used in EIT* as intent-specific information to guide the search towards solutions that are energy efficient to find and execute.

As in AIT*, the performance of EIT* relative to other planners depends on the proportion of the computational cost incurred by the reverse search when calculating the heuristics compared to the computational savings gained by the forward search when leveraging these heuristics. EIT* therefore also performs best relative to other planners when full edge evaluation is computationally expensive.

The performance of EIT* depends less than AIT* on the size of the obstacles relative to the extent of the search space due to the adaptive sparse collision detection in the reverse search of EIT*. These sparse checks allow EIT* to discover smaller obstacles than AIT* without improving its RGG approximation. This results in even more accurate heuristics without incurring much additional computational cost.

EIT* performs especially well relative to other planners when the cost of a solution does not correlate well with the computational effort required to find it. Many other planners either do not pursue anytime performance or implicitly assume that high-quality solutions are fast to find. EIT* instead explicitly distinguishes between the solution quality and the computational effort required to find it and leverages its calculated effort heuristic to prioritize paths that result in fast solutions.

AIT* and EIT* use different algorithms for the reverse and forward searches of their asymmetric bidirectional searches. The change in forward search algorithms from A* in AIT* to AEES in EIT* is motivated by the benefits of intent-specific information in the form of an effort heuristic, which cannot be leveraged with A*. The change in reverse search algorithms from LPA* in AIT* to A* in EIT* is motivated by the observations that repairing the reverse search tree with LPA* is only more efficient than restarting A* for small changes in the search tree (between 1% and 2%; Table 1, Aine and Likhachev, 2016), and that detecting invalid edges with the forward search often results in larger changes in the reverse search tree.

In summary, this chapter presented the following core contributions:

- A review of other sampling-based planners and graph-search algorithms that leverage intent-specific information to align their searches with the priorities of a given problem and a discussion of their differences to EIT* (Section 5.1).
- A detailed description of EIT*, which improves on AIT* by leveraging optimization-, environment-, and intent-specific information to directly reflect problem priorities in its search (Section 5.2).

- A proof of the almost-sure asymptotic optimality of EIT* that combines established results from the literature with a new result on its reverse search (Section 5.3).
- Empirical demonstrations of the benefits of EIT* on abstract, nonholonomic, manipulator, and biomedical problems for two common optimization objectives (Section 5.4).
- A discussion of the empirical results and how the measures introduced in EIT* address some of the shortcomings of AIT* (Section 5.5).

Chapter 6

Conclusion

Summary, results overview, current and future applications

The path planning problem is about finding paths through continuous spaces. It is actively studied (Gammell and Strub, 2021) and appears in many applications in robotics and beyond (Paton et al., 2020; Gonçalves et al., 2021; Yang et al., 2020). Many recent applications are increasingly complex and require improved performance. This simultaneous increase in problem complexity and performance requirements calls for new approaches to the path planning problem.

Path planning problems can be challenging for various reasons. They can be high dimensional, have complicated system constraints, contain adverse obstacle configurations, or be computationally expensive due to complex cost evaluation or collision detection. Some of the most interesting applications combine multiple of these difficulties. For example, NASA/JPL-Caltech's Axel rover system compounds complicated tether constraints with computationally expensive collision detection (Section 3.5). This makes many path planning problems that stem from real-world applications too difficult to solve in an exact manner.

Two popular approaches to solve such problems approximately are graph-based searches and sampling-based planners. These approaches have complementary strengths, but unifying them is difficult. This thesis builds on recent efforts that

successfully incorporates graph-search techniques in sampling-based planning by viewing the sampled states as vertices of an RGG that is embedded in the (continuous) search space of a planning problem. This perspective allows the sampling-based planning algorithms presented in this thesis to build on decades of research from the graph-search literature.

One of the most compelling strengths of graph-search algorithms is that they offer a formal basis to leverage problem-specific information. This information is typically not only specific to a problem but also to a particular component of a problem. Three components of path planning that can often be described with additional information are the cost function (optimization-specific information), the obstacle configuration (environment-specific information), and the set of priorities (intent-specific information). This thesis demonstrates how these three sources of information can be leveraged in a complementary manner in sampling-based planning and presents algorithms that outperform existing planners on many path planning problems from diverse domains.

ABIT* leverages optimization-specific information in all aspects of planning and to a greater extent than previous sampling-based algorithms. It approximates the search space by sampling batches of states and it views these samples as vertices in a series of increasingly dense, edge-implicit RGGs. This sampling-based approximation is focused to the relevant region of the search space with informed sampling (Gammell et al., 2018), which uses optimization-specific information in the form of admissible cost heuristics. This sampling-based approximation is searched with advanced graph-search techniques, such as inflation and truncation, which again leverages optimization-specific information in the form of an admissible cost heuristic.

ABIT* was demonstrated to work on Axel, a next-generation NASA/JPL-Caltech rover that is designed to explore challenging terrain on the Moon and Mars. ABIT* successfully planned paths for Axel during a week-long field trial in the Mojave Desert, California, USA (Figure 6.1a; Paton et al., 2020). ABIT* was

(a) (b) (c)

Figure 6.1: ABIT* and AIT* have been integrated on three real-world systems by researchers at NASA/JPL and ORI. ABIT* has planned paths for two next-generation rover prototypes, Axel (a; Paton et al., 2020) and RoboSimian (a; Reid et al., 2020). AIT* has been integrated in a system that automatically creates high-fidelity models of real-world objects (c; Border and Gammell, 2021). Photographs courtesy of NASA/JPL (a, b) and ORI (c).

independently integrated on RoboSimian, a next-generation NASA/JPL-Caltech rover that is designed to explore the rugged surface of Europa (a moon of Jupiter) and successfully planned paths for RoboSimian on salt-evaporite fields in the Death Valley, California, USA (Figure 6.1b; Reid et al., 2020).

A reference implementation of ABIT* is publicly available in OMPL (Şucan et al., 2012). The work on ABIT* and its extension to non-Markovian systems presented in this thesis was first published in the Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA) and the Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). The associated publications are:

> Strub, M. P. and Gammell, J. D. (2020b). Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–136
>
> Paton, M., Strub, M. P., Brown, T., Greene, R. J., Lizewski, J., Patel, V., Gammell, J. D., and Nesnas, I. A. (2020). Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7034–7041

ABIT* gains environment-specific information when approximating the search space with *valid* samples and checking edges for collision, but fails to use this information during the search and views its sampling-based approximation as an abstract graph. AIT* instead leverages such environment-specific information in all aspects of planning, including its search. As in ABIT*, AIT* samples batches of valid states and views these samples as vertices in a series of increasingly dense, edge-implicit RGGs. It leverages the environment-specific information inherent in the observed distribution of its samples with a hierarchical bidirectional search which is asymmetric in purpose and computational cost and in which both searches continuously inform each other with complementary information.

AIT* is currently being used by colleagues at ORI as part of a system centered around the Surface Edge Explorer (SEE; Border et al., 2018; Border and Gammell, 2020) that automatically creates high-fidelity models of real-world objects (Figure 6.1c; Border and Gammell, 2021). In this application, AIT* plans paths for a manipulator arm and a rotary stage with a total of seven degrees of freedom. AIT* is also being investigated by a colleague at Cornell University for improved performance on tightly coupled task and motion planning problems.

A reference implementation of AIT* is publicly available in OMPL. The work on AIT* presented in this thesis was first published in the Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA) and an extended version of this paper has been accepted for publication in the International Journal of Robotics Research (IJRR). The associated publications are:

> Strub, M. P. and Gammell, J. D. (2020a). Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198
>
> Strub, M. P. and Gammell, J. D. (2021b). AIT* and EIT*: Asymmetric bidirectional sampling-based path planning. *The International Journal of Robotics Research (IJRR)*. To appear, Manuscript #IJR-21-4179

		ABIT*	AIT*	EIT*
Information		Optimization-specific	Optimization-specific & environment-specific	Optimization-specific, environment-specific & intent-specific
Approximation		Series of RGGs	Series of RGGs	Series of RGGs
Reverse search	Purpose	—	Calculate cost	Calculate cost & effort
	Algorithm	—	Vertex-queue LPA*	Edge-queue A*
	Ordering	—	*A priori* cost	*A priori* cost & effort
	Validation	—	None	Sparse collision checks
Forward search	Purpose	Find valid paths	Find valid paths	Find valid paths
	Algorithm	Edge-queue ATD*	Edge-queue A*	Edge-queue AEES
	Ordering	*A priori* cost	Calculated cost	Calculated cost & effort
	Validation	Dense collision checks	Dense collision checks	Dense collision checks

Table 6.1: An overview of the information and algorithm components used by ABIT*, AIT*, and EIT*. All three algorithms approximate the search space with the sames series of increasingly dense RGGs. ABIT* leverages optimization-specific information. It does not have a reverse search and finds valid paths with an edge-queue version of ATD* ordered by the (inflated) total potential solution cost according to an *a priori* admissible cost heuristic. AIT* leverages optimization- and environment-specific information. It calculates an admissible cost heuristic with a reverse LPA* search ordered by the total potential solution cost according to an *a priori* admissible cost heuristic. It finds valid paths with an edge-queue version of A* ordered by the total potential solution cost according to the cost heuristic calculated by its reverse search. EIT* leverages optimization-, environment-, and intent-specific information. It calculates admissible and inadmissible cost and effort heuristics with a reverse A* search ordered by the total potential solution cost and effort according to *a priori* cost and effort heuristics. It finds valid paths with an edge-queue version of AEES ordered by the total potential solution cost and effort according to the heuristics calculated by its reverse search.

AIT* leverages optimization- and environment-specific information but does not directly reflect problem-specific priorities, such as a desire for fast initial solution times, in its search order. Similar to ABIT*, AIT* seeks to accelerate planning with an indirect mechanism that implicitly assumes that the quality of a solution correlates with the computational effort required to find it. EIT* instead directly aligns its search with the priorities of a problem by leveraging intent-specific information. It achieves this similarly as AIT* with a hierarchical bidirectional search, but simultaneously calculates and leverages cost and effort heuristic to guide the search by its ultimate purpose (Table 6.1).

EIT* has not yet been demonstrated on a real-world system, but is currently being investigated by colleagues at Cornell University and TU Berlin for improved multiquery path planning and task and motion planning, respectively.

A reference implementation of EIT* is publicly available in OMPL. The work on EIT* presented in this thesis has been accepted for publication in the International Journal of Robotics Research (IJRR). The associated publication is:

> Strub, M. P. and Gammell, J. D. (2021b). AIT* and EIT*: Asymmetric bidirectional sampling-based path planning. *The International Journal of Robotics Research (IJRR)*. To appear, Manuscript #IJR-21-4179

This thesis identifies and investigates multiple sources of information that can be leveraged to solve continuous planning problems. This is achieved by separating the approximation of the search space from the search of this approximation. The algorithms presented in this thesis approximate planning problems with ever more refined approximations based on batches of samples, similar to BIT*. These sampling-based approximations are searched with advanced graph search techniques, similar to ATD*, LPA*, and AEES, either unidirectionally or with a hierarchical bidirectional search that it asymmetric in purpose and computational cost. This strengthens the conceptual connection between sampling-based planning and graph-based search and improves practical performance on many of the tested problems.

The benefits of ABIT*, AIT*, or EIT* are demonstrated on a wide variety of problems, but it is also demonstrated that they are *not* the best choice for *every* planning problem. End-users should consider problem properties when selecting a planner and the following three paragraphs aim to aid in that selection.

BIT* and ABIT* often outperform planners that build search trees from both directions, such as RRT-Connect, AIT*, and EIT*, if the area around the start is cluttered with obstacles and the area around the goal is mostly obstacle free. For example, such an obstacle configuration emerged in one of the 200 random

Reeds-Shepp problems evaluated in this thesis. RRT-Connect and AIT* were unable to solve this instance in any of their 100 attempts and EIT* only succeeded once in its 100 attempts. BIT* and ABIT* successfully solved this instance in 80 and 76 out of 100 attempts, respectively.

BIT*, ABIT*, and AIT* depend on admissible cost heuristics to guide their searches. If no admissible cost heuristic is available, then they often do not show improved performance over uninformed planners, such as FMT*. Note that EIT* can still find initial solutions quickly, because it initially orders its search on computational effort instead of solution cost.

Finally, the performance improvement of AIT* and EIT* depends on the computational cost of edge evaluation (i.e., collision detection and edge-cost computation). For example, anecdotal tests not presented in this thesis show that the denser the collision detection resolution is on the abstract problems of Sections 3.4.1 and 5.4.1, the better AIT* and EIT* perform relative to other planners. This trend is also evident from the empirical results presented in this thesis, as AIT* and EIT* perform best when collision detection is most expensive (i.e., on the knee replacement dislocation problem of Sections 4.4.3 and 5.4.4). A possible explanation for this is that AIT* and EIT* fully evaluate the least number of edges due to their more accurate heuristics and the computational overhead of improving their heuristics with the reverse search becomes negligible as edge evaluation becomes more expensive.

Ongoing extensions of the work presented in this thesis include applying the ideas developed in AIT* to task an motion planning and modifying EIT* to leverage previous work in a multiquery setting. Future work might look into finding better ways to tune parameter update policies for ABIT* and EIT*, and using different types of intention-specific information in EIT*. EIT* could also be tailored to specific problem domains by learning cost and effort heuristics from data of that domain. Many learning techniques could be used to learn such heuristics since EIT* can leverage cost and effort heuristics even if they are not admissible.

In summary, this thesis presented the following core contributions:

- A conceptual identification of optimization-, environment-, and intent-specific information and a demonstration how these sources of information can be leveraged to improve performance in sampling-based planning.
- A detailed description of ABIT*, which shows how optimization-specific information can be leveraged effectively by incorporating advanced graph-search techniques in sampling-based planning (Chapter 3).
- A detailed description of AIT*, which shows how optimization- and environment-specific information can be leveraged effectively by searching sampling-based approximations with a hierarchical bidirectional search that is asymmetric in purpose and computational cost (Chapter 4).
- A detailed description of EIT*, which shows how optimization-, environment- and intent-specific information can be leveraged to align the search of a problem with its specific priorities (Chapter 5).
- Proofs of the almost-sure asymptotic optimality of ABIT*, AIT*, and EIT* that combine established results from the literature on sampling-based planning and graph-based search (Chapters 3, 4, and 5).
- Empirical demonstrations of the benefits of ABIT*, AIT*, and EIT* on abstract, nonholonomic, manipulator, and biomedical problems with diverse search spaces of up to 16 dimensions, including nonholonomic constraints and optimizing two popular objectives (Chapters 3, 4, and 5).
- Publicly available C++ implementations of ABIT*, AIT*, and EIT* in OMPL (more information at `https://robotic-esp.com/code/`).

Bibliography

Adiyatov, O., Sultanov, K., Zhumabek, O., and Varol, H. A. (2017). Sparse tree heuristics for RRT* family motion planners. In *Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1447–1452. (Cited on page 32.)

Agarwal, P. K., Fox, K., and Salzman, O. (2018). An efficient algorithm for computing high-quality paths amid polygonal obstacles. *ACM Transactions on Algorithms*, 14(4):1–21. (Cited on page 124.)

Aine, S. and Likhachev, M. (2016). Truncated incremental search. *Artificial Intelligence*, 234:49–77. (Cited on pages 25, 52, 53, 79, 80, 85, and 139.)

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V., and Likhachev, M. (2014). Multi-Heuristic A*. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 1–10. (Cited on page 23.)

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V., and Likhachev, M. (2016). Multi-Heuristic A*. *The International Journal of Robotics Research (IJRR)*, 35(1–3):224–243. (Cited on page 23.)

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 on page 32.)

Akinc, M., Bekris, K. E., Chen, B. Y., Ladd, A. M., Plaku, E., and Kavraki, L. E. (2003). Probabilistic Roadmaps of Trees for parallel computation of multiple query roadmaps. In *Proceedings of the International Symposium of Robotics Research (ISRR)*, volume 15, pages 80–89. (Cited on page 29.)

Amato, N. M., Bayazit, O. B., Dale, L. K., Jones, C., and Vallejo, D. (1998a). Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 630–637. (Cited on page 29.)

Amato, N. M., Bayazit, O. B., Dale, L. K., Jones, C., and Vallejo, D. (1998b). OBPRM: An obstacle-based PRM for 3d workspaces. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 155–168. (Cited on page 28.)

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 on page 28.)

Anshelevich, E., Owens, S., Lamiraux, F., and Kavraki, L. E. (2000). Deformable volumes in path planning applications. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2290–2295. (Cited on page 29.)

Apaydin, M. S., Singh, A. P., Brutlag, D. L., and Latombe, J.-C. (2001). Capturing molecular energy landscapes with probabilistic conformational roadmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 932–939. (Cited on page 29.)

Arefin, A. K. S. and Saha, B. A. K. (2010). A new approach of iterative deepening bi-directional heuristic front-to-front algorithm (IDBHFFA). *International Journal of Electrical & Computer Sciences (IJECS-IJENS)*, 10(2):12–20. (Cited on page 21.)

Arslan, O., Berntorp, K., and Tsiotras, P. (2017). Sampling-based algorithms for optimal motion planning using closed-loop prediction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4991–4996. (Cited on page 31.)

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 on page 31.)

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 on page 31.)

Barker, J. K. and Korf, R. E. (2015). Limitations of front-to-end bidirectional heuristic search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1086–1092. (Cited on page 20.)

Barraquand, J., Kavraki, L. E., Latombe, J.-C., Motwani, R., Li, T.-Y., and Prabhakar, R. (1997). A random sampling scheme for path planning. *The International Journal of Robotics Research (IJRR)*, 16(6):759–774. (Cited on page 29.)

Bayazit, O. B., Lien, J.-M., and Amato, N. M. (2002). Probabilistic roadmap motion planning for deformable objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2126–2133. (Cited on page 29.)

Bayazit, O. B., Song, G., and Amato, N. M. (2001a). Enhancing randomized motion planners: Exploring with haptic hints. *Autonomous Robots*, 10(2):163–174. (Cited on page 29.)

Bayazit, O. B., Song, G., and Amato, N. M. (2001b). Ligand binding with OBPRM and haptic user input. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 954–959. (Cited on page 29.)

Bekris, K. E., Chen, B. Y., Ladd, A. M., Plaku, E., and Kavraki, L. E. (2003). Multiple query probabilistic roadmap planning using single query planning primitives. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 656–661. (Cited on page 29.)

Bekris, K. E. and Kavraki, L. E. (2008). Informed and probabilistically complete search for motion planning under differential constraints. In *Proceedings of the International Symposium on Search Techniques in Artificial Intelligence and Robotics*, pages 3–10. (Cited on page 124.)

Berenson, D., Siméon, T., and Srinivasa, S. S. (2011). Addressing cost-space chasms in manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4561–4568. (Cited on page 31.)

Bertram, D., Kuffner Jr., J. J., Dillmann, R., and Asfour, T. (2006). An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1874–1879. (Cited on page 31.)

Bohlin, R. and Kavraki, L. E. (2000). Path planning using Lazy PRM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 521–528. (Cited on page 30.)

Boor, V., Overmars, M. H., and van der Stappen, A. F. (1999). The Gaussian sampling strategy for probabilisitic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1018–1023. (Cited on page 29.)

Border, R. and Gammell, J. D. (2020). Proactive estimation of occlusions and scene coverage for planning next best views in an unstructured representation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4219–4226. (Cited on page 144.)

Border, R. and Gammell, J. D. (2021). The Surface Edge Explorer (SEE): A measurement-direct approach to next best view planning. *In preparation.* (Cited on pages 143 and 144.)

Border, R., Gammell, J. D., and Newman, P. (2018). Surface Edge Explorer: Planning next best views directly from 3D observations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. (Cited on page 144.)

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)*, pages 1481–1487. (Cited on pages 29 and 49.)

Bruce, J. and Veloso, M. (2002). Real-time randomized path planning for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2383–2388. (Cited on page 32.)

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 on page 31.)

Burns, B. and Brock, O. (2003). Information theoretic construction of probabilistic roadmaps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 650–655. (Cited on page 29.)

Burns, B. and Brock, O. (2004). Model-based motion planning. Technical report, University of Massachusetts Amherst. (Cited on page 29.)

Burns, E., Ruml, W., and Do, M. B. (2013). Heuristic search when time matters. *Journal of Artificial Intelligence Research (JAIR)*, 47:697–740. (Cited on page 105.)

Cadmus To, K. Y., Lee, B. K. M., Yoo, C., Anstee, S., and Fitch, R. (2019). Streamlines for motion planning in underwater currents. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4619–4625. (Cited on page 29.)

Cadmus To, K. Y., Yoo, C., Anstee, S., and Fitch, R. (2020). Distance and steering heuristics for streamline-based flow field planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1867–1873. (Cited on page 32.)

Canny, J. F. (1988). *The complexity of robot motion planning*. MIT Press. ISBN: 978-0-262-03136-3. (Cited on page 13.)

Caron, S., Pham, Q.-C., and Nakamura, Y. (2014). Completeness of randomized kinodynamic planners with state-based steering. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5818–5823. (Cited on page 32.)

Chen, C., Rickert, M., and Knoll, A. (2013). Combining space exploration and heuristic search in online motion planning for nonholonomic vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1307–1312. (Cited on page 124.)

Chen, J., Holte, R. C., Zilles, S., and Sturtevant, N. R. (2017). Front-to-end bidirectional heuristic search with near-optimal node expansions. In *Proceedings*

of the International Joint Conference on Artificial Intelligence (IJCAI), pages 489–495. (Cited on page 20.)

Chen, L., Yu, L., Libin, S., and Jiwen, Z. (2021). Greedy BIT* (GBIT*): Greedy search policy for sampling-based optimal planning with a faster initial solution and convergence. In *Proceedings of the IEEE International Conference on Computer, Control, and Robotics (ICCCR)*, pages 30–36. (Cited on page 68.)

Chen, P. C. and Hwang, Y. K. (1998). SANDROS: A dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, 14(3):390–403. (Cited on page 124.)

Cheng, A., Saxena, D. M., and Likhachev, M. (2019). Bidirectional heuristic search for motion planning with an extend operator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7425–7430. (Cited on page 23.)

Cheng, P. and LaValle, S. M. (2001). Reducing metric sensitivity in randomized trajectory design. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 43–48. (Cited on page 31.)

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. (Cited on page 44.)

Cohen, B., Phillips, M., and Likhachev, M. (2014). Planning single-arm manipulations with n-arm robots. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 1–9. (Cited on pages 48 and 72.)

Connell, D. and Manh La, H. (2017). Dynamic path planning and replanning for mobile robots using RRT*. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1429–1434. (Cited on page 32.)

Cortés, J., Jaillet, L., and Siméon, T. (2007). Molecular disassembly with RRT-like algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3301–3306. (Cited on page 32.)

Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics Automation Magazine*, 19(4):72–82. (Cited on pages 8 and 143.)

Culberson, J. C. and Schaeffer, J. (1996). Searching with pattern databases. In *Proceedings of the 11th Conference of the Canadian Society for the Computational Study of Intelligence*, pages 402–416. (Cited on page 26.)

Culberson, J. C. and Schaeffer, J. (1998). Pattern databases. *Computational Intelligence*, 14(3):318–334. (Cited on page 26.)

Dalibard, S. and Laumond, J.-P. (2008). Control of probabilistic diffusion in motion planning. In *Algorithmic Foundations of Robotics VIII*, volume 57, pages 467–481. Springer. (Cited on page 31.)

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton University Press. ISBN: 978-0-691-08000-3. (Cited on pages 18 and 19.)

Das, N. and Yip, M. (2020). Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics (T-RO)*, 36(4):1096–1114. (Cited on page 57.)

Davis, H. W., Pollack, R. B., and Sudkamp, T. (1984). Towards a better understanding of bidirectional search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 68–72. (Cited on page 21.)

de Champeaux, D. (1983). Bidirectional heuristic search again. *Journal of the Association for Computing Machinery (JACM)*, 30(1):22–32. (Cited on page 21.)

de Champeaux, D. and Sint, L. (1977). An improved bidirectional heuristic search algorithm. *Journal of the Association for Computing Machinery (JACM)*, 24(2):177—191. (Cited on page 21.)

Dechter, R. and Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *Journal of the Association for Computing Machinery (JACM)*, 32(3):505–536. (Not cited.)

Dellin, C. M. and Srinivasa, S. S. (2016). A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 459–467. (Cited on page 72.)

Denny, J., Morales, M., Rodriguez, S., and Amato, N. M. (2013). Adapting RRT growth for heterogeneous environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1772–1778. (Cited on page 32.)

Devaurs, D., Siméon, T., and Cortés, J. (2013). Enhancing the Transition-Based RRT to deal with complex cost spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4120–4125. (Cited on page 31.)

Devaurs, D., Siméon, T., and Cortés, J. (2014). A multi-tree extension of the Transition-Based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2991–2996. (Cited on page 31.)

Devaurs, D., Siméon, T., and Cortés, J. (2016). Optimal path planning in complex cost spaces with sampling-based algorithms. *IEEE Transactions on Automation Science and Engineering (TASE)*, 13(2):415–424. (Cited on page 31.)

Diankov, R. (2010). *Automated construction of robotic manipulation programs*. PhD thesis, Carnegie Mellon University. (Cited on page 93.)

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271. (Cited on pages 15, 17, and 42.)

Dillenburg, J. F. and Nelson, P. C. (1994). Perimeter search. *Artificial Intelligence*, 65(1):165–178. (Cited on page 21.)

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412. (Cited on page 19.)

Du, W., Islam, F., and Likhachev, M. (2020). Multi-Resolution A*. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 29–37. (Cited on page 16.)

Edelkamp, S. and Schrödel, S. (2011). *Heuristic search: Theory and applications*. Morgan Kaufmann Publishers, Inc. ISBN: 978-0-12-372512-7. (Cited on page 20.)

Esposito, J. M. and Wright, J. N. (2019). Matrix completion as a post-processing technique for probabilistic roadmaps. *The International Journal of Robotics Research (IJRR)*, 38(2-3):388–400. (Cited on page 29.)

Felner, A., Korf, R. E., and Hanan, S. (2004). Additive pattern database heuristics. *Journal of Artificial Intelligence Research (JAIR)*, 22:279–318. (Cited on page 26.)

Ferguson, D., Kalra, N., and Stentz, A. (2006). Replanning with RRTs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1243–1248. (Cited on page 32.)

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 on page 32.)

Ferguson, D. and Stentz, A. (2007). Anytime, dynamic planning in high-dimensional search spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1310–1315. (Cited on page 32.)

Gammell, J. D. (2017). *Informed anytime search for continuous planning problems*. PhD thesis, University of Toronto. (Cited on page 2.)

Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S. (2018). Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics (T-RO)*, 34(4):966–984. (Cited on pages 4, 31, 46, 49, 75, 106, and 142.)

Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S. (2020). Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research (IJRR)*, 39(5):543–567. (Cited on pages 4, 33, 42, 49, and 84.)

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning 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. (Cited on page 31.)

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. (Cited on pages 4, 33, and 42.)

Gammell, J. D. and Strub, M. P. (2021). Asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):295–318. (Cited on pages 8, 34, and 141.)

García, F. M., Kapadia, M., and Badler, N. (2014). GPU-based dynamic search on adaptive resolution grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. (Cited on page 16.)

Geraerts, R. and Overmars, M. H. (2004). Sampling techniques for probabilistic roadmap planners. Technical report, Utrecht University. (Cited on page 29.)

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 on page 124.)

Glassman, E. and Tedrake, R. (2010). A quadratic regulator-based heuristic for rapidly exploring state space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. (Cited on page 32.)

Gonçalves, V. M., Bolonhez, E. M. B., Campos, G. E. M., and Hoffmann Sathler, L. (2021). Transmission line routing optimization using rapid random trees. *Electric Power Systems Research (EPSR)*, 194:1–11. (Cited on pages 1, 2, and 141.)

Gottschalk, S., Lin, M. C., and Manocha, D. (1996). OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 171–180. (Cited on pages 59 and 93.)

Gunther, T. V., Murray, D. W., Miller, R., Wallace, D. A., Carr, A. J., O'Connor, J. J., McLardy-Smith, P., and Goodfellow, J. W. (1996). Lateral unicompartmental arthroplasty with the Oxford meniscal knee. *The Knee*, 3(1):33–39. (Cited on page 95.)

Hansen, E. A. and Zhou, R. (2007). Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28(1):267–297. (Cited on page 24.)

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 on pages 2, 17, 18, 43, 85, 113, 119, 120, and 121.)

Hatem, M. and Ruml, W. (2014). Simpler bounded suboptimal search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 856–862. (Cited on page 105.)

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 on pages 30, 101, 110, and 137.)

Heiden, E., Palmieri, L., Bruns, L., Arras, K. O., Sukhatme, G. S., and Koenig, S. (2021). Bench-MR: A motion planning benchmark for wheeled mobile robots. *IEEE Robotics and Automation Letters (RA-L)*. (Cited on page 124.)

Helgason, R. V., Kennington, J. L., and Stewart, B. D. (1993). The one-to-one shortest-path problem: An empirical analysis with the two-tree Dijkstra algorithm. *Computational Optimization and Applications*, 2(1). 47–75. (Cited on page 18.)

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)*, pages 1408–1413. (Cited on page 28.)

Holston, A. C., Kim, D.-H., and Kim, J.-H. (2017). Fast-BIT*: Modified heuristics for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1892–1899. (Cited on page 44.)

Holte, R. C., Felner, A., Sharon, G., Sturtevant, N. R., and Chen, J. (2017). MM: A bidirectional search algorithm that is guaranteed to meet in the middle. *Artificial Intelligence*, 252:232–266. (Cited on page 19.)

Holte, R. C., Perez, M. B., Zimmer, R. M., and MacDonald, A. J. (1996). Hierarchical A*: Searching abstraction hierarchies efficiently. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 530–535. (Cited on page 26.)

Hsu, D., Jiang, T., Reif, J., and Sun, Z. (2003). The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4420–4426. (Cited on page 28.)

Hsu, D., Kavraki, L. E., Latombe, J.-C., Motwani, R., and Sorkin, S. (1998). On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 141–154. (Cited on page 28.)

Hsu, D., Latombe, J.-C., and Kurniawati, H. (2006). On the probabilistic foundations of Probabilistic Roadmap planning. *The International Journal of Robotics Research (IJRR)*, 25(7):627–643. (Cited on page 29.)

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)*, pages 495–512. (Cited on page 32.)

Hsu, D., Sánchez-Ante, G., and Sun, Z. (2005). Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3874–3880. (Cited on page 29.)

Huynh, J. (2008). Separating axis theorem for oriented bounding boxes. Technical report, University of California. (Cited on page 59.)

Hwan, J. J., Karaman, S., and Frazzoli, E. (2011). Anytime computation of time-optimal off-road vehicle maneuvers using the RRT. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 3276–3282. (Cited on page 31.)

Islam, F., Narayanan, V., and Likhachev, M. (2016). A*-Connect: Bounded suboptimal bidirectional heuristic search. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2752–2758. (Cited on page 23.)

Isto, P. (2002). Constructing probabilisitic roadmaps with powerful local planning and path optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2323–2328. (Cited on page 29.)

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 on page 31.)

Jaillet, L., Cortés, J., and Siméon, T. (2010). Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics (T-RO)*, 26(4):635–646. (Cited on page 31.)

Jaillet, L., Yershova, A., LaValle, S. M., and Siméon, T. (2005). Adaptive tuning of the sampling domain for dynamic-domain RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2851–2856. (Cited on page 32.)

Janson, L., Ichter, B., and Pavone, M. (2018). Deterministic sampling-based motion planning: Optimality, complexity, and performance. *The International Journal of Robotics Research (IJRR)*, 37(1):46–61. (Cited on pages 29, 30, 42, and 50.)

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 of Robotics Research (ISRR)*, pages 667–684. (Cited on pages 32 and 42.)

Janson, L., Schmerling, E., Clark, A., and Pavone, M. (2015). 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 on pages 33, 42, 50, and 84.)

Kaindl, H. and Kainz, G. (1997). Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research (JAIR)*, 7(7):283–317. (Cited on pages 20 and 21.)

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 on page 11.)

Karaman, S. and Frazzoli, E. (2010a). Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 267–274. (Cited on pages 2 and 31.)

Karaman, S. and Frazzoli, E. (2010b). Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 7681–7687. (Cited on page 31.)

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research (IJRR)*, 30(7):846–894. (Cited on pages 2, 12, 14, 29, 30, 31, 32, 33, 34, 35, 49, 52, 84, 85, and 119.)

Karaman, S. and Frazzoli, E. (2013). Sampling-based optimal motion planning for non-holonomic dynamical systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5041–5047. (Cited on page 31.)

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 on page 31.)

Kavraki, L. E., Kolountzakis, M. N., and Latombe, J.-C. (1998a). Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171. (Cited on page 29.)

Kavraki, L. E., Lamiraux, F., and Holleman, C. (1998b). Towards planning for elastic objects. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 313–325. (Cited on page 29.)

Kavraki, L. E. and Latombe, J.-C. (1994). Randomized preprocessing of configuration for fast path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2138–2145. (Cited on page 28.)

Kavraki, L. E., Latombe, J.-C., Motwani, R., and Prabhakar, R. (1998c). Randomized query processing in robot path planning. *Journal of Computer and System Sciences (JCSS)*, 57(1):50–60. (Cited on page 29.)

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 on pages 28 and 103.)

Kiesel, S. (2016). *Robotics needs non-classical planning.* PhD thesis, University of New Hampshire. (Cited on page 104.)

Kiesel, S., Burns, E., and Ruml, W. (2012). Abstraction-guided sampling for motion planning. In *Proceedings of the Symposium on Combinatorial Search (SOCS)*, pages 162–163. (Cited on page 74.)

Kiesel, S., Gu, T., and Ruml, W. (2017). An effort bias for sampling-based motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2864–2871. (Cited on pages 103 and 104.)

Kim, D., Kwon, Y., and Yoon, S.-E. (2018). Dancing PRM*: Simultaneous planning of sampling and optimization with configuration free space approximation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7071–7078. (Cited on page 124.)

Kingston, Z., Moll, M., and Kavraki, L. E. (2018). Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):159–185. (Cited on page 13.)

Kingston, Z., Moll, M., and Kavraki, L. E. (2019). Exploring implicit spaces for constrained sampling-based planning. *The International Journal of Robotics Research (IJRR)*, 38(10-11):1151–1178. (Cited on page 13.)

Kleinbort, M., Salzman, O., and Halperin (2016). Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, volume 13, pages 624–639. (Cited on pages 101, 110, and 137.)

Kleinbort, M., Salzman, O., and Halperin, D. (2020). Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning. In *Algorithmic Foundations of Robotics XII*, volume 13, pages 624–639. Springer. (Cited on pages 35, 101, 110, and 137.)

Kleinbort, M., Solovey, K., Littlefield, Z., Bekris, K. E., and Halperin, D. (2019). Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):10–16. (Cited on page 32.)

Klemm, S., Oberländer, J., Hermann, A., Roennau, A., Schamm, T., Zollner, J. M., and Dillmann, R. (2015). RRT*-Connect: Faster, asymptotically optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1670–1677. (Cited on page 31.)

Ko, I., Kim, B., and Park, F. C. (2014). Randomized path planning on vector fields. *The International Journal of Robotics Research (IJRR)*, 33(13):1664–1682. (Cited on page 32.)

Koenig, S. and Likhachev, M. (2002). D* Lite. In *Proceedings of the National Conference on Artificial Intelligence*, pages 476–483. (Cited on pages 25 and 103.)

Koenig, S. and Likhachev, M. (2005). Adaptive A*. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1311–1312. (Cited on page 25.)

Koenig, S. and Likhachev, M. (2006). Real-Time Adaptive A*. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 281–288. (Cited on page 25.)

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong Planning A*. *Artificial Intelligence*, 155(1-2):93–146. (Cited on pages 25, 79, 86, 88, and 89.)

Köll, A. L. and Kaindl, H. (1993). Bidirectional best-first search with bounded error: Summary of results. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 217–223. (Cited on page 23.)

Korf, R. E. (1985). Depth-First Iterative-Deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109. (Cited on page 105.)

Korf, R. E. (1997). Finding optimal solutions to Rubik's cube using pattern databases. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 700–705. (Cited on page 26.)

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)*, pages 995–1001. (Cited on page 31.)

Kunz, T. and Stilman, M. (2014). Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 233–244. (Cited on page 32.)

Kwa, J. B. H. (1989). BS*: An admissible bidirectional staged heuristic search algorithm. *Artificial Intelligence*, 38:95–109. (Cited on pages 19 and 20.)

Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. (2000). Fast distance queries with rectanular swept sphere volumes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3719–3726. (Cited on page 132.)

LaValle, S. M. (1998). Rapidly-exploring Random Trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Department, Iowa State University. (Cited on page 30.)

LaValle, S. M., Branicky, M. S., and Lindemann, S. R. (2004). On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research (IJRR)*, 23(7-8):673–692. (Cited on pages 29 and 49.)

LaValle, S. M. and Kuffner Jr., J. J. (1999). Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 473–479. (Cited on pages 2 and 30.)

LaValle, S. M. and Kuffner Jr., J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research (IJRR)*, 20(5):378–400. (Cited on pages 2, 30, and 32.)

Le, D. and Plaku, E. (2014). Guiding sampling-based tree search for motion planning with dynamics via probabilistic roadmap abstractions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 212–217. (Cited on pages 32 and 74.)

Leven, P. and Hutchinson, S. (2002). A framework for real-time path planning in changing environments. *The International Journal of Robotics Research (IJRR)*, 21(12):999–1030. (Cited on page 29.)

Lien, J.-M. and Amato, N. M. (2006). Planning motion in completely deformable environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2466–2471. (Cited on page 32.)

Lien, J.-M., Thomas, S. L., and Amato, N. M. (2003). A general framework for sampling on the medial axis of the free space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4439–4444. (Cited on page 28.)

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005). Anytime Dynamic A*: An anytime, replanning algorithm. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 262–271. (Cited on page 25.)

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2008). Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14):1613–1643. (Cited on page 25.)

Likhachev, M., Gordon, G., and Thrun, S. (2004). ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 767–774. (Cited on page 24.)

Likhachev, M. and Koenig, S. (2005). A generalized framework for Lifelong Planning A* search. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 99–108. (Cited on pages 25 and 79.)

Lim, J., Srinivasa, S. S., and Tsiotras, P. (2021). Lazy lifelong planning for efficient replanning in graphs with expensive edge evaluation. *arXiv*. arXiv:2105.12076 [cs.RO]. (Cited on page 72.)

Lindemann, S. R. and LaValle, S. M. (2004). Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3251–3257. (Cited on page 31.)

Maly, M. R. and Kavraki, L. E. (2012). Low-dimensional projections for SyCLoP. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 420–425. (Cited on page 75.)

Mandalika, A., Choudhury, S., Salzman, O., Srinivasa, S. S., and Srinivasa, S. S. (2019). Generalized Lazy Search for robot motion planning: Interleaving search and edge evaluation via event-based toggles. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 745–753. (Cited on page 72.)

Manzini, G. (1995). BIDA*: An improved perimeter search algorithm. *Artificial Intelligence*, 75(2):347–360. (Cited on page 21.)

Matsuta, K., Kobayashi, H., and Shinohara, A. (2010). Multi Target Adaptive A*. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1065–1072. (Cited on page 25.)

Moore, A. W. and Atkenson, C. G. (1995). The Parti-game Algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233. (Cited on page 16.)

Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, S. (2013). RRT*-SMART: A rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10(7):299–310. (Cited on page 32.)

Natarajan, R., Saleem, M. S., Aine, S., Likhachev, M., and Choset, H. (2019). A-MHA*: Anytime Multi-Heuristic A*. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 192–193. (Cited on page 24.)

Nayak, S. and Otte, M. (2021). Bidirectional sampling-based motion planning without two-point boundary value solution. *arXiv preprint*. arXiv:2010.14692. (Cited on page 31.)

Nesnas, I. A., Matthews, J. B., Abad-Manterola, P., Burdick, J. W., Edlund, J. A., Morrison, J. C., Peters, R. D., Tanner, M. M., Miyake, R. N., Solish, B. S., and Anderson, R. C. (2012). Axel and DuAxel rovers for the sustainable exploration of extreme terrains. *Journal of Field Robotics*, 29(4):663–685. (Not cited.)

Nicholson, T. A. J. (1966). Finding the shortest route between two points in a network. *The Computer Journal*, 9(3):275–280. (Cited on page 19.)

Nielsen, C. L. and Kavraki, L. E. (2000). A two level fuzzy PRM for manipulation planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1716–1721. (Cited on page 30.)

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3d Euclidean signed distance fields for on-board MAV planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. (Cited on page 64.)

Orthey, A., Akbar, S., and Toussaint, M. (2020). Multilevel motion planning: A fiber bundle formulation. *arXiv preprint*. arXiv:2007.09435 [cs.RO]. (Cited on page 75.)

Orthey, A., Escande, A., and Yoshida, E. (2018). Quotient-space motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8089–8096. (Cited on page 75.)

Orthey, A. and Toussaint, M. (2019). Rapidly-exploring Quotient-Space Trees: Motion planning using sequential simplifications. In *Proceedings of the International Symposium of Robotics Research (ISRR)*, pages 1–16. (Cited on page 75.)

Otte, M. and Frazzoli, E. (2015). RRT$^{\text{X}}$: Real-time motion planning/replanning for environments with unpredictable obstacles. In *Algorithmic Foundations of Robotics XI*, volume 107, pages 461–478. (Cited on page 32.)

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 on page 32.)

Overmars, M. H. (1992). A random approach to motion planning. Technical report, Utrecht University. (Cited on page 28.)

Overmars, M. H. and Švestka, P. (1994). A probabilistic learning approach to motion planning. Technical report, Utrecht University. (Cited on page 28.)

Palmieri, L., Bruns, L., Meurer, M., and Arras, K. O. (2020). Dispertio: Optimal sampling for safe deterministic motion planning. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):362–368. (Cited on page 50.)

Pan, J., Chitta, S., and Manocha, D. (2012). FCL: A general purpose library for collision and proximity queries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866. (Cited on page 93.)

Pandit, H., Jenkins, C., Beard, D., Price, A., Gill, R. H. S., Dodd, C., and Murray, D. (2010). Mobile bearing dislocation in lateral unicompartmental knee replacement. *The Knee*, 17(6):392–397. (Cited on pages 2, 95, and 96.)

Paton, M., Strub, M. P., Brown, T., Greene, R. J., Lizewski, J., Patel, V., Gammell, J. D., and Nesnas, I. A. (2020). Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7034–7041. (Cited on pages 1, 8, 41, 62, 63, 64, 65, 66, 68, 141, 142, and 143.)

Pearl, J. and Kim, J. H. (1982). Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, PAMI-4(4):392–399. (Cited on pages 23 and 105.)

Penrose, M. D. (2003). *Random geometric graphs*. Oxford University Press. ISBN: 978-0-19-850626-0. (Cited on page 4.)

Perez, A., Platt Jr., R., Konidaris, G., Kaelbling, L., and Lozano-Perez, T. (2012). LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2537–2542. (Cited on page 32.)

Phillips, J. M., Bedrossian, N., and Kavraki, L. E. (2004). Guided Expansive Spaces Trees: A search strategy for motion- and cost-constrained state spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3968–3973. (Cited on page 32.)

Pisula, C., Hoff III, K., Lin, M. C., and Manocha, D. (2000). Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 1–15. (Cited on page 28.)

Plaku, E. (2013). Robot motion planning with dynamics as hybrid search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1415–1421. (Cited on pages 32 and 74.)

Plaku, E. (2015). Region-guided and sampling-based tree search for motion planning with dynamics. *IEEE Transactions on Robotics (T-RO)*, 31(3):723–735. (Cited on page 74.)

Plaku, E., Bekris, K. E., Chen, B. Y., Ladd, A. M., and Kavraki, L. E. (2005). Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics (T-RO)*, 21(4):597–608. (Cited on page 29.)

Plaku, E., Kavraki, L. E., and Vardi, M. Y. (2010). Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics (T-RO)*, 26(3):469–482. (Cited on page 74.)

Pohl, I. (1969). *Bi-directional and heuristic search in path problems*. PhD thesis, Stanford University, Stanford, California 94305. (Cited on pages 18, 19, 20, and 43.)

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204. (Cited on page 22.)

Pohl, I. (1971). Bi-directional search. *Machine Intelligence*, 6:127–140. (Cited on page 19.)

Politowski, G. and Pohl, I. (1984). D-node retargeting in bidirectional heuristic search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 274–277. (Cited on page 21.)

Qureshi, A. H. and Ayaz, Y. (2015). Intelligent bidirectional Rapidly-exploring Random Trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems*, 68:1–11. (Cited on page 31.)

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 on page 11.)

Reeds, J. A. and Shepp, L. A. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393. (Cited on page 59.)

Reid, W., Paton, M., Karumanchi, S., Chamberlain-Simon, B., Emanuel, B., and Meirion-Griffith, G. (2020). Autonomous navigation over Europa analogue terrain for an actively articulated wheel-on-limb rover. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1939–1946. (Cited on pages 8, 68, and 143.)

Richter, S., Thayer, J. T., and Ruml, W. (2010). The joy of forgetting: Faster anytime search via restarting. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 137–144. (Cited on page 24.)

Rodriguez, S., Xinyu, T., Lien, J.-M., and Amato, N. M. (2006). An obstacle-based Rapidly-exploring Random Tree. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 895–900. (Cited on page 32.)

Ruml, W. and Do, M. B. (2007). Best-first utility-guided search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2378–2384. (Cited on page 105.)

Saha, M., Latombe, J.-C., Chang, Y.-C., and Prinz, F. (2005). Finding narrow passages with probabilistic roadmaps: The small-step retraction method. *Autonomous Robots*, 19(3):301–319. (Cited on page 28.)

Sakcak, B. and LaValle, S. M. (2021). Complete path planning that simultaneously optimizes length and clearance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. (Cited on page 124.)

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 on page 31.)

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 on pages 42 and 72.)

Sánchez, G. and Latombe, J.-C. (2003). A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics Research*, pages 403–417. Springer. (Cited on pages 101, 106, and 110.)

Schmerling, E., Janson, L., and Pavone, M. (2015a). Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 2574–2581. (Cited on page 42.)

Schmerling, E., Janson, L., and Pavone, M. (2015b). Optimal sampling-based motion planning under differential constraints: The driftless case. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2368–2375. (Cited on page 42.)

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research (IJRR)*, 33(9):1251–1270. (Cited on page 11.)

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)*, pages 1–10. (Cited on page 11.)

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 1591–1595. (Cited on pages 33 and 42.)

Shkolnik, A. and Tedrake, R. (2009). Path planning in 1000+ dimensions using a task-space Voronoi bias. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2061–2067. (Cited on page 31.)

Silver, D. (2005). Cooperative pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 117–122. (Cited on pages 26 and 27.)

Singh, A. P., Latombe, J.-C., and Brutlag, D. L. (1999). A motion planning approach to flexible ligand binding. In *Proceedings of the AAAI International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261. (Cited on page 29.)

Sivamurugan, M. S. and Ravindran, B. (2014). RRTPI: Policy iteration on continuous domains using Rapidly-exploring Random Trees. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4362–4367. (Cited on page 32.)

Solovey, K. and Kleinbort, M. (2020). The critical radius in sampling-based motion planning. *The International Journal of Robotics Research (IJRR)*, 39(2-3):266–285. (Cited on pages 29 and 50.)

Solovey, K., Salzman, O., and Halperin, D. (2018). New perspective on sampling-based motion planning via random geometric graphs. *The International Journal of Robotics Research (IJRR)*, 37(10):1117–1133. (Cited on pages 29 and 50.)

Song, G. and Amato, N. M. (2001). Using motion planning to study protein folding pathways. In *Proceedings of the International Conference on Computational Biology (RECOMB)*, pages 287–296. (Cited on page 29.)

Song, G. and Amato, N. M. (2004). A motion-planning approach to folding: From paper craft to protein folding. *IEEE Transactions on Robotics and Automation*, 20(1):60–71. (Cited on page 29.)

Song, G., Miller, S., and Amato, N. M. (2001). Customizing PRM roadmaps at query time. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1500–1505. (Cited on page 30.)

Song, G., Thomas, S., and Amato, N. M. (2003). A general framework for PRM motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4445–4450. (Cited on page 30.)

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 on page 43.)

Starek, J. A., Schmerling, E., Janson, L., and Pavone, M. (2014). Bidirectional Fast Marching Trees: An optimal sampling-based algorithm for bidirectional motion planning. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 1–15. (Cited on page 43.)

Stentz, A. (1995). The Focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1–8. (Cited on page 25.)

Strub, M. P. and Gammell, J. D. (2020a). Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198. (Cited on pages 5, 8, and 72.)

Strub, M. P. and Gammell, J. D. (2020b). Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–136. (Cited on pages 5, 8, 41, and 67.)

Strub, M. P. and Gammell, J. D. (2021a). Admissible heuristics for obstacle clearance optimization objectives. Technical Report TR-2021-MPS001, Estimation, Search, and Planning (ESP) Research Group, University of Oxford. arXiv:2104.02298 [cs.RO]. (Cited on pages 8 and 126.)

Strub, M. P. and Gammell, J. D. (2021b). AIT* and EIT*: Asymmetric bidirectional sampling-based path planning. *The International Journal of Robotics Research (IJRR)*. To appear, Manuscript #IJR-21-4179. (Cited on pages 5, 6, 8, 72, and 102.)

Sturtevant, N. R. and Felner, A. (2018). A brief history and recent achievements in bidirectional search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8000–8006. (Cited on pages 18, 19, 20, and 21.)

Sturtevant, N. R., Shperberg, S., Felner, A., and Chen, J. (2020). Predicting the effectiveness of bidirectional heuristic search. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 281–290. (Not cited.)

Sudhakar, S., Karaman, S., and Sze, V. (2020). Balancing actuation and computing energy in motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. (Not cited.)

Sukharev, A. G. (1971). Optimal strategies of the search for an extremum. *USSR Computational Mathematics and Mathematical Physics*, 11(4):119–137. (Cited on page 13.)

Sun, X. and Koenig, S. (2007). The Fringe-Saving A* search algorithm — A feasibility study. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2391–2397. (Cited on page 25.)

Sun, X., Koenig, S., and Yeoh, W. (2008). Generalized Adaptive A*. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 469–476. (Cited on page 25.)

Thayer, J. T., Benton, J., and Helmert, M. (2012). Better parameter-free anytime search by minimizing time between solutions. In *Proceedings of the Symposium on Combinatorial Search (SOCS)*, pages 120–128. (Cited on pages 24 and 105.)

Thayer, J. T., Dionne, A., and Ruml, W. (2011). Learning inadmissible heuristics during search. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 250–257. (Cited on pages 26 and 27.)

Thayer, J. T. and Ruml, W. (2008). Faster than Weighted A*: An optimistic approach to bounded suboptimal search. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 355–362. (Cited on page 23.)

Thayer, J. T. and Ruml, W. (2010). Finding acceptable solutions faster using inadmissible information. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 1–2. (Cited on pages 23 and 105.)

Thayer, J. T. and Ruml, W. (2011). Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proceedings of the AAAI International Conference on Automated Planning and Scheduling (ICAPS)*, pages 674–679. (Cited on pages 23, 105, and 119.)

Tsao, M., Solovey, K., and Pavone, M. (2020). Sample complexity of probabilistic roadmaps via ϵ-nets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2196–2202. (Cited on page 29.)

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)*, pages 1178–1183. (Cited on page 31.)

Vadlamudi, S., Aine, S., and Chakrabarti, P. P. (2011). MAWA*: A memory-bounded anytime heuristic-search algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(3):725–735. (Cited on page 24.)

van den Berg, J., Shah, R., Huang, A., and Goldberg, K. (2011). ANA*: Anytime Nonparametric A*. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 105–111. (Cited on page 24.)

Wang, Y., Jha, D. K., and Akemi, Y. (2017). A two-stage RRT path planner for automated parking. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 496–502. (Cited on page 32.)

Wein, R., van den Berg, J., and Halperin, D. (2008). Planning high-quality paths and corridors amidst obstacles. *The International Journal of Robotics Research (IJRR)*, 27(11–12):1213–1231. (Cited on page 124.)

Westbrook, M. G. and Ruml, W. (2020). Anytime kinodynamic motion planning using region-guided search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6789–6796. (Cited on page 75.)

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)*, pages 1024–1031. (Cited on page 28.)

Wilt, C. and Ruml, W. (2012). When does Weighted A* fail? In *Proceedings of the Symposium on Combinatorial Search (SOCS)*, pages 137–144. (Cited on page 23.)

Wilt, C. and Ruml, W. (2015). Speedy versus greedy search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4331—4337. (Cited on page 23.)

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 on page 44.)

Yahja, A., Stentz, A., Singh, S., and Brumitt, B. L. (1998). Framed-quadtree path planning for mobile robots operating in sparse environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 650–655. (Cited on page 16.)

Yang, I., Gammell, J. D., Murray, D. W., and Mellon, S. J. (2020). Using a robotics path planning algorithm to assess the risk of mobile bearing dislocation in lateral unicompartmental knee replacement. In *Proceedings of the Annual Meeting of the International Society for Computer Assisted Orthopaedic Surgery (CAOS)*, volume 4 of *EPiC Series in Health Sciences*, pages 301–305. (Cited on pages 1, 95, 135, and 141.)

Yang, I., Gammell, J. D., Murray, D. W., and Mellon, S. J. (2021a). The Oxford Domed Lateral Implant: Increasing tibial component wall height reduces the risk of medial dislocation of the mobile bearing. In *Proceedings of the Annual Meeting of the British Orthopedic Research Society (BORS)*, page 1069. (Cited on page 95.)

Yang, I., Gammell, J. D., Murray, D. W., and Mellon, S. J. (2021b). The Oxford domed lateral unicompartmental knee replacement implant: Increasing wall height reduces the risk of bearing dislocation. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*. (Cited on page 95.)

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)*, pages 4405–4410. (Cited on page 28.)

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 on page 32.)

Yershova, A., Jain, S., LaValle, S. M., and Mitchell, J. C. (2010). Generating uniform incremental grids on SO(3) using the Hopf fibration. *The International Journal of Robotics Research (IJRR)*, 29(7):801–812. (Cited on page 50.)

Yershova, A. and LaValle, S. M. (2004). Deterministic sampling methods for spheres and SO(3). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3974–3980. (Cited on pages 29 and 49.)

Zhou, R. and Hansen, E. A. (2002). Multiple sequence alignment using Anytime A*. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 975–976. (Cited on page 24.)

Zucker, M., Kuffner Jr., J. J., and Branicky, M. S. (2007). Multipartite RRTs for rapid replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1603–1609. (Cited on page 32.)

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 on page 11.)