An Asymptotically-Optimal Sampling-Based Algorithm for Bi-directional Motion Planning

Joseph A. Starek*, Edward Schmerling†, Lucas Janson‡, Marco Pavone*

Abstract—Bi-directional search is a widely used strategy to increase the success and convergence rates of sampling-based motion planning algorithms. Yet, few results are available that merge bi-directional search and asymptotic-optimality together into existing optimal planners, such as PRM*, RRT*, and FMT*. The objective of this paper is to fill this gap. Specifically, this paper presents a bi-directional, sampling-based, asymptotic-optimally algorithm named Bi-directional FMT* (BFMT*) that extends the Fast Marching Trees (FMT*) algorithm to bi-directional search while preserving its key properties, chiefly lazy search and asymptotic optimality through convergence in probability. BFMT* performs a two-source, lazy dynamic programming recursion over a set of randomly-drawn samples, correspondingly generating two search trees: one in cost-to-come space from the initial configuration and another in cost-to-go space from the goal configuration. Multiple strategies are discussed and analyzed for alternating tree growth and interconnecting tree paths. Numerical experiments illustrate the advantages of BFMT* over its unidirectional counterpart, as well as a number of other state-of-the-art planners.

I. INTRODUCTION

Motion planning is a problem central to the field of robotics involving the computation of a path from an initial configuration to a goal configuration that does not collide with nearby obstacles and possibly optimizes an objective function. The problem has a long and rich history, and many algorithmic tools have been developed; we refer the interested reader to [1] and references therein. Arguably, sampling-based algorithms are among the most pervasive, widespread planners available in robotics, including the probabilistic roadmap algorithm (PRM) [2], the expansive space trees algorithm (EST) [3], [4], and the rapidly exploring random trees algorithm (RRT) [5]. Since their development, efforts to improve the “quality” of paths led to asymptotically-optimal variants of RRT and PRM, named RRT* and PRM*, respectively, whereby the cost of the returned solution converges almost surely to the optimum as the number of samples approaches infinity [6]. Recently, a conceptually different asymptotically-optimal, sampling-based motion planning algorithm, called the Fast Marching Trees algorithm (FMT*), has been presented in [7], [8]. Numerical experiments suggested that FMT* converges to an optimal solution faster than PRM* or RRT*, especially in high-dimensional configuration spaces and in scenarios where collision-checking is expensive.

It is a well-known fact that bi-directional search can dramatically increase the convergence rate of planning algorithms, prompting some authors [9] to advocate its use for accelerating essentially any motion planning query. This was first rigorously studied in [10] and later investigated, for example, in [11], [12]. Collectively, the algorithms presented in [9]–[12] belong to the family of non-sampling-based approaches and are more or less closely related to a bi-directional implementation of the Dijkstra Method. More recently, and not surprisingly in light of these performance gains, bi-directional search has been merged with the sampling-based approach, with RRT-Connect and SBL representing the most notable examples [13], [14].

Though such bi-directional versions of RRT and PRM are probabilistically complete, they do not enjoy optimality guarantees. The next logical step in the quest for fast planning algorithms is the design of bi-directional, sampling-based, asymptotically-optimal algorithms. To the best of our knowledge, the only available results in this context are [15] and the unpublished work [16], both discussing bi-directional implementations of RRT*. Both works, however, do not provide a rigorous analysis of optimality. Accordingly, the objective of this paper is to propose and rigorously analyze such an algorithm.

Statement of Contributions: This paper proposes the Bi-directional Fast Marching Trees (BFMT*) algorithm as a tree-based, asymptotically-optimal bi-directional sampling-based planner. BFMT* extends FMT* to bi-directional search and essentially performs a “lazy,” bi-directional dynamic programming recursion over a set of probabilistically-drawn samples in the free configuration space. The contribution of this paper is threefold. First, we present the BFMT* algorithm, for which we propose two strategies for tree growth and two for interconnecting tree paths (yielding four variants total). Secondly, we rigorously prove the asymptotic optimality of BFMT* (under the notion of convergence in probability) and characterize its convergence rate. Finally, we perform numerical experiments across a number of planning spaces that suggest BFMT* converges to an optimal solution at least as fast as FMT*, PRM*, and RRT*, and sometimes significantly faster.

Organization: This paper is structured as follows. Section II defines the path planning problem. Section III then presents BFMT* and discusses alternatives for tree exploration and interconnection. We then follow in Section IV with the proof of asymptotic optimality of BFMT*, a characterization of its convergence rate, and a discussion about extensions to general cost functions and non-uniform sampling strategies. Section V presents results from numerical experiments, and, finally, Section VI concludes with directions for future work.

*Dept. of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, {jstarek, pavone}@stanford.edu.
†Inst. for Computational & Mathematical Engineering, Stanford University, Stanford, CA 94305, schmerling@stanford.edu.
‡Dept. of Statistics, Stanford University, Stanford, CA 94305, ljanson@stanford.edu.

This work was supported in part by NASA ECF Grant #NNX12AQ43G.

The asterisk *, pronounced “star”, is intended to represent asymptotic optimality much like for the RRT* and PRM* algorithms.
II. PROBLEM DEFINITION

Let $X$ be a $d$-dimensional configuration space, and let $X_{\text{obs}}$ be the obstacle region, such that $X \setminus X_{\text{obs}}$ is an open set (we consider $\partial X \subset X_{\text{obs}}$). Denote the obstacle-free space as $X_{\text{free}} = \text{cl}(X \setminus X_{\text{obs}})$, where $\text{cl}(\cdot)$ denotes the closure of a set. A path planning problem, denoted by a triplet $(X_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$, seeks to maneuver from an initial configuration $x_{\text{init}}$ to a goal configuration $x_{\text{goal}}$ through $X_{\text{free}}$. Let a continuous function of bounded variation $\sigma : [0, 1] \rightarrow X$, called a path, be collision-free if $\sigma(\tau) \in X_{\text{free}}$ for all $\tau \in [0, 1]$. A path is called a feasible solution to the planning problem $(X_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$ if it is collision-free, $\sigma(0) = x_{\text{init}}$, and $\sigma(1) = x_{\text{goal}}$.

Let $\Sigma$ be the set of all paths. A cost function for the planning problem $(X_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$ is a function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$ from the set of paths to the nonnegative real numbers; in this paper, we consider as a cost function $c(\sigma)$ the arc length of $\sigma$ with respect to the Euclidean metric in $X$ (the extension to general cost functions will be briefly discussed in Section IV-C).

**Optimal path planning problem:** Given a path planning problem $(X_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$ and an arc length function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, find a feasible path $\sigma^*$ such that $c(\sigma^*) = \min \{c(\sigma) \mid \sigma \text{ is feasible}\}$. If no such path exists, report failure.

Finally, we introduce some definitions concerning the clearance of a path, i.e., its “distance” from $X_{\text{obs}}$ [8]. For a given $\delta > 0$, the $\delta$-interior of $X_{\text{free}}$ is defined as the set of all points that are at least a distance $\delta$ away from any point in $X_{\text{obs}}$. A collision-free path $\sigma$ is said to have strong $\delta$-clearance if it lies entirely inside the $\delta$-interior of $X_{\text{free}}$. A path planning problem with optimal path cost $c^*$ is called $\delta$-robustly feasible if there exists a strictly positive sequence $\delta_n \rightarrow 0$, with $\delta_n \leq \delta \ \forall n \in \mathbb{N}$, and a sequence $\{\sigma_n\}_{n=1}^{\infty}$ of feasible paths such that $\lim_{n \rightarrow \infty} c(\sigma_n) = c^*$ and for all $n \in \mathbb{N}$, $\sigma_n$ has strong $\delta_n$-clearance, $\sigma_n(1) = x_{\text{goal}}$, $\sigma_n(\tau) \neq x_{\text{goal}}$ for all $\tau \in (0, 1)$, and $\sigma_n(0) = x_{\text{init}}$.

III. THE BFMT* ALGORITHM

In this section, we present the Bi-Directional Fast Marching Tree algorithm, BFMT*, represented in pseudocode as Algorithm [1]. To begin, we provide a high-level description of FMT* in Section III-A on which BFMT* is based. We follow in Section III-B with its own high-level description, and then provide additional details in Section III-C.

A. FMT* – High-level description

The FMT* algorithm, introduced in [7], [8], is a unidirectional algorithm that essentially performs a forward dynamic programming recursion over a set of sampled points and correspondingly generates a tree of paths that grow steadily outward in cost-to-come space. The recursion performed by FMT* is characterized by three key features: (1) It is tailored to disk-connected graphs, where two points are considered neighbors (hence connectable) if their distance is below a given bound, referred to as the connection radius; (2) It performs graph construction and graph search concurrently; and (3) For the evaluation of the immediate cost in the dynamic programming recursion, one “lazily” ignores the presence of obstacles, and whenever a locally-optimal (assuming no obstacles) connection to a new sample intersects an obstacle, that sample is simply skipped and left for later (as opposed to looking for other locally-optimal connections in the neighborhood).

The last feature, which makes the algorithm “lazy,” may cause suboptimal connections. A central property of FMT* is that the cases where a suboptimal connection is made become vanishingly rare as the number of samples goes to infinity, which makes the algorithm asymptotically-optimal (AO). This manifests itself into a key computational advantage—by restricting collision detection to only locally-optimal connections, FMT* (as opposed to, e.g., PRM) avoids a large number of costly collision-check computations, at the price of a vanishingly small “degree” of suboptimality. We refer the reader to [7], [8] for a detailed description of the algorithm and its advantages.

B. BFMT* – High-level description

At its core, BFMT* implements a bi-directional version of the FMT* algorithm by simultaneously propagating two wavefronts (henceforth, the leaves of an expanding tree will be referred to as the wavefront of the tree) through the free configuration space. BFMT*, therefore, performs a two-source dynamic programming recursion over a set of sampled points, and correspondingly generates a pair of search trees: one in cost-to-come space from the initial configuration and another in cost-to-go space from the goal configuration (see Fig. 1). Throughout the remainder of the paper, we refer to the former as the forward tree, and to the latter as the backward tree.

![Fig. 1: The BFMT* algorithm generates a pair of search trees: one in cost-to-come space from the initial configuration (blue) and another in cost-to-go space from the goal configuration (purple). The path found by the algorithm is in green color.](image-url)

The dynamic programming recursion performed by BFMT* is characterized by the same lazy feature of FMT* (see Section III-A). However, the time it takes to run BFMT* on a given number of samples can be substantially smaller than for FMT*. Indeed, for not-too-cluttered configuration spaces, the search trees grow hyperspherically, and BFMT* only has to expand about half as far (in both trees) as FMT* in order to return a solution. This is made clear in Fig. 1(a), in which FMT* would have to expand the forward tree twice as far to find a solution. Since runtime scales approximately with edge number, which scales as the linear distance covered by the tree raised to the dimension of the state space, we may expect an approximate speed-up of a factor $2^{d-1}$ over FMT* in $d$-dimensional space (the $-1$ in the exponent is because BFMT* has to expand $2$ trees, so it loses one factor of $2$ advantage).
As for any bi-directional planner, the correctness and computational efficiency of BFMT* hinge upon two key aspects: (i) how computation is interleaved among the two trees (in other words, which wavefront at each step should be chosen for expansion), and (ii) when the algorithm should terminate. In this paper, we compare four possible variations: two options for tree expansions (i.e., item (i)) and two options for termination (i.e., item (ii)).

1) Tree expansion: As explained, BFMT* works by (lazily) propagating two wavefronts. Let $\mathcal{T}$ be the currently-expanding tree and $\mathcal{T}'$ be its companion. Each tree is associated with its own wavefront set, denoted $\mathcal{H}$ and $\mathcal{H}'$ respectively, which contain the nodes on the wavefronts. Here we explore two expansion schemes:

1) “Alternating Trees”: (left column) lines 15–18 in Algorithm 1. At each iteration, the forward and backward trees are “swapped;” that is, the two expanded in turns.

2) “Balanced Trees”: (right column, lines 15–18 in Algorithm 1) The tree with the wavefront node of smallest cost from its root is selected for expansion. If only one tree has a non-empty wavefront, that tree is selected.

The first is the simplest, and emulates the strategy adopted by RRT-Connect. The second, on the other hand, enforces more of a “balanced” search, maintaining equal costs from the root in each wavefront such that the two wavefronts propagate and meet roughly equidistantly in cost-to-go from their roots.

2) “Best Path”: (right column, line 9 in Algorithm 1) The “best path” condition, on the other hand, terminates the algorithm once the node $x_{\text{init}}$, whose selection will be discussed in Section IV. E $\text{COMPANION}(\mathcal{T})$ return the unique path in the tree $\mathcal{T}$ from its root to node $x$. Also, with a slight abuse of notation, let $\text{COMPANION}(x, \mathcal{T})$ return the cost of the unique path in the tree $\mathcal{T}$ from its root to node $x$, and let $\text{COLLISIONFREE}(x, y)$ be a boolean function returning true if the straight-line path between samples $x$ and $y$. Let $\text{PATH}(z, \mathcal{T})$ return the unique path in the tree $\mathcal{T}$ from its root to node $z$. Finally, let $\text{SWAP}(\mathcal{T}, \mathcal{T}')$ be a function that swaps the two trees $\mathcal{T}$ and $\mathcal{T}'$.

Termination: In this paper, we also consider two termination criteria, which affect solution quality and run time:

1) “First Path”: (left column, line 2 in Algorithm 1) The algorithm terminates once the two trees connect; namely, once a node added to tree $\mathcal{T}$ is also in tree $\mathcal{T}'$.

2) “Best Path”: (right column, line 2 in Algorithm 1) The algorithm terminates once the node $z \in \mathcal{H}$ chosen for expansion is in the “interior” of $\mathcal{T}'$ (and hence, already expanded from the standpoint of $\mathcal{T}'$). In abbreviated terms, the algorithm terminates if $z \in \mathcal{V} \setminus \mathcal{H}'$.

Termination with the “first path” criterion returns the first available path discovered, at the moment that the two wavefronts touch (which is not, in general, the lowest cost path). The “best path” condition, on the other hand, terminates at the point where the two wavefronts have propagated sufficiently far through each other that no better solution can be discovered. Intuitively, this is the first moment where the two trees have both selected, at the current iteration or previously, the node $z$ as the minimum cost node from their respective roots.

C. BFMT* – Detailed description

Let $\mathcal{S}$ be a set of points sampled independently and identically from the uniform distribution on $\mathcal{X}_{\text{free}}$, to which $x_{\text{init}}$ and $x_{\text{goal}}$ are added. (The extension to non-uniform sampling distributions is addressed in Section IV-C) Let tree $\mathcal{T}$ be the quadruple $(\mathcal{V}, \mathcal{E}, \mathcal{H}, \mathcal{W})$, where $\mathcal{V}$ is the set of tree nodes, $\mathcal{E}$ is the set of tree edges, and $\mathcal{H}$ and $\mathcal{W}$ are mutually exclusive sets containing the wavefront nodes in $\mathcal{V}$ and the unexplored samples in $\mathcal{S}$, respectively. To be precise, the unexplored set $\mathcal{W}$ stores all samples in the sample set $\mathcal{S}$ that have not yet been considered for addition to the tree of paths. The wavefront set $\mathcal{H}$, on the other hand, tracks in sorted order (by cost from the root) only those nodes which have already been added to the tree that are near enough to tree leaves to actually form better connections. As such, nodes $\mathcal{H}$ and $\mathcal{W}$ play the same role as their counterparts in FMT*, see [7], [8]. However, in this case BFMT* “grows” two such trees, referred to as $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{H}, \mathcal{W})$ and $\mathcal{T}' = (\mathcal{V}', \mathcal{E}', \mathcal{H}', \mathcal{W}')$. Initially, $\mathcal{T}$ is the tree rooted at $x_{\text{init}}$, while $\mathcal{T}'$ is the tree rooted at $x_{\text{goal}}$. Note, however, that the trees are exchanged during the execution of BFMT*, so $\mathcal{T}$ in Algorithm 1 is not always the tree that contains $x_{\text{init}}$.

Before describing BFMT* in detail, we list briefly the basic planning functions employed by the algorithm. Let $\text{SAMPLEFREE}(n)$ be a function that returns a set of $n \in \mathbb{N}$ points sampled independently and identically from the uniform distribution on $\mathcal{X}_{\text{free}}$. Let $\text{COST}(x, y)$ be the cost of the straight-line path between samples $x$ and $y$. Let $\text{PATH}(z, \mathcal{T})$ return the unique path in the tree $\mathcal{T}$ from its root to node $z$. Also, with a slight abuse of notation, let $\text{COST}(x, \mathcal{T})$ return the cost of the unique path in the tree $\mathcal{T}$ from its root to node $x$, and let $\text{COLLISIONFREE}(x, y)$ be a boolean function returning true if the straight-line path between samples $x$ and $y$ is collision free. Given a set of samples $\mathcal{A}$, let $\text{NEAR}(\mathcal{A}, z, r)$ return the set of samples in $\mathcal{A}$ within a ball of radius $r$ centered at sample $z$ (i.e., the set $\{x \in \mathcal{A} | \|x - z\| < r\}$). Pertaining to the tree expansion strategies, let $\text{SWAP}(\mathcal{T}, \mathcal{T}')$ be a function that swaps the two trees $\mathcal{T}$ and $\mathcal{T}'$. Finally, let $\text{COMPANION}(\mathcal{T})$ return the companion tree $\mathcal{T}'$ to $\mathcal{T}$ (or vice versa).

The BFMT* algorithm is represented in Algorithm 1. First, a set of $n$ configurations in $\mathcal{X}_{\text{free}}$ is determined by drawing samples uniformly. Two trees are then initialized using $\text{INITIALIZE}$ as shown in Algorithm 2 with a forward tree rooted at $x_{\text{init}}$ and a reverse tree rooted at $x_{\text{goal}}$. Once complete, tree expansion begins starting with tree $\mathcal{T}$ rooted at $x_{\text{init}}$ using the $\text{EXPAND}$ procedure in Algorithm 3. In the following, the node selected for expansion will be consistently denoted by $z$, while $x_{\text{next}}$ will denote the lowest-cost candidate node for tree connection (i.e., for joining the two trees). The $\text{EXPAND}$ procedure requires the specification of a connection radius parameter, $r_n$, whose selection will be discussed in Section IV. EXPAND implements the “lazy” dynamic programming recursion described (at a high level) in Section III-B making locally-optimal collision-free connections from nodes $x$ near $z$ unexplored by tree $\mathcal{T}$ (those in set $\mathcal{W}$ within search radius $r_n$ of $z$) to wavefront nodes $x'$ near each $x$ (those in set $\mathcal{H}$ within search radius $r_n$ of $x$). Any collision-free edges and newly-connected nodes are then added to $\mathcal{T}$, the connection candidate node $x_{\text{next}}$ is updated, and $z$ is dropped from the list of wavefront nodes. The key feature of the $\text{EXPAND}$ function is that in the execution of the dynamic programming recursion it “lazily” ignores the presence of obstacles (see line 16 – as discussed in Section IV this comes at no loss of (asymptotic) optimality (see also [7], [8]). Note the $\text{EXPAND}$ function is identical to that of unidirectional FMT*, with the exception here of additional lines for tracking of the connection candidate $x_{\text{next}}$.

After expansion, the termination condition is evaluated,
We then provide the (asymptotic) optimality proof for BFMT

Algorithm 1 Bi-directional Fast Marching Tree Algorithm (BFMT*)

1. \( S \leftarrow \{x_{\text{init}}, x_{\text{goal}}\} \cup \text{SAMPLEFREE}(n) \)
2. \( T \leftarrow \text{INITIALIZE}(S, x_{\text{init}}) \)
3. \( T' \leftarrow \text{INITIALIZE}(S, x_{\text{goal}}) \)
4. 
5. \( z \leftarrow x_{\text{init}}, x_{\text{meet}'} \leftarrow \emptyset \)
6. success = false
7. while success = false 
8. \( \text{EXPAND}(T, z, x_{\text{meet}}) \)

**Termination**

**First Path Criterion**

9. if \( x_{\text{meet}} \neq \emptyset \)
10. \( \sigma^* = \text{PATH}(x_{\text{meet}}, T) \cup \text{PATH}(x_{\text{meet}}, T') \)
11. success = true
12.
13. else if \( H' = \emptyset \text{ and } H = \emptyset \)
14. return Failure

**Tree Selection**

15. \( z \leftarrow \text{arg min}_{x' \in H'} \{\text{Cost}(x', T')\} \)
16. \( z_1 \leftarrow \text{arg min}_{x \in H} \{\text{Cost}(x, T)\} \)
17. \( (z, T) \leftarrow \text{arg min}_{(z_1, T'), (z, T')} \{\text{Cost}(x, T)\} \)
18. \( \text{SWAP}(T, T') \)
19. \( T' = \text{COMPANION}(T) \)

**Algorithm 2** Initializes a Fast Marching Tree

1. \( \text{function INITIALIZE}(S, x_0) \)
2. \( V \leftarrow \{x_0\} \)
3. \( E \leftarrow \emptyset \)
4. \( W \leftarrow S \setminus \{x_0\} \)
5. \( H \leftarrow \{x_0\} \)
6. return \( T = (V, E, W, H) \)

IV. ASYMPTOTIC OPTIMALITY OF BFMT*

In this section, we show the optimality of BFMT* for each of the four versions of Algorithm 1 (one for each choice of tree expansion and termination criteria). We begin with a result essentially stating that any path in \( \lambda' \) may be "traced" arbitrarily well by connecting randomly distributed points from a sufficiently large sample set covering the configuration space (we refer to this property as probabilistic exhaustivity). We then provide the (asymptotic) optimality proof for BFMT*, by showing that the algorithm recovers a solution with cost no greater than that of any tracing path. In the following, let \( C_d \) denote the volume of the unit ball in \( d \)-dimensional Euclidean space. Also, the complement of a probabilistic event \( A \) will be denoted by \( A^c \).

A. Probabilistic exhaustivity

Let \( x : [0, 1] \rightarrow \lambda' \) be a path. Given a set of samples (referred to as waypoints) \( \{y_m\}_{m=1}^{M} \subset \lambda' \), we associate a path \( y : [0, 1] \rightarrow \lambda' \) sequentially connecting the nodes \( y_1, \ldots, y_M \) with line segments. We consider the waypoints \( \{y_m\} \) to \((\epsilon, r)-trace\) the path \( x \) if:

a) \([|y_m - y_{m+1}| \leq r \text{ for all } m]\),
b) the cost of \( y \) is bounded as \( c(y) \leq (1 + \epsilon)c(x) \), and
c) the distance from any point of \( y \) to \( x \) is no more than \( r \), i.e., \( \min_{t \in [0, 1]} \|y(t) - x(t)\| \leq r \) for all \( s \in [0, 1] \).

Theorem 4.1 (Probabilistic exhaustivity): Let \( (\lambda'_f, x_{\text{init}}, x_{\text{goal}}) \) be a path planning problem and let \( x : [0, 1] \rightarrow \lambda' \) be a feasible path. Let \( S = \{x_{\text{init}}, x_{\text{goal}}\} \cup \text{SAMPLEFREE}(n, \epsilon) \), and for fixed \( n \) consider the event \( A_n \) that there exist \( \{y_m\}_{m=1}^{M} \subset S \), \( y_1 = x_{\text{init}}, y_M = x_{\text{goal}} \) which \((\epsilon, r)-trace\) \( x \), where

\[
r_n = 4\sqrt{d} \cdot (1 + \eta)^\frac{d}{2} \left( \frac{\mu(\lambda'_f)}{C_d} \right)^\frac{d}{2} \left( \log n \right)^\frac{d}{2} \]

for a parameter \( \eta \geq 0 \). Then, as \( n \rightarrow \infty \), the probability that \( A_n \) does not occur is asymptotically bounded as \( \Pr[A_n^c] = O \left( n^{-\frac{d}{2}} \log^{-\frac{d}{2}} n \right) \).
B. Asymptotic optimality (AO)

We are now in a position to prove the asymptotic optimality of BFMT\(^*\), which represents the main result of this section. We start with an important lemma, that relates the cost of the path returned by BFMT\(^*\) to that of any feasible path.

**Lemma 4.2 (Bi-directional FMT\(^*\) cost comparison):** Let \( x : [0, 1] \rightarrow X_{\text{free}} \) be a feasible path with strong \( \delta \)-clearance. Consider running BFMT\(^*\) to completion using any choice of expansion and termination criteria with \( n \) samples and a connection radius:

\[
r_n = 4 \sqrt{d} (1 + \eta) \cdot 2 \left( \frac{1}{\delta} \right)^{\frac{1}{2}} \left( \frac{\mu(X_{\text{free}})}{\zeta d} \right)^{\frac{1}{4}} \left( \frac{\log(n)}{n} \right)^{\frac{1}{4}},
\]

for a parameter \( \eta \geq 0 \). Let \( c_n \) denote the cost of the path returned by BFMT\(^*\); then for fixed \( \epsilon > 0 \):

\[
P[c_n > (1 + \epsilon)c(x)] = O \left( n^{-\frac{3}{2}} \log^{-\frac{1}{2}} n \right).\]

**Proof:** We address the case of the "first path" termination criterion, in conjunction with any expansion criterion (that is, the two employed in this paper and otherwise). It is clear that the "best path" termination criterion returns a path at least as good as the first path termination criterion. This is because it returns the best path among a number of choices, one of which is the path of the first path criterion. See Remark 4.3 following this proof for further discussion.

If \( X_{\text{init}} = X_{\text{goal}} \), then BFMT\(^*\) immediately terminates with \( c_n = 0 \), trivially satisfying the claim. Thus we assume that \( X_{\text{init}} \neq X_{\text{goal}} \). Consider \( n \) sufficiently large so that \( r_n \leq \min \left\{ \delta / 2, \epsilon \left| X_{\text{init}} - X_{\text{goal}} \right| \right\} / 2 \), and apply Theorem 4.1 to produce, with probability at least \( 1 - O \left( n^{-\frac{3}{2}} \log^{-\frac{1}{2}} n \right) \), a sequence of waypoints \( \{y_m\}_{m=1}^n \subseteq S \), \( y_1 = X_{\text{init}} \), \( y_M \equiv X_{\text{goal}} \), \( \epsilon / 2 \), \( r_n \)-trace \( x \). We claim that in the event that such \( \{y_m\} \) exists, the BFMT\(^*\) algorithm returns a path with cost upper bounded as \( c_n \leq c(y) + r_n \leq (1 + \epsilon / 2)c(x) + (\epsilon / 2)c(x) = (1 + \epsilon)c(x) \). It is clear that the desired result follows from this claim.

Assume the existence of an \((\epsilon / 2, r_n)\)-tracing \( \{y_m\} \). Let \( B(x, r) \) represent a ball of radius \( r \) centered at a sample \( x \). Note that our upper bound on \( r_n \) implies that \( B(y_m, r_n) \) intersects no obstacles. This follows from our choice of \( r_n \) and the distance bound:

\[
\inf_{a \in X_{\text{ch}} \cup \mathcal{V}} ||y_m - a|| \geq 2r_n - r_n = 2r_n.
\]

This fact, along with \( ||y_m - y_{m+1}|| \leq r_n \) for all \( m \), implies that when a connection is attempted for \( y_m \), both \( y_{m-1} \) and \( y_{m+1} \) will be in the search radius without obstacles within that search radius. Running BFMT\(^*\) to completion generates one cost-to-come tree \( T_i(V_i, E_i, H_i, W_i) \) and one cost-to-go tree \( T_g(V_g, E_g, H_g, W_g) \) rooted at \( x_{\text{init}} \) and \( x_{\text{goal}} \), respectively (the subscripts \( i \) and \( g \) are used to identify the root of a tree without ambiguity). The above discussion ensures that the trees will meet and the algorithm will return a feasible path when it terminates – the path outlined by the waypoints \( \{y_m\} \) disallows the possibility of failure.

For each sample point \( z \in S \), let \( c_i(z) := \text{Cost}(z, T_i) \) denote the cost-to-come of \( z \) from \( x_{\text{init}} \) in \( T_i \), and let \( c_g(z) := \text{Cost}(z, T_g) \) denote the cost-to-go from \( z \) to \( x_{\text{goal}} \) in \( T_g \). If \( z \) is not contained in a tree \( T_k \), \( k \in \{i, g\} \), we set \( c_k(z) = \infty \). When the algorithm terminates, we know there exists a sample point \( z_{\text{meet}} \in V_i \cup V_g \) where the two trees meet; indeed we select the particular meeting point \( z_{\text{meet}} = \arg \min_{z \in V_i \cup V_g} c_i(z) + c_g(z) \). Then \( c_n = c(z_{\text{meet}}) + c_g(z_{\text{meet}}) \). We now note a lemma bounding the costs-to-come of the \( \{y_m\} \), the proof of which may be found as an inductive hypothesis (Eq. 5) in Theorem VI.1 [17].

**Lemma 4.3:** Let \( m \in \{1, \ldots, M\} \). If \( c_i(y_m) < \infty \), then \( c_i(y_m) \leq \sum_{k=1}^M ||y_k - y_{k+1}|| \). Otherwise if \( c_i(y_m) = \infty \), then \( c_g(y_m) \leq \sum_{k=1}^M ||y_k - y_{k+1}|| \).

Similarly if \( c_g(y_m) < \infty \), then \( c_g(y_m) \leq \sum_{k=1}^M ||y_k - y_{k+1}|| \). Otherwise \( c_g(y_m) \leq \sum_{k=1}^M ||y_k - y_{k+1}|| \).

To bound the performance \( c_n \) of BFMT\(^*\), there are two cases to consider. Note in either case we find that \( c_n \leq c(y) + r_n \), thus completing the proof.

**Case 1:** There exists some \( y_m \in \mathcal{V}_i \cap \mathcal{V}_g \).

In this case, \( c_n \leq c_i(z_{\text{meet}}) + c_g(z_{\text{meet}}) \leq c_i(y_m) + c_g(y_m) < \infty \) by our choice of \( z_{\text{meet}} \). Then applying Lemma 4.3 we see that \( c_n \leq c_i(y_m) + c_g(y_m) \leq \sum_{k=1}^M ||y_k - y_{k+1}|| = c(y) \).

**Case 2:** \( \mathcal{V}_i \cap \mathcal{V}_g \cap \{y_m\} = \emptyset \).

Consider \( \tilde{m} = \max \left\{ m \mid c_i(y_m) < \infty \right\} \). Then \( y_{\tilde{m}} \in \mathcal{V}_i \) and \( y_{\tilde{m}} \) cannot have been the minimum cost element of \( H_i \) at any point during algorithm execution or else we would have connected \( y_{\tilde{m}+1} \in \mathcal{V}_i \). Let \( z \) denote the minimum cost element of \( H_i \) when \( z_{\text{meet}} \) was added to \( \mathcal{V}_i \). We have the bound:

\[
c_i(z_{\text{meet}}) + c_g(y_m) \leq c_i(z) + r_n \leq c_i(y_m) + r_n \leq \sum_{k=1}^{m-1} ||y_k - y_{k+1}|| + r_n. \tag{1}
\]

By our assumption for this case, \( y_{\tilde{m}} \notin \mathcal{V}_i \). Then by Lemma 4.3 we know that \( c(y_m) \leq \sum_{k=1}^{m-1} ||y_k - y_{k+1}|| \). Combining with the previous inequality yields \( c_n \leq c_i(z_{\text{meet}}) + c_g(z_{\text{meet}}) \leq \sum_{k=1}^{m-1} ||y_k - y_{k+1}|| + r_n = c(y) + r_n \).

**Remark 4.4 (Tightened bound for connection radius):** As discussed in [17], for the sake of clarity the spatial constant \( 4\sqrt{d} \) in the expression for \( r_n \) is greater than is necessary for Theorem 4.1 to hold. A more careful argument along the lines of the original FMT\(^*\) AO proof [7] would suffice to show that \( r_n = (1 + \eta) \cdot 2 \left( \frac{1}{\delta} \right)^{\frac{1}{2}} \left( \frac{\mu(X_{\text{free}})}{\zeta d} \right)^{\frac{1}{4}} \left( \frac{\log(n)}{n} \right)^{\frac{1}{4}} \) satisfies the theorem as well.

**Remark 4.5 (Alternative termination criteria):** If we analyze the "best path" criterion instead of the "first path" criterion (i.e., if BFMT\(^*\) continues until an element of \( \mathcal{V}_i \cap \mathcal{V}_g \) is selected as the minimum cost element of \( H_i \) or \( H_g \)), then the \( r_n \) term may be omitted from inequality (1). In that case we need only consider \( n \) sufficiently large so that \( r_n \leq \delta / 2 \).

We now have everything we need to show that BFMT\(^*\) is asymptotically-optimal. The next theorem defines this formally.

**Theorem 4.6 (BFMT\(^*\) asymptotic optimality):** Assume a \( \delta-\)
robustly feasible path planning problem as defined in Section \[1\] with optimal path \( \sigma^* \) of cost \( c^* \). Then BFMT\(^*\) converges in probability to \( \sigma^* \) as the number of samples \( n \to \infty \). Specifically, for any \( \epsilon > 0 \),

\[
\lim_{n \to \infty} \Pr[c_n > (1 + \epsilon)c^*] = 0
\]

**Proof:** The proof follows as a corollary to Lemma 4.2.

By our \( \delta \)-robustly feasible assumption, we can find a strong \( \delta \)-clearence feasible path \( x \mid [0, 1] \to \mathcal{X}_{\text{free}} \) that approximates \( \sigma^* \) with cost \( c(x) < (1 + \epsilon/3)c^* \) (i.e., less than factor \( \epsilon/3 \) from \( c^* \)), for any \( \epsilon > 0 \). By Lemma 4.2, we can choose \( n \) sufficiently large such that BFMT\(^*\) returns an \( \epsilon/3 \) cost approximation to the approximant:

\[
\Pr[c_n > (1 + \epsilon/3)^2c^*] < \Pr[c_n > (1 + \epsilon/3)c(x)] < O\left(n^{-\frac{2}{3}} \log^{-\frac{1}{3}} n\right)
\]

To approach the optimal path, let the number of samples \( n \to \infty \). It follows that, for any \( \eta \geq 0 \):

\[
\lim_{n \to \infty} \Pr[c_n > (1 + \epsilon/3)^2c^*] < \lim_{n \to \infty} O\left(n^{-\frac{2}{3}} \log^{-\frac{1}{3}} n\right) = 0
\]

Now we relate this to the original claim. First suppose that \( \epsilon \leq 3 \). From \((1 + \epsilon/3)^2 < 1 + \epsilon\), the event \( \{c_n > (1 + \epsilon)c^*\} \) is a subset of the event \( \{c_n > (1 + \epsilon/3)c^*\} \), hence:

\[
\lim_{n \to \infty} \Pr[c_n > (1 + \epsilon)c^*] \leq \lim_{n \to \infty} \Pr[c_n > (1 + \epsilon/3)c^*] = 0.
\]

Because the probability is monotone-decreasing in \( \epsilon \) as \( \epsilon \) increases, the statement holds for all \( \epsilon > 3 \) as well (to see this, apply Lemma 4.2 again for \( m \) sufficiently large to handle \( \epsilon = 3 \); then by similar argument as above \( \Pr[c_m > (1 + \epsilon)c^*] < \Pr[c_n > (1 + \epsilon)c^*] = O\left(m^{-\frac{2}{3}} \log^{-\frac{1}{3}} m\right) \) and take the limit as \( m \to \infty \). Hence \( \lim_{n \to \infty} \Pr[c_n > (1 + \epsilon)c^*] = 0 \) holds for arbitrary \( \epsilon \), and we see that BFMT\(^*\) converges in probability to the optimal path, as claimed.

**Remark 4.7 (Convergence rate):** Note that we can also translate the convergence rate from Lemma 4.2 to the setup of Theorem 4.6 which doesn’t require strong \( \delta \)-clearence. For any \( \epsilon > 0 \), the optimal path can be approximated by a strong- \( \delta \)-cleare path with cost less than \((1 + \epsilon)c(x)\) and we can focus on approximating that path to high-enough precision to still approximate the optimal path to within \((1 + \epsilon)\). Since the convergence rate in Lemma 4.2 only contains \( \epsilon \) in the rate’s constant, the big-O convergence rate remains the same.

**C. Discussion**

It is worth mentioning a few variations on the vanilla version of BFMT\(^*\). First, if one has prior information about areas that the optimal path is unlikely to pass through, one may consider a non-uniform sampling strategy that downsamples these regions. As long as the sampling density is lower-bounded by a positive number over the configuration space, BFMT\(^*\) can be slightly altered (by just increasing \( r_n \) by a constant factor) to ensure it stays AO. The argument is analogous to that made in [8], and essentially proceeds by making the search radius wide enough to balance out the detrimental effect of the lower sampling density (in some areas). Another common problem variation is when the cost is not arc-length, but a different metric or a line integral cost. In either case, BFMT\(^*\) need only consider cost balls instead of Euclidean balls when making connections. Details for adjusting the algorithm and why the AO proof still holds can be derived from [8]. The argument basically shows that the triangle inequality either holds exactly (for metric costs) or approximately, and that this approximation goes away in the limit as \( n \to \infty \). Finally, note that BFMT\(^*\), like FMT\(^*\), is not an anytime algorithm. However, the algorithm can be adjusted analogously to FMT\(^*\) as in [18] in order to make it anytime. Essentially, batches of samples are repeatedly added to the configuration space and connected to the tree until time runs out, with previous computations carefully reused.

**V. Simulations**

In this section, we provide numerical experiments that compare the performance of BFMT\(^*\) with other sampling-based, asymptotically-optimal planning algorithms (namely, FMT\(^*\), RRT\(^*\), and PRM\(^*\)) and existing sampling-based, bi-directional algorithms (namely, RRT-Connect and SBL). To test them, we would like to observe the quality of the solution returned as a function of the execution time allotted to the algorithm. As a basis for quality comparison between incremental or “anytime” planners (such as RRT\(^*\)) and non-incremental planners (such as BFMT\(^*\), which generate solutions via sample batches), we use as our independent variable the number of samples drawn by the planners during the planning process (which in essence serves as a proxy to execution time). Note sample count has a different connotation depending on the planner that will not necessarily be the number of nodes stored in the constructed solution graph – for RRT\(^*\) (with one sample drawn per iteration), this is the number of iterations, while for FMT\(^*\), PRM\(^*\), and BFMT\(^*\), this is the number of free space samples taken.

In the first experiments, we investigate the relative benefits of the four versions of BFMT\(^*\). Then, in the second set, we compare the version of BFMT\(^*\) that appears to perform best (namely, BFMT\(^*\) with the “alternating trees” criterion for expansion and “best path” criterion for termination) with the other planners. To generate simulation data for a given experiment, we queried the planning algorithms once each for a series of sample counts, recording the cost of the solution returned, the planner execution time and whether the planner succeeded or not, then repeated this process over 50 trials. To ensure a fair comparison, each planning algorithm was tested using the Open Motion Planning Library (OMPL) v0.15.0 [19], which provides high-quality implementations of many state-of-the-art planners and a common framework for executing motion plans. In this way, we could ensure that all algorithms employed the exact same primitive routines (e.g., nearest-neighbor search, collision-checking, data handling, etc.), and measure their performances fairly. Regarding implementation, BFMT\(^*\), FMT\(^*\), RRT\(^*\), and PRM\(^*\) used \( \eta = 0 \) from Lemma 4.2 for the nearest-neighbor radius \( r_n \) in order to satisfy the theoretical bounds provided in Section [IV] and [6]. For RRT-Connect, SBL, and RRT\(^*\), we used the default OMPL settings; namely, a steering parameter equal to 20% of the maximum extent of the configuration space, and for RRT\(^*\) a 5% goal bias.

\(^3\)Code for all experiments was written in C++. Corresponding programs were compiled and run on Linux-operated PC, clocked at 2.4 GHz and equipped with 7.5 GB of RAM.
Before proceeding, note that each marker shown on the plots throughout this section represents a single simulation at a fixed sample count. The points on the curves, however, represent the mean cost/time of successful algorithm runs only for a particular sample count, with error bars corresponding to one standard deviation of the 50 run sample mean.

A. Comparison amongst BFMT∗ versions

We first test the four versions of BFMT∗ against a standard path planning problem: a point mass robot moving in a cluttered unit hypercube. The variants permute the termination criterion (whether at the first or best path possible) as well as the expansion criterion (whether swapping trees every time or balancing tree reachable volumes) as outlined in Section III.

Fig. 2: Cost vs. time plots for the four versions of BFMT∗. BFMT∗ with “best path” termination and “alternating trees” expansion seems to represent the best trade-off in exploration vs. exploitation.

Collectively, Fig. 2 shows four facts: (i) all variants of BFMT∗ tend to perform fairly similarly, (ii) the main discriminant in performance is the termination criterion (as curves corresponding to the same expansion criterion are almost overlapping), (iii) the “best path” termination criterion appears to be the best choice fairly uniformly, and (iv) for a given termination criterion, the swapping expansion criterion appears to yield slightly better performance. Fact (iv) exemplifies a classic trade-off in exploration versus exploitation. To clarify, note that “balanced” growth here is analogous to two-source breadth-first search in that the wavefront node of lowest-cost within either tree is chosen for expansion at the next iteration, effectively requiring that both wavefronts fully explore the current cost-curve level set before progressing. These wavefronts, then, spread out equally (which, in the Euclidean distance case, means radially from their sources) at identical rates, with one tree waiting for the other to finish its level set before continuing. This can be a hindrance in cases where one tree has less surrounding free-space to cover than the other – exactly as we see here given a goal located in a corner rather than near the hypercube center. Conversely, when iteratively swapping between trees, wavefronts now explore the level sets of the cost-to-come and cost-to-go curves independently and at “asynchronous” rates; if one tree finishes a level set sooner, it continues on to the next without waiting (a particularly useful quality in cluttered spaces). In light of this and the results in Fig. 2 we selected BFMT∗ with “best path” termination and “alternating trees” for the subsequent experiments.

B. Comparison with the State-of-the-Art

Here we present benchmarking results (average solution cost vs. average execution times and success rates) comparing the best of the four BFMT∗ variants, herein simply denoted as BFMT∗, to other state-of-the-art sampling-based planners.

Three benchmarking test scenarios were considered: (1) a 2D “bug trap” and (2) a 2D “maze” problem for a convex polyhedral robot in the $\mathbb{SE}(2)$ configuration space, as well as (3) a more challenging 3D problem called the “$\alpha$-puzzle” in which we seek to untangle two loops of metal (non-convex) in the $\mathbb{SE}(3)$ configuration space. All problems were drawn directly from OMPL’s bank of tests, and are illustrated in Fig. 3. In each case, collision-checks called upon OMPL’s built-in collision-checking library, FCL.

Fig. 3: Depictions of the three OMPL rigid-body planning problems (a) SE(2) bug trap (b) SE(2) maze (c) SE(3) $\alpha$-puzzle

Figure 4 shows the results for each BFMT∗, FMT∗, RRT∗, and PRM∗ (average costs for RRT-Connect and SBL were roughly 2-4x greater, which occluded the details of other curves; they were thus omitted for clarity). The plots reveal in each case that BFMT∗ tends to an optimal cost at least as fast as the other planners, and sometimes much faster. BFMT∗ noticeably outperforms RRT∗ (despite its goal biasing) as well as PRM∗. Its performance relative to FMT∗ seems somewhat more complex, however.

To compare them in detail, we test their performance as a function of configuration space dimension using the (scalable) unit hypercube test case. Results for 5D and 10D are shown in Fig. 5. Here BFMT∗ substantially outperforms FMT∗, particularly as dimension increases. This suggests that reachable volumes play a significant role in their execution time. The relatively small volume of reachable configurations around the goal at the corner implies that the reverse tree of BFMT∗ expands its wavefront through many fewer states than the forward tree of FMT∗ (which in fact needlessly expands towards the zero-vector); tree interconnection in the bi-directional case prevents its forward tree from growing too large compared to unidirectional search. This is pronounced exponentially as the dimension increases. In trap or maze-like scenarios, however, bi-directionality does not seem to change significantly the number of states explored by the marching trees, leading to comparable performance for the $\mathbb{SE}(2)$ bug-trap and maze. Fortunately, it does offer slight improvements to success rate over FMT∗, and often the other planners as well. Note we expect a greater contrast in execution times as the cost of collision-checking increases, such as with many non-convex obstacles or in time-varying environments.

---

4 Standard deviation of the mean indicates where we expect 1-σ confidence the mean to lie if we ran another set of 50 runs, and is related to the standard deviation of the distribution by $\sigma_n = \sigma/\sqrt{n}$.

5 We populated the space to 50% obstacle coverage with randomly-sized, axis-oriented hyperrectangles, $x_{\text{init}}$ was set to the goal at $[0.5, \ldots, 0.5]$, with the goal $x_{\text{goal}}$ at the one-vector (i.e., $[1, \ldots, 1]$). Collision-checking called upon a two-phase tiling-based hashing scheme.
Importantly, we note that BFMT* misses out on one of the usual advantages of bi-directionality: the improved ability to escape bug traps via two-sided approach. By the nature of BFMT*’s growth, if a forward tree cannot escape the trap, then adding a backward tree will not help. A future area of study is to consider possible remedies to this; for instance, by adaptively-sampling and regrowing near failing wavefronts, it may be possible to quickly salvage a failed run. In this case, a bi-directional approach would be quite advantageous.

Future efforts will study extension of BFMT* to dynamic environments through lazy re-evaluation (leveraging its tree-like forward and reverse path structures) in a way that reuses previous results as much as possible. Maintaining bounds on run-time performance and solution quality in time-varying environments will remain the greatest challenges. Ultimately, we hope that BFMT* will enable fast, easy-to-implement re-planning with proven performance guarantees, analogous to planning in static environments as we have shown here.

Fig. 4: Simulation results for the three OMPL scenarios. The jump in (e) is from two homotopies for the α-puzzle; note that BFMT* finds the cheaper homotopy faster and more often than other planners.

Fig. 5: FMT* and BFMT* comparison for 5D and 10D cluttered hypercubes (50% coverage; all success rates were near 100%).

VI. CONCLUSION

In this paper, we presented a bi-directional, sampling-based, asymptotically-optimal motion planning algorithm named BFMT*, for which we rigorously proved its optimality and characterized its convergence rate – arguably first in the field of bidirectional sampling-based planning. Numerical experiments in $\mathbb{R}^d$, SE(2), and SE(3) revealed that BFMT* tends to an optimal solution at least as fast as its state-of-the-art counterparts, and in some cases significantly faster.

REFERENCES


