Advanced BIT* (ABIT*): Sampling-Based Planning with Advanced Graph-Search Techniques

Marlin P. Strub[1] and Jonathan D. Gammell[1]

***Abstract*— Path planning is an active area of research essential for many applications in robotics. Popular techniques include graph-based searches and sampling-based planners. These approaches are powerful but have limitations.**

This paper continues work to combine their strengths and mitigate their limitations using a unified planning paradigm. It does this by viewing the path planning problem as the two subproblems of search and approximation and using advanced graph-search techniques on a sampling-based approximation.

This perspective leads to Advanced BIT*. ABIT* combines truncated anytime graph-based searches, such as ATD*, with anytime almost-surely asymptotically optimal sampling-based planners, such as RRT*. This allows it to quickly find initial solutions and then converge towards the optimum in an anytime manner. ABIT* outperforms existing single-query, sampling-based planners on the tested problems in $\mathbb{R}^4$ and $\mathbb{R}^8$, and was demonstrated on real-world problems with NASA/JPL-Caltech.

I. Introduction

Popular path planning algorithms in robotics include graph-based searches, such as A* [1] and Dijkstra's [2], and sampling-based planners, such as Rapidly-exploring Random Trees (RRT) [3] and Probabilistic Roadmaps (PRM) [4]. Both graph- and sampling-based approaches have characteristic strengths and limitations. Previous work [5, 6] has separated search and approximation in single-query, almost-surely asymptotically optimal sampling-based planning to combine their strengths and mitigate their limitations. This separation can be leveraged to use advanced graph-based search techniques on an anytime sampling-based approximation to further improve performance.

An important strength of graph-based approaches is their strong theoretical guarantees. A* is *resolution-optimal* (and *resolution-complete*) as well as *optimally efficient*. Any other algorithm guaranteed to find the optimal solution must expand at least as many vertices as A*, given the same problem information [1]. This efficiency is achieved by always expanding the state with the highest potential solution quality but requires an ordering of the search that does not provide any solutions until the optimum is found.

Anytime graph-search algorithms sacrifice the efficiency of A* to find intermediate solutions faster. Anytime Repairing A* (ARA*) [7] finds an initial, potentially suboptimal solution quickly and then uses any remaining computational time to improve it. ARA* does this without duplicated search effort by starting with a heuristic that is inflated and then gradually decreasing this inflation while accounting for changes to state connections (i.e., inconsistent vertices).

[1]M. P. Strub and J. D. Gammell are with the Estimation, Search, and Planning (ESP) Group of the Oxford Robotics Institute (ORI), University of Oxford, United Kingdom. (mstrub|gammell)@robots.ox.ac.uk



Fig. 1: ABIT* on NASA/JPL-Caltech's Axel Rover System [8] during a week-long field test in the Mojave Desert, California, USA (Section V-B).

These graph-based algorithms cannot be applied directly to planning problems with continuously valued state spaces and selecting appropriate *a priori* discretizations is difficult. Coarse resolutions are computationally inexpensive to search but may yield paths of poor quality in a continuous context. Fine resolutions contain paths of higher quality [9] but the associated computational cost becomes prohibitively expensive in high dimensions (i.e., *the curse of dimensionality* [10]).

Sampling-based planners sample states incrementally to avoid picking a resolution *a priori*. Single-query planners such as RRT simultaneously build and search an anytime approximation that improves with computational time. This incremental sampling makes the search dependent on the sampling order and results in a randomly ordered search.

Ordering the search of an incremental planner in a solution-oriented manner requires either ordering the approximation or separating the search from the approximation. Batch Informed Trees (BIT*) [5, 6] achieves the second by sampling batches of states and viewing them as an increasingly dense implicit random geometric graph (RGG) [11]. BIT* uses incremental search techniques, similar to Lifelong Planning A* (LPA*) [12], to process the sampled states in order of potential solution quality.

This paper extends BIT* to further leverage the separation of search and approximation in sampling-based planning. Advanced BIT* (ABIT*) uses advanced graph-search techniques, such as inflation and truncation, to balance exploring and exploiting an increasingly dense RGG approximation. ABIT* is almost-surely asymptotically optimal and outperforms existing single-query, almost-surely asymptotically optimal planners on random and artificially designed problems, especially in high dimensions. The benefits of ABIT* were shown on real-world problems on NASA/JPL-Caltech's Axel (Fig. 1) during a week-long field test in the Mojave Desert testing autonomous navigation on challenging terrain.

II. Background

Multiquery algorithms, such as PRM and PRM* [13], separate the approximation and the search in two phases. These algorithms first sample the entire collision-free state space to approximate it with a graph which is then searched to solve individual problems. This approximation is computationally expensive but justified by its reuse.

Lazy-PRM [14] reduces the approximation cost by initially assuming that all vertices and edges are collision-free. A planning problem is then solved by searching this graph and checking the resulting path for collisions. If it contains collisions then the corresponding vertices and edges are removed and a new search is started on the updated graph.

Single-query problems only require an approximation of the state space relevant to the query. Expansive Space Trees (EST) [15] avoids approximating the entire state space by incrementally growing trees from the start and goal. This focuses the approximation to the relevant region of the state space but couples the search with the approximation and results in a randomly ordered search.

RRT grows a single tree from the start to incrementally approximate the state space. It rapidly expands this tree by biasing its growth towards unexplored regions of the space (i.e., with *Voronoi bias*). The approximation and search are still coupled by the incremental sampling which also results in a randomly ordered search.

RRT-Connect [16] grows trees from the start and goal and biases their growth both towards the unexplored space and each other. The greedy tree connection often results in fast initial solution times but the search of the unexplored space remains randomly ordered by the incremental sampling.

These algorithms provide no solution quality guarantee. RRT* [13] extends RRT to provide *almost-surely asymptotically optimal* solutions. Its solutions converge to the optimum with probability one given infinite computational time by locally rewiring the tree to ensure locally optimal connections. This search for the optimal local connection is performed immediately after each new state is sampled and the global search of the state space remains random.

Improving the convergence rate of RRT* is an active area of research. Informed RRT* [17] and RRT*-Smart [18] are based on different sampling strategies. RRT$^{\#}$ [19] modifies the rewiring procedure to ensure globally optimal connections. Lower Bound Tree-RRT (LBT-RRT) [20] is a near-optimal variant that allows for continuous interpolation between RRT and RRT* by maintaining a tree which stores a lower bound on the cost-to-come to all vertices. These extensions modify how states are used but still process them individually and maintain the random search order of RRT*.

Approximation and search can be separated by processing batches of multiple states. Fast Marching Tree (FMT*) [21] samples a user-specified number of states and uses a Fast Marching Method [22] to grow a tree that connects them in order of increasing cost-to-come. FMT* is not anytime and must be restarted from scratch if no suitable solution is found with the specified number of samples.

BIT* samples multiple batches of states and views the resulting approximation as an increasingly dense edge-implicit RGG. It performs a search ordered on the potential solution quality of the implicit edges of this graph. It efficiently reuses previous search efforts by incorporating techniques from incremental graph-search approaches, similar to LPA*. BIT* is anytime but uses a nonanytime graph-search that only returns a single solution per approximation. This exploits the current approximation without considering that it will be updated once the search is finished, which may be inefficient.

Fast-BIT* [23] is a variation of BIT* that finds initial solutions faster by ordering the initial search solely on cost-to-go. Once a solution is found, it reorders its queues with a full-solution heuristic and reprocesses all vertices. This maintains almost-sure asymptotic optimality but duplicates previous search effort unnecessarily. Fast-BIT* then performs the same incremental search as BIT* to fully exploit each subsequent approximation.

Unlike the multiquery planners, ABIT* builds a problem-specific approximation of the state space. Unlike EST and the RRT-based approaches, ABIT* separates approximation and search in a way that allows it to process states in order of (inflated) potential solution cost. Unlike FMT*, ABIT* improves its approximation in an anytime fashion. Unlike BIT* and Fast-BIT*, ABIT* avoids wasting computational effort to find a resolution-optimal path in an inaccurate approximation that will change.

III. Advanced Batch Informed Trees (ABIT*)

BIT* is an anytime, single-query planner that almost-surely asymptotically finds an optimal solution to a (continuously valued) planning problem. It focuses its approximation of the state space to the region that can improve the current solution using informed sampling [17]. This approximation is separated from the search by sampling batches of states and viewing them as an increasingly dense edge-implicit RGG.

This perspective enables BIT* to perform a series of informed graph-searches in which RGG edges are processed in order of their potential solution quality. This is achieved by sorting an edge queue according to the sum of the current cost-to-come from the start to the edge's parent state, an estimate of the edge cost, and an estimate of the cost-to-go from the edge's child state to a goal. BIT* performs these searches efficiently by reusing information from both previous searches and approximations similar to incremental search algorithms, e.g. LPA*. Full details are in [6, 24].

BIT*'s separation of approximation and search provides a direction for better single-query, almost-surely asymptotically optimal planning algorithms. ABIT* builds on this separation 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 ABIT*'s edge queue. This sacrifices

resolution-optimality but achieves faster initial solution times compared to the incremental search used by BIT*. ABIT*'s initial (potentially suboptimal) solution is subsequently repaired without duplicating search effort by tracking inconsistent states, similar to ARA*. A state is considered inconsistent if its cost-to-come has decreased since its outgoing edges were last inserted into the queue.

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

A. Notation

The state space of the 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 in the tree are associated with valid states and edges in the tree represent valid connections between states. An edge consists of a parent state, $\mathbf{x}_{\text{p}}$, and a child state, $\mathbf{x}_{\text{c}}$, and is denoted as $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$. The set of inconsistent states is denoted by $V_{\text{inconsistent}}$ and the states that are not in the tree make up the set $X_{\text{unconnected}}$.

Let $\mathbb{R}^{\infty}_{\geq 0}$ denote the union of all nonnegative real numbers with infinity. The function $\widehat{g}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents an admissible estimate (i.e., a lower bound) of the cost-to-come from the start to a state, $\mathbf{x} \in X$. The function $g_{\mathcal{T}}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents the cost-to-come from the start to a state, $\mathbf{x} \in X$, through the current search tree, $\mathcal{T}$. This cost is taken to be infinite for any state with no associated vertex in the tree.

The function $\widehat{h}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents an admissible estimate of the cost-to-go from a state to a goal. The function $\widehat{f}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents an admissible estimate of the cost of a path from the start to a goal constrained to go through a state, e.g., $\widehat{f}(\mathbf{x}) \coloneqq \widehat{g}(\mathbf{x}) + \widehat{h}(\mathbf{x})$. This estimate defines the informed set of states that could provide a better solution, $X_{\widehat{f}} \coloneqq \{\mathbf{x} \in X \mid \widehat{f}(\mathbf{x}) \leq c_{\text{current}}\}$, where c_{current} is the current solution cost [17]. The function $c\colon X \times X \to \mathbb{R}^{\infty}_{\geq 0}$ denotes the true edge cost between two states. The function $\widehat{c}\colon X \times X \to \mathbb{R}^{\infty}_{\geq 0}$ is an admissible estimate of this cost.

Let A be any set and let B and C be subsets of A, i.e., $B, C \subseteq A$. The notation $B \xleftarrow{+} C$ is used for $B \leftarrow B \cup C$ and $B \xleftarrow{-} C$ for $B \leftarrow B \setminus C$. The cardinality of a set is denoted by $|\cdot|$ and the minimum of an empty set is taken to be infinity. The Lebesgue measure of a set is denoted by $\lambda(\cdot)$, and the Lebesgue measure of an n-dimensional unit ball by ζ_n. The number of states per batch is denoted by m.

B. Initialization (Algorithm 1, Lines 1-4)

ABIT* starts by initializing the search tree with the start state as its root. The set of unconnected states initially only contains the goal states (line 1). The inflation factor, $\varepsilon_{\text{infl}} \geq 1$, and the truncation factor, $\varepsilon_{\text{trunc}} \geq 1$, are initialized to be infinitely large (line 2). The edge-queue, $\mathcal{Q}$, holds all edges from the start state to the goal states (line 4).

Algorithm 1: $\texttt{ABIT}^*(\mathbf{x}_{\text{start}}, X_{\text{goal}}, m)$

1 $V \leftarrow \{\mathbf{x}_{\text{start}}\}$; $E \leftarrow \emptyset$; $\mathcal{T} \leftarrow (V, E)$; $X_{\text{unconnected}} \leftarrow X_{\text{goal}}$
2 $q \leftarrow |V| + |X_{\text{unconnected}}|$; $\varepsilon_{\text{infl}} \leftarrow \infty$; $\varepsilon_{\text{trunc}} \leftarrow \infty$
3 $V_{\text{closed}} \leftarrow \emptyset$; $V_{\text{inconsistent}} \leftarrow \emptyset$
4 $\mathcal{Q} \leftarrow \texttt{expand}(\{\mathbf{x}_{\text{start}}\}, \mathcal{T}, X_{\text{unconnected}}, \infty)$
5 **repeat**
6 **if** `is_search_marked_finished`()
7 **if** `update_approximation`$(\varepsilon_{\text{infl}}, \varepsilon_{\text{trunc}})$
8 `prune`$(\mathcal{T}, X_{\text{unconnected}}, X_{\text{goal}})$
9 $X_{\text{unconnected}} \xleftarrow{+} \texttt{sample}(m, X_{\text{goal}})$
10 $q \leftarrow |V| + |X_{\text{unconnected}}|$
11 $\mathcal{Q} \leftarrow \texttt{expand}(\{\mathbf{x}_{\text{start}}\}, \mathcal{T}, X_{\text{unconnected}}, r(q))$
12 **else**
13 $\mathcal{Q} \xleftarrow{+} \texttt{expand}(V_{\text{inconsistent}}, \mathcal{T}, X_{\text{unconnected}}, r(q))$
14 $\varepsilon_{\text{infl}} \leftarrow$ `update_inflation_factor`()
15 $\varepsilon_{\text{trunc}} \leftarrow$ `update_truncation_factor`()
16 $V_{\text{closed}} \leftarrow \emptyset$; $V_{\text{inconsistent}} \leftarrow \emptyset$
17 `mark_search_unfinished`()
18 **else**
19 $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) \leftarrow \underset{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{Q}}{\arg\min} \left\{ g_{\mathcal{T}}(\mathbf{x}_i) + \widehat{c}(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon_{\text{infl}} \widehat{h}(\mathbf{x}_j) \right\}$
20 $\mathcal{Q} \xleftarrow{-} (\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$
21 **if** $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) \in E$
22 **if** $\mathbf{x}_{\text{c}} \in V_{\text{closed}}$
23 $V_{\text{inconsistent}} \xleftarrow{+} \mathbf{x}_{\text{c}}$
24 **else**
25 $\mathcal{Q} \xleftarrow{+} \texttt{expand}(\{\mathbf{x}_{\text{c}}\}, \mathcal{T}, X_{\text{unconnected}}, r(q))$
26 $V_{\text{closed}} \xleftarrow{+} \mathbf{x}_{\text{c}}$
27 **else if** $\varepsilon_{\text{trunc}} \left(g_{\mathcal{T}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) + \widehat{h}(\mathbf{x}_{\text{c}}) \right) \leq \min_{\mathbf{x} \in X_{\text{goal}}} \{ g_{\mathcal{T}}(\mathbf{x}) \}$
28 **if** $g_{\mathcal{T}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) < g_{\mathcal{T}}(\mathbf{x}_{\text{c}})$
29 **if** $g_{\mathcal{T}}(\mathbf{x}_{\text{p}}) + c(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) + \widehat{h}(\mathbf{x}_{\text{c}}) \leq \min_{\mathbf{x} \in X_{\text{goal}}} \{ g_{\mathcal{T}}(\mathbf{x}) \}$
30 **if** $g_{\mathcal{T}}(\mathbf{x}_{\text{p}}) + c(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) < g_{\mathcal{T}}(\mathbf{x}_{\text{c}})$
31 **if** $\mathbf{x}_{\text{c}} \in V$
32 $E \xleftarrow{-} \{(\mathbf{x}_{\text{prev}}, \mathbf{x}_{\text{c}}) \in E\}$
33 **else**
34 $X_{\text{unconnected}} \xleftarrow{-} \mathbf{x}_{\text{c}}$
35 $V \xleftarrow{+} \mathbf{x}_{\text{c}}$
36 $E \xleftarrow{+} (\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$
37 **if** $\mathbf{x}_{\text{c}} \in V_{\text{closed}}$
38 $V_{\text{inconsistent}} \xleftarrow{+} \mathbf{x}_{\text{c}}$
39 **else**
40 $\mathcal{Q} \xleftarrow{+} \texttt{expand}(\{\mathbf{x}_{\text{c}}\}, \mathcal{T}, X_{\text{unconnected}}, r(q))$
41 $V_{\text{closed}} \xleftarrow{+} \mathbf{x}_{\text{c}}$
42 **else** `mark_search_finished`()
43 **until** `stop`

C. Approximation (Algorithm 1, Lines 7-13)

ABIT* uses informed sampling [17] to focus its RGG approximation on the relevant region of the state space. The accuracy of this approximation increases with the number of sampled states but so does its complexity. This complexity is reduced by pruning states that cannot improve the current solution (line 8 and Alg. 3) and shrinking the connection radius as more states are sampled. The radius, r, is updated as in [13], using the measure of the informed set, as in [17],

$$r(q) \coloneqq \eta \left(2 \left(1 + \frac{1}{n} \right) \left(\frac{\lambda\left(X_{\widehat{f}}\right)}{\zeta_n} \right) \left(\frac{\log(q)}{q} \right) \right)^{\frac{1}{n}},$$

Algorithm 2: $\texttt{expand}(\{\mathbf{x}_i\}, \mathcal{T}, X_{\text{unconnected}}, r)$

```
1 E_out ← ∅
2 forall x_p in {x_i} do
3     E_out +← {(x_p, x_c) ∈ E}
4     forall x_c in {x ∈ {X_unconnected ∪ V} | ‖x_p − x‖ ≤ r} do
5         if ĝ(x_p) + ĉ(x_p, x_c) + ĥ(x_c) ≤ min_{x∈X_goal} {g_T(x)}
6             if ĝ(x_p) + ĉ(x_p, x_c) ≤ g_T(x_c)
7                 E_out +← (x_p, x_c)
8 return E_out
```

Algorithm 3: $\texttt{prune}\,(\mathcal{T}, X_{\text{unconnected}}, X_{\text{goal}})$

```
1 X_unconnected ← {x ∈ X_unconnected | f̂(x) ≥ min_{x∈X_goal} {g_T(x)}}
2 V ← {x ∈ V | f̂(x) ≥ min_{x∈X_goal} {g_T(x)}}
3 E ← {(x_p, x_c) ∈ E | f̂(x_p) ≥ min_{x∈X_goal} {g_T(x)} or f̂(x_c) >
    min_{x∈X_goal} {g_T(x)}}
4 X_unconnected +← {x_c ∈ V | ∄ x_p ∈ V s. t. (x_p, x_c) ∈ E}
5 V ← {x_c ∈ V | ∄ x_p ∈ V s. t. (x_p, x_c) ∈ E}
```

where q is the number of sampled states in the informed set, $\eta > 1$ is a tuning parameter, and n is the state space dimension. Faster-decreasing radii are provided in [21, 25] but are not used in this paper to isolate the reasons for ABIT*'s improved performance relative to existing algorithms.

D. Search (Algorithm 1, Lines 19-41)

ABIT* delays expensive computation of true edge cost (e.g., collision checks) with a lazy search similar to an edge-queue version of Anytime Truncated D* (ATD*) [26]. This queue is ordered lexicographically by (inflated) potential solution cost and then cost-to-come. A search iteration starts by removing the edge with the lowest queue value from the queue (line 19). If this edge is part of the search tree, then the child state is expanded (i.e., its outgoing edges are added to the queue) if it has not already been expanded during the current search (lines 22–26 and Alg. 2). ABIT* otherwise checks if the new edge can possibly contribute to a solution better than the current one (lines 27–30).

An edge that passes these checks improves the cost-to-come of the child state and possibly the current solution. If the child state is already part of the tree, adding this edge constitutes a rewiring (line 32). Otherwise, this state is removed from the set of unconnected states (line 34) and added to the search tree (line 35). In both cases, the edge is added to the search tree (line 36).

After adding an edge, the child state is expanded unless it has already been expanded during the current search, in which case it is added to the set of inconsistent vertices (lines 37–41).

E. Approximation, Inflation, and Truncation Update Policies

The approximation is updated when a desired bound on the resolution optimality is achieved, which depends on the inflation and truncation factors (line 7). These factors are updated after each search of the current RGG (lines 14 and 15). A high inflation factor biases the search towards the goal and decreases solution times but results in loose bounds on the solution quality. A low inflation factor results in a search that requires more computational effort to complete but achieves tighter bounds on the solution quality. A high truncation factor promotes exploration of the region of the state space that could potentially contain better solutions by truncating the search once a loose bound on the solution quality is achieved, which facilitates adding more samples. A low truncation factor promotes exploiting the current approximation of the state space as the search is not truncated until a tight bound on the solution quality is guaranteed.

The update policies of these two factors are user-tuned parameters that balance exploiting the current RGG with exploring the state space. Section V presents the specific policies used for the experimental results.

IV. Formal Analysis

This paper uses Definition 24 in [13] as the definition of almost-sure asymptotic optimality. Note that any sampling-based planner is almost-surely asymptotically optimal if (i) its underlying graph almost-surely contains an asymptotically optimal path, and (ii) its underlying graph-search is asymptotically resolution-optimal. These conditions are sufficient but not necessary.

A. Almost-Sure Existence of an Asymptotically Optimal Path

ABIT* uses the same increasingly dense RGG approximation as BIT*. Since BIT* is an almost-surely asymptotically optimal algorithm [6], this approximation must almost-surely contain an asymptotically optimal path.

B. Asymptotically Resolution-Optimal Search

Theorem 1 states that ABIT*'s search processes at least all of the edges processed by ATD*, which is an anytime, incremental search algorithm that finds a solution within $\varepsilon_{\text{trunc}}\varepsilon_{\text{infl}}$ of the optimum [26]. Since ABIT* updates the cost-to-come of any vertex under the same condition as ATD*, ABIT* also finds a solution whose cost is within $\varepsilon_{\text{trunc}}\varepsilon_{\text{infl}}$ of the optimum. ABIT* therefore asymptotically finds a resolution-optimal path when the product $\varepsilon_{\text{trunc}}\varepsilon_{\text{infl}}$ tends to one as the number of samples approaches infinity,

$$\lim_{q\to\infty} \varepsilon_{\text{trunc}}\varepsilon_{\text{infl}} = 1.$$

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

Proof. ATD* can handle improved and worsened state connections, but adding states and edges to a graph can only improve connections. Therefore only states with improved cost-to-come (called overconsistent in [26]) are considered.

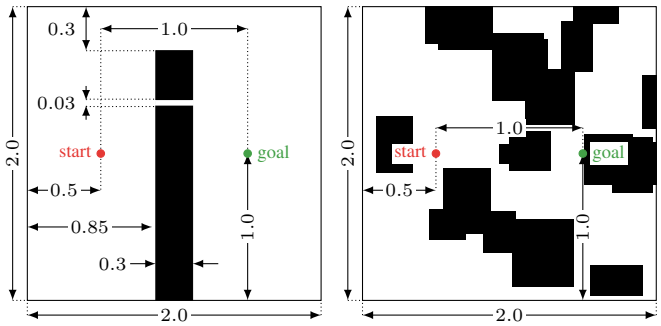
ATD*'s vertex queue is first converted to a compatible edge queue and it is then shown that ATD*'s termination criterion is stricter than that of ABIT*.

ATD* computes a sort key for every state, $\mathbf{x}$, in its queue,

$$\text{key}_{\text{ATD}^*}(\mathbf{x}) \coloneqq g_{\text{p}}[\mathbf{x}] + \varepsilon_{\text{infl}}\widehat{h}(\mathbf{x}).$$

The cost-to-come label, $g_{\text{p}}[\mathbf{x}]$, is recursively defined as

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



(a) Wall gap (b) Random rectangles

Fig. 2: A 2D illustration of the simulated planning problems used in Section V. The state space, $X \subset \mathbb{R}^n$, is bounded by a hypercube of width two for both problems. Ten different instantiations of the random rectangles experiment were tested. The results are presented in Fig. 3.

where $X_{\mathrm{p}}(\mathbf{x})$ denotes the discovered potential parents of $\mathbf{x}$ and the base case is $g_{\mathrm{p}}[\mathbf{x}_{\mathrm{start}}] = 0$. ATD* processes vertices in its vertex queue, $\mathcal{Q}^{\mathrm{V}}_{\mathrm{ATD}^*}$, in order of ascending key values,

$$\mathbf{x}_{\mathrm{next}} = \underset{\mathbf{x} \in \mathcal{Q}^{\mathrm{V}}_{\mathrm{ATD}^*}}{\arg\min} \left\{ \mathrm{key}_{\mathrm{ATD}^*}(\mathbf{x}) \right\}.$$

Whenever a better connection to a state in the queue is found, the key of this state is updated and the queue is resorted.

The queue could alternatively contain multiple instances of the same state, each with a different parent and key value. Selecting the minimum from this queue would ensure that the best discovered connection for each state is considered first. This would be equivalent to an edge version of ATD* where the next connection from the edge queue, $\mathcal{Q}^{\mathrm{E}}_{\mathrm{ATD}^*}$, is

$$(\mathbf{x}_{\mathrm{p}}, \mathbf{x}_{\mathrm{c}})_{\mathrm{next}} = \underset{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{Q}^{\mathrm{E}}_{\mathrm{ATD}^*}}{\arg\min} \left\{ g_{\mathrm{p}}[\mathbf{x}_i] + c(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon_{\mathrm{infl}} \widehat{h}(\mathbf{x}_j) \right\}.$$

ATD*'s inner loop terminates if for any goal $\mathbf{x}_{\mathrm{goal}} \in X_{\mathrm{goal}}$

$$\min_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{Q}^{\mathrm{E}}_{\mathrm{ATD}^*}} \left\{ g_{\mathrm{p}}[\mathbf{x}_i] + c(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon_{\mathrm{infl}} \widehat{h}(\mathbf{x}_j) \right\} \geq \frac{\min\left\{ g_{\mathrm{p}}[\mathbf{x}_{\mathrm{goal}}], g_{\mathcal{T}}(\mathbf{x}_{\mathrm{goal}}) \right\}}{\varepsilon_{\mathrm{trunc}}}.$$

ABIT*'s search terminates if for any goal $\mathbf{x}_{\mathrm{goal}} \in X_{\mathrm{goal}}$

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

This is less strict, as the heuristic, $\widehat{c}$, is admissible, the inflation factor, $\varepsilon_{\mathrm{infl}}$, is greater than or equal to one, and for all states, $\mathbf{x}_i \in X$, it holds that $g_{\mathrm{p}}[\mathbf{x}_i] \geq g_{\mathcal{T}}(\mathbf{x}_i)$ as rewirings can only improve the cost-to-come to states. ABIT* therefore considers at least all edges that ATD* would consider. □

V. Experimental Results

ABIT* was compared against the Open Motion Planning Library (OMPL) [27] versions of RRT-Connect, RRT*, RRT$^{\#}$, LBT-RRT, and BIT* on simulated problems in $\mathbb{R}^4$ and $\mathbb{R}^8$ (Fig. 2)[1]. The objective for the almost-surely asymptotically optimal planners was to minimize path length. The RGG constant η was set to 1.1 for all planners. LBT-RRT used the default value of 0.4 as the approximation factor. RRT$^{\#}$ sampled the entire state space. RRT-based algorithms used a goal bias of 5% and maximum edge lengths of 0.5 and 1.25 in $\mathbb{R}^4$ and $\mathbb{R}^8$, respectively. BIT* and ABIT* sampled 100 states per batch regardless of the state space dimension, had graph pruning turned off, and used Euclidean distance as a heuristic. ABIT* was configured to search each RGG twice. First with a highly inflated heuristic, $\varepsilon_{\mathrm{infl}} = 10^6$, and then again with a lower factor, $\varepsilon_{\mathrm{infl}} = 1 + {}^{10}/_{q}$. A single truncation factor, $\varepsilon_{\mathrm{trunc}} = 1 + {}^{5}/_{q}$ was used for all searches. All parameters were tuned to optimize planner performance on test problems.

[1]The performances were measured with OMPL v1.4.1 on a laptop with 16 GB of RAM and an Intel i7-4910MQ processor running Ubuntu 18.04.

A. Experimental Problems

The planners were tested on two problems in $\mathbb{R}^4$ and $\mathbb{R}^8$. The first consisted of a wall with a narrow gap such that valid paths can only be in one of two homotopy classes (Fig. 2a). Each planner was run 100 times for one second with different random seeds. Figures 3a and 3d show the achieved success rates and median path lengths of all tested planners.

The second consisted of axis-aligned hyperrectangles of random widths placed randomly in the state space (e.g., Fig. 2b). Ten different random problems were generated for each state space dimension and planners were run 100 times on each instantiation. The runtime was limited to one and 40 seconds for problems in $\mathbb{R}^4$ and $\mathbb{R}^8$, respectively. Figures 3b, 3c, 3e, and 3f show the achieved success rates and median path costs of all tested planners for the two problems that resulted in the best and worst performances of ABIT*, as defined by its initial solution time relative to RRT-Connect.

B. Planning for Axel

The benefits of ABIT*'s advanced graph-search techniques were also demonstrated on real-world robotic planning problems during a week-long NASA/JPL-Caltech field test in the Mojave Desert with the Axel Rover System (Fig. 1). Axel is a tethered robotic platform designed for near-vertical surfaces and other challenging or unstable terrain. The complexity of the terrain and its interaction with the tether make for challenging planning problems because state evaluations are computationally expensive. ABIT* typically found initial solutions to these problems in under two seconds. This allowed it to spend the remaining computational time to improve this solution by repairing its search and increasing the density of its approximation. This resulted in 95.12% autonomy by distance, despite the challenging terrain.

VI. Discussion & Conclusion

ABIT* demonstrates that the perspective of separate approximation and search in single-query almost-surely asymptotically optimal sampling-based planning can be used to design algorithms with improved anytime performance. Figure 3 shows that ABIT* outperforms other single-query, almost-surely asymptotically optimal planners by finding initial solutions quickly and converging to an optimal solution in an anytime manner without wasting computational effort. The only tested planner that finds initial solutions faster

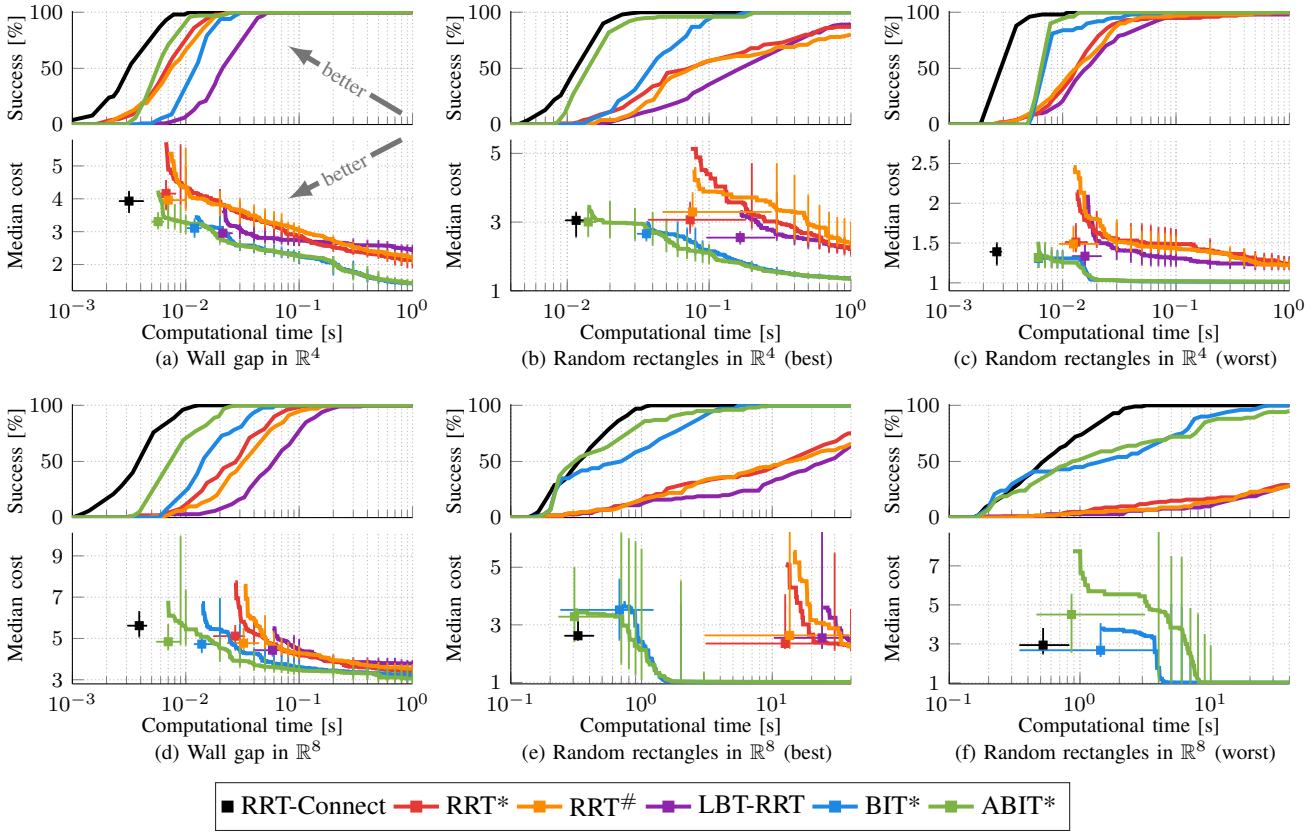


Fig. 3: Results from the experiments described in Section V-A. The results from the wall gap experiment are shown in the plots (a) and (d) for $\mathbb{R}^4$ and $\mathbb{R}^8$, respectively. The plots (b) and (c) show the best and worst instances of ten random rectangles experiments in $\mathbb{R}^4$. The plots (e) and (f) show the best and worst instances in $\mathbb{R}^8$. The squares in the cost plots show the median initial costs and times while the lines show the median cost over time for almost-surely asymptotically optimal planners (unsuccessful runs were taken as infinite costs). The error bars show a nonparametric 99% confidence interval on the solution cost and time. Note that in plot (f) less than 50 trials of LBT-RRT, RRT* and RRT$^{\#}$ succeeded, so their median cost is infinity.

than ABIT* is RRT-Connect, which is not an almost-surely asymptotically optimal algorithm and cannot improve its initial solution when given more computational time.

ABIT* relies on admissible heuristic estimates of edge-costs between states and the cost-to-go from states to a goal. If no heuristics are available, ABIT* can be run with the trivial heuristic, i.e., $\forall \mathbf{x}_i, \mathbf{x}_j \in X$, $\widehat{h}(\mathbf{x}_i) \equiv \widehat{c}(\mathbf{x}_i, \mathbf{x}_j) \equiv 0$. An asymmetric bidirectional search could alternatively be used to simultaneously estimate and exploit a problem-specific heuristic, as in Adaptively Informed Trees (AIT*) [28].

If ABIT* is run with unit inflation and truncation factors, it can be viewed as a simplified but equally performant version of BIT* that cascades rewirings. ABIT* uses a single edge queue instead of BIT*'s dual vertex and edge queues and avoids repeated collision checks by caching checked edges in an object-oriented manner instead of labelling states *old* or *new* as in BIT*. This clarifies the conceptual ideas behind these algorithms and simplifies their implementation without adding any practically significant computational costs.

This improved implementation allows ABIT* to balance exploiting its current approximation of the state space with exploring the relevant regions of the state space. This is achieved using advanced graph-search techniques similar to anytime repairing and truncated search algorithms.

An inflated heuristic biases ABIT*'s search towards the goal and finds initial solutions quickly. Truncating the search once a sufficient bound on the solution quality of the current solution is achieved avoids wasting computational effort fully exploiting 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* is not very sensitive to the exact form of these policies. Results comparable to the ones presented in this paper are achieved whenever the initial search is conducted with a very high inflation factor and both factors asymptotically tend to one as the number of sampled states approaches infinity.

ABIT* also shows the benefits of using advanced graph-search techniques in sampling-based planning on real-world path planning problems posed by Axel, a NASA/JPL-Caltech rover specialized for navigation on challenging terrain.

Information on the OMPL implementation of ABIT* is available at `https://robotic-esp.com/code/`.

VII. Acknowledgements

This research was funded by UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) research hub [EP/R026084/1]. We want to thank the entire NASA/JPL-Caltech Robotic Surface Mobility Group for their support on the Axel Rover System and extend special thanks to Travis, Jacob, Issa, and Mike for their help with integrating ABIT* onto Axel.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[3] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

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

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[6] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, 2020.

[7] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems (NIPS)*, pp. 767–774, 2004.

[8] I. A. D. Nesnas, J. B. Matthews, P. Abad-Manterola, J. W. Burdick, J. A. Edlund, J. C. Morrison, R. D. Peters, M. M. Tanner, R. N. Miyake, B. S. Solish, and R. C. Anderson, "Axel and DuAxel rovers for the sustainable exploration of extreme terrains," *Journal of Field Robotics*, vol. 29, no. 4, pp. 663–685, 2012.

[9] D. Bertsekas, "Convergence of discretization procedures in dynamic programming," *IEEE Transactions on Automatic Control*, vol. 20, no. 3, pp. 415–419, 1975.

[10] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[11] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[12] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

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

[14] R. Bohlin and L. E. Kavraki, "Path planning using Lazy-PRM," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 1, 2000, pp. 521–528.

[15] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 3, 1997, pp. 2719–2726.

[16] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 2, 2000, pp. 995–1001.

[17] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics and Automation*, vol. 34, no. 4, pp. 966–984, 2018.

[18] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-Smart: A rapid convergence implementation of RRT*," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 299, 2013.

[19] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2421–2428.

[20] O. Salzman and D. Halperin, "Asymptotically-optimal motion planning using lower bounds on cost," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4167–4172.

[21] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast Marching Tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[22] A. Valero-Gomez, J. V. Gomez, S. Garrido, and L. Moreno, "The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 111–120, 2013.

[23] A. C. Holston, D.-H. Kim, and J.-H. Kim, "Fast-BIT*: Modified heuristic for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs," in *International Conference on Robotics and Biomimetics*, 2017, pp. 1892–1899.

[24] J. D. Gammell, "Informed anytime search for continuous planning problems," Ph.D. dissertation, University of Toronto, 2017.

[25] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[26] S. Aine and M. Likhachev, "Truncated incremental search," *Artificial Intelligence*, vol. 234, pp. 49–77, 2016.

[27] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[28] M. P. Strub and J. D. Gammell, "Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2020.