Adaptively Informed Trees (AIT*): Fast Asymptotically Optimal Path Planning through Adaptive Heuristics

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

***Abstract*— Informed sampling-based planning algorithms exploit problem knowledge for better search performance. This knowledge is often expressed as heuristic estimates of solution cost and used to order the search. The practical improvement of this informed search depends on the accuracy of the heuristic.**

Selecting an appropriate heuristic is difficult. Heuristics applicable to an entire problem domain are often simple to define and inexpensive to evaluate but may not be beneficial for a specific problem instance. Heuristics specific to a problem instance are often difficult to define or expensive to evaluate but can make the search itself trivial.

This paper presents Adaptively Informed Trees (AIT*), an almost-surely asymptotically optimal sampling-based planner based on BIT*. AIT* adapts its search to each problem instance by using an asymmetric bidirectional search to simultaneously estimate and exploit a problem-specific heuristic. This allows it to quickly find initial solutions and converge towards the optimum. AIT* solves the tested problems as fast as RRT-Connect while also converging towards the optimum.

I. Introduction

Path planning is the problem of finding a continuous sequence of valid states between a start and goal specification. Sampling-based planners, such as Probabilistic Roadmaps (PRM) [1], approximate the state space by sampling discrete states and connecting them with edges. The resulting structure can then be processed by graph-search algorithms to find a sequence of states that connects the start to the goal.

Informed graph-search algorithms, such as A* [2], use knowledge about a problem domain to increase their efficiency. This knowledge is often captured in the form of a *heuristic function*, $\widehat{h}$, which estimates cost-to-go, i.e., the cost to go from any state in the state space to the goal.

The properties of this heuristic directly affect the performance of the search algorithms. An *admissible* heuristic never overestimates the actual cost-to-go. A *consistent* heuristic satisfies a triangle-inequality, such that for any two states, $\mathbf{x}_i, \mathbf{x}_j$, it satisfies $\widehat{h}(\mathbf{x}_i) \leq p(\mathbf{x}_i, \mathbf{x}_j) + \widehat{h}(\mathbf{x}_j)$, where $p(\mathbf{x}_i, \mathbf{x}_j)$ is the best cost of any path from $\mathbf{x}_i$ to $\mathbf{x}_j$. Note that by definition all consistent heuristics are also admissible. A* is guaranteed to find the optimal solution when provided with an admissible heuristic. If the provided heuristic is also consistent, then A* expands the minimum number of states of any informed graph-search algorithm using that heuristic (i.e., it is *optimally efficient* [2]).

Improving the accuracy of a heuristic directly improves the performance of informed search algorithms [3–6], and the search becomes trivial when a perfect heuristic is available.

[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: AIT* uses an asymmetric bidirectional search that is specialized for path planning problems with computationally expensive edge evaluations, such as those posed by NASA/JPL-Caltech's Axel Rover System [7].

Designing and selecting effective heuristics is difficult for many problem domains. This is because heuristics are most effective when they are both accurate and computationally inexpensive to evaluate. Heuristics that are applicable to an entire problem domain are often inexpensive to evaluate but may not be accurate for a specific problem instance. Accurate heuristics can be designed for a specific problem instance during the search, but this can be computationally expensive and may diminish the overall search performance.

Computational cost directly influences the real-world performance of planning algorithms. Sampling-based planners contain a number of computationally expensive basic operations, including state expansions and edge evaluations [8]. State expansions often require nearest neighbor searches that increase in computational cost with the number of samples. Edge evaluations require local planning between two states and detecting collisions on the resulting path.

Lazy sampling-based planners, such as Lazy PRM [9], reduce this computational cost by avoiding the evaluation of every edge. These algorithms first perform an inexpensive search on a simplified approximation without collision detection. This allows them to only evaluate the edges that are believed to be on an optimal path, and reduce the number of evaluated edges. This improves performance, especially for problems with computationally expensive edge evaluations, such as those considered in this paper.

This paper presents Adaptively Informed Trees (AIT*), a lazy, almost-surely asymptotically optimal sampling-based planner that uses an asymmetric bidirectional search to simultaneously estimate and exploit an accurate, problem-specific heuristic. AIT* estimates this heuristic by performing a lazy reverse search on the current sampling-based approximation. This heuristic is then used to order the forward search of this approximation while considering complete edge

evaluations. The results of the computationally expensive edge evaluations performed by this forward search inform the reverse search, which creates increasingly accurate heuristics. This allows AIT* to efficiently share information between the two individual searches.

Efficiently estimating and exploiting a problem-specific heuristic allows AIT* to outperform existing sampling-based planning algorithms when edge evaluations are expensive. AIT* finds initial solutions to the tested problems at least as fast as RRT-Connect while still almost-surely converging to the optimal solution, which RRT-Connect does not.

II. Background

Creating and adapting heuristics to improve informed graph-search algorithms is an active area of research (Sec. II-A). Heuristics are applied in sampling-based planning to reduce search effort by ordering the search and focus the approximation to the relevant region of the state space (Sec. II-B). Lazy search algorithms separately focus on reducing search effort by avoiding collision detection (Sec. II-C).

A. Improved Heuristics for Informed Search Algorithms

Improving the heuristic for informed graph-search algorithms has been shown to be effective for many problem domains, including path planning [6].

Pattern Databases [4] are precomputed tables of exact solution costs to simplified subproblems of a problem domain. An informed algorithm can use this database during the search to create admissible heuristics. Additive Pattern Databases [5] extend this approach to combine database entries into more accurate heuristics that are still admissible. This approach speeds up the search of problems for which simplified subproblems can be created and solved, but requires creating databases for every problem domain *a priori* to the search.

Heuristic accuracy can alternatively be improved by using the error in the heuristic values of states as they are discovered. Thayer et al. [10] use the error of each state expansion to update the heuristic during the search. This can be applied to any problem domain and does not require any preprocessing, but the resulting heuristic is not guaranteed to be admissible.

Adaptive A* [11, 12] is an incremental search algorithm that updates its heuristic function based on the cost-to-come values of previous searches of similar problems. This results in ever more accurate and admissible heuristics but can not be used for the initial search of a graph.

Kaindl. et al [13] use the *Add method* to inform a forward search with a partial reverse search. The reverse search reveals errors in the heuristic values, the minimum of which is added to all unexpanded states. This results in a more informed but still admissible heuristic, but requires the user to specify how many states to expand during the reverse search and increases the heuristic uniformly for all unexpanded states. Wilt et al. [14] present an updated version of this method which does not require a user-specified parameter.

Unlike these approaches, AIT* does not need a predefined database, creates a consistent heuristic during the search, can be used on the initial search of a graph, and adaptively estimates the heuristic for each state individually.

B. Sampling-Based Planning with Heuristics

Heuristics have been used in sampling-based planning to guide the search and focus the approximation. RRT-Connect [15] builds on Rapidly-exploring Random Trees (RRT) [16] by incrementally growing two trees, one rooted in the start state and one in the goal state. These trees each explore the state space around them but are also guided towards each other by a *connect heuristic*. This approach can result in very fast initial solution times but is not almost-surely asymptotically optimal and does not improve the solution quality with more computational time.

Informed RRT* [17] incorporates an ellipsoidal heuristic into the almost-surely asymptotically optimal RRT* [18]. This improves the convergence rate by focusing the incremental approximation to the relevant region of the state space but does not guide the search.

Sakcak et al. [19] incorporate a heuristic into a version of RRT* which is based on motion-primitives [20]. This can improve the performance for kinodynamic systems but requires preprocessing and relies on an *a priori* discretization which suffers from the *curse of dimensionality* [21].

Batch Informed Trees (BIT*) [22, 23] samples batches of states and views these sampled states as an increasingly dense edge-implicit random geometric graph (RGG) [24]. This allows BIT* to use a series of informed graph-searches to process the states in order of potential solution quality. BIT* efficiently reuses information from both previous searches and approximations by using incremental search techniques but does not update its heuristic during the search.

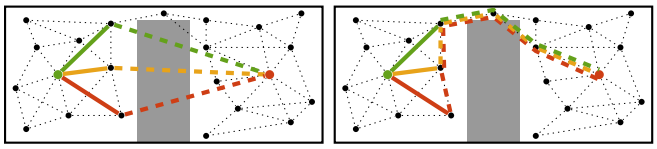
Unlike these approaches, AIT* improves its solution quality with more computational time, uses its heuristic to guide the search, does not rely on an *a priori* discretization, and updates its heuristic during the search.

C. Sampling-Based Planning with Lazy Collision Detection

Path planning algorithms employ lazy collision detection to avoid spending computational resources on edges that are unlikely to be on an optimal path. Lazy PRM [9] approximates the entire state space with an RGG without collision detection and searches this RGG for a path from the start state to a goal state. This path is then checked for collisions. If collisions are detected, then the corresponding edges and vertices are removed from the graph and a new search must be started from scratch. There also exist almost-surely asymptotically optimal variants of Lazy PRM [25, 26].

Lazy Shortest Path (LazySP) [27] is a class of algorithms that reduces the number of edges checked for collisions. It first finds a path from the start to the goal using an inexpensive estimate of the edge costs. Once a path is found, it uses an *edge selector* function which determines the order in which these edges are checked for collision. An example of a LazySP algorithm is Lazy Receding Horizon A* (LRHA*) [28].

Unlike these approaches, AIT* does not restart its search from scratch upon detecting collisions, and uses admissible heuristics to focus its approximation.



(a) Euclidean heuristic (b) Adaptive heuristic

Fig. 2: BIT* approximates the state space of a problem by sampling batches of states (black dots), which define an RGG with implicit edges (dotted connections). These edges are processed in order of potential solution quality, according to a heuristic cost-to-go estimate (thick dashed lines). *A priori* heuristics are often not problem specific and provide poor estimates of the cost-to-go. The Euclidean norm assigns the lowest potential solution cost to the yellow edge despite the presence of the obstacle (a). By using a lazy reverse search, AIT* is instead able to provide a more accurate heuristic adapted to the current approximation of the specific problem (b).

III. Adaptively Informed Trees (AIT*)

BIT* is an almost-surely asymptotically optimal sampling-based planner that builds a discrete approximation of a state space by sampling batches of states. This approximation can be focused to the region of the state space that can possibly improve a current solution with informed sampling [17].

BIT* views the states it samples as an increasingly dense, edge-implicit RGG. It processes the implicit RGG edges in order of potential solution quality, similar to an edge-queue version of Lifelong Planning A* (LPA*) [29]. The true edge costs are evaluated lazily by maintaining a queue ordered by the sum of the current cost-to-come from the start to the edge's parent state, a heuristic of the edge cost, and a heuristic of the cost-to-go from the edge's child state to the goal state. Full details of BIT* are in [23, 30].

AIT* builds on BIT* by improving the accuracy of the used heuristic, which improves performance on problems with expensive edge evaluations. It uses an asymmetric bidirectional search to efficiently estimate and exploit an accurate heuristic for each problem instance (Fig. 2). The forward search is the same as in BIT* but uses the heuristic provided by a computationally inexpensive reverse search. This heuristic can be updated efficiently when the forward search reveals that it contains errors by using an incremental algorithm, such as LPA*, on the reverse search. Algorithm 1 presents a conceptual overview of BIT* and AIT*. The full algorithmic details are provided in Algorithms 2–6.

Since BIT* is almost-surely asymptotically optimal when given an admissible heuristic [23] and the reverse search of AIT* results in an admissible heuristic for each approximation, AIT* is almost-surely asymptotically optimal as well.

A. Notation

The state space of the planning problem is denoted by $X = \mathbb{R}^n, n \in \mathbb{N}$, the start by $\mathbf{x}_{\text{init}} \in X$, and the goals by $X_{\text{goal}} \in X$. The sampled states are denoted by X_{sampled}.

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 connections between states, while the edges in the reverse tree, $E_{\mathcal{R}} \subseteq V_{\mathcal{R}} \times V_{\mathcal{R}}$, can lead through invalid regions of the problem domain. An edge consists of a parent state, $\mathbf{x}_{\text{p}}$, and a child state, $\mathbf{x}_{\text{c}}$, and is denoted as $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$.

Algorithm 1: Concept of BIT* **with changes for AIT***

```
initialize search, queue, and approximation
update the heuristic
repeat
    get and remove the best edge from the queue
    if the edge can possibly improve the solution
        if the edge is valid
            compute the true cost of the edge
            update the search tree with the edge
        else
            update the heuristic
    else
        update the approximation
        update the heuristic
        populate the queue
until stopped
```

Let $\mathbb{R}^{\infty}_{\geq 0}$ denote the union of the nonnegative real numbers with infinity. The function $\widehat{g}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents an admissible heuristic of the cost-to-come from the start to a state. The function $g_{\mathcal{F}}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents the cost-to-come from the start state to a state through the current forward tree. This cost is taken to be infinite for any state without an associated vertex in the forward tree.

The function $\widehat{h}\colon X \to \mathbb{R}^{\infty}_{\geq 0}$ represents an admissible heuristic 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., $\widehat{f}(\mathbf{x}) := \widehat{g}(\mathbf{x}) + \widehat{h}(\mathbf{x})$. This estimate defines the informed set of states that could provide a better solution, $X_{\widehat{f}} := \{\mathbf{x} \in X \mid \widehat{f}(\mathbf{x}) \leq c_{\text{current}}\}$, where c_{current} is the current solution cost. 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 a set and let B, C be subsets of 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 number of states sampled per batch is denoted by m.

B. Approximation

AIT* samples batches of states to build a discrete approximation of the state space. It uses informed sampling to focus its approximation to the region of the state space that can possibly improve the current solution (Alg. 2, line 23).

States that are within a radius, r, of each other are treated as neighbors (Alg. 4, line 1). Graph complexity is limited as states are sampled by decreasing this radius as in [18], using the measure of the informed set, as in [17],

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

where q is the number of sampled states in the informed set, $\eta > 1$ is a tuning parameter, and $\lambda(X_{\widehat{f}})$ and ζ_n are the Lebesgue measures of the informed set and an n-dimensional unit ball, respectively. Faster-decreasing radii are provided in [31, 32] but are not used in AIT* for fairer comparison to existing algorithms as they are presented in the literature.

AIT* always includes the existing connections in both the forward and the reverse search trees in its approximation (Alg. 4, lines 2, 3), and removes invalid edges, regardless of the distance (Alg. 5, line 8; Alg. 4, line 6).

Algorithm 2: BIT* with changes for AIT*

1 $V_{\mathcal{F}} \leftarrow \mathbf{x}_{\text{init}}$; $E_{\mathcal{F}} \leftarrow \emptyset$; $\mathcal{F} \leftarrow (V_{\mathcal{F}}, E_{\mathcal{F}})$; $c_{\text{current}} \leftarrow \infty$
2 $X_{\text{sampled}} \leftarrow X_{\text{goal}} \cup \{\mathbf{x}_{\text{init}}\}$; $\mathcal{Q}_{\text{F}} \leftarrow \texttt{expand}(\mathbf{x}_{\text{init}})$;
3 **update_heuristic**()
4 **repeat**
5 $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) \leftarrow \texttt{pop_best}(\mathcal{Q}_{\text{F}})$
6 **if** $g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) + \widehat{h}(\mathbf{x}_{\text{c}}) < c_{\text{current}}$
7 **if** $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) \in E_{\mathcal{F}}$
8 $\mathcal{Q}_{\text{F}} \xleftarrow{+} \texttt{expand}(\mathbf{x}_{\text{c}})$
9 **else if** $g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) < g_{\mathcal{F}}(\mathbf{x}_{\text{c}})$
10 **if** $\texttt{is_valid}((\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}))$
11 **if** $g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + c(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) + \widehat{h}(\mathbf{x}_{\text{c}}) < c_{\text{current}}$
12 **if** $g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + c(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) < g_{\mathcal{F}}(\mathbf{x}_{\text{c}})$
13 **if** $\mathbf{x}_{\text{c}} \notin V_{\mathcal{F}}$
14 $V_{\mathcal{F}} \xleftarrow{+} \mathbf{x}_{\text{c}}$
15 **else**
16 $E_{\mathcal{F}} \xleftarrow{-} (\texttt{parent}_{\mathcal{F}}(\mathbf{x}_{\text{c}}), \mathbf{x}_{\text{c}})$
17 $E_{\mathcal{F}} \xleftarrow{+} (\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$
18 $\mathcal{Q}_{\text{F}} \xleftarrow{+} \texttt{expand}(\mathbf{x}_{\text{c}})$
19 $c_{\text{current}} \leftarrow \min_{\mathbf{x}_{\text{goal}} \in X_{\text{goal}}} \{g_{\mathcal{F}}(\mathbf{x}_{\text{goal}})\}$
20 **else**
21 **update_heuristic**$(\{(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})\})$
22 **else**
23 $X_{\text{sampled}} \xleftarrow{+} \texttt{sample}(m, c_{\text{current}})$
24 $\mathcal{Q}_{\text{F}} \leftarrow \texttt{expand}(\mathbf{x}_{\text{init}})$
25 **update_heuristic**()
26 **until** stopped

Algorithm 3: $\texttt{expand}(\mathbf{x})$

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

Algorithm 4: $\texttt{neighbors}(\mathbf{x})$

$V_{\text{neighbors}} \leftarrow \{\mathbf{x}_i \in X_{\text{sampled}} \mid \|\mathbf{x} - \mathbf{x}_i\| \leq r(q)\}$
$V_{\text{neighbors}} \xleftarrow{+} \texttt{parent}_{\mathcal{F}}(\mathbf{x})$; $V_{\text{neighbors}} \xleftarrow{+} \texttt{parent}_{\mathcal{R}}(\mathbf{x})$
$V_{\text{neighbors}} \xleftarrow{+} \texttt{children}_{\mathcal{F}}(\mathbf{x})$; $V_{\text{neighbors}} \xleftarrow{+} \texttt{children}_{\mathcal{R}}(\mathbf{x})$
for all $\mathbf{x}_i \in V_{\text{neighbors}}$ **do**
if $(\mathbf{x}, \mathbf{x}_i) \in E_{\text{invalid}}$ **or** $(\mathbf{x}_i, \mathbf{x}) \in E_{\text{invalid}}$
$V_{\text{neighbors}} \xleftarrow{-} \mathbf{x}_i$
return $V_{\text{neighbors}}$

C. Reverse Search

AIT* estimates a heuristic specific to the current approximation by performing a lazy reverse search with LPA* (Alg. 5 and Alg. 6). This search uses a vertex-queue, $\mathcal{Q}_{\text{R}}$, which sorts states according to a lexicographical key,

$$\texttt{key}_{\text{R}}(\mathbf{x}) := \left(\min\left\{\widehat{h}_{\text{con}}[\mathbf{x}], \widehat{h}_{\text{exp}}[\mathbf{x}]\right\} + \widehat{g}(\mathbf{x}), \min\left\{\widehat{h}_{\text{con}}[\mathbf{x}], \widehat{h}_{\text{exp}}[\mathbf{x}]\right\}\right),$$

where $\widehat{h}_{\text{con}}[\mathbf{x}]$ denotes the cost-to-go of $\mathbf{x}$ when it was last connected to the reverse tree and $\widehat{h}_{\text{exp}}[\mathbf{x}]$ denotes the cost-to-go of $\mathbf{x}$ when it was last expanded in the reverse search. These are the g and v values in a forward LPA* search [33].

If Algorithm 5 is called without an argument (Alg. 2, lines 3, 25), it sets the $\widehat{h}_{\text{con}}$ and $\widehat{h}_{\text{exp}}$ values of all states except the goals to infinity and inserts the goal states into the queue (Alg. 5, lines 1–6). This restarts LPA*, which is more efficient than repairing the search when large changes

Algorithm 5: **update_heuristic**$((\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}))$

1 **if** $(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) = \emptyset$
2 **for all** $\mathbf{x} \in V_{\mathcal{R}}$ **do**
3 $\widehat{h}_{\text{con}}[\mathbf{x}] \leftarrow \infty$; $\widehat{h}_{\text{exp}}[\mathbf{x}] \leftarrow \infty$
4 **for all** $\mathbf{x}_{\text{goal}} \in X_{\text{goal}}$ **do**
5 $\widehat{h}_{\text{con}}[\mathbf{x}_{\text{goal}}] \leftarrow 0$
6 $\mathcal{Q}_{\text{R}} \leftarrow \mathbf{x}_{\text{goal}}$
7 **else**
8 $E_{\text{invalid}} \xleftarrow{+} (\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}})$
9 $\texttt{update_state}(\mathbf{x}_{\text{p}})$
10 **while** $\min_{\mathbf{x} \in \mathcal{Q}_{\text{R}}} \{\texttt{key}_{\text{R}}(\mathbf{x})\} < \texttt{key}_{\text{R}}(\mathbf{x}_{\text{init}})$
11 **or** $\widehat{h}_{\text{exp}}[\mathbf{x}_{\text{init}}] < \widehat{h}_{\text{con}}[\mathbf{x}_{\text{init}}]$
12 **or** $\mathcal{Q}_{\text{F}}$ contains an edge with an unprocessed vertex **do**
13 $\mathbf{x} \leftarrow \texttt{pop_best}(\mathcal{Q}_{\text{R}})$
14 **if** $\widehat{h}_{\text{con}}[\mathbf{x}] < \widehat{h}_{\text{exp}}[\mathbf{x}]$
15 $\widehat{h}_{\text{exp}}[\mathbf{x}] \leftarrow \widehat{h}_{\text{con}}[\mathbf{x}]$
16 **else**
17 $\widehat{h}_{\text{exp}}[\mathbf{x}] \leftarrow \infty$
18 $\texttt{update_state}(\mathbf{x})$
19 **for all** $\mathbf{x}_i \in \texttt{neighbors}(\mathbf{x})$ **do**
20 $\texttt{update_state}(\mathbf{x}_i)$

Algorithm 6: **update_state**$(\mathbf{x})$

1 **if** $\mathbf{x} \neq \mathbf{x}_{\text{init}}$
2 $\mathbf{x}_{\text{p}} \leftarrow \arg\min_{\mathbf{x}_i \in \texttt{neighbors}(\mathbf{x})} \left\{\widehat{h}_{\text{exp}}[\mathbf{x}_i] + \widehat{c}(\mathbf{x}_i, \mathbf{x})\right\}$
3 $\widehat{h}_{\text{con}}[\mathbf{x}] \leftarrow \widehat{h}_{\text{exp}}[\mathbf{x}_{\text{p}}] + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x})$
4 **if** $\widehat{h}_{\text{con}}[\mathbf{x}] \neq \widehat{h}_{\text{exp}}[\mathbf{x}]$
5 **if** $\mathbf{x} \notin \mathcal{Q}_{\text{R}}$
6 $\mathcal{Q}_{\text{R}} \xleftarrow{+} \mathbf{x}$
7 **else if** $\mathbf{x} \in \mathcal{Q}_{\text{R}}$
8 $\mathcal{Q}_{\text{R}} \xleftarrow{-} \mathbf{x}$

in the graph are expected [33–35]. LPA*'s initial search is equivalent to A* and results in a consistent and admissible estimate of the heuristic in the current approximation.

If Algorithm 5 is called with an invalid edge (Alg. 2, line 21), then it adds the edge to the set of invalid edges, E_{invalid}, and updates the cost-to-go of the parent state (Alg. 5, lines 8, 9). LPA* then repairs the reverse search tree and increases the cost-to-go values, $\widehat{h}_{\text{con}}$, as necessary. This results in an updated heuristic which is still admissible for the current approximation and can be used by the forward search. Full details of LPA* are available in [29, 33, 34].

D. Forward Search

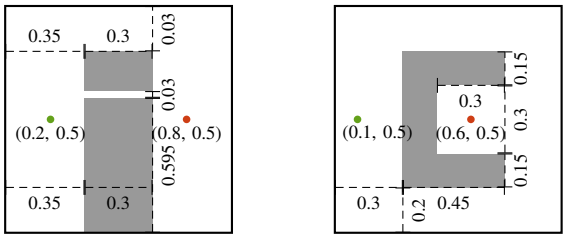
The forward search of AIT* uses an edge-queue, $\mathcal{Q}_{\text{F}}$, which sorts edges according to a lexicographical key,

$$\texttt{key}_{\text{F}}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) := \left(g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}) + \widehat{h}(\mathbf{x}_{\text{c}}), g_{\mathcal{F}}(\mathbf{x}_{\text{p}}) + \widehat{c}(\mathbf{x}_{\text{p}}, \mathbf{x}_{\text{c}}), g_{\mathcal{F}}(\mathbf{x}_{\text{p}})\right),$$

where the cost-to-go values from the reverse search are used as heuristic for the forward search, i.e., $\widehat{h}(\mathbf{x}) := \widehat{h}_{\text{con}}[\mathbf{x}]$.

An iteration begins by getting the best edge from the queue and checking whether it can possibly improve the current solution (Alg. 2, line 6). If it can and is already part of the forward tree, its child state is expanded (Alg. 2, line 8).

If it is not in the forward tree but can possibly improve it (Alg. 2, line 9), then the edge is checked for validity (Alg. 2, line 10). If the edge is invalid, then the heuristic is updated



(a) Wall Gap (b) Goal Enclosure

Fig. 3: Two dimensional illustrations of the experiments on which the planners were tested. The green (●) and red (●) dots represent the positions of the start and goal states, respectively. Grey regions represent invalid states. Each state space dimension was bounded to the interval [0, 1], which is illustrated by the black bounding boxes.

by the reverse search (Alg. 2, line 21; Alg 5). If the edge is valid, then it is completely evaluated. The search then checks whether it can actually improve the current solution and the forward tree (Alg. 2, lines 11, 12).

The child state of a new edge that can improve the current solution and the forward tree is added to the tree if it is not already in the tree (Alg. 2, line 14). If it is, then the new edge is a rewiring and the old edge is removed (Alg. 2, line 16). The new edge is added to the tree and its child state is expanded in both cases (Alg. 2, lines 17, 18).

The iteration finishes by updating the current solution cost (Alg. 2, line 19). In practice this is done efficiently by only checking the goal states in the forward search tree.

If the forward search processes an edge that can not possibly improve the current solution, then a new search on a refined approximation is started (Alg. 2, lines 6, 23–25).

IV. Experimental Results

AIT* was compared against the Open Motion Planning Library (OMPL) [36] implementations of RRT-Connect, RRT*, RRT$^{\#}$ [37], and BIT* on simulated problems[1]. RRT* and RRT$^{\#}$ used a goal bias of 5% and RRT$^{\#}$ used rejection sampling. All RRT-based algorithms used maximum edge lengths of 0.5, 1.25, and 3.0 in $\mathbb{R}^4, \mathbb{R}^8$, and $\mathbb{R}^{16}$, respectively. BIT* and AIT* sampled 100 states per batch regardless of dimension and used the Euclidean norm for all *a priori* heuristics. The RGG constant η was 1.001 for all planners.

A. Abstract Problems

The planners were tested on two abstract problems with different obstacle configurations in $\mathbb{R}^4, \mathbb{R}^8$, and $\mathbb{R}^{16}$ (Fig. 3). Each planner was run 100 times with different random seeds on each instantiation. Planners were given one second to solve problems in $\mathbb{R}^4$, ten seconds in $\mathbb{R}^8$, and 100 seconds in $\mathbb{R}^{16}$. The collision detection resolution was set to 10^{-6} to make evaluating edge costs computationally expensive. The optimization objective was path length. Figure 4 shows the achieved preformances of all tested planners on all problems.

One abstract problem consisted of a wall with a narrow gap, such that in all dimensions only two homotopy classes exist (Fig. 3a). This shows AIT*'s performance on a problem containing a hard-to-find optimal homotopy class (Figs. 4a–c).

[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.

The other abstract problem consisted of a hollow, axis-aligned hyperrectangle enclosing the goal state configured such that even in higher dimensions the goal can only be reached through the face of the hyperrectangle farthest from the start state (Fig. 3b). This problem is challenging for AIT* because there are many invalid edges close to the root of the reverse search tree which means that often large parts of it must be repaired (Figs. 4d–f).

B. Planning for Axel

The benefits of AIT*'s asymmetric bidirectional search were also tested on simulated planning problems for NASA/JPL-Caltech's Axel Rover System (Fig. 1), which is specialized for challenging terrain. These problems require sequences of $\mathrm{SE}(3)$ poses settled on the surface manifold of the terrain. This makes edge evaluations expensive, as every state along an edge has to be projected onto the manifold.

BIT* and AIT* were run 100 times to plan a path down a steep slope with a line-of-sight distance of 30.97 meters between the start and goal positions (Fig. 5a). The linear and angular collision detection resolutions were set to 2 cm and 0.1 rad, respectively. BIT* and AIT* optimized for path length and roll. Figure 5 shows the achieved performances.

V. Discussion & Future Work

AIT* was designed for planning problems with expensive edge evaluations. These often occur when the search has to consider dynamic constraints (e.g., two-point boundary value problems) or complex robot and obstacle interactions (e.g., difficult collision detection) for each edge, as found on NASA/JPL-Caltech's Axel. In future work, Axel will consider tether-terrain interaction and physics-based stability checks based on the anchor history of the tether, which will further increase the edge evaluation cost.

These expensive edge evaluations were simulated in the abstract problems by increasing the collision detection resolution, providing a simple way to increase the edge evaluation cost and evaluate AIT* on illustrative obstacle configurations.

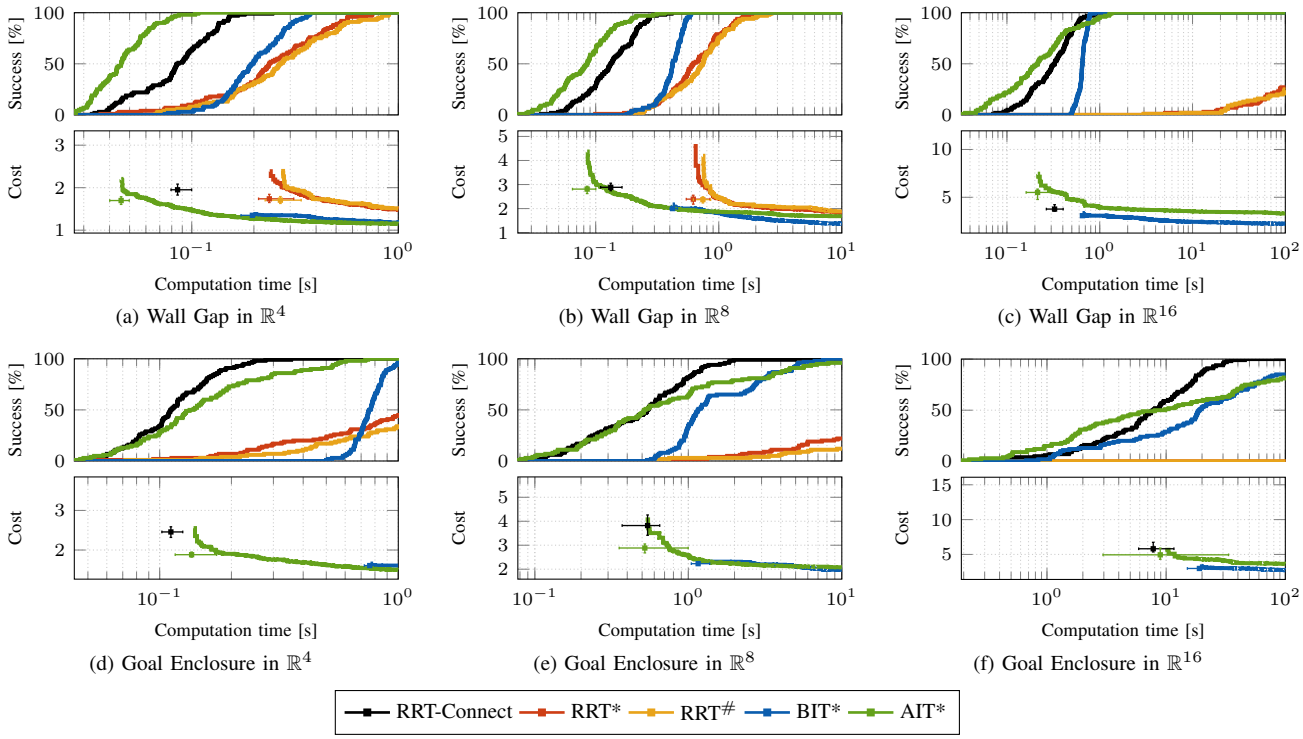
The adaptive heuristic of AIT* is less effective when the lazy reverse search connects many states through invalid edges, especially if these edges are near the root of the reverse search tree. This was illustrated with the goal enclosure experiment (Fig. 3b). Future work could use sparse collision detection on the reverse search to mitigate this problem.

The reverse search of AIT* could also be used to estimate the search effort instead of the solution cost. The forward search could then be replaced with an anytime search that explicitly tries to minimize the time to the next solution, similar to [38], which could speed up initial solution times.

Another way to speed up initial solution times of AIT* would be to inflate the heuristic term in the key of the forward queue, as in Advanced BIT* (ABIT*) [39].

VI. Conclusion

Informed sampling-based algorithms use heuristic knowledge about a problem domain to improve their performance. Heuristics that are applicable to all problems in a domain



(a) Wall Gap in $\mathbb{R}^4$ (b) Wall Gap in $\mathbb{R}^8$ (c) Wall Gap in $\mathbb{R}^{16}$

(d) Goal Enclosure in $\mathbb{R}^4$ (e) Goal Enclosure in $\mathbb{R}^8$ (f) Goal Enclosure in $\mathbb{R}^{16}$

Fig. 4: Planner performances on the abstract problems described in Section IV-A. Results from the wall gap experiments are presented in plots (a), (b), and (c), and from the goal enclosure experiment in plots (d), (e), and (f). The squares in the cost plots show the median times and costs of the initial solutions with a nonparametric 99% confidence interval. The lines show the median cost over time for almost-surely asymptotically optimal planners (unsuccessful runs were taken as infinite costs). Note that in plots (c), (d), (e), and (f) less than 50 trials of RRT* and RRT# were successful, so the median solution cost is infinite for these planners. AIT* finds initial solutions faster than RRT-Connect on four out of six problems, always faster than BIT* with the Euclidean heuristic, and around an order of magnitude faster than RRT* and RRT#. The goal enclosure is a challenging problem for AIT*, because many states close to the goal are initially connected through invalid edges, which results in large updates of the reverse tree.

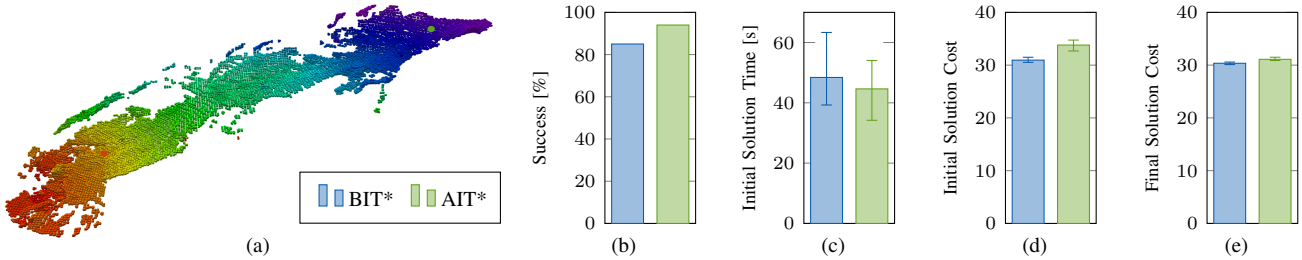


Fig. 5: Results from 100 trials of BIT* and AIT* on a problem for NASA/JPL-Caltech's Axel. The challenge was to plan down a steep slope (a) from the green dot (top right) to the red dot (bottom left), through two narrow passages. The map is colored by elevation. The plot (b) shows the achieved success rates after running for 100 seconds. The plots (c), (d), and (e) show the medians with nonparametric 99% confidence intervals of the initial solution times, the initial solution costs, and the final solution costs, respectively. Unsuccessful runs were taken as infinite costs. AIT* achieves a higher success rate and slighly faster initial solution times but higher initial solution costs than BIT*. Both planners achieve similar final solution costs.

are often simple to define and inexpensive to evaluate but seldom accurate for a specific problem instance. Problem-specific heuristics can be very accurate but the computational cost to estimate and/or evaluate them can often outweigh the improved search efficiency.

This paper presents AIT*, an almost-surely asymptotically optimal sampling-based planner that simultaneously estimates and exploits an accurate heuristic specific to each problem instance. AIT* uses an asymmetric bidirectional search to efficiently share information between the individual searches. The computationally inexpensive reverse search informs the expensive forward search by providing accurate heuristics specific to the current approximation of each problem instance. The forward search informs the reverse search by providing information about invalid edges, which results in ever more accurate heuristics. This is done efficiently by using LPA* as the reverse search algorithm.

This approach is promising for path planning problems with expensive edge evaluations, such as those posed by NASA/JPL-Caltech's Axel. AIT* outperforms existing sampling-based algorithms on the tested abstract problems by finding an initial solution quickly and converging to the optimum in an anytime manner. These problems show the robustness of AIT* with respect to expensive edge evaluations and encourage more thorough evaluations of states which could be used in more advanced optimization objectives.

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

References

[1] 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.

[2] 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.

[3] R. E. Korf, "Finding optimal solutions to Rubik's cube using pattern databases," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 1997, pp. 700–705.

[4] J. C. Culberson and J. Schauffer, "Pattern databases," *Computational Intelligence*, vol. 14, pp. 318–334, 1998.

[5] A. Felner, R. E. Korf, and S. Hanan, "Additive pattern database heuristics," *Journal of Artificial Intelligence Research (JAIR)*, vol. 22, pp. 279–318, 2004.

[6] B. Paden, V. Varricchio, and E. Frazzoli, "Verification and synthesis of admissible heuristics for kinodynamic motion planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 648–655, 2017.

[7] 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 *et al.*, "Axel and DuAxel rovers for the sustainable exploration of extreme terrains," *Journal of Field Robotics*, vol. 29, no. 4, pp. 663–685, 2012.

[8] M. Kleinbort, O. Salzman, and D. Halperin, "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)*, 2016.

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

[10] J. T. Thayer, A. Dionne, and W. Ruml, "Learning inadmissible heuristics during search," in *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[11] S. Koenig and M. Likhachev, "Adaptive A*," in *In Proceedings of the International Converence on Autonomous Agents and Multiagent Systems (AAMAS)*, 2005, pp. 1311–1312.

[12] S. Koenig and M. Likhachev, "A new principle for incremental heuristic search: Theoretical results," in *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2006.

[13] H. Kaindl and G. Kainz, "Bidirectional heuristic search reconsidered," *Journal of Artificial Intelligence Research (JAIR)*, vol. 7, no. 7, pp. 283–317, 1997.

[14] C. Wilt and W. Ruml, "Robust bidirectional search via heuristic improvement," in *Proceedings the Association for the Advancement of Artificial Intelligence Conference International Conference (AAAI)*, 2013.

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

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

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

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

[19] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "An admissible heuristic to improve convergence in kinodynamic planners using motion primitives," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 175–180, 2020.

[20] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, vol. 43, no. 7, pp. 1715–1732, 2019.

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

[22] 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 IEEE International Conference of Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[23] 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 (IJRR)*, 2020.

[24] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003, vol. 5.

[25] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2951–2957.

[26] D. Kim, Y. Kwon, and S.-e. Yoon, "Adaptive lazy collision checking for optimal sampling-based motion planning," in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 320–327.

[27] C. M. Dellin and S. S. Srinivasa, "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)*, 2016.

[28] A. Mandalika, O. Salzman, and S. S. Srinivasa, "Lazy receding horizon A* for efficient path planning in graphs with expensive-to-evaluate edges," in *Proceedings of the*

AAAI International Conference on Automated Planning and Scheduling (ICAPS), 2018.

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

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

[31] 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.

[32] 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.

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

[34] M. Likhachev and S. Koenig, "A generalized framework for lifelong planning a* search," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.

[35] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613–1643, 2008.

[36] 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.

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

[38] J. T. Thayer, J. Benton, and M. Helmert, "Better parameter-free anytime search by minimizing time between solutions," in *Proceedings of the Symposium on Combinatorial Search (SoCS)*, 2012.

[39] M. P. Strub and J. D. Gammell, "Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.