

Nearest-Neighbourless Asymptotically Optimal Motion Planning with Fully Connected Informed Trees (FCIT*)

Tyler S. Wilson¹, Wil Thomason², Zachary Kingston^{2,3}, Lydia E. Kavraki^{2,4}, and Jonathan D. Gammell¹

Abstract— Improving the performance of motion planning algorithms for high-degree-of-freedom robots usually requires reducing the cost or frequency of computationally expensive operations. Traditionally, and especially for asymptotically optimal sampling-based motion planners, the most expensive operations are local motion validation and querying the nearest neighbours of a configuration.

Recent advances have significantly reduced the cost of motion validation by using single instruction/multiple data (SIMD) parallelism to improve solution times for satisficing motion planning problems. These advances have not yet been applied to asymptotically optimal motion planning.

This paper presents Fully Connected Informed Trees (FCIT*), the first fully connected, informed, anytime almost-surely asymptotically optimal (ASAO) algorithm. FCIT* exploits the radically reduced cost of edge evaluation via SIMD parallelism to build and search fully connected graphs. This removes the need for nearest-neighbours structures, which are a dominant cost for many sampling-based motion planners, and allows it to find initial solutions faster than state-of-the-art ASAO (VAMP, OMPL) and satisficing (OMPL) algorithms on the MotionBenchMaker dataset while converging towards optimal plans in an anytime manner.

I. INTRODUCTION

Planning low-cost motions for high-degree-of-freedom (DoF) robots quickly is a fundamental area of research in robotics. These high-DoF robots are described by a continuous *configuration space*, but motion planning requires both a discrete approximation of this space and the ability to efficiently search this approximation. Graph-based planners, such as Dijkstra’s algorithm [1] and A* [2], require the configuration space (i.e., search space) to be discretized *a priori*, and both their planning time and the quality of their solution depends on the resolution of this discretization.

Sampling-based motion planners, such as Probabilistic Roadmaps (PRM) [3], Rapidly-exploring Random Trees (RRT) [4], and RRT-Connect [5], avoid this *a priori* discretization by incrementally sampling the search space which constructs a random geometric graph (RGG) online as a discrete approximation of this search space. Anytime almost-surely asymptotically optimal (ASAO) sampling-based motion planners, such as RRT* [6] and Batch Informed Trees (BIT*) [7], extend sampling-based motion planning by

¹Estimation, Search, and Planning (ESP) Research Group, Queen’s University, Kingston ON, Canada. {18tswl,gammell}@queensu.ca.

²Department of Computer Science, Rice University, Houston TX, USA {wbthomason, zak, kavraki}@rice.edu

³Department of Computer Science, Purdue University, West Lafayette IN, USA

⁴Ken Kennedy Institute, Rice University, Houston TX, USA

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2024-06637], and the National Science Foundation (NSF) [2336612].

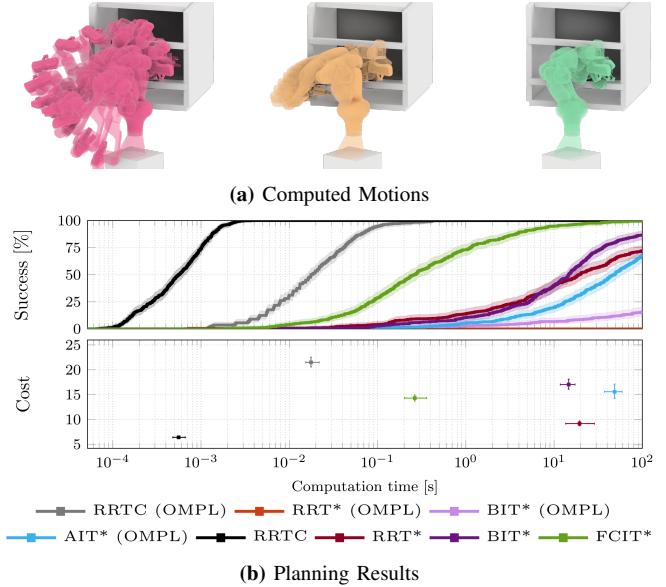


Fig. 1: Results for the 7-DoF Panda [8] on the *cage* environment from the MotionBenchMaker [9] dataset (Sec. IV). (a) Final computed paths for RRT-Connect (pink), RRT-Connect with standard path simplification heuristics applied (orange), and FCIT* (green) on the *cage* problem evaluated in Fig. 3a. (b) Planning results of all planners on all problems in the *cage* environment. The upper plot shows the percentage of runs that found a solution at a given time with Clopper-Pearson 99% confidence intervals. The lower plot shows the median initial path length with nonparametric 99% confidence intervals. The only tested planner that finds initial solutions faster than FCIT* in this environment is RRT-Connect, which is not an ASAO algorithm and cannot improve its initial solution with additional computational time.

continually improving their sampled approximations even after finding an initial solution in order to converge probabilistically towards an optimal solution. This search in BIT* is ordered by potential solution cost which minimizes the required number of edge evaluations (i.e., local motion validations). This, in turn, reduces planning time because edge evaluation traditionally dominates execution time in sampling-based motion planning.

If connectivity in a planner’s sampled approximation is too low, then the graph maintained by the planner *almost never* (i.e., with probability zero) contains a solution [6], [10], [11], but if it is too high, the graph becomes expensive to search due to the high branching factor and resulting high number of edges. As such, although incremental sampling avoids the need for *a priori* approximations, sampling-based ASAO planners still require a user-defined connectivity metric between samples (e.g., connection radius) for efficiency.

A significant body of work has refined the bounds necessary for almost-sure connectivity [10], [12]–[14], but using these bounds has practical tradeoffs. Considering only a subset of

possible edges makes *incomplete* use of the set of samples, potentially lowering the quality of the best solution that can be found without additional sampling. Implementing a partially connected RGG also requires nearest-neighbours structures to make it more efficient to repeatedly query the incident edges of a given sample.

Vector-Accelerated Motion Planning (VAMP) [15] has reduced the computational cost of edge evaluations using single instruction/multiple data (SIMD) parallelism, drastically accelerating solution times for feasible (i.e., satisfying) motion planning (Tbl. I), with RRT-Connect performing up to 500x faster than the previous state of the art [16], [17].

The original VAMP work did not address ASA motion planning. This paper shows that its radically decreased edge evaluation cost can also guide algorithmic advances on this class of problems. Specifically, we revisit the assumptions that edge evaluations are computationally expensive, and consequently that fully connected graphs are prohibitively expensive to search because of their high branching factor and the resulting large number of edges to evaluate.

This insight leads us to Fully Connected Informed Trees (FCIT*), an ASA motion planning algorithm that accelerates motion planning by searching a fully connected graph. It does this efficiently by leveraging SIMD parallelism to reduce the cost of edge evaluations and using a distributed edge queue to address the high branching factor of fully connected graphs. These inexpensive edge evaluations allow FCIT* to build an approximation with maximum connectivity (i.e., a fully connected, or complete, graph) instead of limiting connections to near a theoretical lower bound required for probabilistic guarantees. This removes the need for costly nearest-neighbours structures and improves convergence. The benefits of FCIT* are demonstrated on the MotionBenchMaker (MBM) [9] dataset, where it finds better solutions faster than all tested non-VAMP ASA algorithms (e.g., Fig. 1) and solves more problems, with better solutions, than all tested VAMP ASA algorithms (Tbl. I).

II. RELATED WORK

Informed graph-based searches, such as A* [2], search a discrete graph approximation of a continuously valued search space in order of potential solution quality using an admissible heuristic (i.e., an underestimate of the true cost). A* is both *resolution optimal*, finding the optimal solution in an approximation if one exists, and *optimally efficient*, expanding the minimum number of vertices to find the resolution optimal solution, but requires the continuous search space be discretized *a priori*.

Sampling-based planners, like PRM [3] and RRT [4], place samples to create an approximation of the search space. PRM first builds a graph before searching it for a solution to a specific pair of vertices, while RRT incrementally constructs a tree from the starting vertex until it is connected to the goal, and then extracts the solution. These planners are *anytime* in their approximation by sampling incrementally, avoiding the need for *a priori* approximation.

Bidirectional planners like RRT-Connect [5] perform simultaneous searches from the start and goal by building two trees

to explore the search space and trying to connect them. This bidirectional search performs well on many difficult problems and results in fast initial solution times, but is not ASA and provides no guarantees on solution quality.

Lazy planners, like LazySP [18], Lazy-PRM [19], and Lower Bound Tree-RRT [20], reduce planning time by delaying costly operations like edge evaluation until necessary. Edges are only evaluated when they are a part of a potential solution or to satisfy optimality bounds, reducing the number of these costly operations but requiring that the search be restarted after finding an invalid edge along the path.

Anytime ASA planning allows for an initial solution to be found quickly on a sparse approximation of the search space, and then higher quality solutions to be found as the resolution of the approximation increases with additional sampling.

RRT* [6], BIT* [7], and other similar ASA planners are anytime in their approximation, doing away with the need for *a priori* approximation and relatively sampling the search space and searching the graph. BIT* is also informed in its search, using a heuristic to evaluate edges in order of potential solution quality. This avoids unnecessary computational costs and improves planning convergence, allowing BIT* to efficiently search its increasingly accurate RGG approximation.

Planners like Advanced BIT* (ABIT*) [21], Greedy BIT* (GBIT*) [22], and Greedy RRT* (G-RRT*) [23] leverage a greedy heuristic search to better exploit the current approximation and find an initial solution faster than BIT*. This avoids the high cost of finding an optimal initial solution by computing one that is ‘good enough’ and then later searching the approximation for the resolution optimal solution. This greedy search helps balance exploitation and exploration, but is done to avoid computationally expensive edge evaluations.

Lazy bidirectional ASA planners, like AIT* [24], use a reverse search to calculate a heuristic based on the current approximation that is then used to order the forward search. This extra effort to calculate an accurate heuristic helps reduce the number of computationally expensive edge evaluations, improving the time to find initial solutions.

Flexible Informed Trees (FIT*) [25] uses an adaptive batch size to improve performance by using dense batches to first find an initial solution, and then using sparse batches to optimize it. Placing dense batches helps ensure coverage of difficult-to-sample regions, making a good initial solution more likely, while sampling sparsely after finding a solution avoids computationally expensive operations like edge validation.

Edge evaluations can also be reduced by considering fewer outgoing edges per vertex. Significant research has focused on deriving the lower bound of connectivity necessary to maintain almost-sure asymptotic optimality [10], [12]–[14]. This reduces computational costs by considering fewer edges but will not fully exploit the current approximation and can lead to lower-quality solutions for a given set of samples. If this connectivity is too low then the planner will quickly exploit the approximation but *almost never* find a valid path [6], [10], [11], and if it is too high then the graph will almost surely contain a high quality solution but becomes increasingly expensive to search as the number of edges increases.

Algorithm 1: FCIT*

```

1  $X_{\text{smpl}} \leftarrow \{x_g\}; T \equiv \{V, E\}; V \leftarrow \{x_{\text{start}}\};$ 
2  $E \leftarrow \emptyset; E_{\text{invalid}} \leftarrow \emptyset; Q_{\text{open}} \leftarrow \emptyset; Q_{\text{local}}[x_{\text{start}}] \leftarrow \emptyset;$ 
3 while not Done do
4    $Q_{\text{local}}[x_{\text{start}}] \leftarrow \{(x_{\text{start}}, x \in X_{\text{smpl}}) \mid x \neq x_{\text{start}}\};$ 
5    $Q_{\text{open}} \leftarrow \{\text{NextBestEdge}(x_{\text{start}})\};$ 
6   while  $Q_{\text{open}} \neq \emptyset$  do
7      $(x_p, x_c) \leftarrow \underset{(x,y) \in Q_{\text{open}}}{\operatorname{argmin}} (\hat{f}(x, y));$ 
8      $Q_{\text{open}} \leftarrow \{x_p, x_c\};$ 
9      $Q_{\text{open}} \leftarrow \{\text{NextBestEdge}(x_p)\};$ 
10    if  $p(x_c) \equiv x_p$ 
11       $Q_{\text{local}}[x_c] \leftarrow \{(x_c, x \in X_{\text{smpl}}) \mid x \neq x_c\};$ 
12       $Q_{\text{open}} \leftarrow \{\text{NextBestEdge}(x_c)\};$ 
13    else if  $\hat{f}(x_p, x_c) \leq \min_{x_g \in X_g} \{g_T(x_g)\}$ 
14      if  $g_T(x_p) + \hat{c}(x_p, x_c) \leq g_T(x_c)$ 
15        if IsValid( $x_p, x_c$ )
16          if  $g_T(x_p) + c(x_p, x_c) + \hat{h}(x_c) \leq \min_{x_g \in X_g} \{g_T(x_g)\}$ 
17            if  $g_T(x_p) + c(x_p, x_c) \leq g_T(x_c)$ 
18              if  $x_c \in V$ 
19                 $E \leftarrow (p(x_c), x_c);$ 
20              else
21                 $V \leftarrow x_c;$ 
22                 $Q_{\text{local}}[x_c] \leftarrow \{(x_c, x \in X_{\text{smpl}}) \mid x \neq x_c\};$ 
23                 $E \leftarrow (x_p, x_c);$ 
24                 $Q_{\text{open}} \leftarrow \{\text{NextBestEdge}(x_c)\};$ 
25      else if  $(x_p, x_c) \notin E_{\text{invalid}}$ 
26         $E_{\text{invalid}} \leftarrow \{(x_p, x_c), (x_c, x_p)\};$ 
27    else
28       $Q_{\text{open}} \leftarrow \emptyset;$ 
29     $X_{\text{smpl}} \leftarrow \text{AddSamples}(n);$ 
30 return  $T$ 

```

In comparison, FCIT* reduces the cost of edge evaluations with SIMD parallelism and avoids nearest-neighbours structures by considering a fully connected graph. This completely exploits the current set of samples and is done efficiently via an informed search over a distributed edge queue. FCIT* orders its search by potential solution quality like A*, but does not require *a priori* discretization. Unlike other VAMP algorithms, FCIT* almost-surely converges asymptotically to the optimal solution with additional computational time. FCIT* evaluates edges in order of potential solution cost to efficiently search its RGG approximation like BIT*, but does not connect samples within a critical connection radius or require nearest-neighbours structures, instead considering every possible edge in the graph.

Algorithm 2: NextBestEdge(x_p)

```

1 while  $Q_{\text{local}}[x_p] \neq \emptyset$  do
2    $x_c \leftarrow \underset{(x,y) \in Q_{\text{local}}[x_p]}{\operatorname{argmin}} (\hat{f}(x, y));$ 
3    $Q_{\text{local}}[x_p] \leftarrow \{x_c\};$ 
4   if  $g_T(x_p) + \hat{c}(x_p, x_c) < g_T(x_c)$ 
5   return  $(x_p, x_c);$ 

```

III. FULLY CONNECTED INFORMED TREES (FCIT*)

FCIT* extends the BIT* algorithm by leveraging the hardware-accelerated parallelized approach to edge validation of VAMP [15] and revisiting the assumption that a high branching factor and the high cost of edge evaluation makes fully connected graphs prohibitively expensive to construct and search. It builds and searches a fully connected approximation of the continuous search space, as opposed to relying on a sparsely connected approximation built using nearest-neighbours structures. Building a fully connected approximation makes complete use of all placed samples, potentially containing a higher quality solution than could be found in an approximation with connectivity near a theoretical lower bound [10].

Constructing and searching a fully connected graph also removes the need to maintain expensive nearest-neighbours structures by instead considering all possible edges between sampled states. These edges *may* all be evaluated in the limit, but in practice many will not be because they exist in disconnected components, or belong to a more expensive path than the current best solution.

FCIT*'s search is ordered by potential solution quality (Alg. 1 L. 7). It distributes its ordered queue of unprocessed (i.e., open) edges across a set of local queues stored by each vertex (Alg. 1 L. 22). Only the most promising edges from each local queue are added to the global queue and sorted (Alg. 1 L. 24), resulting in a more efficient search. Pseudocode for FCIT* is provided in Algs. 1 and 2.

A. Notation and Preliminaries

We denote the search space by $X \subseteq \mathbb{R}^n$ and the invalid and valid subsets as $X_{\text{invalid}} \subseteq X$ and $X_{\text{valid}} := \text{closure}(X \setminus X_{\text{invalid}})$, respectively. Let $x \in X$ be a state, $x_{\text{start}} \in X_{\text{valid}}$ be the start state, and $x_{\text{goal}} \in X_{\text{valid}}$ be the goal state. The set of all sampled valid states is denoted as $X_{\text{smpl}} \subset X_{\text{valid}}$. We store the search as a tree, $T = (V, E)$, comprising a set of vertices from the sampled states, $V \subseteq X_{\text{smpl}}$, and a set of edges, $E \subseteq V \times V$. Each edge connects two states, $x_p, x_c \in X_{\text{smpl}}$, which we refer to as the edge's parent and child, respectively.

The planner also tracks the set of invalidated edges, denoted by $E_{\text{invalid}} \subseteq V \times X_{\text{smpl}}$, and maintains a queue of open edges denoted as $Q_{\text{open}} \subseteq V \times X_{\text{smpl}}$. Each vertex, $x \in V$, has an associated local outgoing edge queue stored in a lookup table, $Q_{\text{local}}(x)$, such that $Q_{\text{local}}(x) := \{(x, y \in X_{\text{smpl}}) \mid y \neq x\}$. The functions $p(x)$ and $g_T(x)$ respectively calculate the parent and cost-to-come from the start through the tree, T , for a

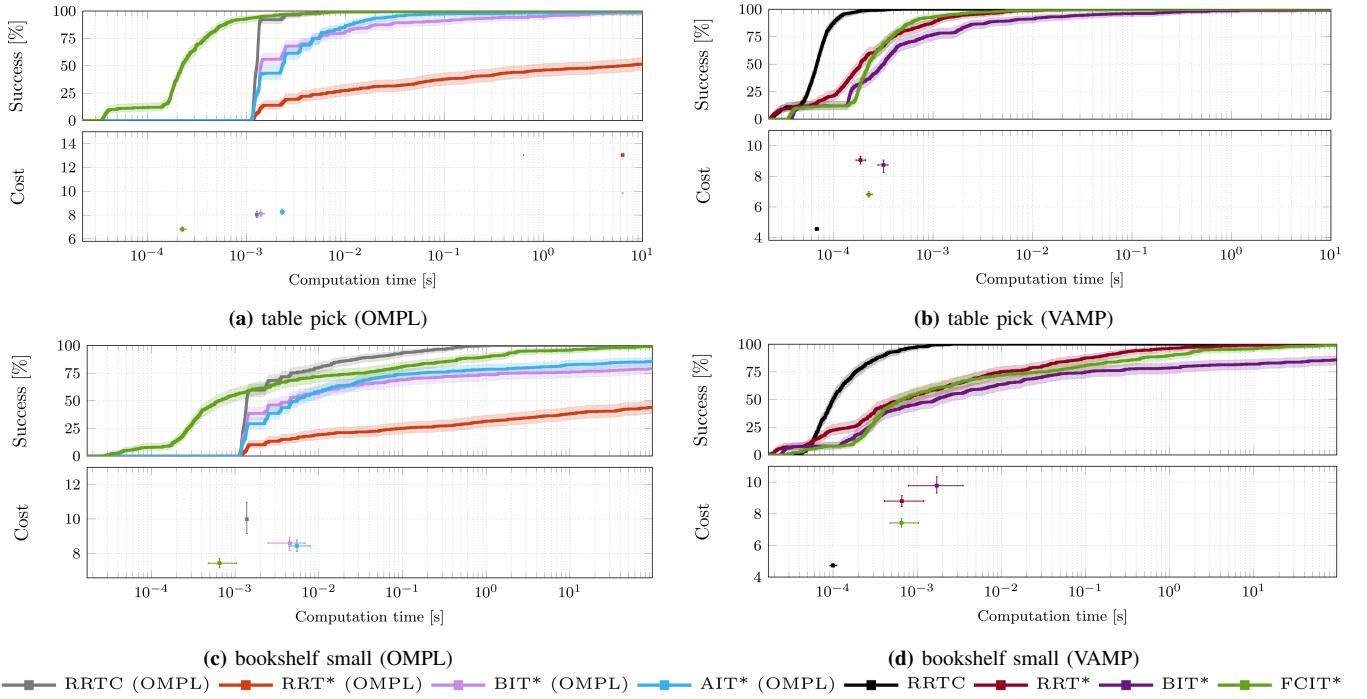


Fig. 2: Results for the 7-DoF Panda [8] on all problems in the *table pick* and *bookshelf small* environments (Sec. IV). Plots (a) and (c) compare FCIT* to several OMPL planners, while plots (b) and (d) compare FCIT* to several VAMP planners. For each plot, the top shows the percentage of runs that found a solution at any given time with Clopper-Pearson 99% confidence intervals, and the bottom shows the median initial path length with nonparametric 99% confidence intervals. The only tested planner that finds initial solutions significantly faster than FCIT* in these environments is RRT-Connect, which is not an ASAO algorithm and cannot improve its initial solution with additional computational time.

state $x \in X_{\text{smpl}}$. These functions return infinity if the state is not in the tree, i.e., $x \notin V$.

Following the formulation in BIT* [7], the function $c : X \times X \rightarrow [0, \infty)$ represents the computed edge cost between two states. The function $\hat{c} : X \times X \rightarrow [0, \infty)$ is an admissible estimate of this edge cost, where $\forall x, y \in X, \hat{c}(x, y) \leq c(x, y)$. The function $\hat{h} : X \rightarrow [0, \infty)$ represents the estimated cost-to-go from the state x to the goal, e.g., $\hat{h}(x) = \min_{x_g \in X_g} (\hat{c}(x, x_g))$.

The function $\hat{f} : V \times X \rightarrow [0, \infty)$ represents an admissible heuristic estimate of the cost of a solution constrained to pass through an edge given the current tree. It is calculated as the sum of the current cost-to-come through the tree, the estimated edge cost, and the estimated cost-to-go, i.e., $\hat{f}(x_p, x_c) := g_T(x_p) + \hat{c}(x_p, x_c) + \hat{h}(x_c)$.

Let A and B be two sets. The notation $A \leftarrow^+ B$ is shorthand for the operation $A \leftarrow A \cup B$, and $A \leftarrow^- B$ is shorthand for the operation $A \leftarrow A \setminus B$. The number of states sampled in each batch is denoted by n .

B. Local Edge Queue

FCIT* maintains a sorted open queue of edges to be expanded, Q_{open} , ordered by potential solution cost. This open queue has to be sorted each time a new edge is added to it. As more edges are added, it becomes longer and takes more time to sort. To reduce the computational cost of this frequent sort, FCIT* distributes the total set of open edges such that each vertex, $x \in V$, keeps its own local queue of outgoing edges, $Q_{\text{local}}(x)$. These local edge queues are sorted *once* each by

the admissible heuristic estimate of solution cost, \hat{f} , through each edge. The open queue, Q_{open} , is then populated with the most promising edge from each vertex's local edge queue (Alg. 1 L. 5). The open queue is thus the ordered set of only the *best* open edge outgoing from each vertex, $Q_{\text{open}} := \{(x, y) \in V \times X_{\text{smpl}} \mid (x, y) \leq \operatorname{argmin}_{(x, y) \in Q_{\text{local}}(x)} \{\hat{f}(x, y)\}\}$. When an edge outgoing from a vertex is removed from the open queue, it is replaced by the next best outgoing edge from that vertex's local queue that could potentially improve the current tree (Alg. 1 L. 24, Alg. 2). This ensures that the open queue always contains the most promising unevaluated outgoing edges from all the vertices in the search tree.

Each vertex in a fully connected RGG has a number of potential outgoing edges equal to $|X_{\text{smpl}}| - 1$, where $|\cdot|$ is the cardinality of a set. Expanding vertices in the tree quickly inflates the total number of open edges, in the worst case increasing it to $|X_{\text{smpl}}|^2$ elements. These local queues reduce the worst case size of the open queue from $|X_{\text{smpl}}|^2$ to $|X_{\text{smpl}}|$. Expanding a new vertex only adds one new edge to the open queue to be frequently sorted, storing the other $|X_{\text{smpl}}| - 2$ edges in that vertex's local queue and sorting them only once. The frequent cost of sorting the open queue, Q_{open} , is thus reduced, and the one-time cost of sorting a given vertex's local queue, $Q_{\text{local}}(x)$, is amortized over the runtime of the planner.

1) *Nearest-Neighbours Structures*: The set of outgoing edges from a given vertex is determined by the connectivity of the graph, i.e., the neighbouring vertices with which it shares edges. In contrast to traditional algorithms, which typically maintain an expensive nearest-neighbours structure,

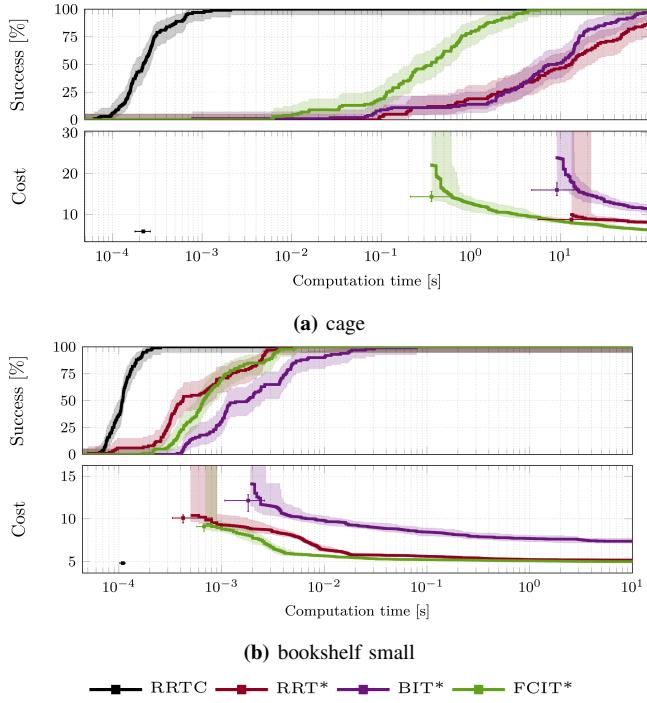


Fig. 3: Convergence results for VAMP planners with the 7-DoF Panda [8] over 100 trials on a single problem from the *cage* (a) and *bookshelf small* (b) environments from the MotionBenchMaker [9] dataset (Sec. IV). For each plot, the top shows the percentage of runs that found a solution by a given time with Clopper-Pearson 99% confidence intervals, and the bottom shows the median initial path length and median path length over time with nonparametric 99% confidence intervals. The only tested planner that finds initial solutions significantly faster than FCIT* in these environments is RRT-Connect, which is not ASAO and cannot improve its initial solution with additional computational time.

finding neighbours is trivial in a fully connected graph since the neighbours of any given vertex are every other sampled state in the graph. FCIT* therefore iterates over all sampled points, X_{smpl} , when populating a vertex’s local edge queue, $Q_{\text{local}}(x) := \{(x, y \in X_{\text{smpl}}) \mid y \neq x\}$, avoiding the computational cost of maintaining a nearest-neighbours structure and reducing planning time.

C. Analysis

This paper uses Definition 24 in [6] 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 resolution optimal. These conditions are sufficient but not necessary.

Since there is a non-zero probability of sampling every state in the search space, the probability that the solution found by FCIT* will asymptotically converge to the optimum approaches one as the number of samples approaches infinity, regardless of whether the sampling used is pseudorandom or deterministic [13]. This holds true for a fully connected graph since it trivially satisfies the lower bound on connectivity, as demonstrated by the simplified-PRM [26] ASAO proof with an infinite r-disc graph [6]. Since the search is performed in an informed order as in BIT*, it is also *resolution*

optimal [7], finding the best possible solution with respect to the current approximation. FCIT* is therefore almost surely asymptotically optimal.

D. Implementation

In practice, each vertex locally stores its set of invalid edges, similar to its local edge queue, instead of maintaining a global list. The cost-to-come function, $g_T(x)$, and parent function, $p(x)$, are implemented as lookups, storing and updating the values to reduce time per iteration. The open and local queues are both implemented as sorted structures.

IV. EXPERIMENTS

FCIT* was evaluated against Open Motion Planning Library (OMPL) [16] and VAMP [15] baselines. We compared to OMPL implementations of RRT-Connect, RRT*, BIT*, and AIT*, as well as VAMP implementations of RRT-Connect, RRT*, and BIT*. Note that the VAMP implementation of BIT* uses the same local edge queue presented in Sec. III-B, but performs a traditional r -disc nearest-neighbours search rather than using a fully connected graph. The VAMP implementation of RRT-Connect is a dynamic domain [27] balanced [28] RRT-Connect [29]. The reported initial solution costs and times for RRT-Connect include path smoothing with randomized shortcircuiting [30], [31] and B-spline smoothing [32].

The planners were tested on the MBM [9] dataset, which consists of 7 difficult planning environments each containing 100 pregenerated problems. These environments cover a range of planning problems, including reaching (*bookshelf tall*, *bookshelf small*, and *bookshelf thin*), highly constrained reaching (*box* and *cage*), and tabletop manipulation (*table pick* and *table under pick*)¹.

All experiments were run using a simulated 7-DoF Panda robotic arm [8]. Each problem was evaluated 5 times by each planner to mitigate the effect of machine noise on the results². All planners use the default VAMP and OMPL samplers with different seeding for each trial. Planners were all given the same time constraints to evaluate each problem in a given environment. The time constraints per environment were 10 seconds on *box*, *table pick*, *table under pick*, *bookshelf thin*, and *bookshelf tall*; and 100 seconds on *bookshelf small* and *cage*.

Experimental results for all problems in a given environment are shown in Figs. 1 and 2, and are summarized in Tbl. I. The time axis for all figures is in logarithmic scale. Results for each environment are presented separately because all the environments in the MBM dataset pose significantly different planning problems from each other. While Tbl. I includes initial solution results for all planners across all environments, Figs. 1 and 2 show results for all problems in *cage*, *table pick*, and *bookshelf small*. These results were chosen because they are the most indicative of relative qualitative planner performance. Fig. 3 shows convergence results for 100 trials of each VAMP planner on a single problem from both the

¹One of the problems in *table pick* is invalid with respect to the robot simulation and is disregarded, giving these experiments 699 total problems.

²All tests were run in Ubuntu 22.04 on a Intel i7-9750H CPU with 32GB of RAM, and the planning algorithms are implemented in C++17.

TABLE I: Summary of all planning results. The **top** results in each row are for the VAMP implementation of the given planner, while the **bottom** results are for the OMPL implementation of that planner. The only tested planner that solves more problems than FCIT* is RRT-Connect, which is not an ASAO algorithm and cannot improve its initial solution with additional computational time. Only VAMP RRT-Connect reliably finds initial solutions faster than FCIT*. Each result indicates the percentage of problems solved in bold, the median initial solution time across all problems on that environment, and the median initial path length across all problems on that environment.

Planner	bookshelf small	bookshelf tall	bookshelf thin	table pick	table under pick	box	cage
RRTConnect	100% 0.1ms (4.7)	100% 0.1ms (4.7)	100% 0.1ms (4.6)	100% 0.1ms (4.6)	100% 0.1ms (6.0)	100% 0.2ms (4.4)	100% 0.6ms (6.4)
	100% 1.4ms (10.0)	100% 1.3ms (8.8)	100% 1.3ms (8.7)	100% 1.3ms (8.0)	100% 1.3ms (12.6)	100% 1.4ms (11.3)	100% 18ms (21.5)
RRT*	99% 0.9ms (8.8)	100% 0.9ms (8.7)	100% 0.7ms (9.3)	100% 0.2ms (9.1)	100% 0.3ms (14.9)	100% 1.7ms (8.1)	72% 19460ms (9.2)
	44% ∞ (∞)	39% ∞ (∞)	51% 8827ms (8.7)	52% 6304ms (13.0)	33% ∞ (∞)	17% ∞ (∞)	0% ∞ (∞)
BIT*	86% 1.7ms (9.8)	95% 1.7ms (9.3)	98% 1.8ms (9.4)	99% 0.3ms (8.7)	100% 0.4ms (12.4)	100% 1.8ms (10.5)	87% 14644ms (17.0)
	79% 4.5ms (8.6)	91% 3.4ms (8.6)	93% 4.4ms (8.3)	98% 1.4ms (8.1)	98% 2.2ms (11.4)	98% 7.7ms (9.9)	16% ∞ (∞)
AIT*	86% 5.5ms (8.4)	95% 4.4ms (8.5)	98% 4.5ms (8.3)	99% 2.3ms (8.3)	100% 3.4ms (11.5)	100% 11.1ms (10.3)	68% 48541ms (15.6)
FCIT*	99% 0.7ms (7.4)	100% 0.4ms (7.5)	100% 0.5ms (7.8)	100% 0.2ms (6.8)	100% 0.4ms (10.7)	100% 0.9ms (8.1)	100% 264ms (14.3)

cage and *bookshelf small* environments. This figure omits OMPL planner convergence results because FCIT* finds initial solutions significantly faster and of higher quality than *all* ASAO OMPL planners (Fig. 1, 2a, and 2c).

V. DISCUSSION

FCIT* revisits fundamental assumptions about ASAO planning in light of the computationally inexpensive local motion validation introduced by VAMP [15]. Planners traditionally seek to reduce the number of edges in their approximation by setting connectivity near a theoretical lower bound. This requires maintaining a computationally expensive nearest-neighbours structure. Moreover, this limits the connectivity of the resulting graph, preventing solutions from being found without additional sampling. FCIT* is able to avoid this lower bound because of the reduced cost of edge evaluation, instead searching a fully connected approximation. This removes the need for maintaining a nearest-neighbours structure, instead considering all possible edges in the graph in an informed order. It uses this fully connected RGG approximation to find better solutions with fewer required samples.

FCIT* outperforms all other tested ASAO planners, both VAMP and OMPL, on the difficult *cage* environment (Fig. 1). The only tested planners that finds initial solutions faster than FCIT* in this environment are the VAMP and OMPL implementations of RRT-Connect, which do not converge towards an optimal solution given additional planning time.

FCIT* demonstrates better performance than other tested VAMP ASAO planners. FCIT* consistently outperforms the VAMP BIT* planner in all environments, finding better initial solutions in less time (Fig. 2). VAMP RRT* shows similar initial solution times to FCIT* on the simpler environments, but consistently yields lower quality solutions, and performs much worse than FCIT* in the difficult *cage* environment, only solving 72% of problems (Tbl. I).

FCIT* solves more problems than any other tested planner barring the two implementations of RRT-Connect, only occasionally failing to solve one difficult problem in the *bookshelf small* environment. It also outperforms OMPL RRT-Connect on all environments other than *cage*, finding faster initial solutions with additional ASAO guarantees. The initial solution time for FCIT* is consistently within an order of magnitude of that of VAMP RRT-Connect on all

environments except for the difficult *cage* environment (Tbl. I). This environment seems well suited for bidirectional planners, as also evidenced by the performance of AIT* relative to the other OMPL ASAO planners.

FCIT* outperforms all tested OMPL ASAO planners on all environments, finding higher quality solutions in less time. All tested OMPL planners show a delay before finding initial solutions (Fig. 2). This can be attributed to overhead in OMPL, even though only planning time is reported for these planners.

We believe that the introduction of VAMP poses an open question of how to best leverage trivialized edge evaluation, since so much existing planning research has been focused on avoiding these operations under the assumption that they are computationally expensive. FCIT* is an initial answer to this question, but it is clear there is further research to be done on the topic. We are particularly interested in finding ways to apply the benefits of bidirectional search to FCIT*, potentially closing the gap to RRT-Connect’s performance on difficult environments.

VI. CONCLUSIONS

Motion planning is an ongoing and important area of research in robotics. This paper leverages VAMP to reduce the cost of edge evaluation and presents FCIT*, the first fully connected, informed, anytime almost-surely asymptotically optimal planner. FCIT* leverages inexpensive edge evaluations to build and search a fully connected graph instead of limiting connections to near a theoretical lower bound. This allows it to fully exploit all samples in a given approximation without requiring nearest-neighbours structures, instead considering every possible edge in the approximation and yielding better solutions in less time and with fewer samples placed.

The benefits of leveraging VAMP to search a fully connected graph are demonstrated on hundreds of problems across seven different planning environments. FCIT* demonstrates performance comparable to that of the fastest planner, VAMP’s RRT-Connect, on almost all environments tested and with additional guarantees. It outperforms all other tested VAMP and OMPL ASAO planners, finding initial solutions faster, of higher quality, and more consistently, and outperforms OMPL’s RRT-Connect on all but the most difficult class of problems tested, all while maintaining ASAO guarantees.

Information on the implementation of FCIT* is available at <https://robotic-esp.com/code/fcitstar/>.

REFERENCES

- [1] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [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, pp. 100–107, 1968.
- [3] L. Kavraki, P. Svestka, J.-C. Latombe, and M. 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.
- [4] S. LaValle, “Rapidly-exploring random trees : a new tool for path planning,” *Research Report 9811*, 1998.
- [5] J. Kuffner and S. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 995–1001 vol.2, 2000.
- [6] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.
- [7] 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*, vol. 39, pp. 543–567, 2017.
- [8] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, “Motion policy networks,” 2022.
- [9] C. Chamzas, C. Quintero-Peña, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, “MotionBenchMaker: A tool to generate and benchmark motion planning datasets,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, 2022.
- [10] K. Solovey and M. Kleinbort, “The critical radius in sampling-based motion planning,” 2018.
- [11] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 05 2003.
- [12] L. Janson 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, pp. 883–921, 2013.
- [13] L. Janson, B. Ichter, and M. Pavone, “Deterministic sampling-based motion planning: Optimality, complexity, and performance,” *The International Journal of Robotics Research*, vol. 37, pp. 46–61, 2015.
- [14] M. W. Tsao, K. Solovey, and M. Pavone, “Sample complexity of probabilistic roadmaps via ϵ -nets,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2196–2202, 2019.
- [15] W. Thomason, Z. Kingston, and L. E. Kavraki, “Motions in microseconds via vectorized sampling-based planning,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8749–8756, 2024.
- [16] 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.
- [17] S. Chitta, I. Sucan, and S. Cousins, “Moveit!,” *IEEE robotics & automation magazine*, vol. 19, no. 1, pp. 18–19, 2012.
- [18] A. Mandalika, O. Salzman, and S. Srinivasa, “Lazy receding horizon A* for efficient path planning in graphs with expensive-to-evaluate edges,” 2018.
- [19] K. K. Hauser, “Lazy collision checking in asymptotically-optimal motion planning,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2951–2957, 2015.
- [20] O. Salzman and D. Halperin, “Asymptotically near-optimal RRT for fast, high-quality, motion planning,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4680–4685, 2013.
- [21] M. P. Strub and J. D. Gammell, “Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 130–136, 2020.
- [22] L. Chen, L. Yu, S. Libin, and Z. Jiwen, “Greedy BIT* (GBIT*): greedy search policy for sampling-based optimal planning with a faster initial solution and convergence,” *International Conference on Computer, Control and Robotics (ICCCR)*, pp. 30–36, 2021.
- [23] P. T. Kyaw, A. V. Le, L. Yi, V. Prabakaran, M. R. Elara, D. T. Vo, and M. B. Vu, “Greedy heuristics for sampling-based motion planning in high-dimensional state spaces,” *ArXiv*, vol. abs/2405.03411, 2024.
- [24] M. P. Strub and J. D. Gammell, “Adaptively Informed Trees (AIT*) and Effort Informed Trees (EIT*): Asymmetric bidirectional sampling-based path planning,” *The International Journal of Robotics Research (IJRR)*, vol. 41, pp. 390–417, Apr. 2022.
- [25] L. Zhang, Z. Bing, K. Chen, L. Chen, K. Cai, Y. Zhang, F. Wu, P. Krumbholz, Z. Yuan, S. Haddadin, and A. Knoll, “Flexible informed trees (FIT*): Adaptive batch-size approach in informed sampling-based path planning,” 2023.
- [26] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, “Analysis of probabilistic roadmaps for path planning,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.
- [27] L. Jaillet, A. Yershova, S. La Valle, and T. Simeon, “Adaptive tuning of the sampling domain for dynamic-domain RRTs,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2851–2856, 2005.
- [28] J. Kuffner and S. LaValle, “An efficient approach to path planning using balanced bidirectional rrt search,” *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep.*, 2005.
- [29] C. W. Ramsey, Z. Kingston, W. Thomason, and L. E. Kavraki, “Collision-affording point trees: Simd-amenable nearest neighbors for fast collision checking,” in *Robotics: Science and Systems (RSS)*, 2024.
- [30] R. Geraerts and M. H. Overmars, “Creating high-quality paths for motion planning,” *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.
- [31] K. Hauser and V. Ng-Thow-Hing, “Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts,” in *2010 IEEE International Conference on Robotics and Automation*, pp. 2493–2498, 2010.
- [32] J. Pan, L. Zhang, and D. Manocha, “Collision-free and smooth trajectory computation in cluttered environments,” *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1155–1175, 2012.