

Tree-Based Grafting Approach for Bidirectional Motion Planning with Local Subsets Optimization

Liding Zhang¹, Yao Ling¹, Zhenshan Bing^{2,1}, Fan Wu¹, Sami Haddadin¹, *Fellow, IEEE*,
Alois Knoll¹, *Fellow, IEEE*

Abstract—Bidirectional motion planning often reduces planning time compared to its unidirectional counterparts. It requires connecting the forward and reverse search trees to form a continuous path. However, this process could fail and restart the asymmetric bidirectional search due to the limitations of lazy-reverse search. To address this challenge, we propose Greedy GuILD Grafting Trees (G3T*), a novel path planner that grafts invalid edge connections at both ends to re-establish tree-based connectivity, enabling rapid path convergence. G3T* employs a greedy approach using the minimum Lebesgue measure of guided incremental local densification (GuILD) subsets to optimize paths efficiently. Furthermore, G3T* dynamically adjusts the sampling distribution between the informed set and GuILD subsets based on historical and current cost improvements, ensuring asymptotic optimality. These features enhance the forward search's growth towards the reverse tree, achieving faster convergence and lower solution costs. Benchmark experiments across dimensions from $\mathbb{R}^2$ to $\mathbb{R}^8$ and real-world robotic evaluations demonstrate G3T*'s superior performance compared to existing single-query sampling-based planners. A video showcasing our experimental results is available at: <https://youtu.be/3mfCRL5SQUU>.

Index Terms—Bidirectional-tree grafting, greedy local subsets, optimal motion planning, sampling-based path planning

I. INTRODUCTION

ROBOT motion planning focuses on finding collision-free paths between start and goal configurations while avoiding obstacles [1]. In high-dimensional configuration spaces (C -spaces), sampling-based algorithms are popular in robotics for their simplicity, fast exploration, and robustness against the curse of dimensionality. [2], [3]. In single-query motion planning, unidirectional planners grow a single tree from either the start or goal state, while bidirectional planners grow two trees—one forward from the start and one backward from the goal [4]. Bidirectional planners are generally more efficient in finding solutions, their performance is often hindered by edge checks connecting the two trees, particularly in narrow passages and uneven sampling densities. We propose a grafting approach to regionally reconnect invalid edges from both trees, expediting the path search. In this paper, we aim to solve the motion planning problem as quickly as possible.

Manuscript received: January, 2, 2025; Revised March, 10, 2025; Accepted April, 10, 2025. This paper was recommended for publication by Editor Júlia Borrás Sol upon evaluation of the Associate Editor and Reviewers' comments. (Corresponding author: Zhenshan Bing.)

¹ Liding Zhang, Yao Ling, Zhenshan Bing, Fan Wu, Sami Haddadin, Alois Knoll are with the School of Computation, Information and Technology (CIT), Technical University of Munich, 80333 Munich, Germany liding.zhang@tum.de

² Zhenshan Bing is also with the State Key Laboratory for Novel Software Technology and the School of Science and Technology, Nanjing University (Suzhou Campus), China.

Digital Object Identifier (DOI): see top of this page.

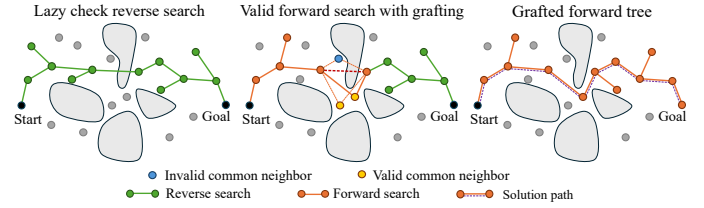


Fig. 1. Illustrates the grafting process in asymmetric bidirectional search. The forward search tree (orange) and the lazy-reverse search tree (green) grow independently from the start and goal configurations. Due to the limitations of lazy search, invalid edges can hinder the search process. Common neighbors are extracted and evaluated based on their heuristics to bypass obstacles (gray regions). Invalid common neighbors (blue) fail to bypass the obstacle, whereas valid common neighbors (yellow) successfully connect the two trees.

A. Related Work

Graph-based algorithms, such as Dijkstra's [5], explore all possible paths to compute the shortest route, while A* [6] leverages heuristic functions to guide exploration. Sampling-based methods, including Rapidly-exploring Random Trees (RRT) [7] and Probabilistic Roadmaps (PRM) [8] efficiently solve high-dimensional, multi-constrained path planning problems. RRT grows a tree incrementally from the start toward the goal, where RRT-Connect [9] enhances efficiency by growing trees from both ends. RRT* [10] offering anytime solutions with guaranteed optimality by nearest neighbors [11] and rewiring. Informed RRT* [12] accelerates path optimization by limiting the search space via elliptical informed subsets [13]. Batch Informed Trees (BIT*) [14], [15] integrates Informed RRT* and Fast Marching Trees (FMT*) [16], advancing sampling techniques by grouping states (*batch*) into compact implicit *random geometric graphs* (RGGs) [17]. Adaptively Informed Trees (AIT*) [18], [19] and Effort Informed Trees (EIT*) [18] employ asymmetrical *forward-reverse* search with sparse collision checks in the reverse search, enhancing exploration efficiency. Flexible Informed Trees (FIT*) [20] utilize adaptive batch sizes for faster initial convergence. Fully Connected Informed Trees (FCIT*) [21] utilize single instruction/multiple data parallelism to reduce the computational cost of edge checks. However, those asymmetric bidirectional planners inefficiently restart the lazy-reverse tree when the forward tree encounters invalid edges. Our grafting method (Fig. 1) can utilize neighbor states of invalid edges to continue the forward search, thereby increasing the convergence rate.

The greedy informed set [22] improves convergence to the optimal by basing the informed set on the maximum admissible estimated cost of the path. Guided Incremental Local Densification (GuILD) [23] further improves path optimization by refining the sampling process within local subsets of the problem domain. However, the informed set may converge

slowly in complex environments with narrow passages. The greedy informed set and GuILD subsets are not asymptotically optimal, which can lead to suboptimal solutions. Furthermore, GuILD's dependence on carefully tuned beacon point selection increases the risk of getting trapped in local optima. To address these limitations, we propose a mathematically defined greedy beacon selector with a historically non-uniform distributed sampling strategy that maintains asymptotic optimality while incorporating greedy path optimization principles.

B. Proposed Algorithm and Original Contribution

This paper proposes Greedy GuILD Grafting Trees (G3T)*, a novel asymmetric bidirectional planner that accelerates motion planning in high-dimensional spaces. G3T* features a tree-based grafting method to reuse neighbor states for bridging invalid edges, avoiding costly tree restarts, and enhancing convergence in narrow passages. G3T* further employs beacon selection based on Lebesgue measures, efficiently targeting high-potential sampling regions to improve path quality and reduce computational overhead. This ensures robust performance in complex environments. To maintain asymptotic optimality, G3T* uses a historical sampling strategy guided by the current and historical cost improvement metrics. Experiments with state-of-the-art (SOTA) planners on cross-dimensional benchmarks and real-world robotic manipulation tasks showcase G3T*'s effectiveness in converging faster initial solutions (i.e., path cost) and enhanced path quality (i.e., path optimization).

The contributions of this work are summarized as follows:

- 1) This paper presents a new tree-based grafting method to improve the convergence rate for bidirectional planners.
- 2) The optimal selection of beacons and sampling regions via the Lebesgue measure for greedy GuILD subsets.
- 3) Historical distributed sampling strategy based on cost improvements to ensure asymptotic optimality.
- 4) Demonstrate the G3T* effectiveness across multiple domains with continuous goal regions in robotic systems.

II. PROBLEM FORMULATION AND PRELIMINARIES

A. Problem Definition

The Optimal Path Planning Problem [10]: Given a planning problem with state space $\mathcal{X} \subseteq \mathbb{R}^n$, where $\mathcal{X}_{\text{obs}}$ denotes states in collision with obstacles, and $\mathcal{X}_{\text{free}} = \text{cl}(\mathcal{X} \setminus \mathcal{X}_{\text{obs}})$ represents permissible states, where $\text{cl}(\cdot)$ represents the *closure* of a set. The initial state is $v_{\text{start}} \in \mathcal{X}_{\text{free}}$, and the desired final states are in $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$. A continuous map $\xi : [0, 1] \mapsto \mathcal{X}$ represents a collision-free path, and Σ is the set of all nontrivial paths.

The optimal solution, denoted as ξ^* , is the path minimizing a chosen cost function $c : \Sigma \mapsto \mathbb{R}_{\geq 0}$. This path connects the initial state v_{start} to any goal state $v_{\text{goal}} \in \mathcal{X}_{\text{goal}}$ through the free space:

$$\xi^* = \arg \min_{\xi \in \Sigma} \{c(\xi) | \xi(0) = v_{\text{start}}, \xi(1) \in v_{\text{goal}}, \forall t \in [0, 1], \xi(t) \in \mathcal{X}_{\text{free}}\}. \quad (1)$$

where $\mathbb{R}_{\geq 0}$ denotes non-negative real numbers. The cost of the optimal path is c^* . Considering a discrete set of states, $\mathcal{X}_{\text{sample}} \subset \mathcal{X}$, represented as a graph with edges determined algorithmically by a transition function, we can characterize

Algorithm 1: Greedy GuILD Grafting Trees (G3T*)

Input : Start vertex v_{start} , goal region $\mathcal{X}_{\text{goal}}$
Output : Optimal feasible path $\mathcal{T}_{\mathcal{F}}$

```

1  $\mathcal{X}_{\text{sample}} \leftarrow \{v_{\text{goal}}\}$ ,  $E_{\mathcal{F}} \leftarrow \emptyset$ ,  $\mathcal{T}_{\mathcal{F}} = (V_{\mathcal{F}}, E_{\mathcal{F}})$ 
2 while not terminateCondition() do
3    $\mathcal{X}_{\text{sample}} \stackrel{\text{red}}{\leftarrow} \text{historicalSampling}()$ 
4    $r \leftarrow \text{getRadius}()$  ▷ calculate RGG radius
5    $\mathcal{T}_{\mathcal{R}} \leftarrow \text{lazyReverseSearch}()$ 
6   while couldImproveForwardSearch( $\mathcal{T}_{\mathcal{R}}$ ) do
7      $E_{\mathcal{F}} \leftarrow \text{forwardSearch}(\mathcal{T}_{\mathcal{R}})$ 
8     if isCollide( $E_{\mathcal{F}}$ ) then
9        $\hat{E}_{\mathcal{P}} \leftarrow \text{grafting}(E_{\mathcal{F}}, r)$ 
10      if  $\hat{E}_{\mathcal{P}} \neq \emptyset$  then
11        continueForwardSearch( $\hat{E}_{\mathcal{P}}$ )
12      else
13        updateLazyReverseSearch()
14      else
15         $\mathcal{T}_{\mathcal{F}} \stackrel{\text{red}}{\leftarrow} E_{\mathcal{F}}$  ▷ add the valid edges into the tree
16    prune( $\mathcal{X}_{\text{sample}}$ )
17 return  $\mathcal{T}_{\mathcal{F}}$ 
```

its properties using a probabilistic model, specifically implicit dense RGGs, i.e., $\mathcal{X}_{\text{sample}} = v \sim \mathcal{U}(\mathcal{X})$, as discussed in [17].

B. Notation

The state space for the planning problem is denoted by $\mathcal{X} \subseteq \mathbb{R}^n$, where $n \in \mathbb{N}$. The start vertex is represented by $v_{\text{start}} \in \mathcal{X}$, and the goal states are denoted by $v_{\text{goal}} \in \mathcal{X}$. The set of sampled states is denoted by $\mathcal{X}_{\text{sample}}$. The forward and reverse search trees are denoted by $\mathcal{T}_{\mathcal{F}}$ and $\mathcal{T}_{\mathcal{R}}$, respectively. The vertices and edges in these trees, denoted by $V_{\mathcal{F}}$ and $V_{\mathcal{R}}$, correspond to valid states. The edges in the forward tree, $E_{\mathcal{F}} \subseteq V_{\mathcal{F}} \times V_{\mathcal{F}}$, represent valid connections between states. Each edge connects a source state, v_s , and a target state, v_t , denoted as $E(v_s, v_t)$. An admissible estimate of the cost of a path is denoted by $\hat{f} : \mathcal{X} \rightarrow [0, \infty)$. For sets A , B , and C , where $B, C \subseteq A$, the notation $B \stackrel{\text{red}}{\leftarrow} C$ denotes the operation $B \leftarrow B \cup C$, and $B \stackrel{\text{blue}}{\leftarrow} C$ denotes the operation $B \leftarrow B \setminus C$.

G3T-specific Notation:* The grafting edges set $E_{\mathcal{P}} \subset E_{\mathcal{F}}$, which are defined by the common neighbor queue $\mathcal{Q}_{\text{comNeigh}}$. The optimal heuristic-sorted grafting edge is denoted as $\hat{E}_{\mathcal{P}}$. The beacon vertex is represented as $v_{\text{beacon}} \in V_{\mathcal{F}}$. The front and back greedy GuILD subsets are denoted by $\mathcal{X}_{G^2_f}$ and $\mathcal{X}_{G^2_b}$, with the corresponding greedy costs denoted as c_{greedy_f} and c_{greedy_b} , respectively. The greedy GuILD subsets are denoted as $\mathcal{X}_{G^2} \subseteq \mathcal{X}$, where $\mathcal{X}_{G^2}$ is a subset of the informed set $\mathcal{X}_{\hat{f}}$. The current and historical cost improvements are denoted by Υ_{CCI} and Υ_{HCI} , respectively. The batch size of the current batch is denoted as $\mathcal{M}_{\text{curr}}$, with the distribution of $\mathcal{M}_{\mathcal{X}_{G^2}}$ in the greedy GuILD and $\mathcal{M}_{\mathcal{X}_{\hat{f}}}$ in the informed set.

III. GREEDY GUILD GRAFTING TREES (G3T*)

In this section, we first introduce the concept of grafting. Next, we illustrate GuILD's greedy approach. Then, we propose the historical sampling. Finally, we analyze the probabilistic completeness and asymptotic optimality of G3T*.

Algorithm 2: G3T*: Grafting

Input : (v_s, v_t) - The invalid edge, r - RGG radius
Output: $\hat{E}_P$ - Valid edge-pair

```

1  $N_{\text{source}} \leftarrow \text{nearest}(v_s, r), N_{\text{target}} \leftarrow \text{nearest}(v_t, r)$ 
2  $Q_{\text{comNeigh}} \leftarrow \emptyset$   $\triangleright$  initialize common neighbor set
3 foreach  $v_i \in N_{\text{source}}$  do
4   if  $v_i \in N_{\text{target}}$ 
5      $Q_{\text{comNeigh}} \leftarrow^+ v_i$   $\triangleright$  common neighbors of invalid-edge
6 if  $Q_{\text{comNeigh}} \neq \emptyset$ 
7   foreach  $v_i \in Q_{\text{comNeigh}}$  do
8      $\hat{E}_P \leftarrow \text{bestEdgePair}(\{(v_s, v_i), (v_i, v_t)\})$   $\triangleright$  Eq. 9
9     if  $\text{lazyCheck}(\hat{E}_P)$   $\triangleright$  inexpensive edge-pair filter
10      if  $\text{fullCheck}(\hat{E}_P)$   $\triangleright$  valid edge-pair
11         $V_F \leftarrow v_i$   $\triangleright$  insert into forward tree
12         $\mathcal{T}_F \leftarrow^+ \text{expand}(v_i)$   $\triangleright$  add parent and children
13      return  $\hat{E}_P$ 
14 return null

```

A. Bidirectional Tree-based Grafting Method

The main idea behind our proposed grafting is to expand the source and target endpoints of invalid edges in the bidirectional tree search. First, we obtain the common neighbor queue for both the source and target vertices and then form two new edges as an edge-pair to continue the forward search.

Our algorithm is built on FIT*, which utilizes lazy checking in the reverse search to quickly explore the $\mathcal{C}$ -spaces (Alg. 1), while performing a full collision check in the forward search to ensure the validity of the path. We define the connection radius $r(m)$ during the construction of the RGGs. This defines the nearest neighbor samples and the density of the graph. The radius decreases as the number of sampling vertices increases, leading to a denser RGG. Where $r(m)$ can be formulated as:

$$r(m) := \gamma \left(\frac{\log m}{m} \right)^{1/n}, \quad (2)$$

$$\gamma := 2\eta \left(1 + \frac{1}{n} \right)^{\frac{1}{n}} \left(\frac{\min\{\lambda(\mathcal{X}_{\text{free}}), \lambda(\mathcal{X}_{\hat{f}})\}}{B_{1,n}} \right)^{1/n}, \quad (3)$$

here, γ is a scaling factor tied to problem dimensions and the measure of the region, which defines the connectivity of RGGs [24]. The term $\min\{\lambda(\mathcal{X}_{\text{free}}), \lambda(\mathcal{X}_{\hat{f}})\}$ ensures γ is computable from $\mathcal{X}_{\text{free}}$ without an initial solution. m represents the number of sampled states, $\eta > 1$ is the RGG constant, n is the dimensionality of the problem domain, and $\lambda(\cdot)$ denotes the Lebesgue measure (i.e., geometric volume). $B_{1,n}$ represents the volume of the unit n -dimensional ball, $\mathcal{X}_{\hat{f}}$ is the informed set. In RGG, for any undirected graph $\mathcal{G}_n$, two vertices v and v' are said to be connected if there exists a path in $\mathcal{G}_n$ that links v to v' . The connectivity of $\mathcal{G}_n$ is:

$$\lim_{m \rightarrow \infty} \mathbb{P}(\mathcal{G}_n \text{ is connected}) := \begin{cases} 0 & \text{if } \gamma < \gamma^* \\ 1 & \text{if } \gamma > \gamma^* \end{cases}, \quad (4)$$

where $\gamma^* = 2(2n \cdot B_{1,n})^{-1/n}$. If every pair of vertices in the graph is connected, the graph is considered connected [24]. In fully connected RGG, the nearest neighbor is the set of all

vertices within a specified radius $r(m)$ of a given vertex v . Mathematically, the neighborhood $N(v)$ is expressed as:

$$N(v) := \{v' \in \mathcal{X}_{\text{sample}} : \|v - v'\|_2 \leq r(m)\}, \quad (5)$$

where $\|v - v'\|_2$ denotes the distance between vertices v and v' . If a vertex v' is a neighbor of both v_s and v_t , it is defined as a common neighbor of v_s and v_t . The common neighbor queue Q_{comNeigh} of v_s and v_t is defined as:

$$Q_{\text{comNeigh}}(v_s, v_t) = \{v' \in \mathcal{X}_{\text{sample}} \mid v' \in N(v_s) \cap N(v_t)\}, \quad (6)$$

the grafting method (Alg. 2) applies when the forward and reverse trees are connected in a bidirectional search. If the connecting edge $E(v_s, v_t)$ fails full collision detection (e.g., intersects an obstacle), the traditional asymmetric bidirectional search increases the lazy check resolution and restarts the reverse search. The new reverse tree is expanded from the goal and may have a different structure due to varying resolutions in lazy checking. It will continue attempting to connect to the source vertex of the invalid edge, v_s , making the search slower and more prolonged. To address this limitation, the algorithm extracts the common neighbors of the search source vertex v_s and the target vertex v_t and sorts these common neighbors according to the search heuristics. Each common neighbor is evaluated to identify a valid edge-pair to bypass the obstacle. An edge-pair consists of two edges: (v_s, v') and (v', v_t) . All edge-pairs form an edge queue E_P described as:

$$E_P := \bigcup_{v' \in Q_{\text{comNeigh}}} \{(v_s, v'), (v', v_t)\}, \quad (7)$$

where v' represents a common neighbor of v_s and v_t , which is lexicographically ordered based on the heuristic function value of its corresponding edge-pair using the heuristic key:

$$\text{key}_{Q_{\text{comNeigh}}}^{\text{G3T}^*}(v') := \begin{cases} \hat{c}(v_s, v') + \hat{c}(v', v_t), \\ \bar{e}(v_s, v') + \bar{e}(v', v_t), \end{cases} \quad (8)$$

here, $\hat{c}$ represents the cost function used to evaluate the edge cost, and $\bar{e}$ is the heuristic effort required to validate the edge. By sorting the common neighbors v' based on $\text{key}_{Q_{\text{comNeigh}}}^{\text{G3T}^*}(v')$, the algorithm efficiently identifies optimal edge-pairs that bypass obstacles and re-connect the forward and reverse trees.

$$\hat{E}_P := \arg \min_{\{(v_s, v'), (v', v_t)\} \in E_P} \{\text{key}_{Q_{\text{comNeigh}}}^{\text{G3T}^*}(v') \mid v' \in Q_{\text{comNeigh}}, \forall Q_{\text{comNeigh}} \in \mathcal{X}_{\text{sample}}\}, \quad (9)$$

where $\hat{E}_P$ represents the optimal edge-pairs. G3T* extracts the highest-priority common neighbor v' from the common neighbor queue Q_{comNeigh} to construct the edge-pairs E_P . The validity of the edge-pair is first evaluated through a less computationally expensive lazy check, followed by the more costly full check (Alg. 2, lines 8-10). If the edge-pair passes the full check, it returns to continue the forward search, bypassing obstacles and connecting the forward and reverse trees. This approach accelerates pathfinding, and eliminates the need for restarting the reverse search.

B. Greedy Approach for Beacon Selector in GuILD

In informed path planning, once the solution path ξ is found, its cost $c(\xi)$ defines the informed set $\mathcal{X}_{\hat{f}}$ for sampling and

Algorithm 3: G3T*: Greedy GuILD Subsets

Input : $v_{\text{start}}, v_{\text{goal}}, \xi$ - Path of solution from start to goal
Output: Greedy measure subset $\mathcal{X}_{G^2}$

```

1  $\mathcal{X}_{\text{GuILD}_f} \leftarrow \mathcal{X}_{\text{free}}, \mathcal{X}_{\text{GuILD}_b} \leftarrow \mathcal{X}_{\text{free}}$   $\triangleright$  initialize GuILD sets
2 foreach  $v_i \in \xi$  do
3    $\xi_{f_i} \leftarrow \text{pathBetween}(v_{\text{start}}, v_i)$   $\triangleright$  front-part path
4    $\xi_{b_i} \leftarrow \text{pathBetween}(v_i, v_{\text{goal}})$   $\triangleright$  back-part path
5    $c_{f_i} \leftarrow c(\xi_{f_i}), c_{b_i} \leftarrow c(\xi_{b_i})$   $\triangleright$  calculate path costs
6    $\mathcal{X}_{f_i} \leftarrow \text{hyperEllipsoid}(v_{\text{start}}, v_i, c_{f_i})$ 
7    $\mathcal{X}_{b_i} \leftarrow \text{hyperEllipsoid}(v_i, v_{\text{goal}}, c_{b_i})$ 
8    $\lambda(\mathcal{X}_{v_i}) \leftarrow \lambda(\mathcal{X}_{f_i}) + \lambda(\mathcal{X}_{b_i})$   $\triangleright$  total Lebesgue measures
9   if  $\lambda(\mathcal{X}_{v_i}) \leq \lambda(\mathcal{X}_{\text{GuILD}_f}) + \lambda(\mathcal{X}_{\text{GuILD}_b})$   $\triangleright$  Eq. 14
10     $v_{\text{beacon}} \leftarrow v_i$   $\triangleright$  minimal measure beacon vertex
11     $\xi_f \leftarrow \xi_{f_i}, \xi_b \leftarrow \xi_{b_i}$   $\triangleright$  update minimal paths
12     $\mathcal{X}_{\text{GuILD}_f} \leftarrow \mathcal{X}_{f_i}, \mathcal{X}_{\text{GuILD}_b} \leftarrow \mathcal{X}_{b_i}$   $\triangleright$  update subsets
13  $c_{\text{greedy}_f} = 0, c_{\text{greedy}_b} = 0$   $\triangleright$  initialize greedy costs
14 foreach  $v_j \in \xi_f$  do  $\triangleright$  get greedy cost for front-part path
15    $d(v_j) \leftarrow \|v_{\text{start}} - v_j\|_2 + \|v_j - v_{\text{beacon}}\|_2$   $\triangleright$  front-distance
16   if  $d(v_j) \geq c_{\text{greedy}_f}$   $\triangleright$  Eq. 20
17      $v_f \leftarrow v_j$   $\triangleright$  set front-part greedy vertex
18      $c_{\text{greedy}_f} \leftarrow d(v_j)$ 
19 foreach  $v_k \in \xi_b$  do  $\triangleright$  get greedy cost for back-part path
20    $d(v_k) \leftarrow \|v_{\text{beacon}} - v_k\|_2 + \|v_k - v_{\text{goal}}\|_2$   $\triangleright$  back-distance
21   if  $d(v_k) \geq c_{\text{greedy}_b}$   $\triangleright$  Eq. 21
22      $v_b \leftarrow v_k$   $\triangleright$  set back-part greedy vertex
23      $c_{\text{greedy}_b} \leftarrow d(v_k)$ 
24  $\mathcal{X}_{G^2_f} \leftarrow \text{hyperEllipsoid}(v_{\text{start}}, v_{\text{beacon}}, c_{\text{greedy}_f})$ 
25  $\mathcal{X}_{G^2_b} \leftarrow \text{hyperEllipsoid}(v_{\text{beacon}}, v_{\text{goal}}, c_{\text{greedy}_b})$ 
26  $\mathcal{X}_{G^2} \leftarrow \mathcal{X}_{\text{free}} \cap (\mathcal{X}_{G^2_f} \cup \mathcal{X}_{G^2_b})$   $\triangleright$  update greedy GuILD sets
27 return  $\mathcal{X}_{G^2}$ 

```

search. The informed set includes all paths with a cost less than $c(\xi)$, ensuring that any optimal path lies within it [13]. This reduces the search space and improves search efficiency by focusing on regions likely to contain optimal solutions:

$$\mathcal{X}_{\hat{f}} = \{v \in \mathcal{X}_{\text{free}} \mid \|v - v_{\text{start}}\|_2 + \|v_{\text{goal}} - v\|_2 < c(\xi)\}, \quad (10)$$

where $\|\cdot\|_2$ denotes the L_2 norm (i.e., the Euclidean distance) between two vertices. The L_2 informed set is defined as the intersection of the free space $\mathcal{X}_{\text{free}}$ and an n -dimensional hyperellipsoid, which is symmetric along its transverse axis, we define it as **Symmetric HyperEllipsoid (SHE)**:

$$\mathcal{X}_{\hat{f}} := \mathcal{X}_{\text{free}} \cap \mathcal{X}_{\text{SHE}}, \quad (11)$$

$$\mathcal{X}_{\text{SHE}} := \{v \in \mathbb{R}^n \mid \|v - v_{\text{start}}\|_2 + \|v_{\text{goal}} - v\|_2 < c(\xi)\}, \quad (12)$$

where the SHE of n -dimensional $\mathcal{C}$ -space ($\mathbb{R}^n$) is defined by the start vertex, the goal vertex, and the current path cost $c(\xi)$. The Lebesgue Measure (i.e., hypervolume) of the informed set in an n -dimensional Euclidean space $\mathbb{R}^n$ is given by:

$$\lambda(\mathcal{X}_{\hat{f}}) \leq \lambda(\mathcal{X}_{\text{SHE}}) = \frac{c(\xi) (c(\xi)^2 - c_{\min}^2)^{\frac{n-1}{2}} B_{1,n}}{2^n}, \quad (13)$$

where $B_{1,n} := \pi^{\frac{n}{2}} / \Gamma(\frac{n}{2} + 1)$ is the Lebesgue measure of a n -dimensional unit ball, with the theoretical minimum cost $c_{\min} := \|v_{\text{goal}} - v_{\text{start}}\|_2$ [13]. The informed set often provides limited improvements in sample efficiency, as its measure, $\lambda(\mathcal{X}_{\hat{f}})$, depends on the current solution cost. A high-cost solu-

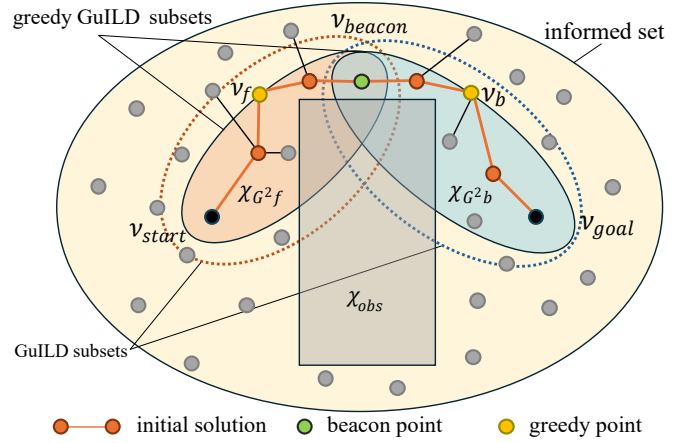


Fig. 2. Illustrates the concept of GuILD and greedy GuILD. The problem domain is reduced to a smaller region by selecting a beacon point (green) along the solution path (orange). The first hyper-spheroid focuses on the start point and the beacon point, while the second hyper-spheroid focuses on the beacon point and the goal. Greedy vertices (yellow) are selected to evaluate cost values and enhance path optimization.

tion may lead to a large informed set, potentially comparable to or larger than the full state space measure $\lambda(\mathcal{X})$, which slows convergence, increases computational costs, and reduces path optimization efficiency. GuILD defines a vertex as a beacon vertex v_{beacon} , decomposing the planning problem into two smaller subproblems. This reduces the search domains and accelerates optimization convergence, partitioning the sampling space into two n -dimensional prolate hyperspheroids, $\mathcal{X}_{\text{GuILD}_f}$ and $\mathcal{X}_{\text{GuILD}_b}$. The greedy approach for beacon point selection v_{beacon} in GuILD is defined via vertices along the solution path ξ is selected as the beacon vertex that minimizes the sum of the Lebesgue measures of $\mathcal{X}_{\text{GuILD}_f}$ and $\mathcal{X}_{\text{GuILD}_b}$ (Alg. 3):

$$v_{\text{beacon}} := \arg \min_{v \in \xi} \{\lambda(\mathcal{X}_{\text{GuILD}_f}) + \lambda(\mathcal{X}_{\text{GuILD}_b})\}, \quad (14)$$

$\mathcal{X}_{\text{GuILD}_f}$: Front GuILD is defined by the start vertex v_{start} and the beacon vertex v_{beacon} , with the front-path cost c_f of the first hyperspheroid as its transverse diameter.

$\mathcal{X}_{\text{GuILD}_b}$: Back GuILD is defined by the beacon vertex v_{beacon} and the goal vertex v_{goal} , with the back-path cost c_b of the second hyperspheroid as its transverse diameter.

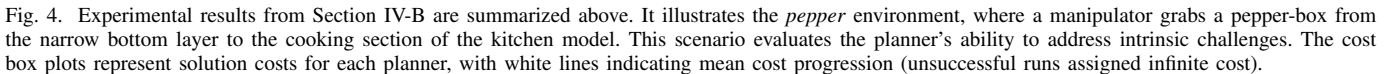
$$\mathcal{X}_{\text{GuILD}_f} := \{v \mid v \in \mathcal{X}, \|v_{\text{start}} - v\|_2 + \|v - v_{\text{beacon}}\|_2 \leq c_f\}, \quad (15)$$

$$\mathcal{X}_{\text{GuILD}_b} := \{v \mid v \in \mathcal{X}, \|v_{\text{goal}} - v\|_2 + \|v - v_{\text{beacon}}\|_2 \leq c_b\}, \quad (16)$$

where the cost c_f is defined as the cost along the current solution path ξ from v_{start} to v_{beacon} and c_b is the cost from v_{beacon} to v_{goal} , respectively. The combination is defined as the total path cost $c(\xi) := c_f + c_b$. By employing the beacon vertex v_{beacon} and the path ξ , the greedy GuILD subsets $\mathcal{X}_{G^2}$ are defined as:

$$\mathcal{X}_{G^2} := \mathcal{X}_{\text{free}} \cap (\mathcal{X}_{G^2_f} \cup \mathcal{X}_{G^2_b}), \quad (17)$$

$\mathcal{X}_{G^2_f}$: Front greedy GuILD is defined by the start vertex v_{start} and the beacon vertex v_{beacon} , with the greedy cost c_{greedy_f} (Alg. 3, line 13-18) as its transverse diameter. The greedy cost c_{greedy_f} is determined by identifying the front greedy vertex v_f along the path ξ_f , which connects v_{start} to v_{beacon} . This vertex v_f maximizes the sum of the L_2 heuristic values, comprising



Υ_{HCI} . Then the marginal solution cost improvement level is reset (Alg. 4, line 18), ensuring that new sample vertices are generated within $\mathcal{X}_{\hat{f}}$. This mechanism guarantees asymptotic optimality. The optimization speed and efficiency are improved while ensuring that the path-planning algorithm eventually converges to the optimal solution over time.

Most informed sampling-based path-planning algorithms are probabilistically complete and asymptotically optimal, and G3T* upholds these properties. By leveraging uniform sampling, G3T* ensures that as the number of iterations i approaches infinity, the entire $\mathcal{C}$ -space is explored:

this indicates that if a feasible path exists, G3T* will identify it, thereby guaranteeing probabilistic completeness.

By Lemmas 56, 71, and 72 in [10], the following holds in the limit of infinitely dense sampling:

where q is the number of samples, $\Sigma_q \subset \Sigma$ represents the set of valid paths identified by the planner, $c : \Sigma \rightarrow [0, \infty)$ is the cost function, and c^* denotes the optimal cost. This indicates that G3T* guarantees asymptotic optimality, but only converges to the optimal path as the number of samples approaches infinity.

To benchmark the performance of motion planning algorithms, this study utilized the Planner-Arena benchmark database [25], along with Planner Developer Tools (PDT) [26] and MoveIt [27]. The proposed G3T* algorithm was compared against SOTA methods in both simulated random scenarios (Fig. 5) and real-world manipulation problems (Fig. 4). The benchmark included multiple algorithms sourced from the Open Motion Planning Library (OMPL) [28], such as RRT-Connect, Informed RRT*, BIT*, AIT*, EIT*, and FIT*. The tests covered simulated environments ranging in dimensionality from $\mathbb{R}^2$ to $\mathbb{R}^8$, focusing primarily on path length (cost) minimization. Key parameters were standardized across planners: the RGG constant η was set to 1.001, and the rewire factor to 1.2. For RRT-based planners, a goal bias of 5%

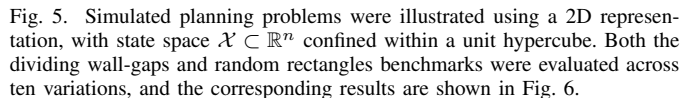


TABLE I
BENCHMARKS EVALUATION COMPARISON (100 RUNS) (FIG. 6)

was implemented, with maximum edge lengths set at 0.5, 1.25, and 3.0 for $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$, respectively. The level-up threshold θ_{thold} for historical sampling is set at three. All planners employed Euclidean distance as the primary metric and effort as heuristics. G3T* leverages tree-based grafting to reconnect invalid edges during the search. It then uses a greedy GuILD subset approach to accelerate path optimization in critical target areas, followed by a historical distribution sampling strategy to ensure asymptotic optimality. The implementation of G3T* planner into OMPL framework is available at: https://github.com/Liding-Zhang/ompl_g3t.git.

The planners were evaluated across two distinct benchmarks in three domains: $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$. In the first scenario, a constrained environment resembling dividing wall gaps (DW) includes multiple corridors with spacious regions, testing the ability of motion planners to navigate multi-narrow spaces

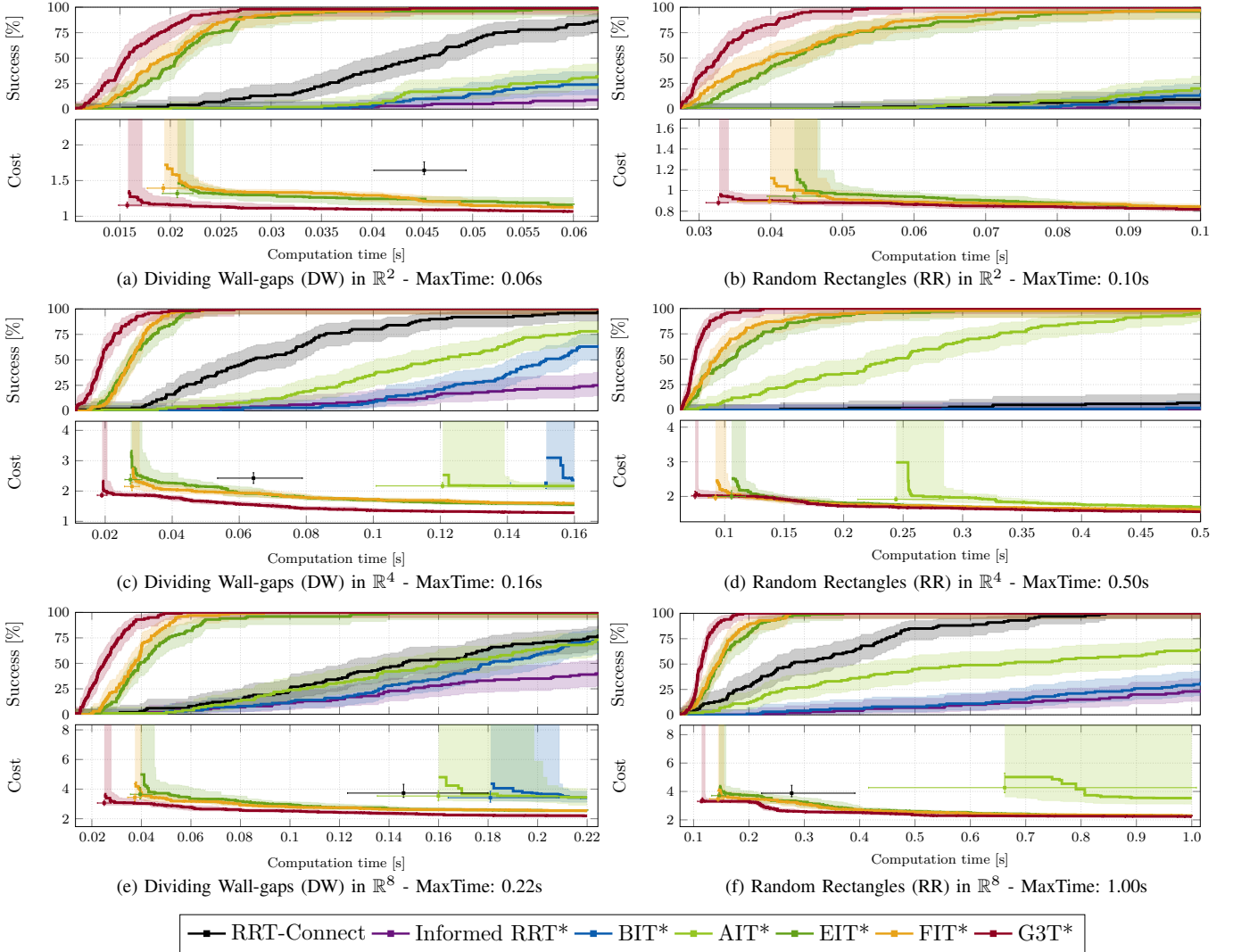


Fig. 6. Detailed experimental results from Section IV-A are presented above. MaxTime is the planner’s maximum allotted planning time. Fig. (a), (c), and (e) depict test benchmark dividing wall-gaps (DW) outcomes in $\mathbb{R}^2$ to $\mathbb{R}^8$, respectively. Panel (b) showcases random rectangle (RR) experiments in $\mathbb{R}^2$, while panels (d) and (f) demonstrate in $\mathbb{R}^4$ and $\mathbb{R}^8$. In the cost plots, boxes represent solution cost and time, with lines showing cost progression for optimal planners (unsuccessful runs have infinite cost). Error bars provide nonparametric 99% confidence intervals for solution cost and time.

while identifying an optimal path to the goal (Fig. 5a). Each planner was tested over 100 runs with computation times for each optimal planner detailed in the labels, using random seeds. The overall success rates and median path lengths for all planners are presented in Fig. 6a, Fig. 6c, and Fig. 6e. Results indicate that G3T* quickly finds an initial solution across dimensions with faster path convergence.

In the second test scenario, random widths were assigned to *axis-aligned hyperrectangles*, which were generated arbitrarily within the *C-space* (Fig. 5b). For each dimension of the *C-space*, random rectangle problems were created, and each planner underwent 100 runs for every instance. Figures 6b, 6d, and 6f demonstrate that the proposed method achieved the highest success rates and the lowest median path costs within the given computation time, compared to the other planners. This suggests that G3T* can optimize paths more quickly and converge faster through grafting. G3T* outperformed the other planners and was able to quickly find an initial solution.

As observed in benchmark Table I, the notable improvement (marked in red) in the median initial times over 100 runs is across different benchmark scenarios, with performance trends

correlating with dimensionality. In the DW – $\mathbb{R}^4$ scenario, G3T* achieves a lower initial median time (0.019s, calculated over 100 trials) compared to FIT* (0.028s). Furthermore, in higher-dimensional scenarios such as RR – $\mathbb{R}^8$ and DW – $\mathbb{R}^8$, G3T* consistently outperforms FIT* by achieving lower initial median times with higher success rate. Specifically, in the DW – $\mathbb{R}^8$ scenario, G3T* demonstrates a notable initial median time improvement of approximately 33.15% over FIT*.

Overall, G3T* shows faster initial path convergence compared to the SOTA planner, thereby enhancing the search efficiency of sampling-based path planning algorithms.

B. Real-world Path Planning Tasks

To evaluate the algorithm’s performance in real-world scenarios, kitchen model pepper problem experiments (Fig. 4) are conducted on an 8-DoF base manipulator (DARKO) to demonstrate the efficiency and scalability of G3T*, compared to three SOTA path planning algorithms: Batch Informed Trees (BIT*) [14], [15], Effort Informed Trees (EIT*) [18], and Flexible Informed Trees (FIT*) [20]. We compare G3T* with EIT* and FIT* in robot manipulation tasks to evaluate

performance in terms of convergence to optimal solution cost and success rate over 30 runs. The *Kitchen-pepper* experiment is conducted on the DARKO robot in a cluttered environment with narrow spaces. A collision-free path is required from the lower-shelf section of the kitchen model (start config.) to the upper-left cooking pan area (goal region). G3T* demonstrates effective grafting for path convergence with greedy GuILD sets for path optimization throughout the experimental tasks.

Each planner was allocated 1.0 seconds to solve this confined, limited space pull-out and manipulate problem. Across 30 trials, EIT* managed a 33.33% success rate with a median solution cost of 20.7864. FIT* had a 70% success rate with a median solution cost of 22.6911. G3T* achieved an 80% success rate with a median solution cost of 17.9354.

In summary, compared to EIT* and FIT*, G3T* achieves the best performance in both finding the initial solution and converging to the optimal solution.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduce the Greedy GuILD Grafting Trees (G3T*), a novel path-planning algorithm that employs a tree-based grafting method for bidirectional search. G3T* defines a common neighbor set for the invalid edges' endpoints and forms new optimal edge-pairs to continue the search. For fast path optimization, G3T* defines a greedy GuILD approach based on the maximal admissible cost of the front-back path as a two-subdomain problem. G3T* utilizes historical distribution sampling to prevent premature convergence to local minima and adjusts the sampling level based on both current and historical cost improvements. By historical sampling, the probabilistic completeness and asymptotic optimality were guaranteed. Simulations across varying dimensions and real-world experiments validate that our algorithm achieves faster initial convergence and shorter path lengths than the benchmarks.

Future work could integrate safety constraints for human-interactive scenarios, and utilize single instruction/multiple data parallelism methods [21] to improve motion planning.

REFERENCES

- [1] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 295–318, 2021.
- [2] L. Zhang, K. Cai, Z. Sun, Z. Bing, C. Wang, L. Figueredo, S. Haddadin, and A. Knoll, "Motion planning for robotics: A review for sampling-based planners," *Biomimetic Intelligence and Robotics*, vol. 5, no. 1, p. 100207, 2025.
- [3] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, no. 1, 2024.
- [4] S. Nayak and M. W. Otte, "Bidirectional sampling-based motion planning without two-point boundary value solution," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3636–3654, 2022.
- [5] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [6] 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.
- [7] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.
- [8] L. E. Kavraki, P. Svestka, 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.
- [9] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, 2000, pp. 995–1001 vol.2.
- [10] 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.
- [11] L. Zhang, Z. Bing, Y. Zhang, K. Cai, L. Chen, F. Wu, S. Haddadin, and A. Knoll, "Elliptical k-nearest neighbors: Path optimization via coulomb's law and invalid vertices in c-space obstacles," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12 032–12 039, 2024.
- [12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014.
- [13] 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.
- [14] 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 *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.
- [15] 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, no. 5, 2020.
- [16] 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.
- [17] M. Penrose, *Random geometric graphs*. OUP Oxford, 2003, vol. 5.
- [18] 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*, vol. 41, no. 4, pp. 390–417, 2022.
- [19] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3191–3198.
- [20] 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," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3146–3152, 2024.
- [21] T. S. Wilson, W. Thomason, Z. Kingston, L. E. Kavraki, and J. D. Gammell, "Nearest-neighbourless asymptotically optimal motion planning with fully connected informed trees (FCIT*)," *arXiv*, 2024. [Online]. Available: <https://arxiv.org/abs/2411.17902>
- [22] P. T. Kyaw, A. V. Le, P. Veerajagadheswar, M. R. Elara, T. T. Thu, N. H. K. Nhan, P. Van Duc, and M. B. Vu, "Energy-efficient path planning of reconfigurable robots in complex environments," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2481–2494, 2022.
- [23] R. Scalise, A. Mandalika, B. Hou, S. Choudhury, and S. S. Srinivasa, "GuILD: Guided incremental local densification for accelerated sampling-based motion planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 212–10 218.
- [24] K. Solovey, O. Salzman, and D. Halperin, "New perspective on sampling-based motion planning via random geometric graphs," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1117–1133, 2018.
- [25] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.
- [26] J. D. Gammell, M. P. Strub, and V. N. Hartmann, "Planner developer tools (PDT): Reproducible experiments and statistical analysis for developing and testing motion planners," in *Proceedings of the Workshop on Evaluating Motion Planning Performance (EMPP), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
- [27] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "Moveit! task constructor for task-level motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 190–196.
- [28] 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.