

Bidirectional Incremental Generalized Hybrid A*

Sidharth Talia, Oren Salzman, and Siddhartha Srinivasa

Abstract—We focus on the problem of efficient anytime kinodynamic planning for systems with complex dynamics in unstructured environments that make using precomputed motion primitives infeasible. Directly applying A* here is computationally infeasible due to the curse of dimensionality. Methods such as Hybrid A* (HA*) address this by pruning the search tree by discretizing the state space, but the coupling between pruning and discretization resolution can eliminate a vertex on the solution path. The Incremental Generalized Hybrid A* (IGHA*) breaks this coupling by organizing anytime search over a hierarchy of resolutions and by *freezing* vertices for later expansion rather than pruning. However, IGHA* can still freeze a vertex on the solution path, forcing the search to spend expansions elsewhere before reaching it. Our *key insight* is that bidirectional-IGHA* (Bi-IGHA*) not only gains the expected reduction in effective search depth from classical bidirectionality, but also mitigates this *frozen-vertex barrier*: when one search is blocked, the opposing search can reach it at a low resolution. We formalize this structural separation between Bi-IGHA* and IGHA* and empirically show a reduction in effective branching factor beyond the theoretical expectation from classic bidirectionality alone. Through open-loop experiments in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^6$, we show that Bi-IGHA* substantially reduces expansions over IGHA*. Closed-loop experiments further demonstrate improved performance and feasibility in simulation and on a real-world robotic system. [Link to Website.](#)

I. INTRODUCTION

Planning for autonomous systems in complex, unstructured environments—such as high-speed off-road autonomy [1], [2] and aerial robotics [3]—is challenging due to complex, nonlinear dynamics. We are motivated by kinodynamic planning for high-speed off-road autonomous vehicles, where precomputing motion trees is often infeasible because the terrain, friction, and obstacle geometry continuously evolve [1], [4]. In many such settings, the planner has access only to a computationally expensive forward dynamics model rather than a fast analytical solution to the two-point boundary value problem (2pBVP).

For such problems, directly applying A* search in continuous kinodynamic spaces is computationally prohibitive due to the curse of dimensionality. Hybrid A* (HA*) [5] addressed this challenge by using *approximate dominance* [6]; discretizing the state space into grid cells at some *resolution*, with each cell only retaining the lowest-cost vertex to reduce the effective branching factor (Sec. VI-C). The underlying assumption is that the *best cost* from any vertex in the grid cell to the goal is identical. However, this assumption may not always hold, and HA* can even prune a goal-bearing vertex (a vertex on the path to the goal) from the search tree. Incremental Generalized Hybrid A* (IGHA*) [2] avoids this by organizing search over a hierarchy of resolutions in an anytime fashion and *freezing* vertices for expansion in

later search iterations rather than pruning them. However, at a particular resolution, IGHA* can still freeze a goal-bearing vertex, creating a barrier the search overcomes only through many more expansions. We illustrate this through Fig. 1 and its caption, and term this the *frozen vertex barrier*.

Our *key insight* is that when we run two IGHA* searches, one *forward* and one *backward*, yielding solutions when the opposing trees *near meet* [7] (Def. 5.2), then, in addition to the expected reduction in effective branching factor from classical bidirectionality [8], we also get an additional reduction from *bidirectional mitigation* of the frozen vertex barrier. Even if one search gets blocked by frozen vertices, the other search can still meet it at a low resolution (Fig. 2).

In this work, we identify the frozen vertex barrier as a consequence of approximate dominance-based search (Sec. IV, Obs. 6.1), propose *bidirectional-IGHA** (Bi-IGHA*, Sec. V) that preserves IGHA*'s guarantees on monotonic cost improvement and termination with finite expansions (Thm. 6.1), and formally show the structural separation between IGHA* and Bi-IGHA* as a result of the bidirectional mitigation (Sec. VI-B). We empirically show that Bi-IGHA* provides $\approx 10\times$ reduction in expansions on planning problems in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^6$ (Fig. 8) and validate that claim regarding the additional reduction in effective branching factor (Fig. 4). Using closed-loop planning and control experiments in simulation ($\mathbb{R}^4$) as well as in the real world high speed off-road autonomy, we show that Bi-IGHA* significantly outperforms IGHA*.

II. RELATED WORK

A. Bidirectional Graph Search

Bidirectional graph search (see, e.g., [10]–[16]) algorithms have long been employed to improve search efficiency. Recent theoretical advances in this domain have primarily followed two distinct conceptual frameworks [17], [18]. The first, grounded in the theory of Must-Expand Pairs (MEP) [19], led to algorithms like Near-Optimal Bidirectional Search (NBS) [20]. NBS introduced mechanisms to certify optimality even when the two search frontiers do not meet exactly at a vertex. The second framework focuses on the Meet-in-the-Middle Property (MMP), which guarantees that no state is expanded if its cost exceeds half the optimal solution cost [8], with recent work [18] showing ways to reduce computational burden of computing termination conditions [21], [22]. However, both MEP and MMP approaches typically assume a static graph in which vertices can be checked for exact overlap. This assumption often breaks down in kinodynamic planning for off-road autonomy

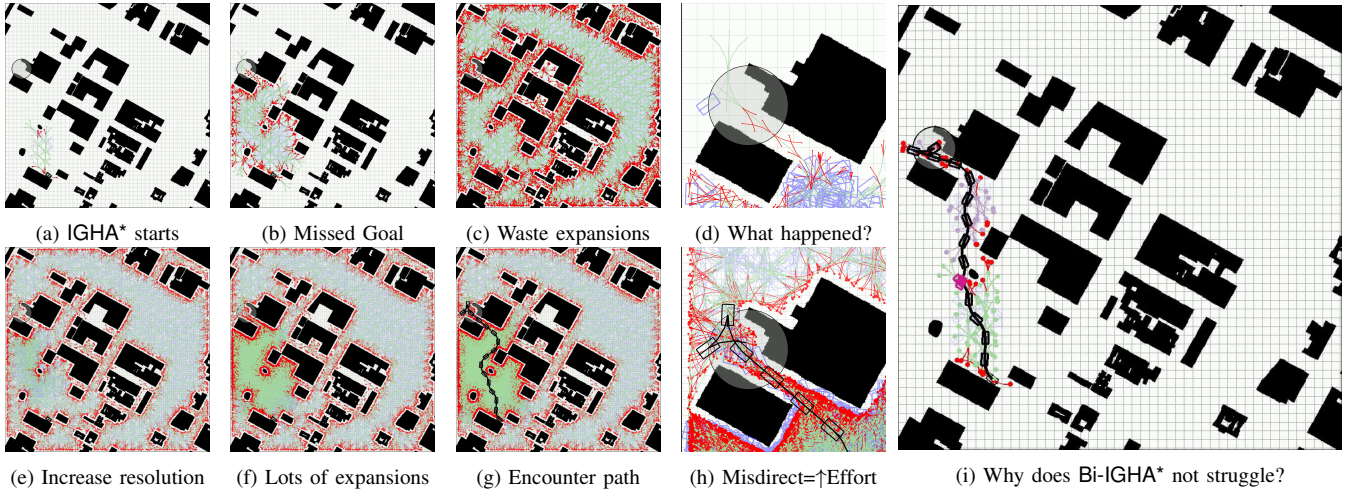


Fig. 1: Comparison of IGHA* (a)–(h) and Bi-IGHA* (i). For IGHA*, forward expansions are green, frozen (non-dominant) vertices blue, and collision states red. For Bi-IGHA*, FORWARD search uses the same color, BACKWARD search is purple, lighter shades denote frozen vertices, and the *near-meet* is pink. In both, the final path is black. The goal condition is the Reeds-Shepp [9] distance to $v_g \in \mathbb{R}^3$. Although IGHA* appears to pass through the goal region in (b), the true goal configuration is in (x, y, θ) , and the goal condition requires the Reeds-Shepp distance to be within a threshold (the circle is a visual aid), it freezes the critical vertex leading to v_g (d), forcing exploration of the remaining free space (c), resolution refinement after exhaustion (e, f), and eventual discovery only after many expansions (g). Bi-IGHA* avoids this failure mode (i) by mitigating the frozen vertex barrier.

B. Kinodynamic Bidirectional Motion-Planning

In kinodynamic planning, unlike geometric graph search, the forward and backward trees in continuous state space do not share exact states. Consequently, establishing a connection requires solving a local BVP to bridge the gap between the two trees [23]. Early work on Kinodynamic Bi-RRT utilized the property of a *local controllability radius* (LCR) to assume that if two states are sufficiently close, they can be connected [7]. Similarly, recent sampling-based methods verify a connection between vertices using trajectory optimization [24], or exploit the reverse tree only when vertices fall within a specific radius [25]. Approaches such as idB-A* [26], [27] rely on the concept of bounded discontinuity for connecting motion primitives. At their core, they all rely on the ability to connect vertices within some LCR, either by assuming that a connection is possible or at least to filter out unpromising candidates. Our algorithm also uses the notion of LCR, but remains orthogonal to how the connection is actually verified.

C. Kinodynamic Planning & Off Road Autonomy

For unstructured and ever-changing environments with complex dynamics, such as in the case of off-road autonomy [1], [28], caching motion primitive rollouts [26], [27] is often infeasible [2] as the dynamics depend on the terrain geometry, and would affect the rollout as soon as the perception system provides a new map. Algorithms like Hybrid A* (HA*) [5] and SST [29] commit to tree search with the system’s dynamics model, and leverage the notion of *approximate dominance* [6] to reduce the computational complexity of these problems; however, these approaches couple the tree discovery with the discretization resolution. While the IGHA* [2] breaks this coupling outside the inner search process, it is still susceptible to the frozen vertex barrier. In this work, we extend the IGHA* framework into the bidirectional setting and show how it mitigates this issue.

III. PROBLEM STATEMENT

We consider the problem of computing near-optimal paths over two implicitly-defined opposing trees $T_D = (V_D, E_D)$ for $D \in \{F, B\}$ (FORWARD, BACKWARD resp.), $V_D \subseteq \mathbb{R}^N$. For the D search, each edge $(u, v) \in E_D$ has an associated positive cost given by a function $w : E_D \rightarrow \mathbb{R}^+$. T_D is rooted at a (given) root vertex v_r^D , and the objective is to reach any vertex in a target set V_T^D while minimizing the accumulated edge cost, where V_T^D denotes the subset of V_D satisfying the goal condition. The tree structures are defined implicitly by a successor function SUCC that returns the set of valid children for any given vertex. With slight abuse of notation, we set $w(\pi)$ to be the cost of a path $\pi = \langle v_1, v_2, \dots, v_n \rangle$. Namely, $w(\pi) := \sum_{i=1}^{n-1} w(v_i, v_{i+1})$ and define $w(\emptyset) = \infty$ to denote the cost of failure. We assume access to a heuristic function $h_D : V_D \rightarrow \mathbb{R}_{\geq 0}$ that is admissible, providing an estimate of the remaining cost to the goal for the D search. In FORWARD search, $v_r^F \leftarrow v_s \in V_s \subseteq V_F$ and the target set $V_T^F \leftarrow V_g$, and in BACKWARD search, $v_r^B \leftarrow v_g \in V_g \subseteq V_B$ and $V_T^B \leftarrow V_s$, where s, g refer to start and goal resp. We assume that every BACKWARD edge corresponds to a FORWARD realizable edge of equal cost, so reversing a BACKWARD edge preserves feasibility. Notationally, let $\bar{D}$ imply the opposing direction; i.e., $T_{\bar{D}}$ is T_D ’s opposing tree.

IV. ALGORITHM BACKGROUND–IGHA*

Recall that A* [30] on an implicitly defined tree $T_D = (V_D, E_D)$ maintains a priority queue Q_v^D ordered by f -value, where $v.f = v.g + h_D(v)$, $v.g$ is the cost-to-come, and $h_D(v)$ is the heuristic cost-to-go. It repeatedly expands vertices from Q_v^D using SUCC until a goal-satisfying vertex is expanded. This is prohibitively expensive in the continuous state space setting as exact state overlap is measure zero. Hybrid A* (HA* [5]) reduces this burden through *approximate dominance*. The state space is discretized at resolution R , and $\hat{v} : V_D \rightarrow V_D$ maps each vertex v to the vertex currently dominating its region (implemented as a grid cell). When a

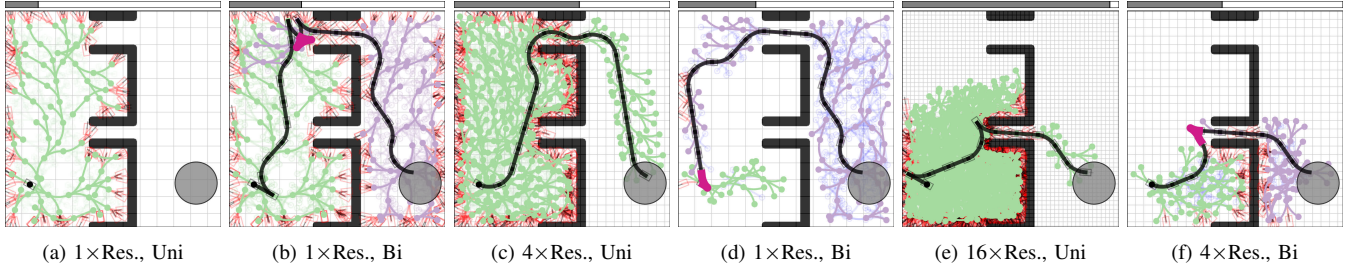


Fig. 2: Comparison of IGHA* and Bi-IGHA* illustrating equivalent solutions at lower resolutions, manifesting bidirectional mitigation. The color scheme matches Fig. 1; the grey bar indicates total expansions at each snapshot. For clarity, we only render the vertices expanded at an iteration, and use an ACTIVATE variant that resets Q_v^D , which removes the advantage of vertex reuse. At the base resolution, IGHA* fails ((a)), whereas the BACKWARD search in Bi-IGHA* succeeds ((b)). Unidirectional IGHA* must refine the resolution to obtain first a suboptimal ((c)) and then an optimal solution ((e)). In contrast, Bi-IGHA* recovers equivalent-cost solutions at lower resolutions via near-meets (—) ((d), (f)), using much fewer expansions.

new vertex enters a dominated region, the lower- g vertex is retained and the other pruned. This assumes the *best cost* from any vertex in a cell to the goal is identical, coupling pruning to R : a fine R approaches the computational burden of A^* , while a coarse R can prune the goal-bearing vertex.

IGHA* (Alg. 1; left half of Fig. 3, excluding green portions) breaks this coupling by generalizing approximate dominance to N levels $\ell = [\ell_0, \ell_1, \dots, \ell_N]$ with corresponding resolutions $R = [R_0, R_1, \dots, R_N]$ (we use “level” and “resolution” interchangeably), with $\hat{v} : V_D \times N \rightarrow V_D$ indexing the level ℓ_D . Unlike HA^* , IGH A* retains non-dominant vertices but marks them *InActive* or *Frozen*, preventing their expansion during the current search iteration. The search starts at level 0 with root v_r^D . The planner ACTIVATES v_r^D and begins an inner *search iteration* (Line 7), expanding vertices from Q_v^D (Line 13) until the goal condition is satisfied, $Q_v^D.Active = \emptyset$, or a resolution SHIFT occurs (Lines 7, 8, 9, 12). Upon termination, BOUND (Line 27) removes vertices worse than the best cost so far, $w(\hat{\pi})$; the remaining vertices are PROJECTED to the new resolution (Line 28), and the inner search is bootstrapped with newly ACTIVATED vertices (Line 6). This repeats until $Q_v^D = \emptyset$.

Although IGH A* avoids HA^* ’s pruning of goal-bearing vertices, it can still freeze one at a given resolution, forcing many expansions of vertices that do not lead to the goal before refinement; the goal-bearing vertex may remain frozen even after refinement. This creates a barrier that the search overcomes only through additional expansions (Fig. 1), which we term the *frozen vertex barrier*. Here we consider running two IGH A* searches to mitigate this, one FORWARD and one BACKWARD, so that either search can make progress toward the other even when one becomes blocked by frozen vertices.

V. BIDIRECTIONAL IGH A* (BI-IGH A*)

A. Considerations for Making IGH A* Bidirectional

While bidirectional search offers the potential of exponential speed-ups [8], [18] through the reduction of the search depth, these techniques typically assume a static graph, where vertices overlap exactly, and that the dynamics are symmetric forward and backward [31]. Once a vertex is reached, it is assumed that calling SUCC would produce the same neighbours from both sides, and so we no longer have to continue the search *through* that vertex. The requirement

of exact overlap is important for efficiency in finding *a path*, and the requirement of producing the same neighbours on both sides is important for reducing search effort *after* finding the best path (early termination). We leverage the notion of a *local controllability radius*, standard in kinodynamic planning [23], for detecting and verifying *near meets* to achieve efficiency in finding *a path*.

Definition 5.1 (Local Controllability Radius): A system has local controllability radius R_{LCR} if for any states $v \in V_D, \bar{v} \in V_{\bar{D}}$ with $\|v - \bar{v}\|_2 < R_{LCR}$, there exists a dynamically feasible trajectory connecting v to $\bar{v}$.

Definition 5.2 (NEARMEET): for a $v \in V_D$, NEARMEET returns *true* if $\exists \bar{v} \in V_{\bar{D}}$ such that $\|v - \bar{v}\|_2 < R_{LCR}$ and the dynamically feasible trajectory connecting v to $\bar{v}$ is collision free. It returns *false* otherwise.

While various ways of detecting a NEARMEET exist (assuming a connection is possible [23], [27], or trajectory optimization [24]), we treat it as a black box. *After* this connection, we can’t assume that the neighbours from either direction for the connected vertices will be identical, as the vertices aren’t *exactly* the same. Thus, the techniques mentioned earlier for reducing the search effort *after* the connection can’t be used within our setting. As the dynamically generated (sub)-trees in IGH A* are always subject to activation and approximate dominance, the g value of the vertex is always an *upper bound*, not a *lower bound*; thus, it would make for an inadmissible heuristic and cannot be used in our setting.

B. The Bi-IGH A* Framework

The Bidirectional IGH A* (Bi-IGH A*) (Alg. 1, Fig. 3) runs two (FORWARD, BACKWARD) IGH A* searches that share tree information for detecting near meets and the current best cost $w(\hat{\pi})$. This information sharing is an ever-present background process. For simplicity, we pop vertices from the FORWARD and BACKWARD Q_v^D s symmetrically.

Definition 5.3 (FORWARD-BACKWARD-Paths): When NEARMEET ($v, R_{LCR}, T_{\bar{D}}$) returns true for a vertex $v \in V_D$ with $\bar{v} \in V_{\bar{D}}$, (Lines 22), we call the resulting path a FORWARD-BACKWARD path denoted as π_{FB} , and denote the set of such paths as Π_{FB} .

When there exist multiple π_{FB} for a vertex $v \in V_D$ and multiple opposing vertices $V_{NEARMEET} \subseteq V_{\bar{D}}$, GETNEARMEETVERTEX returns the vertex $\bar{v}^* = \arg \min_{\bar{v} \in V_{NEARMEET}} (v.g + \bar{v}.g + w(v, \bar{v}))$, i.e., the one minimizing the FORWARD-BACKWARD path cost for

Algorithm 1 Bidirectional IGHA* (Bi-IGHA*)**Input:** Start set V_s , Goal set V_g , Resolution sequence $\mathcal{R}$ **Output:** Least-cost path or $\emptyset$ if FAILURE

```

1:  $\hat{\pi} \leftarrow \emptyset$  ▷ Shared amongst the two searches
   Procedure for D search;  $\bar{D}$  search runs in the background
2:  $v_r^D.\text{Active} \leftarrow \text{true}$ 
3:  $Q_v^D \leftarrow \{v_r^D\}$  ▷ Priority:  $v.g + h_D(v)$ 
4:  $l_D \leftarrow 0, l'_D \leftarrow 0$ 
5: while  $Q_v^D \neq \emptyset$  do ▷ Iterate over D search
6:   ACTIVATE()
7:   while  $Q_v^D.\text{Active} \neq \emptyset$  do
8:     if  $Q_v^D.\text{Active}.\text{PEEK}().f \geq w(\hat{\pi})$  then break
9:     if  $Q_v^D.\text{Active}.\text{PEEK}() \in V_T^D$  then
10:       $\hat{\pi} \leftarrow \text{EMITPATH}()$ 
11:      break
12:     if SHIFT() then break ▷ Internally update  $l'$ 
13:      $u \leftarrow Q_v^D.\text{Active}.\text{POP}()$ 
14:     for all  $v \in \text{SUCC}(u)$  do
15:       if  $v.g < \hat{v}(v, R_l).g$  then
16:          $\hat{v}(v, R_l).\text{Active} \leftarrow \text{false}$ 
17:          $\hat{v}(v, R_l) \leftarrow v$ 
18:          $v.\text{Active} \leftarrow \text{true}$ 
19:       else
20:          $v.\text{Active} \leftarrow \text{false}$ 
21:          $Q_v^D.\text{INSERT}(v)$ 
22:       if NEARMEET( $v, R_{\text{LCR}}, T_{\bar{D}}$ ) then
23:          $\bar{v} \leftarrow \text{GETNEARMEETVERTEX}()$ 
24:          $w(\pi_{\text{LCR}}) \leftarrow v.g + \bar{v}.g + w(v, \bar{v})$ 
25:         if  $w(\pi_{\text{LCR}}) < w(\hat{\pi})$  then
26:            $\hat{\pi} \leftarrow \text{EMITPATH}()$ 
27:       BOUND( $w(\hat{\pi})$ ) ▷  $\forall v \in Q_v^D$  Remove  $v : v.f > w(\hat{\pi})$ 
28:       PROJECT( $R_l$ ) ▷  $l_D \leftarrow l'_D$ , update  $R_l$ , and recompute  $\hat{v}$ 
29: return  $\hat{\pi}$ 

```

$\pi_{\text{LCR}} \in \Pi_{\text{FB}}$ (Line 24). Note that the FORWARD and BACKWARD IGHA* are not level-locked; they can change levels independently. If one terminates before the other, the remainder continues running to termination, as a path may exist in one tree that does not exist in the other direction's tree. Note that both searches share $\hat{\pi}$. One IGHA* does not directly affect the other IGHA*'s SHIFT or ACTIVATE.

Our key insight is that bidirectionality alters how goal-reachability information propagates through the incremental dominance structure, mitigating resolution-induced freezing effects; we term this phenomenon Bidirectional Mitigation, and illustrate it through Fig. 2. When it comes to Bi-IGHA*, even when the resolution is too coarse for IGHA* to find any path from v_s to V_g in the forward tree, requiring resolution refinement, Bi-IGHA* may find a path from v_s to v_g through Π_{FB} (Fig. 2). *Bidirectional Mitigation* allows at least one of the searches in Bi-IGHA* to encounter an equivalent cost solution at a lower resolution. At lower resolutions, the effective branching factor of Bi-IGHA*'s FORWARD and BACKWARD searches can be much lower via approximate dominance, which also contributes to the speed-up. We analyse this formally in Sec. VI-B, show how this results in better-than-expected reduction in the effective branching factor in Sec. VI-C, and show empirically in Sec. VII that this (mitigation) happens very often when there is a speed-up. **Note:** we use **FORWARD** and **BACKWARD** to refer to Bi-IGHA*'s internal searches, and **forward/reverse** to refer to instances of (unidirectional) IGHA*.

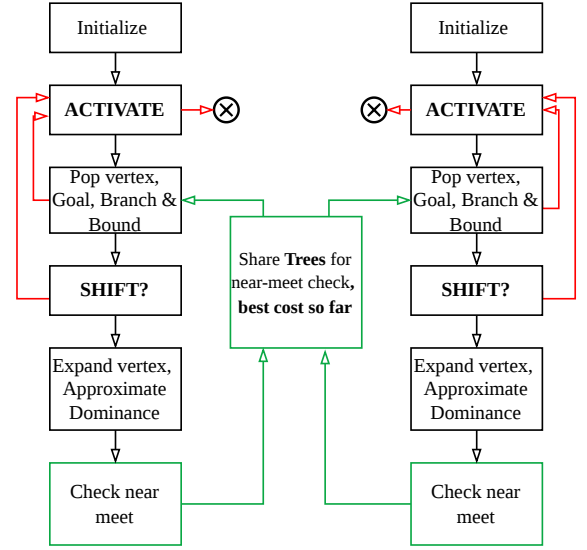


Fig. 3: Bi-IGHA* runs two anti-parallel IGHA* searches with information sharing while checking for near-meets between the trees of them

VI. FORMAL GUARANTEES AND STRUCTURAL ANALYSIS

We first show that Bi-IGHA* preserves the core guarantees of IGHA* on monotonic cost improvement and termination with finite expansions. We then formalize the structural interaction between discretization and freezing that motivates bidirectionality. Throughout this section we assume:

- A1** The heuristic h_D never overestimates the true cost-to-go to V_T^D .
 - A2** Each vertex has at most $b < \infty$ successors under **SUCC**.
 - A3** Every edge cost satisfies $w(u, v) \geq \epsilon > 0$.
 - A4** Every **BACKWARD** edge corresponds to a **FORWARD** realizable edge of equal cost.
- A1-A3** are identical to the assumptions in [2]. From [2], the (relevant) invariants of IGHA* are:
- I1** Only Active vertices are expanded during search.
 - I2** At least one vertex is **ACTIVATED** until $Q_v^D = \emptyset$.

A. Preserving IGHA*'s Guarantees

Theorem 6.1 (IGHA* Preservation): Given **A1-A4**, **I1-I2**, Bi-IGHA* retains IGHA*'s guarantees of monotonic cost improvement and termination with finite expansions.

Proof: (Sketch) If a path is emitted from the FORWARD or BACKWARD search, either by reaching their respective V_T^D (Line 10) or a **NEARMEET** (Line 26), then this is because the search found a valid path that is guaranteed to be better than the existing best estimate $w(\hat{\pi})$ (Lines 8, 25). Given this, it follows from **A1-A4** and **I1, I2** that Bi-IGHA* retains IGHA*'s guarantees of monotonic cost improvement and termination with finite expansions. ■

B. Freezing as a Structural Barrier

Observation 6.1 (Frozen-Vertex Barrier): **I1** implies that if a vertex is frozen ($v.\text{Active} = \text{false}$) at a given iteration then no descendant of v can be generated at that iteration; consequently, no solution path passing through v can be discovered at that iteration.

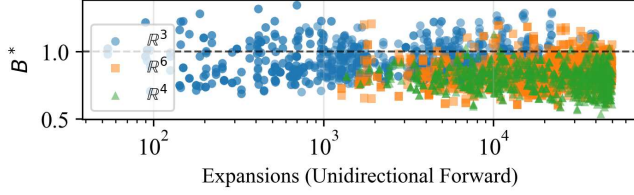


Fig. 4: B^* (see Sec. VI-C) scatter plot for Bi-IGHA* vs. IGHA* w.r.t. forward expansions, for best cost path. We usually expect $B^* \leq 1$; however, through bidirectional mitigation, Bi-IGHA* appears to cross this limit.

We **construct** a family of planning instances for the forward search that expose the frozen vertex barrier (Obs 6.1) when there exists a *bottleneck vertex* and the search does not have access to information about whether the vertex is on the solution path or not (the search is naive).

Definition 6.1 (Bottleneck Vertex): A vertex v_b is considered a bottleneck if every path from the root vertex v_r^D to the target set V_T^D must traverse through it.

Definition 6.2: (Naive Rule) A Naive Rule [2] refers to a combination of a naive SHIFT and a naive ACTIVATE method, where either method is considered naive if it only uses information in Q_v^D .

Definition 6.3 (Bottleneck Family $\{\mathcal{P}_k\}$): For a unidirectional IGHA* with a Naive Rule, for each $k \geq 1$, let $\mathcal{P}_k$ denote a planning instance with k sequential bottleneck vertices v_b^k , such that another vertex $v_{\text{dead}}^i = \hat{v}(v_b^i, R_i)$, at level i (See IV), dominates it. Here, v_{dead}^i does not lead to V_T^D , but rather to a dead-end subtree containing $M_i > M \forall i$ vertices, none of which lead to V_T^D , and M is an arbitrarily large constant. Finally, for a naive Bi-IGHA*, assume that at any iteration $|\Pi_{\text{FB}}| \geq 1$, where the FORWARD-BACKWARD paths that do not require forward expansion of v_b^0 and require $O(1)$ expansions beyond those required to reach v_b^0 .

Observation 6.2 (Unidirectional Freezing Blowup): On $\mathcal{P}_k$, it follows from Obs. 6.1 that for k bottlenecks, IGHA* expands at least $\Omega(M_0)$ expansions if SHIFT refines to the highest possible resolution after the first iteration and at least $\Omega(\sum_{i=0}^k M_i)$ if not (from Inv. 12).

The sequence of plots in Fig. 1(a)-1(h) illustrates **this blowup for $k = 1$** , where IGHA*'s search tree passes near the goal region, but freezes a crucial vertex leading to the goal, explores all the dead-end nodes before refining resolution and finally expanding the crucial vertex. **The blowup arises from freezing, not from search depth alone.**

Observation 6.3: On $\mathcal{P}_k$, through Π_{FB} 's found via NEARMEET, Bi-IGHA* finds a path from v_s to v_g in $O(1)$ additional expansions beyond reaching v_b^0 .

Corollary 6.1 (Structural Separation): There exists a family of planning instances for which unidirectional IGHA* may require at least $\Omega(\sum_i M_i)$ expansions, while Bi-IGHA* requires $O(1)$ additional expansions beyond reaching v_b^0 .

This separation shows that the improvement of Bi-IGHA* is not solely due to reduced effective depth, and we will empirically show in Sec. VI-C that this separation exists in practice.

C. Surprising Branching factors

We defer experimental details to Sec. VII, and **empirically** illustrate the structural separation (Corr. 6.1) of Bidirectional mitigation. Across $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^6$ systems, Fig. 5 shows that, as intuition may suggest, increasing R_{LCR} raises the likelihood of near-meets and thus the speed-up, until we examine effective branching factors.

We define the *empirical effective unidirectional branching factor* b_{uni} for a given run of a unidirectional search as follows: Let E_{uni} be the number of node expansions and d_{uni} be the number of nodes of the path returned. Then, b_{uni} is the solution to the following equation: $E_{\text{uni}} = \sum_{n=0}^{d_{\text{uni}}-1} (b_{\text{uni}})^n$. Given b_{uni} , we can compute $E_{\text{bi}}^{\text{ideal}}$, the number of expected node expansion under an optimal symmetric bidirectional search. Specifically, $E_{\text{bi}}^{\text{ideal}} := 2 \sum_{n=0}^{d_{\text{uni}}/2-1} (b_{\text{uni}})^n$. Given $E_{\text{bi}}^{\text{ideal}}$, we define the *estimated effective bidirectional branching factor* as the solution to the following equation: $E_{\text{bi}}^{\text{ideal}} = \sum_{n=0}^{d_{\text{bi}}-1} (b_{\text{bi}}^{\text{ideal}})^n$. In a completely analogous way to the unidirectional setting, we define the *empirical effective bidirectional branching factor* b_{bi} for a given run of a bidirectional search as follows: Let E_{bi} be the number of node expansions and d_{bi} be the number of nodes of the path returned. Then, b_{bi} is the solution to the following equation: $E_{\text{bi}} = \sum_{n=0}^{d_{\text{bi}}-1} (b_{\text{bi}})^n$.

Now, consider the ratio $B^* = \frac{b_{\text{bi}}^{\text{ideal}}}{b_{\text{bi}}}$. Since $b_{\text{bi}}^{\text{ideal}}$ assumes an optimal symmetric bidirectional search, we expect that $B^* \leq 1$. Surprisingly, empirical values recorded and reported in Fig. 4 demonstrate instances for which $B^* > 1$. The reason for this phenomenon is that in the computation of B^* , we implicitly assumed that both planners operate on the same search structure; this is typically true for search on a static graph, but is not the case here due to mitigation of the frozen vertex barrier. Nonetheless, as Bi-IGHA* does not explicitly enforce MMP, at a first glance of Alg. 1 and flow chart Fig. 3, one would *never* expect this result from it.

VII. EXPERIMENTS

We evaluate Bi-IGHA* in both open and closed-loop settings, consider IGHA* as our baseline, and use 95% confidence intervals. We count the expansions of the forward and reverse search separately. For planning, we use dynamics models satisfying A4 throughout, with particular care to avoid irreversible operations such as state clamping, constraint projection, or slip-induced information loss. For physical dimensions, we use S.I units. For NEARMEET checks, we define $\|u - v\|_2 \triangleq \|(u - v)/S\|_2$ for $u \in V_D, v \in V_{\bar{D}}$, where division by the normalizing vector S is component-wise.

A. Open Loop Evaluations

We present both an ablation of Bi-IGHA* with respect to LCR and a comparison against IGHA* across three systems. First, a kinematic car [33] in $\mathbb{R}^3$ (x, y, θ) on urban city maps from the Moving AI benchmark [34] (e.g., Fig. 1), with $S = (1.0, 1.0, \pi/2)$. Second, a kinodynamic car [35] in $\mathbb{R}^4$ (x, y, θ, v), assuming no-slip and no body-frame vertical velocity, evaluated on off-road maps taken from

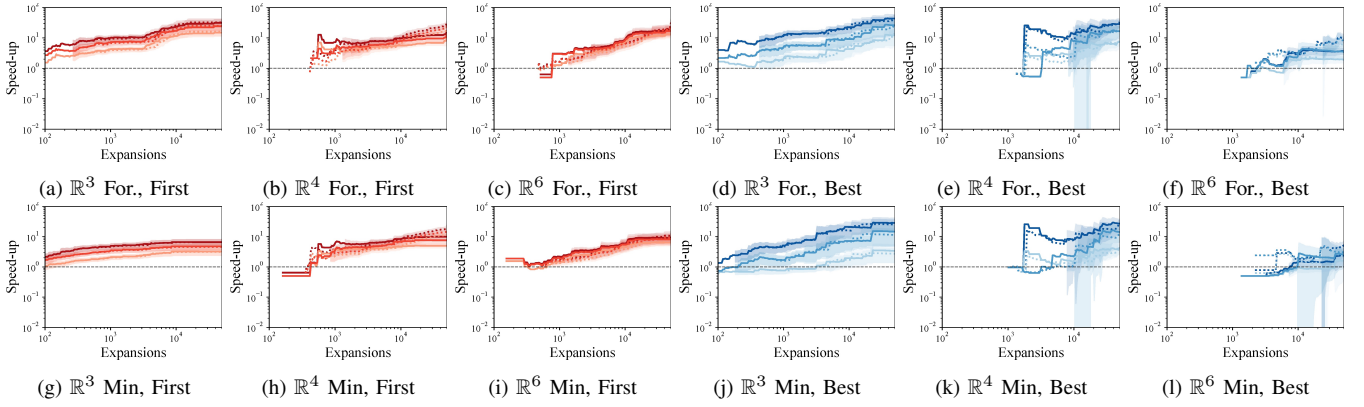


Fig. 5: Here, — denotes Speed-up to the first solution and — to best solution; lighter shades correspond to smaller LCR values (LCR, LCR/8, LCR/64). Solid lines compare Bi-IGHA* ∞ vs. IGHA* ∞ , dotted/dashed lines compare Bi-IGHA* ∞ vs. IGHA* ∞ . Confidence intervals are shown only when more than 30 measurements are available. Comparisons are against Forward-only (For.) ((a)-(f)) IGHA* and the faster (Min.) of forward/reverse IGHA* ((g)-(l)). We observe roughly order-of-magnitude Speed-ups; gains reduce as LCR is reduced, when comparing to the Min., and increased complexity of higher dimensions shifts the curves right.

	First Path (For. / Min.)						Best Path (For. / Min.)					
$\mathbb{R}^3$	0.88 $\pm$ 0.03	0.60 $\pm$ 0.05	0.92 $\pm$ 0.03	0.76 $\pm$ 0.04	0.94 $\pm$ 0.02	0.85 $\pm$ 0.03	0.59 $\pm$ 0.05	0.39 $\pm$ 0.05	0.75 $\pm$ 0.04	0.56 $\pm$ 0.05	0.85 $\pm$ 0.03	0.79 $\pm$ 0.04
$\mathbb{R}^4$	0.91 $\pm$ 0.03	0.89 $\pm$ 0.03	0.92 $\pm$ 0.03	0.90 $\pm$ 0.03	0.97 $\pm$ 0.02	0.96 $\pm$ 0.02	0.59 $\pm$ 0.05	0.47 $\pm$ 0.05	0.71 $\pm$ 0.04	0.63 $\pm$ 0.05	0.86 $\pm$ 0.03	0.84 $\pm$ 0.04
$\mathbb{R}^6$	0.92 $\pm$ 0.03	0.83 $\pm$ 0.04	0.94 $\pm$ 0.02	0.87 $\pm$ 0.03	0.95 $\pm$ 0.02	0.88 $\pm$ 0.03	0.51 $\pm$ 0.05	0.33 $\pm$ 0.05	0.63 $\pm$ 0.05	0.47 $\pm$ 0.05	0.68 $\pm$ 0.05	0.57 $\pm$ 0.05
	LCR/64		LCR/8		LCR		LCR/64		LCR/8		LCR	

Fig. 6: $\hat{p}(\text{Speed-up} > 1)$ across LCR values are reported as mean $\pm$ 95% CI, shades of red represent the values for the first path, and shades of blue represent the best path. For each LCR, the left half represents the Speed-up probability compared to forward (For.) IGHA*, and the right half represents it against the faster of forward/reverse IGHA* (Min.). As LCR increases, $\hat{p}(\text{Speed-up} > 1)$ increases across the board.

	First Path ($\mu_{\leq} / \mu_{<}$)			Best Path ($\mu_{\leq} / \mu_{<}$)		
$\mathbb{R}^3$	1.00 / 0.99	1.00 / 0.98	1.00 / 0.98	0.95 / 0.61	0.98 / 0.76	1.00 / 0.86
$\mathbb{R}^4$	1.00 / 0.94	1.00 / 0.94	1.00 / 0.93	0.99 / 0.96	1.00 / 0.93	1.00 / 0.96
$\mathbb{R}^6$	1.00 / 0.99	1.00 / 0.99	1.00 / 0.99	1.00 / 0.84	0.99 / 0.89	1.00 / 0.88
	LCR/64	LCR/8	LCR	LCR/64	LCR/8	LCR

Fig. 7: Showing $\mu_{\leq}$, $\mu_{<}$ for $\mathbb{R}^N$ for the Min. of forward/reverse IGHA*; the likelihood that one of the Bi-IGHA*'s resolution levels was lower than IGHA*'s for the first or the best solution is almost always true. In the $\mathbb{R}^3$ setting, $\mu_{<}$ is lower because often the solution is available at a low enough resolution to both Bi-IGHA* and IGHA*.

BeamNG [36] (e.g. Fig. 9(d)), with $S = (1.0, 1.0, \pi/2, 2.5)$. Third, a hovercraft [7] in $\mathbb{R}^6$ ($x, y, \theta, v_x, v_y, \dot{\theta}$), with maps from Moving AI [34] and BeamNG [36], with $S = (1.0, 1.0, \pi/2, 2.5, 2.5, \pi/2)$. In all environments, the vehicle dimensions (length, width) are 2.6×1.6 m². Each environment contains 200 problems, with a 50k expansion limit. We evaluate two extreme hysteresis settings ($\infty, 250$; see [2] for details) for IGHA* [2] and use the same values for Bi-IGHA* (both directions). To compare across $\mathbb{R}^3/\mathbb{R}^4/\mathbb{R}^6$, we report LCR in terms of induced volume: LCR ($R_{\text{LCR}} = 1$), LCR/8 ($1/8^{\text{th}}$ volume), and LCR/64 ($1/64^{\text{th}}$ volume). Note that our edges are usually longer than 1.0 meter (≈ 5 m), meaning near meets join vertices that are already closer to each other than a single edge. Here, for simplicity, NEARMEET declares (u, v) feasible if $\|u - v\|_2 < R_{\text{LCR}}$, where R_{LCR} is the radius at the corresponding resolution.

Let $n_{\text{first}}(\text{ALG})$ be the number of expansions required by an algorithm (either IGHA* or Bi-IGHA*) to obtain a solution. Here, $\text{Speed-up} := n_{\text{first}}(\text{IGHA*})/n_{\text{first}}(\text{Bi-IGHA*})$. Let $n_{\text{best}}(\text{IGHA*})$ be the number of expansions required by IGHA* to reach its best path cost within the expansion budget, and let $n_{\text{best}}(\text{Bi-IGHA*})$ be the expansions required by

Bi-IGHA* to reach an equivalent-or-better cost. Here, $\text{Speed-up} := n_{\text{best}}(\text{IGHA*})/n_{\text{best}}(\text{Bi-IGHA*})$. We additionally record the solution level ($\text{level}_{\text{Bi-IGHA*}}, \text{level}_{\text{IGHA*}}$) for both first and best solutions. We are interested in knowing how often bidirectional mitigation happens when there is a Speed-up. Bidirectional mitigation manifests empirically as follows: when a solution is found, either FORWARD or BACKWARD search operates at a level ($\text{level}_{\text{Bi-IGHA*}}$) lower than IGHA*'s ($\text{level}_{\text{IGHA*}}$). We define $\mu_{\leq} := \hat{p}(\text{level}_{\text{Bi-IGHA*}} \leq \text{level}_{\text{IGHA*}} \mid \text{Speed-up} > 1)$ and $\mu_{<} := \hat{p}(\text{level}_{\text{Bi-IGHA*}} < \text{level}_{\text{IGHA*}} \mid \text{Speed-up} > 1)$. We make the above comparisons against the best of forward and reverse IGHA*, as if an oracle informed IGHA* on whether to run in forward or reverse. Since the first solution admits any cost, we compare Bi-IGHA* against the faster of forward/reverse to the first (any) solution. For the best solution, we compare the expansions taken by Bi-IGHA* with those of the forward/reverse IGHA* that achieves the lowest cost. We do this comparison to show that Bi-IGHA* is not fast just by hedging a bet in both directions. In this section, we test the following hypotheses:

H1 A variant of Bi-IGHA* achieves Speedup > 1 .

H2 $\hat{p}(\text{Speed-up} > 1) > 0.5$.

H3 $\mu_{\leq} > 0.5$.

H4 $\mu_{<} > 0.5$.

Result: Fig. 5 shows that generally Speedup > 1 for the first and best path, often approaching an order of magnitude, confirming **H1**. Fig. 6 shows that **H2** is always true for the first path, and often true for the best path. For both of the above plots, increasing LCR increases speed up, and the speed up (or its likelihood) dips when compared to the best of forward/reverse IGHA*. Fig. 7 shows that **H3**, **H4** are always true, and provides context on the stricter condition of $\mu_{<}$.

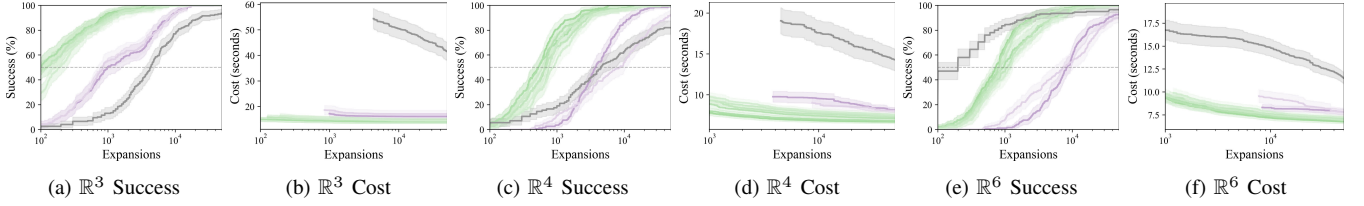


Fig. 8: Here, —: Bi-IGHA*, —: IGHA*, where darker shades imply bigger LCR, and additionally —: SST (from OMPL [32]) as a reference point. Mean cost is computed for problems succeeding in the first 50 %. In Fig. 5, 6, results are *somehow* pessimistic because we measure the expansions at which we get the same or better cost – often the cost found by Bi-IGHA* is higher but very close ($< \epsilon$, recall A3), to IGHA*'s – this plot shows that Bi-IGHA*'s cost is consistently a left-shifted version of IGHA*'s by $\approx$ an order of magnitude. We show SST as an additional reference – SST is good at rapidly exploring high-dimensional spaces; however, in mission-critical settings, solution quality dominates in-the-loop performance (Fig. 9(a)–9(c) and [2]).

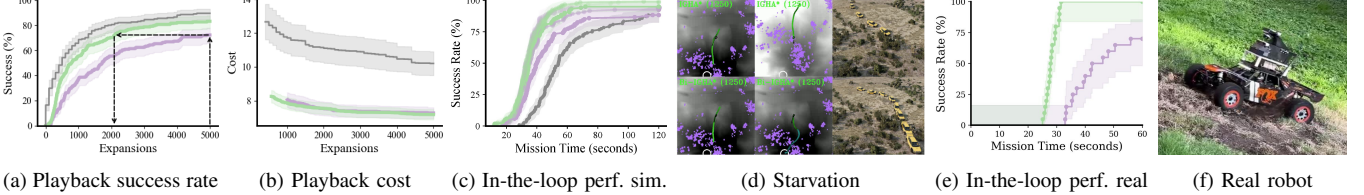


Fig. 9: Here, —: Bi-IGHA*, —: IGHA*. The playback success (a), and cost (b) metrics were used to estimate a reasonable range of expansions to use in-the-loop to exaggerate the difference between Bi-IGHA* and IGHA*. The mission-success rate plot shows Bi-IGHA* and IGHA* running with 5k, 2.5k, and 1.25k expansions, respectively, with darker shades implying **fewer** expansions. We include —: SST's performance data from [2] as reference (5k exp). Even with just 1.25k expansions, Bi-IGHA* is able to maintain mission performance on par with IGHA* using 5k expansions; while IGHA* gets stalled due to starvation Bi-IGHA*'s continues to provide high-quality solutions ((d)). Real-world experiments with a small-scale car-like robot ((f)), reaching speeds of 4.6 m/s in an off-road environment, show the gap widening (e); the *slowest* Bi-IGHA* time is comparable to the *fastest* IGHA* time.

Fig. 8 shows a more conventional comparison and provides additional nuance to show that the results in Fig. 5, 6 are pessimistic.

B. Closed Loop Evaluations

We evaluate the planners using the $\mathbb{R}^4$ kinodynamic car model in off-road environments. Planning runs in closed loop with MPPI [37] (1.5 s horizon, 1024 rollouts) in BeamNG [36], assuming ground-truth state and body-centric bird's-eye-view cost and elevation maps ($100 \times 100 \text{ m}^2$, 25 Hz). We use the same benchmarking setup as used by the authors in [2] (30 scenarios, 4 trials) and reuse its reported results for IGHA*, along with SST [29] as a point of reference. Here, similar to [27], NEARMEET for (u, v) additionally collision checks interpolated states between u and v to avoid near-meets over thin obstacles. The vehicle dimensions are $2.6 \times 1.6 \text{ m}^2$, and the $S = (1.0, 1.0, \pi/8, 1.25) \approx \text{LCR}/8$. We use the same hysteresis value of 100 as used by IGHA* in [2], and the rate of expansions ($\approx 20 \text{ k/sec}$) on our computer matches what was reported in [2]. To bring out the difference between IGHA* and Bi-IGHA*, we operate both at 5k, 2.5k, and 1.25k expansions – we arrive at this range of expansions by looking at the success/cost plots on playback data (Fig. 9(a), 9(b)). When running the planner with fewer than 5k expansions, we slow down the rate of expansions to simulate a slower computer, such that the overall update frequency remains identical (4 Hz, as in [2]). We additionally evaluate the planner on a smaller ($0.9 \times 0.6 \text{ m}^2$) car-like off-road robot (a modified version of the HOUND platform [38]), using $S = (0.5, 0.5, \pi/8, 2.0)$, in an off-road environment, with an average trial distance of 80 meters over 20 trials. Planning runs at a locked 4 Hz with a budget of 625 expansions in closed loop with MPPI (1.0 s horizon, 1024 rollouts, 50 Hz) and online perception with a $30 \times 30 \text{ m}^2$ map at 10 Hz. All modules run onboard

an NVIDIA AGX Orin 64GB. The hypothesis we want to test here is:

H5 Bi-IGHA* achieves higher success rates than IGHA* for a given expansion budget.

Result: Fig. 9(c) shows that while IGHA*'s mission speed (slope of the curves) is generally close to Bi-IGHA*'s, Bi-IGHA* consistently attains higher success rates under equal compute budgets, reaching comparable reliability in less time. Fig. 9(d) illustrates this difference anecdotally. Fig. 9(e) shows a similar pattern to Fig. 9(c), with a wider gap between Bi-IGHA* and IGHA*. This confirms H5 in both simulation and real-world experiments, demonstrating the realizability of Bi-IGHA* and its benefits at lower computational budgets.

VIII. DISCUSSION

This work proposes Bi-IGHA* and shows how bidirectionality overcomes a structural limitation of IGHA*, in that the improvement does not come from the reduction in search depth alone, as is usually expected with bidirectional search. Under assumptions A1–A4, Bi-IGHA* preserves IGHA*'s guarantees on monotonic cost improvement and termination with finite expansions. Our work keeps the FORWARD and BACKWARD search identical in both the rate of expansions and the SHIFT and ACTIVATE methods used; however, the guarantees of Bi-IGHA* don't rely on this. For Bi-IGHA* to be practically useful, a time-reversible dynamics model (for A4) and a reliable NEARMEET implementation are required. We leverage simplified time-reversible dynamics models as *good enough* proxies for the true model and treat the NEARMEET method as a black box. We show that useful speed-ups are achievable with a range of LCR values, show that the chosen values are reasonable by using the middle value (LCR/8) in our simulated closed-loop experiment, with a fairly simple collision-checked NEARMEET, and use the real-world experiments to show the realizability of our work.

ACKNOWLEDGEMENTS

This work was (partially) funded by the National Science Foundation NRI (#2132848) and CHS (#2007011), DARPA RACER (#HR0011-21-C-0171), the Office of Naval Research (#N00014-17-1-2617-P00004 and #2022-016-01 UW), Amazon, Technion Autonomous Systems Program (TASP), and the United States-Israel Binational Science Foundation (BSF) grant (#2021643). We thank NVIDIA for providing the Jetson AGX Orin used in the real-world experiments.

REFERENCES

- [1] T. Han, A. Liu, A. Li, A. Spitzer, G. Shi, and B. Boots, “Model Predictive Control for Aggressive Driving Over Uneven Terrain,” in *Robotics: Science and Systems*, 2024, pp. 1–16.
- [2] S. Talia, O. Salzman, and S. Srinivasa, “Incremental Generalized Hybrid A*,” *IEEE Robotics and Automation Letters*, vol. 11, no. 2, pp. 1586–1593, 2025.
- [3] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, “Real-time Planning with Multi-fidelity Models for Agile Flights in unknown Environments,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 725–731.
- [4] X. Meng, N. Hatch, A. Lambert, A. Li, N. Wagener, M. Schmittle, J. Lee, W. Yuan, Z. Chen, S. Deng, G. Okopal, D. Fox, B. Boots, and A. Shaban, “Terrainet: Visual Modeling of Complex Terrain for High-speed, Off-road Navigation,” in *Robotics: Science and Systems*, 2023, pp. 1–14.
- [5] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path Planning for Autonomous Vehicles in Unknown Semi-Structured Environments,” *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
- [6] R. Bellman, “Dynamic Programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [7] S. M. LaValle and J. J. Kuffner Jr, “Randomized Kinodynamic Planning,” *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [8] R. C. Holte, A. Felner, G. Sharon, N. R. Sturtevant, and J. Chen, “MM: A Bidirectional Search Algorithm that is Guaranteed to Meet in the Middle,” *Artificial Intelligence*, vol. 252, pp. 232–266, 2017.
- [9] J. Reeds and L. Shepp, “Optimal Paths for a Car that goes both Forwards and Backwards,” *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.
- [10] L. Siag, S. S. Shperberg, A. Felner, and N. R. Sturtevant, “Front-to-End Bidirectional Heuristic Search with Consistent Heuristics: Enumerating and Evaluating Algorithms and Bounds,” in *International Joint Conference on Artificial Intelligence*, 2023, pp. 5631–5638.
- [11] E. Shaham, A. Felner, J. Chen, and N. R. Sturtevant, “The Minimal Set of States that Must Be Expanded in a Front-to-End Bidirectional Search,” in *International Symposium on Combinatorial Search*, 2017, pp. 82–90.
- [12] E. Shaham, A. Felner, N. R. Sturtevant, and J. S. Rosenschein, “Minimizing Node expansions in Bidirectional Search with Consistent Heuristics,” in *International Symposium on Combinatorial Search*, 2018, pp. 81–89.
- [13] S. S. Shperberg, A. Felner, S. E. Shimony, N. R. Sturtevant, and A. Hayoun, “Improving Bidirectional Heuristic Search by Bounds Propagation,” in *International Symposium on Combinatorial Search*, 2019, pp. 106–114.
- [14] S. S. Shperberg, A. Felner, N. R. Sturtevant, S. E. Shimony, and A. Hayoun, “Enriching Non-parametric Bidirectional Search Algorithms,” in *AAAI Conference on Artificial Intelligence*, 2019, pp. 2379–2386.
- [15] N. R. Sturtevant, S. S. Shperberg, A. Felner, and J. Chen, “Predicting the Effectiveness of Bidirectional Heuristic Search,” in *International Conference on Automated Planning and Scheduling*, 2020, pp. 281–290.
- [16] V. Alcázar, P. J. Riddle, and M. Barley, “A Unifying View on Individual Bounds and Heuristic Inaccuracies in Bidirectional Search,” in *AAAI Conference on Artificial Intelligence*, 2020, pp. 2327–2334.
- [17] N. Sturtevant and A. Felner, “A Brief History and Recent Achievements in Bidirectional Search,” in *AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 8000–8006.
- [18] Y. Wang, E. Weiss, B. Mu, and O. Salzman, “Bidirectional Search while Ensuring Meet-In-The-Middle via Effective and Efficient-to-Compute Termination Conditions,” in *International Joint Conference on Artificial Intelligence*, 2025, pp. 8987–8995.
- [19] ü. Eckerle, J. Chen, N. Sturtevant, S. Zilles, and R. Holte, “Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search,” in *International Conference on Automated Planning and Scheduling*, vol. 27, 2017, pp. 79–87.
- [20] J. Chen, R. C. Holte, C. Bäckström, and N. R. Sturtevant, “New Bidirectional Search: Algorithms and Improvements,” in *International Conference on Automated Planning and Scheduling*, 2017, pp. 224–232.
- [21] S. K. Sadhukhan, “A New Approach to Bidirectional Heuristic Search using Error Functions,” in *International Conference on Intelligent Infrastructure*, 2012.
- [22] E. C. Sewell and S. H. Jacobson, “Dynamically Improved Bounds Bidirectional Search,” *Artificial Intelligence*, vol. 291, p. 103405, 2021.
- [23] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [24] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, “Regionally Accelerated Batch Informed Trees (RABIT*): A Framework to Integrate Local Information into Optimal Path Planning,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 4207–4214.
- [25] 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.
- [26] J. Ortiz-Haro, W. Hoenig, V. N. Hartmann, and M. Toussaint, “iDB-A*: Iterative Search and Optimization for Optimal Kinodynamic Motion Planning,” *IEEE Transactions on Robotics*, 2024.
- [27] J. Ortiz-Haro, W. Hönig, V. N. Hartmann, M. Toussaint, and L. Righetti, “iDB-RRT: Sampling-based Kinodynamic Motion Planning with Motion Primitives and Trajectory Optimization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2024, pp. 10 702–10 709.
- [28] H. Lee, T. Kim, J. Mun, and W. Lee, “Learning Terrain-Aware Kinodynamic Model for Autonomous Off-road Rally Driving with Model Predictive Path Integral Control,” *IEEE Robotics and Automation Letters*, 2023.
- [29] Y. Li, Z. Littlefield, and K. E. Bekris, “Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning,” in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 263–282.
- [30] 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.
- [31] J. B. Kwa, “BS*: An Admissible Bidirectional Staged Heuristic Search Algorithm,” *Artificial Intelligence*, vol. 38, no. 1, pp. 95–109, 1989.
- [32] I. A. Şucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, <https://ompl.kavrakilab.org>.
- [33] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, “Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design,” in *2015 IEEE Intelligent Vehicles Symposium*. IEEE, 2015, pp. 1094–1099.
- [34] N. Sturtevant, “Benchmarks for Grid-Based Pathfinding,” *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012. [Online]. Available: <http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf>
- [35] T. Fork, H. E. Tseng, and F. Borrelli, “Models for Ground Vehicle Control on Nonplanar Surfaces,” *Vehicle System Dynamics*, pp. 1–25, 2023.
- [36] BeamNG GmbH, “BeamNG.tech,” 2022. [Online]. Available: <https://www.beamng.tech/>
- [37] G. Williams, A. Aldrich, and E. A. Theodorou, “Model Predictive Path Integral Control: From Theory to Parallel Computation,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [38] S. Talia, M. Schmittle, A. Lambert, A. Spitzer, C. Mavrogiannis, and S. S. Srinivasa, “Demonstrating Hound: A Low-cost Research Platform for High-speed Off-road Underactuated Nonholonomic Driving,” in *Robotics: Science and Systems*, 2024, pp. 1–12.