Bidirectional Heuristic Search for Motion Planning with an Extend Operator

Allen Cheng, Dhruv Mauria Saxena, and Maxim Likhachev

***Abstract*— Sampling-based approaches are often favored in robotics for high-dimensional motion planning for their fast exploration of the search space. However, at best they offer asymptotic guarantees on solution quality due to their inherent stochasticity. While planning, the majority of effort is often spent near the start and goal configurations with a large amount of free space in between. Bidirectional approaches such as RRT-Connect exploit this fact by greedily extending and connecting search frontiers that simultaneously propagate from the start and goal configurations of a planning problem. In this work, we use such an extend operator for bidirectional heuristic search-based planners, which typically struggle with high-dimensionality. In doing so, we address the difficulty that these bidirectional planners face with connecting frontiers of both search efforts while providing suboptimality bounds on solution quality. We validate our simple approach on high-dimensional manipulation tasks, demonstrating significantly reduced search effort when compared against other popular bidirectional algorithms, both search-based and sampling. Our algorithm maintains theoretical guarantees on suboptimality and completeness for a given resolution. In addition, the solutions found by our planner are of higher quality compared to those found by the other baseline algorithms.**

I. Introduction

Consider the manipulation problem of pick-and-place in cluttered environments with a high degree-of-freedom (DoF) robotic arm, a challenge commonly faced in industrial settings. Successful grasping often requires tight and minimal tolerances due to clutter around the start and goal configurations. This demands a large amount of search effort around the start and goal region, but the majority of the intermediate path is typically collision-free. Naive search-based planners require procedural reasoning over the possible arm configurations when planning through this free space to maintain optimality guarantees. To increase planning efficiency in these scenarios, biased heuristics may be used, but they suffer when presented with cluttered environments that induce local minima. Planning with adaptive dimensionality [1] exploits this insight to quickly find paths in low-dimensional representations before executing a high-dimensional search in a tunnel around those paths. Despite these advances, sampling-based approaches are often still favored for planning in high-dimensions due to their attractive planning speeds. By abandoning deterministic guarantees on optimality of found solutions, sampling-based planners randomly probe the configuration space to quickly find feasible trajectories. As a result, the solutions found by these planners can be inconsistent and exhibit high variance, whereas those found by the more deterministic search-based planners are fairly consistent with each other.[1]

Nicholson first proposed bidirectional search as two separate searches initialized from both the start and goal [2]. In theory, these planners can exponentially reduce the number of expanded states, making them an attractive class of algorithms. It is well-known that informative heuristics significantly improve search efficiency for unidirectional search, so using them for both searches in a bidirectional algorithm is an intuitive extension. In practice, however, the frontiers of both search efforts often miss each other entirely, without any overlap in the search space. The searches can end up expending significantly more effort in these cases, contrary to the motivating intuition. This problem is exacerbated in high-dimensional state spaces, which are commonly found in robotic motion planning problems. Bidirectional sampling-based approaches on the other hand, such as the RRT-Connect algorithm [3], offer fast planning performance and scale well with high-dimensionality but at the cost of deterministic theoretical guarantees on solution quality and completeness.

In our work, we present a simple but effective extension that aims to overcome the known shortcomings of bidirectional heuristic search with the use of an extend operator. The extend operator is used to connect the search frontiers being explored simultaneously but independently from the start and goal configurations. This operator was originally proposed as a greedy heuristic for connecting forward and backward searches in bidirectional sampling-based planners [3]. Motivated by the example described earlier, we adopt this operator for the heuristic search setting to efficiently reason through free space in high DoF motion planning. With our modification, we maintain consistency and deterministic bounds on the suboptimality of found solutions, for a given resolution of action and state space.

II. Related Work

A. Bidirectional Planning Algorithms

1) RRT-Connect: The RRT-Connect algorithm [3] first introduced the extend function in the context of motion planning without kinodynamic constraints. The sampling-based planner dynamically constructs two rapidly-exploring random trees [4] (from the start and goal configurations). It attempts to connect both trees using an EXTEND operator.

All authors are affiliated with The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA {allenc1, dsaxena, maxim}@cs.cmu.edu

[1] *Consistency* refers to the likeness between paths with similar start-goal pairs with respect to some metric.

This operator takes a single, fixed-size step towards the nearest-neighbor in the opposite tree. A CONNECT function repeatedly executes EXTEND until a node in the opposite tree is reached or an obstacle blocks any further extension attempts. The greedy nature of RRT-Connect leads to fast performance and its sampling behavior allows it to translate well to high-dimensional motion planning problems. However, the algorithm has no guarantees on optimality or bounded suboptimality, and solution quality can vary greatly. Recent work [5], [6] has attempted to address this by adopting the local rewiring procedure from RRT* [7], but these approaches can only guarantee asymptotic optimality.

2) Heuristic Search-Based Algorithms: Pohl first explored the combination of bidirectional and heuristic search with the classical front-to-back bidirectional heuristic search algorithm, BHPA [8]. However, the theoretical benefits were never consistently reflected in experiments. It was later noted that the frontiers for forward and backward search often missed or crossed through each other in practice [9]. This observation, considered to be crux of bidirectional heuristic search, motivated Champeaux and Sint to develop the Bidirectional Heuristic Front-to-Front Algorithm (BHFFA) [10]. BHFFA achieves reduced expansions compared to BHPA at the cost of having to exhaustively compute a front-to-front distance for entire frontiers. For high-dimensional motion planning, this is prohibitively expensive to compute. Recent research has focused on ensuring the two search efforts "meet in the middle" through the use of a unique priority function [11]. However, this property has no clear use in the context of complex motion planning, where frontier intersection may not always be in the middle.

3) A-Connect:* The recently introduced A*-Connect algorithm [12] leverages a Multi-Heuristic A* (MHA*) [13] framework to guide search bidirectionally towards both the opposite root and frontier in parallel. MHA* provides a structured framework enabling the use of multiple inadmissible heuristic functions while still providing guarantees on completeness and bounds on suboptimality via a consistent anchor heuristic. MHA* maintains separate priority queues containing the frontier states for each heuristic for each search. Specifically, A*-Connect runs two Improved MHA* searches [14]. For each, the consistent anchor heuristic serves as a front-to-back estimate. Additional heuristics, referred to as *connect heuristics*, are used to estimate the front-to-front cost to the opposite frontier. However, instead of exhaustively computing the front-to-front heuristic between all pairs of states in the frontier of the opposite search, A*-Connect computes the connect heuristic for selected pivot states. Pivot states are the last expanded states of the opposite search for each heuristic, and serve as an estimate for the "most promising" state for frontier connection. Effectively, the connect heuristic is a fast-to-compute estimate of the cost to the opposite frontier. A*-Connect produces bounded suboptimal solutions by addressing the issue of crossing frontiers in an efficient, structured manner. Our proposed algorithm similarly uses front-to-end heuristics for structured bidirectional search, but instead of an additional connect heuristic to guide front-to-front search, an extend operator is used to greedily extend to the opposite tree.

B. Tree Extension

1) EXTEND/CONNECT Operators: The extend operator responsible for connecting pairs of states has been widely explored in robotic motion planning. The extend operator's job is two-fold: it must select a state as a local goal using a distance metric and perform the extension itself. For bidirectional sampling-based methods, nearest-neighbors (NN) are selected as extension targets. For problems with kinematic or kinodynamic constraints, the extension must solve a two-point boundary value problem (BVP). Lavalle [15] proposes motion primitives that describe a predefined set of discretized control actions, but this does not fully solve the BVP. Prior research has offered solutions that include the use of optimal controllers [16] and splines [17]. Since we use the same nearest-neighbors extend operator in our approach, these methodologies and adaptations to various domains can be easily adopted into our algorithm.

2) Adaptive Motion Primitives: Cohen, et al. use adaptive motion primitives [18] that are generated on-the-fly during search. In the example scenario we introduced previously, they are used to connect the search frontier to the goal configuration. The goal may be underdefined vis-á-vis degrees-of-freedom of the robot and thus differ from the dimensionality of the search space. For example, we may specify a 6 DoF goal pose in $SE(3)$ while planning for a 7 DoF robot arm (i.e. planning for 7 joints). The redundant joint allows for a set of states that satisfy the goal constraint. An analytical inverse kinematics (IK) solver is used to generate a valid state when the search is sufficiently close to the goal. A dynamically constructed motion primitive is used to reach the goal state. For the robotic arm, this translates into to a linearly-interpolated path between the valid search state and the goal state. The use of adaptive motion primitives is shown to improve search efficiency while preserving the planner's theoretical guarantees. The extend operator can be viewed as an adaptive motion primitive to connect more generally to any state in the opposite frontier.

III. BIDIRECTIONAL HEURISTIC SEARCH WITH AN EXTEND OPERATOR

A. Notation and Assumptions

We assume the planning problem can be formulated as a graph-search problem, where S refers to the finite set of states for the planning domain. The start and goal states are denoted as s_{start} and s_{goal} respectively. The cost function $c(s,s')$ denotes the cost of the edge between states s and s'. If no such edge exists, $c(s,s') = \infty$. We further assume that $c(s,s') \geq 0 \;\forall\; s,s'$ pairs. Let $g(s)$ denote the current best path cost from s_{start} to s. The successor function, $\textsc{Succ}(s) := \{s' \in S | c(s,s') \neq \infty\}$, denotes the set of reachable successors for state s. The optimal path between s and s' has cost $c^*(s,s')$. The optimal path from s_{start} to s has cost $g^*(s)$.

The heuristic for state s is denoted as $h(s)$ and is used to estimate the best path cost from s to s_{goal}. $h(s)$ is considered *admissible* if it never overestimates the path cost to s_{goal}, that is $h(s) \leq c^*(s, s_{goal}) \forall s \in S$, and *consistent* if it satisfies $h(s_{goal}) = 0$ and $h(s) \leq h(s') + c(s, s') \forall s, s'$ where $s' \in \textsc{Succ}(s)$, $s \in S$, and $s \neq s_{goal}$. OPEN denotes a priority queue typically ordered with priority $f(s) = g(s) + h(s)$ or $f(s) = g(s) + \omega h(s)$ given an inflation factor $\omega \geq 1$. This is used to store frontier states, and CLOSED denotes states that have been expanded.

We use dir and $\overline{dir}$ to denote the current and opposite search directions respectively. If $s \notin (\text{OPEN}_{dir} \cup \text{CLOSED}_{dir})$, $g_{dir}(s) = \infty$. An extension function $\textsc{NewState}(s, s_{NN}, s')$ (Alg. 1, line 7), takes in search state s and the nearest neighbor in the opposite tree s_{NN}. It greedily moves from s towards s_{NN} by taking a step of ε-distance (or smaller if distance from s to s_{NN} is smaller) and is successful if resulting state s' is reachable and collision-free.

B. Algorithm

At a high level, our algorithm runs two A*-like algorithms from both start and goal configurations while using an extend operator to guide front-to-front connection. We choose to analyze our strategy using weighted A* (WA*) [19] to run the forward and backward searches. In effect, an inflation factor $\omega \geq 1$ is used to bias the front-to-back heuristic. Since our approach is bidirectional, we initialize separate OPEN and CLOSED sets for each search direction. s_{start} is initialized as the start configuration for the forward search, and the goal configuration for the backward search. s_{goal} follows in a complementary manner. The main loop iteratively runs WA* where CONNECT is called after every expansion. If connection is successful, the connecting state in the opposite frontier is added to the current OPEN list. After each iteration, the search direction is swapped (Alg. 2 line 29). Search stops when s_{min} satisfies the TERMINATIONCRITERION (Alg. 2 line 3). This occurs when the considered state exists in both search efforts. In other words, s_{min} is either in the OPEN or CLOSED lists for both directions (Alg. 2 line 4). At this point, the search terminates, and the solution path is reconstructed by recursively tracing the parents of the connecting state in both directions.

C. Extend Operator

The connection process, depicted in Fig. 1, is a slightly modified version of the one used in RRT-Connect and can be viewed as a greedy front-to-front heuristic to directly address the problem of missing frontiers. Given a state and the current search direction, CONNECT first selects the nearest-neighbor s_{NN} in the opposite search tree using a domain-dependent distance function, $\triangle$ (Alg. 1, line 4). It then attempts to connect to the chosen state by repeatedly applying EXTEND until either s_{NN} is reached or no further extensions are possible. Feasible extension states from NEWSTATE are treated similarly to successor states in A* expansion and added to OPEN_{dir} with priority PRIORITY-C if s' is not in CLOSED_{dir}. For intermediate states (where $s' \neq s_{NN}$), this is would be equivalent to PRIORITY (Alg. 2, line 1). In

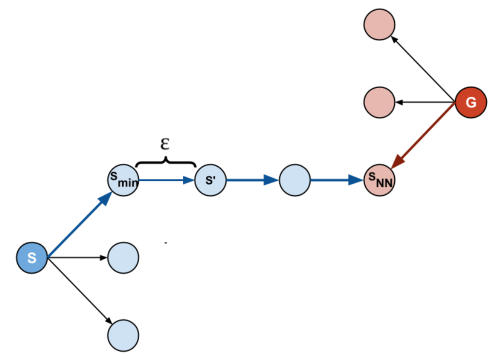

Fig. 1: *Extend Operator* On expansion, s_{min} takes ε-size step to successor states, s', towards the nearest-neighbor state, s_{NN}. If a connection is made, s_{NN} has the root-to-state cost for forward and backward search, as shown with blue and red edges respectively.

Algorithm 1 Extend Operator

1: **procedure** PRIORITY-C(s, dir)
2: **return** $g_{dir}(s) + \min(\omega \cdot h_{dir}(s), g_{\overline{dir}}(s))$
3: **procedure** NEARESTNEIGHBOR(s_{cur}, dir)
4: **return** $\underset{s \in \text{OPEN}_{\overline{dir}} \cup \text{CLOSED}_{\overline{dir}}}{\arg\min} \triangle(s, s_{cur})$
5: **procedure** EXTEND(s, s_{NN}, dir)
6: EXTENDSTATUS $\leftarrow$ *Trapped*
7: **if** NEWSTATE(s, s_{NN}, s') **then**
8: **if** $s' = s_{NN}$ **then**
9: EXTENDSTATUS $=$ *Reached*
10: **else**
11: EXTENDSTATUS $=$ *Advanced*
12: **if** $s' \notin \text{OPEN}_{dir}$ **then**
13: $g_{dir}(s') = \infty$
14: **if** $g_{dir}(s') > g_{dir}(s) + c(s, s')$ **then**
15: $g_{dir}(s') = g_{dir}(s) + c(s, s')$
16: **if** $s' \notin \text{CLOSED}_{dir}$ **then**
17: Insert/Update s' in OPEN_{dir} with PRIORITY-C(s', dir)
18: **return** EXTENDSTATUS
19: **procedure** CONNECT(s, dir)
20: $s_{NN} \leftarrow$ NEARESTNEIGHBOR(s, dir)
21: **repeat**
22: EXTENDSTATUS $\leftarrow$ EXTEND(s, s_{NN}, dir)
23: **until not** EXTENDSTATUS $=$ *Advanced*

the case that a successful connection between both searches exists (when $s' = s_{NN}$), PRIORITY-C additionally considers $g_{\overline{dir}}(s')$, calculated in the opposite search, when estimating cost-to-go for s'.

Variations to the algorithm naturally arise as one may choose the frequency at which valid intermediate states s' are added to the open list. In the context of heuristic search, intermediate states added from extend iterations may not lie on the discretized graph induced by the SUCC function. For this reason, we focus on the variation that does not add these intermediate successors, effectively requiring that EXTENDSTATUS $=$ *Reached* before adding to OPEN_{dir} in line 17 in Algorithm 1.

Algorithm 2 Bidirectional WA* with Extend Operator

1: **procedure** PRIORITY(s, dir)
2:     **return** $g_{dir}(s) + \omega \cdot h_{dir}(s)$
3: **procedure** TERMINATIONCRITERION(s)
4:     **return** $s \in (\text{OPEN}_f \cup \text{CLOSED}_f) \cap (\text{OPEN}_b \cup \text{CLOSED}_b)$
5: **procedure** EXPAND(s, dir)
6:     Remove s from OPEN_{dir}
7:     $\text{CLOSED}_{dir} \leftarrow \text{CLOSED}_{dir} \cup \{s\}$
8:     **for** $s' \in \text{SUCC}_{dir}(s)$ **do**
9:         **if** $s' \notin \text{OPEN}_{dir}$ **then**
10:             $g_{dir}(s') = \infty$
11:         **if** $g_{dir}(s') > g_{dir}(s) + c(s, s')$ **then**
12:             $g_{dir}(s') = g_{dir}(s) + c(s, s')$
13:             **if** $s' \notin \text{CLOSED}_{dir}$ **then**
14:                 Insert/Update s' in OPEN_{dir} with PRIORITY-C(s', dir)
15: **procedure** MAIN($args$)
16:     $\text{OPEN}_f \leftarrow \emptyset, \text{OPEN}_b \leftarrow \emptyset$
17:     $\text{CLOSED}_f \leftarrow \emptyset, \text{CLOSED}_b \leftarrow \emptyset$
18:     $g_f(s_{start}) \leftarrow 0, g_f(s_{goal}) \leftarrow \infty$
19:     $g_b(s_{start}) \leftarrow \infty, g_b(s_{goal}) \leftarrow 0$
20:     Insert s_{start} into OPEN_f with PRIORITY-C(s_{start}, f)
21:     Insert s_{goal} into OPEN_b with PRIORITY-C(s_{goal}, b)
22:     $dir = f$
23:     **while not** OPEN_{dir}.EMPTY() **do**
24:         $s_{min} \leftarrow \underset{s \in \text{OPEN}_{dir}}{\arg\min}$ PRIORITY-C(s, dir)
25:         **if** TERMINATIONCRITERION(s_{min}) **then**
26:             **return** EXTRACTPATH(s_{min})
27:         EXPAND(s_{min}, dir)
28:         CONNECT(s_{min}, dir)
29:         $dir = \overline{dir}$

D. Theoretical Analysis

These properties hold for a bidirectional heuristic-search using any A* variant for forward and backward search, but we consider the case of using WA*. As our framework follows closely using forward and backward WA* (*without* re-expansions), we borrow the proofs of [20].

Theorem 1: On expansion for each direction, it holds that g-values are at most ω-suboptimal, i.e., $g_{dir}(s) \leq \omega g^*_{dir}(s) \forall s \in \text{CLOSED}_{dir}$.

Proof: (Sketch) As WA* is used for both searches here, we directly borrow this property from [20].

Theorem 2: On expansion for each direction, it holds that $\text{PRIORITY}(s_{min}, dir) \leq \omega g^*(s_{goal})$ for $s_{min} = \underset{s \in \text{OPEN}_{dir}}{\arg\min} \text{PRIORITY}(s, dir)$.

Proof: (Sketch) This proof directly follows Theorem 2 of [13].

Theorem 3: (Bounded suboptimality) When the main search exits, the returned solution (if one exists) has a cost which is at most ω-suboptimal, or $g(s_{goal}) \leq \omega g^*(s_{goal})$.

Proof: (Sketch) The strategy terminates either when both OPEN lists are exhausted (no solution exists) or when TERMINATIONCRITERION is satisfied, which occurs when expanded state s_{min} exists in both searches (Alg. 2 line 25). There are two cases for which s_{min} is expanded and it exists in both searches:

1) s_{min} was expanded in search dir, and was generated as a reachable successor of another state via SUCC in search $\overline{dir}$
2) s_{min} was expanded in search dir, and a successful connection was made to it via EXTEND from search $\overline{dir}$

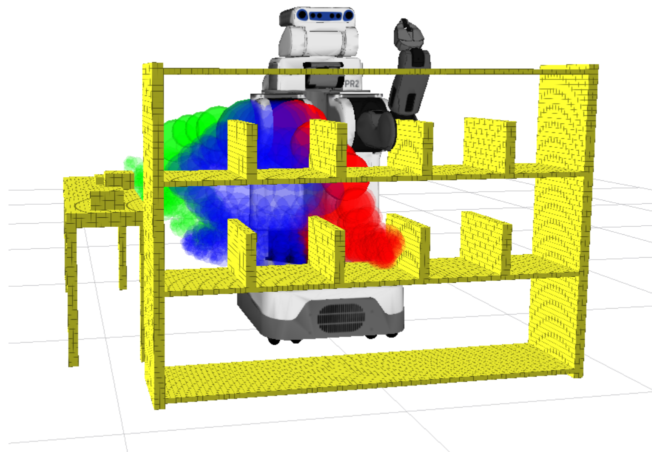

Fig. 2: Visualization of WA*-Extend for the single-arm planning domain. The found trajectory is colored to denote the different regions of search. Green and red denote the forward and backward search respectively, and blue highlights the connection between frontiers.

The first scenario is a direct extension of WA*, so we can apply the same proof as described in [20]. Let us now consider the second scenario where a connection is made. Since connection was successful, there are two possible priorities of s_{min} stemming from Alg. 1 line 2. The unique case arises when $\text{PRIORITY-C}(s_{min}, dir) = g_{dir}(s_{min}) + g_{\overline{dir}}(s_{min})$. By definition, $\text{PRIORITY-C}(s_{min}, dir) \leq \text{PRIORITY}(s_{min}, dir)$. When s_{min} is expanded with a priority based on two g-values, then $g(s_{goal}) \leq g_{dir}(s_{min}) + g_{\overline{dir}}(s_{min}) = \text{PRIORITY}(s_{min}, dir) \leq \omega g^*(s_{goal})$, by Theorem 2.

IV. EXPERIMENTAL RESULTS

We compare our approach to search-based and sampling-based planners in the context of high-dimensional, single-arm manipulation. Since any graph search algorithm may be used in our framework, we choose to run our algorithm, that is weighted A* without re-expansions, bidirectionally with an extend operator (WA*-Extend). We evaluate WA*-Extend's performance against unidirectional WA*, A*-Connect, BHPA$_w$ [21], RRT-Connect, and RRT*. These include unidirectional, bidirectional, search-based, and sampling-based planners.

A. Single-Arm Motion Planning

We evaluate the use of the extend operator in bidirectional heuristic search in the domain of high-dimensional manipulation. All experiments use a simulated model of a PR2 robot's 7 DoF arm. Each test involves generating collision free trajectories to a specified goal. Start configurations are randomly generated such that they lie above the cluttered

TABLE I: Performance for Single-Arm Motion Planning

	WA*-Extend	WA*	A*-Connect	BHPA	RRT-Connect	RRT*
Success (%)	99	58	100	98	100	98
State Expansions	**2027.24**	119284.95	21364.71	43638.84	-	-
Solution Cost (rad)	14.43	**11.17**	11.59	12.77	22.49	16.55
Total Plan Time(s)	3.47	31.58	6.44	17.51	**0.09**	7.09
Post Processing Time (ms)	6.95	7.97	6.63	6.07	5.54	**4.44**
Processed Solution Cost (rad)	11.68	9.59	**9.04**	10.88	12.89	10.41

table top, and goal configurations are similarly generated to be within a feasible region in the shelf. The presented setup in Fig. 2 reproduces manipulation tasks that are common in industrial settings where large amounts of free space exists between start and goal configurations. In the presence of high clutter all throughout the workspace, the search would less likely be able to utilize the extend operator, leading to performance degradation.

1) Heuristics: For high-dimensional planning, it is well-known that solutions to simplified, lower-dimensional problems can be used as informative heuristic for the original problem [22]. In the context of single-arm planning, a 6 DoF goal constraint describes the desired end effector position and orientation in $SE(3)$. The solution to the relaxed problem of only considering the goal position $(x,y,z) \in \mathbb{R}^3$ performs well as a proxy heuristic for the full dimensional representation. Cohen et al [23], discretize the environment using 3D voxels and find the shortest feasible path to the simplified goal using breadth first search, referred to as h_{BFS}. In practice, this heuristic proves to outperform other commonly used estimates such as h_{euc}, the Euclidean distance between states, when applied to obstacles in cluttered workspaces.

2) Motion Primitives: Motion primitives are used to encode kinematic constraints of a robot and ensure that a valid transition between adjacent states exists. In search-based planning, the robot's action space is discretized to control the branching factor of search. For the 7 DoF arm, we use a base set of 14 static motion primitives, corresponding to moving each joint by a small, fixed amount in both directions.

3) Extend Operator: The extend operator used in this experiment selects the nearest-neighbor in joint space and attempts a direct extension using a linearly interpolated path. For all our experiments, the bidirectional search we implement executes the extend operator until either a connection to s_{NN} is made, or the extension is trapped due to an obstacle and cannot go any further. Another option would be to directly interpolate between end-effector poses similar to the approach of Cohen et al [18] for adaptive snapping to goal configuration. This requires repeatedly computing inverse kinematics solutions which can be computationally costly.

B. Implementation Details

Since our strategy requires extensions to the opposite frontier, we use the k-d tree library nanoflann [24] to dynamically add states and perform efficient NN queries. The search-based planners in our tests use an inflation factor ω of 100. h_{BFS} is used as the front-to-back, anchoring heuristic for

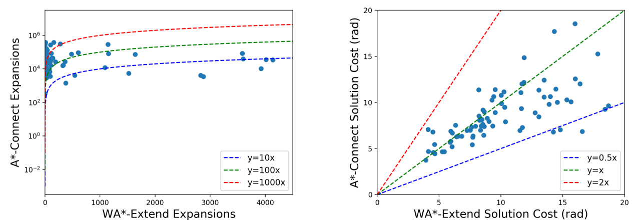

Fig. 3: In the left (right) plot, each data point illustrates WA*-Extend state expansions (post-processed solution cost) against A*-Connect for a single run.

A*-Connect as well as the main informative heuristic for remaining heuristic search-based planners. For the connect heuristic, we use the fast-to-compute h_{euc} as Islam suggests [12]. For bidirectional planners, we swap directions after expanding 10 times [12], but attempt to connect frontiers on each iteration WA*-Extend. Our baseline sampling-based planners, RRT-Connect and RRT*, use implementations provided from the Open Motion Planning Library (OMPL) [25]. RRT* is conditioned to terminate after a first valid solution is found. We post process all solutions using existing OMPL implementations to simplify and smooth paths.

C. Results

Table I compares our approach against a suite of planners. We consider 100 different planning problems with randomly generated start and goal pairs such that all planners find a solution for them given a large timeout. We then say a planner *failed* if that solution took longer than 120 seconds to find. Except for the success percentage, only the successful trials are used to compute the statistics in Table I.

1) Comparison with Search-Based Planners: Bidirectional variants are faster to find solutions compared to unidirectional planners for the high DoF planning problems we consider. Compared to the other search-based planners, our approach has the largest averaged solution cost but outperforms its search-based counterparts in terms of speed and reduced state expansions. In Fig. 3, we compare our approach to A*-Connect, which is more competitive compared to the other bidirectional heuristic search, $BHPA_w$. The left plot shows how WA*-Extend commonly expands 100 to 1000 times less states than A*-Connect on each run. The right plot illustrates that comparable solution cost is achieved after post-processing. Ultimately, this highlights the extend operator's effectiveness in efficiently connecting frontiers. In our example domain, this translates to interpolated paths in

TABLE II: Evaluating Solution Consistency

	WA*-Extend	WA*	A*-Connect	BHPA$_w$	RRT-Connect	RRT*
μ_c	12.79	8.81	9.30	15.92	27.43	12.71
σ_c	-	-	-	-	8.06	2.11

free regions as shown in Fig. 2 where the backward search is able to extend to the opposite frontier after planning out of the constrained cubby region near the goal configuration.

2) Comparison with Sampling-Based Planners: While RRT-Connect is fastest in terms of planning time, our solutions are competitive in other criteria. We borrow the distance metric used by Cohen et al. [18] to measure solution consistency across planners. This is defined as the ratio between the unprocessed solution path length and euclidean distance between start and goal configurations. We execute 50 runs for each planner, using the same start and goal configurations each time. The mean and standard deviation for this consistency metric, μ_c and σ_c, are recorded in Table II. For search-based planners, this value is deterministic. Without post-processing, solutions found with RRT-Connect are of poor quality as first observed in Table I. With respect to the defined distance metric, RRT-Connect has high variance for the same start and goal configurations. Inconsistent solutions are undesirable in applications that require high repeatability such as automated manufacturing. In highly cluttered environments, where post-processing methods might not always be able to converge, RRT-Connect can produce inconsistent solutions. However, our approach is grounded from using a search-based framework and finds consistent solutions.

V. Conclusion

In this work, we presented a simple extension to bidirectional heuristic search that enables fast and bounded suboptimal solutions. Bidirectional heuristic search faces the trade-off of ensuring intersecting frontiers with competitive planning efficiency. Our strategy capitalizes on large free space in high DoF motion planning to connect frontiers using an extend operator popular in sampling-based planners. These extend operators may need to be adapted to take into account the kinodynamic constraints of robots and take advantage of redundant degrees-of-freedom of complicated robots. This leaves open the possibility of further research into better operators and extension strategies. By running WA* bidirectionally with an informative front-to-end heuristic, we maintain deterministic, theoretical guarantees that sampling-based planners lack. We also provide consistent solution paths as a result of using deterministic, search-based methods. From our simulated experiments, our approach achieves significantly faster search efficiency compared with other search-based strategies and makes strides towards closing the performance gap between heuristic-based and sampling-based planners in high DoF motion planning.

Acknowledgements

This work was in part supported by ONR grant N00014-15-1-2129

References

[1] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *Fourth annual symposium on combinatorial search*, 2011.

[2] T. A. J. Nicholson, "Finding the shortest route between two points in a network," *The computer journal*, vol. 9, no. 3, pp. 275–280, 1966.

[3] J. J. Kuffner Jr and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000.

[4] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[5] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[6] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.

[7] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics: Science and Systems VI, Universidad de Zaragoza, Zaragoza, Spain, June 27-30, 2010*, 2010. [Online]. Available: http://www.roboticsproceedings.org/rss06/p34.html

[8] I. Pohl, "Bidirectional and heuristic search in path problems," Tech. Rep., 1969.

[9] J. B. Kwa, "Bs: An admissible bidirectional staged heuristic search algorithm," *Artificial Intelligence*, vol. 38, no. 1, pp. 95–109, 1989.

[10] D. de Champeaux and L. Sint, "An improved bidirectional heuristic search algorithm." *J. ACM*, vol. 24, no. 2, pp. 177–191, 1977.

[11] R. C. Holte, A. Felner, G. Sharon, and N. R. Sturtevant, "Bidirectional search that is guaranteed to meet in the middle," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[12] F. Islam, V. Narayanan, and M. Likhachev, "A*-connect: Bounded suboptimal bidirectional heuristic search," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2752–2758.

[13] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[14] V. Narayanan, S. Aine, and M. Likhachev, "Improved multi-heuristic a* for searching with uncalibrated heuristics," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[15] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[16] D. J. Webb and J. v. d. Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," *arXiv preprint arXiv:1205.5088*, 2012.

[17] K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim, and S. Joo, "Spline-based rrt path planner for non-holonomic robots," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 763–782, 2014.

[18] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5478–5485.

[19] I. Pohl, "First results on the effect of error in heuristic search," *Machine Intelligence*, vol. 5, pp. 219–236, 1970.

[20] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara: formal analysis," 2003.

[21] A. L. Koll and H. Kaindl, "Bidirectional best-first search with bounded error: Summary of results," in *Proceedings of the 13th international joint conference on Artifical intelligence-Volume 1*. Morgan Kaufmann Publishers Inc., 1993, pp. 217–223.

[22] B. Bonet, G. Loerincs, and H. Geffner, "A robust and fast action selection mechanism for planning," in *AAAI/IAAI*, 1997, pp. 714–719.

[23] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2902–2908.

[24] J. L. Blanco and P. K. Rai, "nanoflann: a c++ header-only fork of flann, a library for nearest neighbor (nn) wih kd-trees," 2014.

[25] 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.