

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

# AORRTC: Almost-Surely Asymptotically Optimal Planning with RRT-Connect

Tyler S. Wilson<sup>1</sup>, Wil Thomason<sup>2</sup>, Zachary Kingston<sup>3</sup>, and Jonathan D. Gammell<sup>1</sup>

**Abstract**—Finding high-quality solutions quickly is an important objective in motion planning. This is especially true for high-degree-of-freedom robots. Satisficing planners have traditionally found feasible solutions quickly but provide no guarantees on their optimality, while almost-surely asymptotically optimal (a.s.a.o.) planners have probabilistic guarantees on their convergence towards an optimal solution but are more computationally expensive.

This paper uses the AO- $x$  meta-algorithm to extend the satisficing RRT-Connect planner to optimal planning. The resulting Asymptotically Optimal RRT-Connect (AORRTC) finds initial solutions in similar times as RRT-Connect and uses additional planning time to converge towards the optimal solution in an anytime manner. It is proven to be probabilistically complete and a.s.a.o.

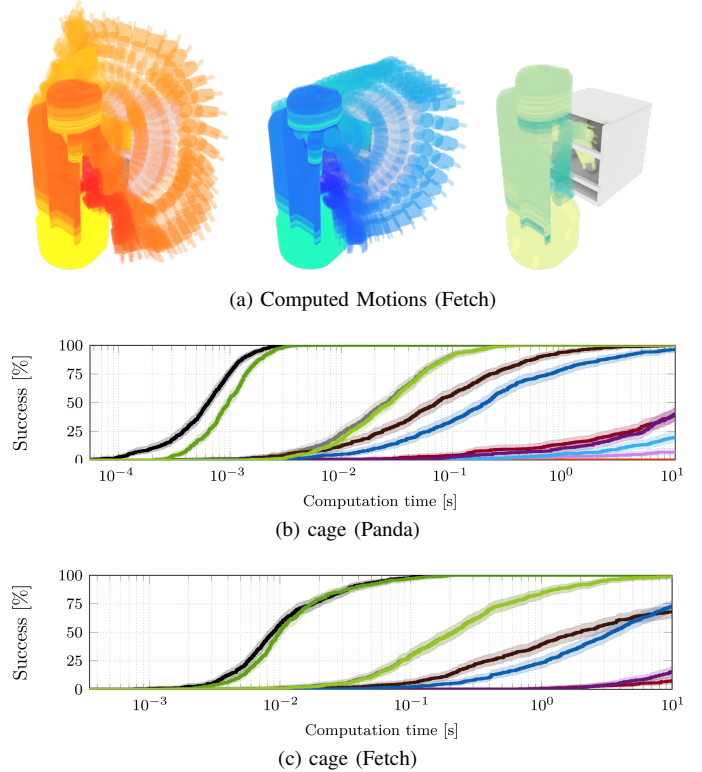
AORRTC was tested with the Panda (7 DoF) and Fetch (8 DoF) robotic arms on the MotionBenchMaker dataset. These experiments show that AORRTC finds initial solutions as fast as RRT-Connect and faster than the tested state-of-the-art a.s.a.o. algorithms while converging to better solutions faster. AORRTC finds solutions to difficult high-DoF planning problems in *milliseconds* where the other a.s.a.o. planners could not consistently find solutions in seconds. This performance was demonstrated both with and without single instruction/multiple data (SIMD) acceleration.

**Index Terms**—Manipulation Planning, Motion & Path Planning

## I. INTRODUCTION

**M**OTION planning seeks to quickly find high-quality solutions to a given problem, especially when planning for high degree-of-freedom (DoF) robots or in real time. Motion planning algorithms search a discrete approximation of the robot’s continuous *configuration space* (i.e., search space).

Motion planning algorithms approximate the search space in different ways. Graph-based planners, such as Dijkstra’s algorithm [1] and A\* [2], require *a priori* discretization of the search space. High-resolution approximations usually contain high quality solutions but are computationally expensive to search, while low-resolution approximations are cheaper to search but may only contain low quality solutions, or no solution at all. Trajectory optimization methods, such as CHOMP [3] and TrajOpt [4], are less dependent on their approximation but



— RRTC (OMPL) — RRT\* (OMPL) — BIT\* (OMPL) — AIT\* (OMPL)  
— RRTC\* (OMPL) — RRTC — RRT\* — BIT\* — FCIT\* — AORRTC (OMPL) — AORRTC

Figure 1: Results for all planners on the 7 DoF Panda problems, and a subset of planners on the 8 DoF Fetch problems, from the MotionBenchMaker [5] dataset. All planners are implemented in either OMPL (labelled with *OMPL*) or are implemented in VAMP. The paths for the challenging 8 DoF Fetch *cage* problem (Fig. 3c) are shown in (a) with the VAMP RRT-Connect solution found after 62ms in orange, the initial VAMP AORRTC solution (i.e., RRT-Connect with path simplification) found after 63ms in blue, and the last VAMP AORRTC solution found after 3.5s in green. The percentage of trials that found a solution on the *cage* environment within the given time are shown in (b) and (c) with Clopper-Pearson 99% confidence intervals [6]. Both VAMP and OMPL AORRTC find initial solutions significantly faster than all other tested VAMP and OMPL a.s.a.o. planners.

only provide local guarantees and may not find a solution when getting stuck in local minima on difficult planning problems [4].

Sampling-based planners, such as Probabilistic Roadmaps (PRM) [7] and Rapidly-exploring Random Trees (RRT) [8], avoid the need for an *a priori* approximation by incrementally sampling the search space. This allows them to search an increasingly accurate approximation of the underlying continuous space and has made them a popular and effective choice for high-dimensional problems. Sampling-based planners provide probabilistic guarantees. Algorithms are said to be probabilistically complete if their probability of finding a

Manuscript received: May, 15, 2025; Revised August, 10, 2025; Accepted September, 11, 2025.

This paper was recommended for publication by Editor Júlia Borràs Sol upon evaluation of the Associate Editor and Reviewers’ comments.

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

<sup>1</sup>Estimation, Search, and Planning (ESP) Research Group, Queen’s University, Kingston ON, Canada. {18tsw1, gammell}@queensu.ca

<sup>2</sup>wil.thomason@gmail.com

<sup>3</sup>Department of Computer Science, Purdue University, West Lafayette IN, USA zkingston@purdue.edu

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

©2026 IEEE

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.**

solution goes to one as their number of samples approaches infinity, if a solution exists. They are said to be almost-surely asymptotically optimal (a.s.a.o.) if they have probability one of asymptotically converging towards the optimal solution with an infinite number of samples [9].

RRT-Connect [10] extends RRT to interleave searching for a feasible path from both the start and goal. This bidirectional satisficing planner is widely used because of its simple implementation and fast initial solution time. RRT-Connect provides no solution quality guarantees and does not improve its solution with more samples (i.e., it is not a.s.a.o.). Path smoothing or simplification [11–14] can improve the cost of the often low-quality solutions found by RRT-Connect (Fig. 1a) but provide no global quality guarantees.

Anytime a.s.a.o. planners, such as RRT\* [9] and Batch Informed Trees (BIT\*) [15], are probabilistically guaranteed to find a solution and then converge to the optimum. These algorithms use additional planning time to improve their current solution, but the overhead required to guarantee almost-sure asymptotic optimality can increase the time required to find an initial solution. Research has focused on how to improve initial solution times and the rate of convergence towards the optimum [16].

The AO- $x$  meta-algorithm [17] poses an alternative framing of the optimal planning problem. Instead of requiring planners to optimize the cost of a feasible path in configuration space, AO- $x$  asks planners to find a feasible path through a search space that has been augmented to include an extra dimension. This extra dimension represents the cost to reach each configuration. Calling a satisficing planner on a *sequence* of these augmented search spaces with appropriately decreasing limits in the cost dimension has been proven to be a.s.a.o. [17, 18].

Asymptotically Optimal RRT-Connect (AORRTC)<sup>1</sup> applies the ideas of AO- $x$  and RRT-Connect to perform a bidirectional search in a cost-augmented search space. This finds initial solutions as fast as RRT-Connect and converges towards high-quality solutions orders of magnitudes faster than other a.s.a.o. algorithms. The performance of AORRTC is tested with both Open Motion Planning Library (OMPL) [19] and Vector-Accelerated Motion Planning (VAMP) [20] implementations. These demonstrate the effectiveness of the approach with and without single instruction/multiple data (SIMD) acceleration and specifically show that AORRTC can converge close to the optimum of high-dimensional planning problems in microseconds with SIMD acceleration.

Both implementations of AORRTC were evaluated on the 7 DoF Panda and 8 DoF Fetch planning problems from the MotionBenchMaker (MBM) dataset [5]. AORRTC found initial solutions as fast as RRT-Connect and faster than all tested a.s.a.o. planners while consistently solving more problems (Figs. 1 and 2). It also converged to better solutions faster than the tested a.s.a.o. planners. These relative performances held for both OMPL and VAMP implementations of AORRTC (Sec. V).

## II. RELATED WORK

Improving the performance of real-world motion planning often focuses on finding higher-quality solutions in less time. There has been significant work improving sampling-based motion planning with these goals in mind.

### A. Almost-Surely Asymptotically Optimal Planning

Anytime a.s.a.o. planners use additional planning time to improve their approximation of the search space and find better solutions. These planners asymptotically converge to the optimal solution with probability one (i.e., almost surely).

Nearest neighbour lookups and edge evaluations are major computational costs for sampling-based motion planning [21]. These computationally expensive operations are often performed more frequently in a.s.a.o. planners to guarantee almost-sure asymptotic optimality which can make them slower to find initial solutions than satisficing planners.

Anytime a.s.a.o. planners, such as RRT\* [9] and BIT\* [15], iteratively sample the search space to improve both their approximation and solution. RRT\* incrementally rewires new vertices to reduce path costs while BIT\* searches a batch of samples in an order informed by heuristics to consider states in order of potential solution quality. These planners require computational effort to maintain their a.s.a.o. guarantees and significant work has focused on improving their performance.

Several RRT\* extensions have improved the planner's convergence to an optimal solution. RRT<sup>#</sup> [22] builds a search tree that contains information about the best cost-to-come to all vertices that could possibly belong to an optimal solution instead of only locally rewiring vertices. RRT\*-Smart [23] and Informed RRT\* [24] instead improve the planner's performance by leveraging problem specific information. These planners use information from prior search efforts to generate samples more intelligently in order to reduce planning time when improving a solution but this informed sampling cannot help find an initial solution.

Fully Connected Informed Trees (FCIT\*) [25] leverages the reduced cost of edge evaluations enabled by VAMP's SIMD parallelism to search a fully connected graph. This fully exploits the approximation of the search space and avoids the need to maintain costly nearest neighbour structures. FCIT\* finds high-quality solutions faster than non SIMD-accelerated planners, including RRT-Connect, but cannot find solutions as fast as SIMD-accelerated RRT-Connect.

AORRTC is a.s.a.o. like RRT\* and BIT\* and improves the quality of its found solution given additional planning time, but finds initial solutions as fast as RRT-Connect. Unlike RRT\* or RRT<sup>#</sup>, AORRTC does not have to rewire its tree to maintain its a.s.a.o. guarantees, instead randomly sampling lower cost bounds when adding a new vertex to potentially connect it to a lower-cost parent. AORRTC leverages informed sampling as is done in Informed RRT\*, but restarts its search after finding a solution rather than continuing to plan with the same tree.

### B. Bidirectional Planning

Bidirectional sampling-based planners, such as RRT-Connect [10], explore the search space by extending a tree

<sup>1</sup>Pronounced aortic (ā-'ôr-tik).

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

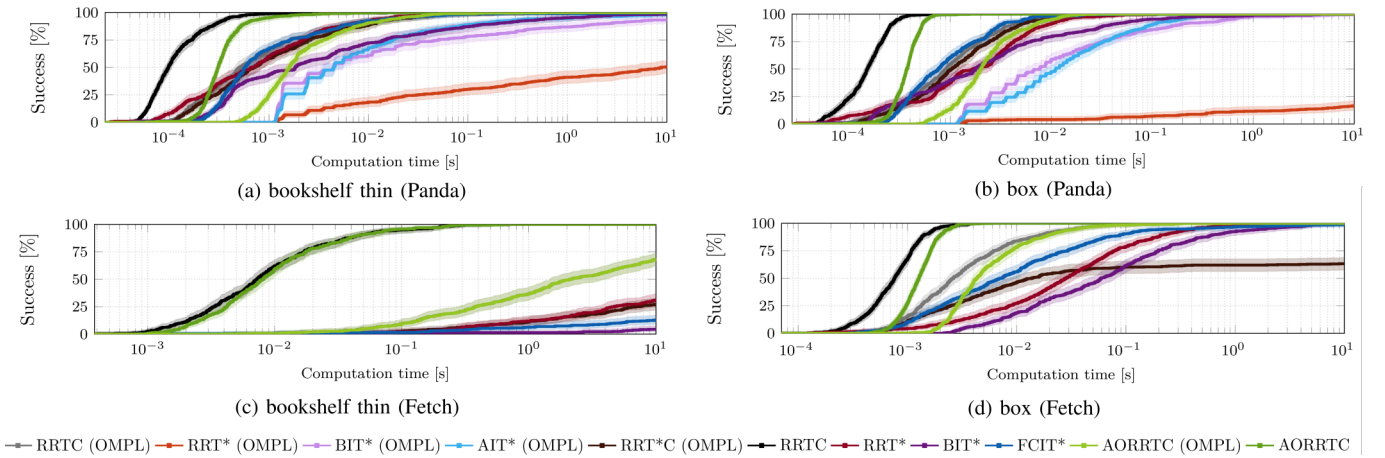


Figure 2: Initial solution results for 5 trials of all planners on the 7 DoF Panda, and a subset of these planners on the 8 DoF Fetch, for all problems from the *bookshelf thin*, (a) and (c), and *box*, (b) and (d), environments from the MotionBenchMaker [5] dataset (Sec. IV). Each plot shows the percentage of runs that found a solution at any given time with Clopper-Pearson 99% confidence intervals [6]. VAMP AORRTC finds initial solutions faster than all other tested a.s.a.o. planners.

from both the start and goal vertices and trying to connect these trees. RRT-Connect finds initial solutions significantly faster than other planning algorithms on most real-world manipulation problems [26] but offers no guarantees on solution quality and often finds low quality solutions.

Some bidirectional a.s.a.o. planners, such as Bidirectional RRT\* (B-RRT\*) [27] and RRT\*-Connect [28], extend the fast bidirectional search of RRT-Connect with the a.s.a.o. guarantees of RRT\*. These algorithms require RRT\*-style rewiring throughout the entire search in order to guarantee almost-surely asymptotic optimality. This increase in computational cost increases the time required to find an initial solution relative to RRT-Connect.

Other bidirectional a.s.a.o. planners, such as Adaptively Informed Trees (AIT\*) and Effort Informed Trees (EIT\*) [16], instead use an asymmetric bidirectional search where information is shared between the reverse and forward searches. The lightweight reverse search calculates heuristics from the current samples to inform the computationally expensive forward search, which finds a solution and passes collision checking information to the reverse search to update the heuristics. This reduces edge evaluation costs but its utility depends on the cost of edge evaluations relative to nearest neighbour lookups.

AORRTC performs an a.s.a.o. bidirectional search similar to B-RRT\* and RRT\*-Connect but does not require RRT\*-style rewiring to maintain its a.s.a.o. guarantees. Unlike AIT\* and EIT\*, AORRTC grows both trees and tries to connect them instead of using the reverse search to calculate heuristics.

### C. Augmented Search Spaces

Augmented search spaces have been used in other planners to improve planning. Time-based RRT (TB-RRT) [29] augments the configuration space with a time dimension for planning with dynamic obstacles or goals. This time-augmented space transforms dynamic obstacles in the original problem into static obstacles in the time-augmented search space [30] and simplifies planning in dynamic environments but offers no guarantees for solution time or quality.

Other planners, such as Windowed Hierarchical Cooperative A\* (WHCA\*) [31] and Safe Interval Path Planning (SIPP) [32], abstract a time-augmented search space to reduce planning time. WHCA\* plans in the time-augmented search space up to a user-specified threshold and then excludes the time dimension and any dynamic obstacles for later times. SIPP discretizes the continuous time dimension into discrete *safe intervals*, which describe a duration during which a configuration is considered valid, and plans in this simplified state-*safe interval* augmented search space. These search space abstractions help improve planner performance, but offer no guarantees on solution quality.

Some bidirectional planners, such as Space-Time RRT\* (ST-RRT\*) [33] and Safe-Interval RRT (SI-RRT) [34], extend RRT-Connect to search in a time-augmented search space. ST-RRT\* searches an incrementally increasing range in the time dimension to quickly find initial solutions in unbounded time spaces but requires RRT\*-style rewiring to maintain its guarantees on solution quality. SI-RRT instead searches in a simplified state-*safe interval* augmented search space to quickly find initial solutions in high-dimensional dynamic environments but offers no guarantees on solution quality.

The meta-algorithm AO-*x* [17] extends satisficing kinodynamic planners to include a.s.a.o. guarantees through the use of a cost-augmented search space. This search space consists of the  $n$  dimensions of the configuration space and a  $(n + 1)$ <sup>th</sup> dimension that describes the cost to reach each state. Satisficing planners can almost-surely converge asymptotically to the optimal solution by finding a series of feasible plans in this augmented search space when the cost function and the dynamics of the robot are Lipschitz continuous [18]. This meta-algorithm has been applied to the Expansive Space Tree (EST) [17] and RRT [18].

AORRTC searches in an augmented space, like TB-RRT and ST-RRT\*, but this space is augmented with a cost dimension instead of a time dimension. Unlike WHCA\*, SIPP, and SI-RRT, AORRTC does not search an abstraction of its augmented search space.

## III. ASYMPTOTICALLY OPTIMAL RRT-CONNECT

AORRTC applies the ideas of AO- $x$  and RRT-Connect to create an anytime a.s.a.o. planner that finds initial solutions as fast as RRT-Connect and asymptotically converges towards the optimal solution with additional planning time in an anytime manner. Pseudocode for AORRTC is presented in Alg. 1 with changes to RRT-Connect (Lines 11 to 47) marked in red.

AORRTC runs RRT-Connect iteratively on a series of problems in the augmented search space with an open upper-bound on solution cost. This bound is first determined by RRT-Connect's initial solution after path simplification techniques (*e.g.*, randomized shortcutting [11, 12] and B-spline smoothing [13]) have been applied (Line 2). AORRTC then uses this bound to pose a new planning problem where the cost-dimension of the augmented space is limited by the cost of the current best solution (Line 5). AORRTC applies path simplification each time a better solution is found and this bound is lowered accordingly (Lines 5 to 8).

AORRTC grows trees from the start and goal vertices in the augmented search space where each vertex consists of a configuration and its cost-to-come in the respective tree (Lines 12 and 13). The current tree is grown towards a random sample from the augmented search space consisting of a random configuration (Line 15) and a randomly sampled cost bound (Line 16). This sampled cost bound is the upper limit on cost for a connection to the sample and limits connections to only those that could contribute to a solution that is higher-quality than the current best solution. After a vertex is connected to a tree its sampled cost bound is randomly resampled from a lower range to see if the vertex can be easily connected with lower cost (Lines 27 to 32).

The underlying RRT-Connect search finds incrementally higher-quality solutions by forcing samples to only attempt connections that would satisfy their randomly sampled cost bound (Line 46). Once a solution has been found it is simplified and a new upper bound on solution cost is determined. The search is then restarted with this tighter cost bound. This restricts each subsequent search to paths with higher quality than the current solution and allows the planner to almost-surely asymptotically converge to an optimal solution even if its underlying search is not a.s.a.o.

## A. Notation

We denote the configuration space by  $X \subseteq \mathbb{R}^n$  and the invalid and valid subsets as  $X_{\text{invalid}} \subseteq X$  and  $X_{\text{valid}} := \text{closure}(X \setminus X_{\text{invalid}})$ , respectively. Let  $x \in X$  be a configuration,  $x_{\text{start}} \in X_{\text{valid}}$  be the starting configuration, and  $x_{\text{goal}} \in X_{\text{valid}}$  be the goal configuration. Let  $c \in \mathbb{R}^+$  be the cost of an edge or set of consecutive edges (*i.e.*, a path) in configuration space. We store the searches as trees,  $T := (V, E)$ , each comprising a set of vertices,  $V \subseteq \mathbb{R}^{n+1}$ , and a set of edges,  $E \subseteq V \times V$ . Each edge,  $e := (v_p, v_c) \in V \times V$ , connects two vertices which we refer to as the edge's parent and child, respectively. We denote a solution as the sequence of edges,  $\sigma = (x_s, v_0), (v_0, v_1), (v_1, v_2), \dots, (v_q, x_g)$ , where  $(v_i, v_j) \in E$ .

The function  $c : X \times X \rightarrow [0, \infty)$  computes the edge cost between two configurations or, with a slight abuse

## Algorithm 1: AORRTC

---

```

1 Function aorrtc()
2    $\sigma_0 \leftarrow \text{simplify}(\text{rrt-connect}(\infty));$ 
3    $\sigma_{\text{best}} \leftarrow \sigma_0; c_{\text{min}} \leftarrow c(\sigma_0);$ 
4   repeat
5      $\sigma_i \leftarrow \text{simplify}(\text{rrt-connect}(c_{\text{min}}));$ 
6     if  $\sigma_i \neq \emptyset$  then
7        $\sigma_{\text{best}} \leftarrow \sigma_i;$ 
8        $c_{\text{min}} \leftarrow c(\sigma_i);$ 
9   until stop;
10  return  $\sigma_{\text{best}};$ 

11 Function rrt-connect( $c_{\text{max}}$ )
12   $V_a \leftarrow \{(x_{\text{start}}, 0)\}; E_a \leftarrow \emptyset; T_a := \{V_a, E_a\};$ 
13   $V_b \leftarrow \{(x_{\text{goal}}, 0)\}; E_b \leftarrow \emptyset; T_b := \{V_b, E_b\};$ 
14  repeat
15     $x_{\text{rand}} \sim \mathcal{U}(X_{\hat{f}});$ 
16     $c_{\text{rand}} \sim \mathcal{U}(\left(\hat{g}_{T_a}(x_{\text{rand}}), c_{\text{max}} - \hat{h}_{T_a}(x_{\text{rand}})\right));$ 
17     $x_{\text{near}} \leftarrow \text{nearest}(T_a, x_{\text{rand}}, c_{\text{rand}});$ 
18     $x_{\text{new}} \leftarrow \text{steer}(x_{\text{near}}, x_{\text{rand}});$ 
19    if validate( $x_{\text{near}}, x_{\text{new}}$ ) then
20       $c_{\text{new}} \leftarrow \text{extend}(T_a, x_{\text{near}}, x_{\text{new}});$ 
21      if connect( $T_b, x_{\text{new}}, c_{\text{new}}$ ) then
22        return path( $T_a, T_b$ );
23    swap( $T_a, T_b$ );
24  until timeout;
25  return  $\emptyset;$ 

26 Function extend( $T, x_{\text{near}}, x_{\text{new}}$ )
27  repeat
28     $x_p \leftarrow x_{\text{near}};$ 
29     $c_{\text{new}} \leftarrow g_T(x_p) + \hat{c}(x_p, x_{\text{new}});$ 
30     $c_{\text{rand}} \sim \mathcal{U}(\left(\hat{g}_T(x_{\text{new}}), c_{\text{new}}\right));$ 
31     $x_{\text{near}} \leftarrow \text{nearest}(T, x_{\text{new}}, c_{\text{rand}});$ 
32  until  $x_{\text{near}} \equiv x_p$  or not validate( $x_{\text{near}}, x_{\text{new}}$ );
33   $E \stackrel{\pm}{\leftarrow} (x_p, x_{\text{new}});$ 
34   $V \stackrel{\pm}{\leftarrow} (x_{\text{new}}, c_{\text{new}});$ 
35  return  $c_{\text{new}};$ 

36 Function connect( $T_b, x, c$ )
37  repeat
38     $x_{\text{near}} \leftarrow \text{nearest}(T_b, x, c_{\text{max}} - c);$ 
39     $x_{\text{new}} \leftarrow \text{steer}(x_{\text{near}}, x);$ 
40    if validate( $x_{\text{near}}, x_{\text{new}}$ ) then
41      extend( $T_b, x_{\text{near}}, x_{\text{new}}$ );
42  until  $x_{\text{new}} \equiv x$  or not validate( $x_{\text{near}}, x_{\text{new}}$ );
43  return validate( $x_{\text{near}}, x_{\text{new}}$ );

44 Function nearest( $T, x, c$ )
45   $x_{\text{near}} \leftarrow$ 
46  arg min  $\{w_x \|x - x_v\|^2 + w_c |c - c_v|^2 \mid c_v + \hat{c}(x_v, x) < c\};$ 
47  return  $x_{\text{near}};$ 

```

---

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.**

Table I: Summary of all planning results on the Panda (7 DoF) and Fetch (8 DoF) robotic arms for the MotionBenchMaker [5] dataset. Each result indicates the percentage of problems solved in the given environment, the median initial solution time across all problems on that environment, and the median initial path length across all problems on that environment. The best result for a given robot and framework on each environment is shown in bold.

	Planner	bookshelf small	bookshelf tall	bookshelf thin	table pick	table under pick	box	cage	
Panda	OMPL	RRT-Connect	100% 0.7ms (10.3)	100% 0.6ms (9.2)	100% 0.5ms (8.9)	100% 0.3ms (7.7)	100% 0.3ms (13.4)	100% 0.9ms (10.9)	100% 25ms (20.8)
		RRT*	38% $\infty$ ( $\infty$ )	39% $\infty$ ( $\infty$ )	51% 8827ms (13.2)	52% 6304ms (13.0)	33% $\infty$ ( $\infty$ )	17% $\infty$ ( $\infty$ )	0% $\infty$ ( $\infty$ )
		BIT*	76% 4.5ms (8.7)	91% 3.4ms (8.6)	93% 4.4ms (8.3)	98% 1.4ms (8.1)	98% 2.2ms (11.4)	100% 7.7ms (9.9)	7% $\infty$ ( $\infty$ )
		AIT*	83% 5.5ms (8.8)	95% 4.4ms (8.5)	98% 4.5ms (8.3)	99% 2.3ms (8.3)	100% 3.4ms (11.5)	100% 11.1ms (10.3)	20% $\infty$ ( $\infty$ )
		RRT*-Connect	98% 0.8ms (8.3)	100% 0.7ms (8.3)	99% 0.7ms (8.2)	100% 0.3ms (8.0)	100% 0.3ms (13.1)	100% 1.0ms (9.7)	100% 72.1ms (14.1)
		AORRTC	100% 1.8ms (5.2)	100% 1.7ms (5.0)	100% 1.6ms (5.0)	100% 1.2ms (4.9)	100% 1.5ms (7.2)	100% 2.0ms (5.7)	100% 28.1ms (8.3)
	VAMP	RRT-Connect	100% 0.1ms (7.0)	100% 0.1ms (6.8)	100% 0.1ms (6.8)	100% 0.1ms (6.4)	100% 0.1ms (10.0)	100% 0.2ms (9.0)	100% 0.6ms (15.4)
		RRT*	99% 0.7ms (8.8)	100% 0.9ms (8.7)	100% 0.7ms (9.3)	100% 0.2ms (9.1)	100% 0.3ms (14.9)	100% 1.7ms (8.1)	40% $\infty$ ( $\infty$ )
		BIT*	82% 1.7ms (9.9)	95% 1.7ms (9.3)	98% 1.8ms (9.4)	99% 0.3ms (8.7)	100% 0.4ms (12.4)	100% 1.8ms (10.5)	40% $\infty$ ( $\infty$ )
		FCIT*	96% 0.7ms (7.2)	100% 0.5ms (7.3)	100% 0.6ms (7.3)	100% 0.3ms (6.5)	100% 0.5ms (10.4)	100% 0.8ms (7.8)	96% 201ms (14.9)
		AORRTC	100% 0.3ms (4.7)	100% 0.3ms (4.7)	100% 0.3ms (4.5)	100% 0.3ms (4.5)	100% 0.3ms (6.4)	100% 0.4ms (4.5)	100% 1.0ms (7.1)
		Fetch	OMPL	RRT-Connect	77% 798ms (29.1)	84% 569ms (29.4)	68% 2137ms (30.3)	99% 1.5ms (18.8)	100% 2.6ms (19.5)
RRT*-Connect	42% $\infty$ $\infty$			53% 7696ms 19.4	27% $\infty$ $\infty$	79% 3.7ms 17.0	79% 9.5ms 16.7	63% 13.8ms 23.5	68% 1952ms 24.7
AORRTC	79% 804ms (11.8)			84% 570ms (12.6)	68% 2139ms (12.3)	99% 3.1ms (9.2)	100% 4.1ms (9.0)	100% 4.3ms (11.5)	98% 214ms (17.0)
VAMP	RRT-Connect		100% 5.5ms (20.1)	100% 6.6ms (21.6)	100% 7.3ms (18.0)	100% 0.3ms (15.1)	100% 0.4ms (16.6)	100% 0.8ms (20.8)	100% 8.9ms (29.9)
	RRT*		36% $\infty$ ( $\infty$ )	36% $\infty$ ( $\infty$ )	31% $\infty$ ( $\infty$ )	99% 6.5ms (12.9)	99% 13.3ms (12.2)	100% 28.0ms (13.1)	7% $\infty$ ( $\infty$ )
	AORRTC		100% 6.4ms (11.0)	100% 7.1ms (11.8)	100% 8.1ms (10.3)	100% 0.8ms (8.8)	100% 0.8ms (9.1)	100% 1.3ms (11.0)	100% 10ms (14.3)

of notation, the cost of a given solution. The function  $\hat{c}: X \times X \rightarrow [0, \infty)$  is an admissible estimate of the edge cost, i.e.,  $\forall x_p, x_c \in X, \hat{c}(x_p, x_c) \leq c(x_p, x_c)$ . The function  $g_T: V_T \rightarrow [0, \infty)$  represents the cost-to-come through a tree,  $T$ , from the root to the given vertex,  $v$ . The function  $\hat{g}_T: X \rightarrow [0, \infty)$  represents an admissible estimate of this cost-to-come, i.e.,  $\forall x \in X, \hat{g}_T(x) \leq g_T(x)$ . The function  $\hat{h}_T: X \rightarrow [0, \infty)$  represents the estimated cost-to-go from the configuration,  $x$ , to the goal of the tree,  $T$ . Note that the goal of one tree,  $T_a$ , is the root of the other,  $T_b$ , i.e.,  $\forall x \in X, \hat{h}_{T_a}(x) \equiv \hat{g}_{T_b}(x)$ . Let  $X_{\hat{f}} \subseteq X_{\text{free}}$  be the informed set of free configurations, i.e.,  $\forall x \in X_{\hat{f}}, g_T(x) + \hat{h}_T(x) < c_{\max}$ .

Let  $w_x$  and  $w_c$  be the distance function weighting for configuration and cost distance, respectively. Let  $A$  and  $B$  be two sets. The notation  $\mathcal{U}(A)$  is shorthand for drawing a sample uniformly from a set,  $A$ . The notation  $A \stackrel{\pm}{\leftarrow} B$  is shorthand for the set compounding operation,  $A \leftarrow A \cup B$ .

### B. Augmented Search Space

AORRTC searches an augmented  $(n + 1)$ -dimensional search space where the first  $n$  dimensions are the configuration space and the  $(n + 1)$ <sup>th</sup> dimension is the cost to reach that configuration. New configurations are sampled uniformly from configuration space (Line 15). The upper bound on the cost of a new state,  $c_{\text{rand}}$ , is randomly sampled uniformly between the minimum and maximum possible cost of a solution passing through the state such that the state is feasible and useful, i.e.,  $\hat{g}(x) + \hat{h}(x) \leq c_{\text{rand}} < c_{\max}$  (Line 16). This cost is an upper bound for a connection to the configuration to be considered valid in the augmented space and the true cost of a configuration will be calculated from its parent configuration.

The nearest neighbour in the augmented search space is defined as the state closest in both configuration and cost

(Line 46) [17, 18]. This distance function means samples with high cost bounds are closest to vertices with high costs-to-come, and can inflate the cost of reaching the new configuration. To address this, the cost of a newly added vertex is resampled to search for lower cost neighbours after it is added to the tree (Lines 27 to 32). This resampling continues until an invalid edge or the same parent is found. The cost of a new vertex,  $v$ , in the tree is the cost-to-come through its parent vertex,  $v_p$ , i.e.,  $g_T(v) = g_T(v_p) + c(v_p, v)$ .

### C. Analysis

The AO- $x$  meta-algorithm has been proven by [18] to be a.s.a.o. when the cost function and robot dynamics are Lipschitz continuous and the planner used on the augmented search space is probabilistically complete. AORRTC is therefore a.s.a.o. if the version of RRT-Connect it uses is probabilistically complete. The original RRT-Connect is probabilistically complete [10] and the added cost resampling (Lines 27 to 32) in AORRTC does not change the vertices in the trees but only the cost of their connections. This maintains the probabilistic completeness of RRT-Connect and AORRTC is therefore probabilistically complete and a.s.a.o. with respect to the original planning problem.

### D. Anytime RRT-Connects

A naïve approach to making RRT-Connect an anytime a.s.a.o. planner would be to run it iteratively with informed sampling to try and find higher-quality solutions. This bidirectional extension of Anytime RRTs [35] converges to a reasonable solution on many problems but is not sufficient to probabilistically converge to optimal solutions and is provably *not* a.s.a.o. [9]. The performance of this *Anytime RRT-Connects* is presented in Fig. 3, even though it provides no formal optimality guarantees despite its practical performance.

## IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

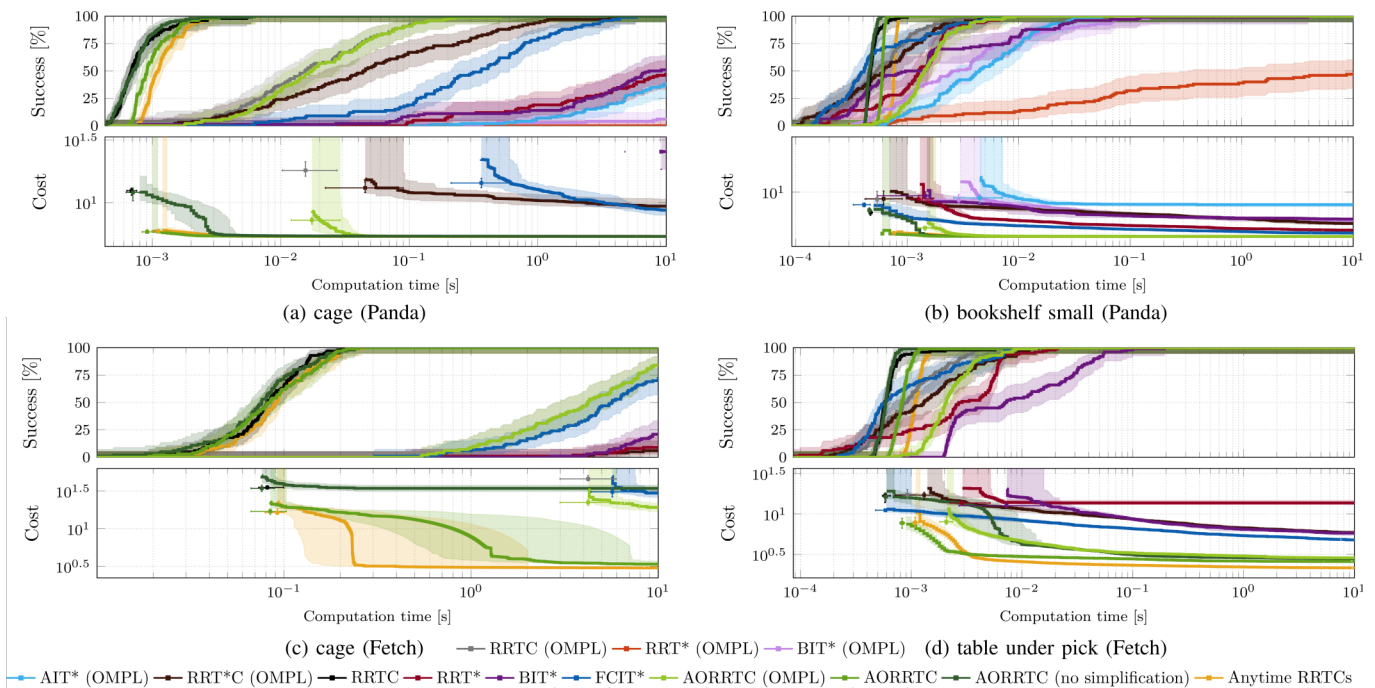


Figure 3: Convergence results for 100 trials of all planners on the 7 DoF Panda, and a subset of these planners on the 8 DoF Fetch, for a single problem from the *cage*, (a) and (c), *bookshelf small*, (b), and *table under pick*, (d), environments from the MotionBenchMaker [5] dataset (Sec. IV). The top plot of each subfigure shows the percentage of runs that found a solution by a given time with Clopper-Pearson 99% confidence intervals, and the bottom shows the median initial path length and median path length over time with nonparametric 99% confidence intervals [6].

### E. Implementation

Alg. 1 describes a conceptual version of AORRTC that leaves room for several improvements. AORRTC should sample configurations using direct informed sampling [24] when available or rejection sampling otherwise, and should use a balanced bidirectional search [36]. Nearest neighbour structures and lookups should be used where appropriate.

## IV. EXPERIMENTS

OMPL and VAMP implementations of AORRTC were evaluated against OMPL [19] implementations of RRT-Connect, RRT\*, BIT\*, AIT\*, and RRT\*-Connect and VAMP [20] implementations of RRT-Connect, RRT\*, BIT\*, and FCIT\*. We also show convergence results for a VAMP implementation of Anytime RRT-Connects (Sec. III-E) and a variation of VAMP AORRTC without simplification in Fig. 3. The reported solution costs and times for AORRTC include the computational costs of randomized shortcutting [11, 12] and B-spline smoothing [13]. The reported solution costs and times for the implementations of RRT-Connect do not include simplification but these can be inferred from the initial solution time and cost of AORRTC since this is a single simplified RRT-Connect search. Both RRT-Connect and AORRTC used a balanced bidirectional search [36]. The implementations of AORRTC and RRT-Connect uses an edge length of 2 (VAMP) and 2.6 (OMPL). All planners calculate their RGG constant with Theorem 38 from [9] when applicable. A value of 1.0 was used for both distance function weights,  $w_x$  and  $w_c$ .

The planners were tested on the MBM [5] dataset, which contains 7 planning environments for different robots where

each environment has 100 different planning problems<sup>2</sup>. These environments cover a range of manipulation planning scenarios, including reaching (*bookshelf tall*, *bookshelf small*, and *bookshelf thin*), constrained reaching (*box* and *cage*), and tabletop manipulation (*table pick* and *table under pick*).

The OMPL and VAMP planners were run on the 7 DoF Panda robotic arm MBM problems while only the VAMP planners and OMPL versions of AORRTC, RRT\*-Connect, and RRT-Connect were run on the harder 8 DoF Fetch Mobile Manipulator MBM problems. Most OMPL planners were not tested on the harder Fetch problems because they were not able to reliably find solutions in the available planning time. All tests were run in Ubuntu 22.04 on a Intel i7-9750H CPU with 32GB of RAM, and the planning algorithms are implemented in C++17. All planners use the default VAMP and OMPL samplers with different seeding for each trial. The OMPL planners use VAMP's collision checking backend. The planners were given 10 seconds to evaluate each problem. The time and cost axes for all figures are in logarithmic scale.

The success rate for all problems in a given environment are shown in Figs. 1 and 2. Fig. 3 shows the success rate and median solution cost over time for 100 trials of each planner on a single problem for the Panda and Fetch robotic arms.

Fig. 4 shows the success rate and median time to converge to a *near-optimal* solution for 5 trials of each planner on 100 problems from the *cage*, *bookshelf small*, and *table pick* environments for the Panda robotic arm. These results compare the number of problems where a solution was found that falls within a suboptimality factor,  $\epsilon$ , of an empirical estimate of

<sup>2</sup>Some of the problems are invalid for the Panda and Fetch robots resulting in 699 and 679 total problems, respectively.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

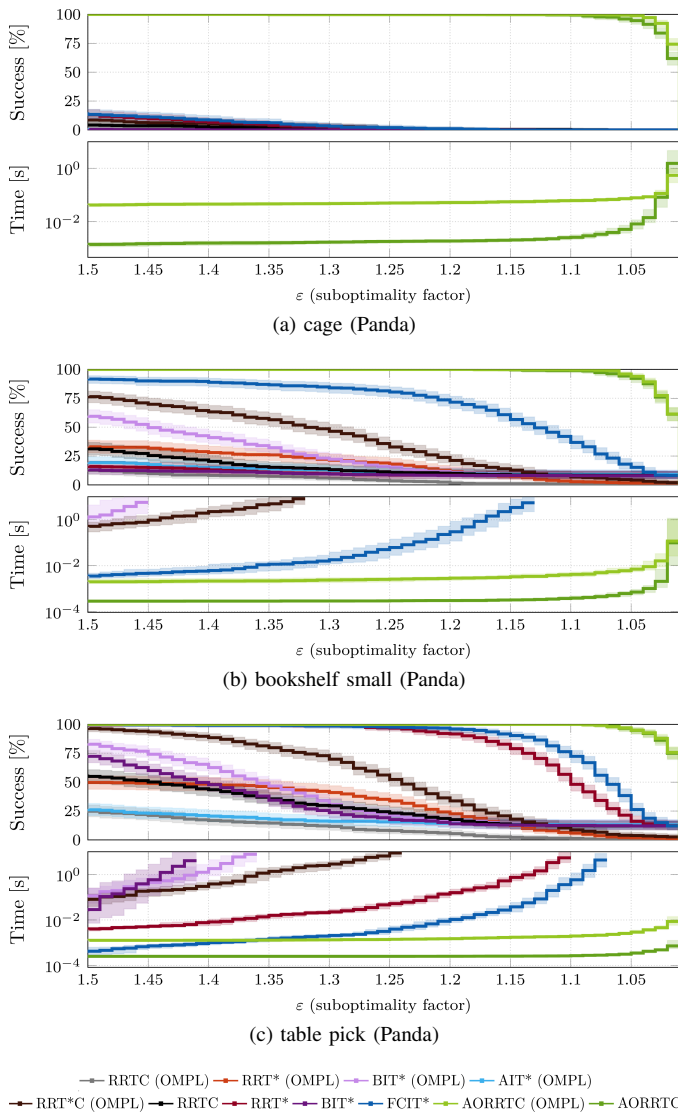


Figure 4: Near-optimality results for 5 trials of all planners on the 7 DoF Panda for all problems from the *cage*, (a), *bookshelf small*, (b), and *table pick*, (c), environments from the MotionBenchMaker [5] dataset (Sec. IV). The top plots show the percentage of runs that found a solution within a suboptimality factor,  $\epsilon$ , of the empirically determined optimum,  $c^*$ , with Clopper-Pearson 99% confidence intervals. The bottom plots show the median time to find a solution within a suboptimality factor,  $\epsilon$ , of the empirically determined optimum,  $c^*$ , with nonparametric 99% confidence intervals.

the optimum cost,  $\hat{c}^*$ , as well as the median time to find a solution that satisfies that bound. The empirical optimum for a given problem was taken as the minimum cost found across more than 700 trials during development and experiments.

## V. DISCUSSION

OMPL and VAMP AORRTC were the only tested a.s.a.o. planners that found solutions to all Panda problems. VAMP AORRTC was also the only tested a.s.a.o. planner that found solutions to all Fetch problems. The other tested a.s.a.o. planners struggled to find solutions on the difficult Fetch problems. VAMP RRT-Connect also found solutions to all tested problems, but is not a.s.a.o. Although OMPL AORRTC was not able to find solutions to all the Fetch problems in the allowed time, it found solutions to significantly more Fetch problems

and found initial solutions to these problems in less time than all tested planners other than RRT-Connect and VAMP AORRTC.

The VAMP implementation of AORRTC outperforms all other tested a.s.a.o. planners and finds initial solutions in significantly less time on all environments (Table I). None of the tested a.s.a.o. planners converge to higher quality solutions than those found by VAMP AORRTC (Fig. 3). OMPL AORRTC found initial solutions faster than most tested OMPL a.s.a.o. planners. Some VAMP a.s.a.o. planners, including VAMP AORRTC, were able to find solutions faster than OMPL AORRTC (Fig. 2). This is because of the performance improvements of VAMP’s SIMD-accelerated edge evaluation.

The only OMPL a.s.a.o. planner with similar initial solution performance to OMPL AORRTC was OMPL RRT\*-Connect. The VAMP and OMPL implementations of RRT-Connect also demonstrated similar initial solution performance to the VAMP and OMPL implementations of AORRTC, respectively. RRT-Connect and RRT\*-Connect found initial solutions microseconds faster than AORRTC. This is expected because the RRT-Connect and RRT\*-Connect results do not include the computational cost of path simplification while the initial result found by AORRTC is equivalent to a simplified solution found by RRT-Connect. This is supported by the variation of AORRTC with no simplification, which finds initial solutions as fast as RRT-Connect and of similar quality (Fig. 3). RRT-Connect cannot improve its initial solution given additional planning time. OMPL RRT\*-Connect converged slower and to significantly lower-quality solutions than OMPL AORRTC (Fig. 3).

Both OMPL and VAMP AORRTC converged to better solutions in significantly less time than all other tested planners (Fig. 4). AORRTC reliably converged to solutions that were close to the empirical optimum within milliseconds, even on the difficult *cage* problems, and converged to these solutions in less time than any other planner. The variation of VAMP AORRTC with no simplification also converged to better solutions in significantly less time than all other tested a.s.a.o. planners other than VAMP AORRTC on almost all problems (Fig. 3). This demonstrates that AORRTC’s convergence does not rely on simplification, but is accelerated by it.

## VI. CONCLUSION

AORRTC uses the ideas of the AO- $x$  meta-algorithm [17, 18] and RRT-Connect [10] to design a bidirectional anytime a.s.a.o. algorithm that quickly finds initial solutions and then converges towards the optimal solution.

AORRTC searches an  $(n+1)$ -dimensional augmented search space, where the first  $n$  dimensions are the configuration space and the  $(n+1)$ <sup>th</sup> dimension is its cost-to-come. It calls a satisficing planner on a sequence of these augmented search spaces with the cost dimension bounded by the current best solution cost to iteratively find higher-quality solutions. This has been proven to be a.s.a.o. [17, 18].

AORRTC finds initial solutions on the same order of magnitude as RRT-Connect, which is faster than almost all other a.s.a.o. planners. It then uses the remaining planning time to find higher quality solutions than all other tested a.s.a.o. algorithms in an anytime manner. This is demonstrated

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.**

with and without SIMD-acceleration on hundreds of problems across seven different planning environments for both the 7 DoF Panda and 8 DoF Fetch robotic arms.

AORRTC represents a different approach to a.s.a.o. planning. The majority of previous a.s.a.o. algorithms approximated the continuously valued search space with increasing accuracy in order to find better solutions with additional planning time and almost-surely asymptotically converge towards the optimum. AORRTC instead quickly samples from the set of feasible solutions that could provide a better solution than the current one. This avoids the computational cost of maintaining high-resolution approximations of the search space and allows AORRTC to find initial solutions quickly and then rapidly converge towards the optimum. AORRTC shows the promise of this alternative approach to a.s.a.o. planning and we expect that future work will explore its full implications.

Future work will explore the performance of different distance functions and apply AORRTC to optimize cost functions other than path length, including multivariate and non-smooth cost functions, as well as extending AORRTC to plan in non-Euclidean spaces, or in real-world dynamic environments. Information on the implementation of AORRTC is available at <https://robotic-esp.com/code/aorrtc/>.

## REFERENCES

- [1] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271.
- [2] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Trans. on Systems Science and Cybernetics* 4 (1968), pp. 100–107.
- [3] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. "Chomp: Covariant hamiltonian optimization for motion planning". In: *IJRR* 32.9-10 (2013), pp. 1164–1193.
- [4] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. "Motion planning with sequential convex optimization and convex collision checking". In: *IJRR* 33.9 (2014), pp. 1251–1270.
- [5] C. Chamzas, C. Quintero-Peña, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki. "MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets". In: *IEEE RA-L* 7.2 (2022), pp. 882–889.
- [6] 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, IEEE/RSJ IROS*. 2022.
- [7] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Trans. on Robot. and Automat.* 12.4 (1996), pp. 566–580.
- [8] S. LaValle. "Rapidly-exploring random trees : a new tool for path planning". In: *Research Report 9811* (1998).
- [9] S. Karaman and E. Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *IJRR* 30 (2011), pp. 846–894.
- [10] J. Kuffner and S. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings of the IEEE ICRA*. Vol. 2. 2000, 995–1001 vol.2.
- [13] J. Pan, L. Zhang, and D. Manocha. "Collision-free and smooth trajectory computation in cluttered environments". In: *IJRR* 31.10 (2012), pp. 1155–1175.
- [14] B. Raveh, A. Enosh, and D. Halperin. "A little more, a lot better: Improving path quality by a path-merging algorithm". In: *IEEE Trans. on Robot.* 27.2 (2011), pp. 365–371.
- [11] R. Geraerts and M. H. Overmars. "Creating High-quality Paths for Motion Planning". In: *IJRR* 26.8 (2007), pp. 845–863.
- [12] K. Hauser and V. Ng-Thow-Hing. "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts". In: *IEEE ICRA*. 2010, pp. 2493–2498.
- [15] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa. "Batch Informed Trees (BIT\*): Informed asymptotically optimal anytime search". In: *IJRR* 39 (2017), pp. 543–567.
- [16] M. P. Strub and J. D. Gammell. "Adaptively Informed Trees (AIT\*) and Effort Informed Trees (EIT\*): Asymmetric bidirectional sampling-based path planning". In: *IJRR* 41.4 (Apr. 2022), pp. 390–417.
- [17] K. Hauser and Y. Zhou. "Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space". In: *IEEE Trans. on Robot.* 32.6 (2016), pp. 1431–1443.
- [18] M. Kleinbort, E. Granados, K. Solovey, R. Bonalli, K. E. Bekris, and D. Halperin. "Refined analysis of asymptotically-optimal kinodynamic planning in the state-cost space". In: *ICRA*. IEEE. 2020, pp. 6344–6350.
- [19] I. A. Sucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.
- [20] W. Thomason, Z. Kingston, and L. E. Kavraki. "Motions in Microseconds via Vectorized Sampling-Based Planning". In: *Proceedings of the IEEE ICRA* (2024), pp. 8749–8756.
- [21] M. Kleinbort, O. Salzman, and D. Halperin. "Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning". In: *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer. 2020, pp. 624–639.
- [22] O. Arslan and P. Tsiotras. "Use of relaxation methods in sampling-based algorithms for optimal motion planning". In: *IEEE ICRA*. 2013, pp. 2421–2428.
- [23] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad. "RRT\*-SMART: A Rapid Convergence Implementation of RRT\*". In: *IJARS* 10.7 (2013), p. 299.
- [24] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa. "Informed Sampling for Asymptotically Optimal Path Planning". In: *IEEE Trans. on Robot.* 34.4 (2018), pp. 966–984.
- [25] 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\*)". In: *Proceedings of the IEEE ICRA*. 19–23 5 2025.
- [26] A. Orthey, C. Chamzas, and L. E. Kavraki. "Sampling-based motion planning: A comparative review". In: *Annual Review of Control, Robotics, and Autonomous Systems* 7 (2023).
- [27] B. Akgun and M. Stilman. "Sampling heuristics for optimal motion planning in high dimensions". In: *IEEE/RSJ IROS*. 2011, pp. 2640–2645.
- [28] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann. "RRT\*-Connect: Faster, asymptotically optimal motion planning". In: *IEEE ROBOTICS*. 2015, pp. 1670–1677.
- [29] A. Sintov and A. Shapiro. "Time-based RRT algorithm for rendezvous planning of two dynamic systems". In: *IEEE ICRA*. 2014, pp. 6745–6750.
- [30] J. van den Berg and M. Overmars. "Roadmap-based motion planning in dynamic environments". In: *IEEE Trans. on Robot.* 21.5 (2005), pp. 885–897.
- [31] D. Silver. "Cooperative Pathfinding". In: *Proceedings of the AAAI Conference on AIIDE* 1.1 (Sept. 2021), pp. 117–122.
- [32] M. Phillips and M. Likhachev. "SIPP: Safe interval path planning for dynamic environments". In: *IEEE ICRA*. 2011, pp. 5628–5635.
- [33] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint. "ST-RRT\*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time". In: *ICRA*. 2022, pp. 3314–3320.
- [34] N. Kerimov, A. Onegin, and K. Yakovlev. "Safe Interval Randomized Path Planning For Manipulators". In: *Submitted to The 35th ICAPS* (2024). arXiv:2412.19567.
- [35] D. Ferguson and A. Stentz. "Anytime RRTs". In: *IEEE/RSJ IROS*. IEEE. 2006, pp. 5369–5375.
- [36] J. Kuffner and S. LaValle. "An efficient approach to path planning using balanced bidirectional RRT search". In: *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep* (2005).