Homotopy-Aware Efficiently Adaptive State Lattices for Mobile Robot Motion Planning in Cluttered Environments*

Ashwin Satish Menon, Eric R. Damm, and Thomas M. Howard

***Abstract*—Mobile robot navigation architectures that employ a planning algorithm to provide a single optimal path to follow are flawed in the presence of unstructured, rapidly changing environments. As the environment updates, optimal plans often oscillate around discrete obstacles, which is problematic for path following controllers that are biased to follow the planned route. A potentially better approach involves the generation of multiple plans, each optimal within their own homotopy class, to provide a more comprehensive approximation of cost to goal for a path-following controller. In this paper, we present Homotopy-Aware Efficiently Adaptive State Lattices (HAEASL), which uses multiple open lists to bias search towards routes with distinct homotopy classes. Experiments are presented that measure the number, the optimality, and the diversity of solutions generated across 3,200 planning problems in 80 randomly generated environments. The performance of HAEASL is benchmarked against two previous approaches: Search-Based Path Planning with Homotopy Class Constraints (A*HC) and Homotopy-Aware RRT* (HARRT*). Experimental results demonstrate that HAEASL can generate a greater number of paths and more diverse paths than A*HC without a significant reduction of optimality. Additionally, results demonstrate that HAEASL generates a greater number of paths and ones with lower costs than HARRT*. A final demonstration of HAEASL generating multiple solutions subject to temporal, resource, and kinodynamic constraints using data collected from an off-road mobile robot illustrates the suitability of the approach for the motivating example.**

***Index Terms*—Motion and path planning, field robots**

I. Introduction

MOTION planning algorithms for mobile robots that search for the optimal route given an environment and motion model are applied across various domains [5] [10] [11]. In the context of off-road navigation, where perceptual information is dynamically updated, global plans must be robust to the presence of newly appearing obstacles and hazards. Often, search for a minimum-cost plan may result in a route that is homotopically distinct from the prior solution but negligibly lower in cost. A potentially better approach involves generating multiple solutions, each optimal within their homotopy class, so that solutions can be more effectively compared with prior selections to determine if the guidance to a path-following controller [7] [18] should be updated.

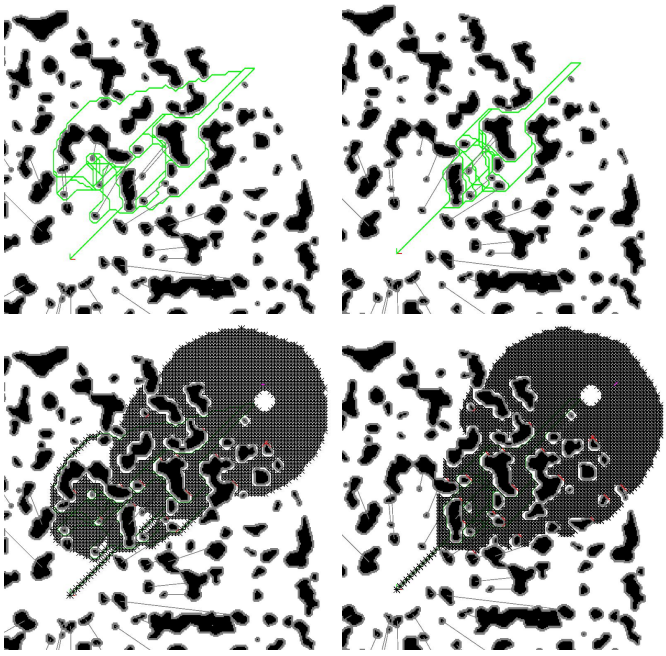

Fig. 1: Top row: Paths generated from start to goal for HAEASL (left) and A* with Homotopy Constraints (A*HC) (right) through an obstacle field. Bottom row: Paths sampled by HAEASL (left) and A*HC (right). In this example, HAEASL generates a larger number of and more diverse solutions than A*HC.

In the same way as [13], we adopt a definition of homotopy as a formalism to compare the similarity of paths. Given two paths that start and end at the same states, the paths belong to the same homotopy class if one can be continuously deformed into the other without encroaching on any obstacle. This paper introduces Homotopy-Aware Efficiently Adaptive State Lattices (HAEASL), which, unlike previous approaches, uses multiple open lists to bias exploration toward multiple solutions that are optimal within their own homotopy classes. Figure 1 illustrates solutions generated and the search space explored by HAEASL, and a search algorithm based

Manuscript received: July 15, 2024; Revised: October 14, 2024; Accepted: November 26, 2024.

This paper was recommended for publication by Editor Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments.

*Research was sponsored by the Defense Advanced Research Projects Agency (DARPA), and was accomplished under Contract Number HR001121C0189. Any opinions, findings, conclusions or recommendations expressed herein are those of the author(s) and do not reflect the views of the Defense Advanced Research Projects Agency or Carnegie Mellon University. Research was also sponsored by the DEVCOM Army Research Laboratory (ARL) and was accomplished under Cooperative Agreement Number W911NF-20-2-0106. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Ashwin Satish Menon, Eric R. Damm, and Thomas M. Howard are with the Robotics and Artificial Intelligence Laboratory, Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA amenon4@ur.rochester.edu

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

on Search-Based Path Planning with Homotopy Class Constraints, which we refer to as A*HC. Here, HAEASL generates more solutions and solutions that exhibit a greater degree of path diversity than A*HC, where path diversity is measured as a function of deviation from the optimal path generated by the search algorithm. This metric quantifies the difference between trivial perturbations of the optimal path versus truly distinct routes.

HAEASL is built upon Efficiently Adaptive State Lattices (EASL) [5], a recombinant search space that uses heuristic search to generate optimal paths for mobile robot navigation in cluttered environments. HAEASL is also compatible with Kinodynamic Efficiently Adaptive State Lattices (KEASL) [2], which extend the representation of the search space to satisfy position and orientation dependent velocity constraints to generate trajectories between boundary states. Although these approaches improve the efficiency, fidelity, and optimality of heuristic search for mobile robots in cluttered environments, they are not formulated to exploit homotopy constraints or return multiple distinct solutions. This paper presents an approach that can generate multiple homotopically distinct trajectories in cluttered environments while also efficiently searching under temporal, resource, and kinodynamic constraints.

II. Related Work

Mobile robot motion planning algorithms are generally classified into two methods: probabilistic sampling [12] [10] [11] and deterministic graphs [16]. The stochastic nature of sampling-based approaches can lead to inefficiencies in how the state space is explored, resulting in multiple different routes to roughly the same state. Recombinant graphs make use of control sets which recombine at fixed intervals in the state space. Robot states are represented by nodes in the graph. More recent recombinant search spaces such as the Adaptive State Lattice [6], EASL [5], and KEASL [2], locally optimize the location of sampled nodes in the graph using approximations of the aggregate costs of local edges to improve the relative optimality of heuristic search. EASL specifically discretizes the state space adaptations considered, thus limiting the set of feasible motions which could arise when search is performed. By doing this, the online trajectory generation process [8], which can be expensive if applied extensively in large search spaces, is eliminated. This enables the precomputation of swaths for edge cost evaluation, resulting in paths that are of comparable cost to the ASL solutions, but are found in less time. Correspondingly, the EASL planner returns paths which have lower cost than a standard state lattice-based planner yet with similar computation time. In this contribution, we make use of the EASL algorithm but apply it to solving multiple planning problems, each one in a different homotopy class.

Other representations for robot motion planning have explored the use of homotopic constraints. The authors of [1] introduce the *L*-augmented graph, which is a representation based on [16] that enables any search algorithm paired with it to maintain awareness of the homotopy class in which the robot's planned path exists. In addition to the generation of paths that satisfy homotopy constraints, this algorithm can generate multiple homotopically distinct paths by letting heuristic search continue past the generation of the initial path. The search algorithm described in [1], which we refer to as A* with Homotopy Constraints (A*HC), maintains a single open list and performs node expansion by the nodes' associated f-costs [4]. Contrarily, HAEASL uses multiple open lists that are added to the search space as it encounters new homotopy classes.

The algorithm proposed in [19] leverages geometry in tandem with homotopy and demonstrated significant reductions in planning time when compared against [1]. However, [19] functions only when the shortest path is the optimal one. When optimality also incorporates velocity information, like in Figure 9 in Section V, the method in [19] is infeasible.

In [20] multiple paths are generated using an RRT*-based graph by tracking the homotopy class of edges as the state space is explored. This homotopic class-tracking enables a bidirectional RRT* algorithm to pair with a heuristic that can restrict search to given homotopy classes in the search space. The authors of HARRT* [20] based their homotopic equivalence-checking algorithm on Jenkins' approach [9]. This approach partitions the map into disjoint regions, and the lines which do the partitioning are known as reference frames. The reference frames originate from the map's closest point to center which does not exist in an obstacle.

We implemented the search methods described in [1] and [20] as the two benchmarks for evaluating our contribution, as further discussed in Section V. We use the homotopic equivalence-checking algorithm described in [20] and [9] for checking the homotopy class of all approaches. The contributions of this paper are as follows:

- HAEASL, the first state lattice-based search algorithm that uses multiple open lists to generate multiple solutions in distinct homotopy classes.
- An evaluation of HAEASL, A*HC [1], and HARRT* [20], that measured the number of solutions, the optimality of solutions, and average path diversity.
- Demonstration of HAEASL generating multiple paths for mobile robot navigation using data collected from a physical robot in an off-road environment subject to resource, temporal, and kinodynamic constraints.

III. Technical Approach

Rather than the *h-signature* approach used in [1], HAEASL makes use of the mathematical formalisms presented in [20] that describe how homotopy constraints are defined. This involves segmenting the robot's environment by drawing lines in obstacle maps which are anchored by a single center point and each obstacle's centroid. The lines are treated as separate "reference frames". This process is illustrated in Figures 2 through 5 and explained here.

First, the obstacle map is parsed to find the location of obstacles. If an obstacle if found to exist at a cell, the 8-connected neighboring cells will be recursively searched and combined into the same obstacle cluster until no neighboring

cells are obstacles. The clusters are stored in a vector of obstacle clusters, where the cluster's centroid serves as the anchor point for the reference frame. Figure 2 illustrates how reference frames are then drawn with the two points being obstacle centroid (the yellow dot within the red square obstacles) and the map's sampled center point (the pink dot). We include only sampled lines which are on the far side of the obstacle relative to the center of the map, which reduces the number of reference frame crossings to track. If the frame encounters a new obstacle on its way to the map boundaries, the frame ends. Each frame has a unique string identifier for identifying which reference frames a path may have crossed. For algorithmic purposes, it is referred to as the *homotopy class*. If a path crosses reference frame α and then reference frame β as shown in Figure 2, that path will have $\alpha\beta$ as its associated *homotopy class*.

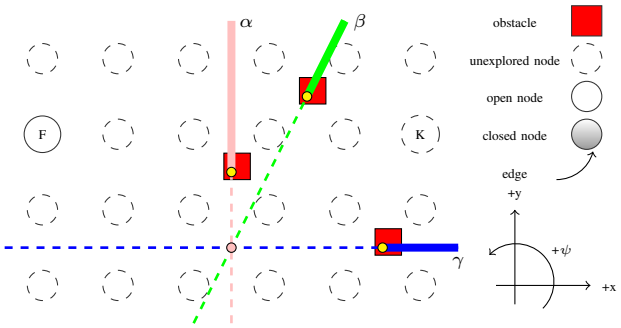

Fig. 2: HAEASL follows the procedure in [20] by sampling points in the obstacle and generating lines from the center point of the map that do not result in coincident lines.

Figure 3 shows the first three expansions of the search process, which is based on A* search [4]. The path begins with no reference frame crossings, so we describe the path as belonging to the homotopy class $\varnothing$. As search expands from $(H, 0°)$ to $(I, 0°)$ the reference frame α is crossed, meaning that the path from $(F, 0°)$ to $(I, 0°)$ now contains edges that belong to two different homotopy classes, $\varnothing$ and α. Note that the second value in each of these node pairs refers to the node's heading. To ensure that we continue searching along paths that go to the left and right of the first encountered obstacle, we add a second priority queue that only includes nodes belonging to the α homotopy class.

Figure 4 shows two expansions for each of the two priority queues defined by homotopy classes $\varnothing$ and α. A solution is found from $(F, 0°)$ to $(K, 0°)$ in homotopy class α. Search continues in the other priority queues to explore alternative routes.

Figure 5 shows a second route belonging to homotopy class $\varnothing$ after two additional expansions of that priority queue. HAEASL continues until no more priority queues are capable of expanding or a maximum time of search is reached.

The search process illustrated in Figures 2 through 5 is formalized in Algorithm 1. The start node is added to the first open list, and consequently, this open list with associated homotopy class $\varnothing$ is added to $\mathcal{O}$, a map of open lists indexed by the homotopy class. We term the *satisfied* list, initialized

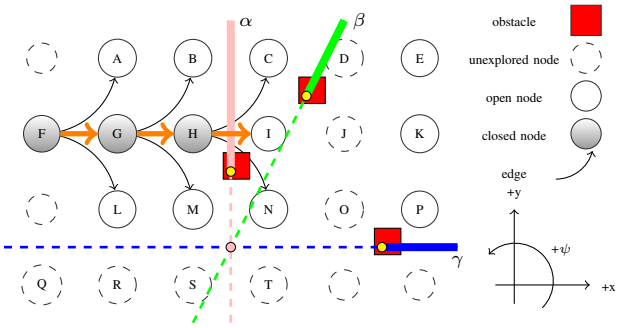

Fig. 3: HAEASL search begins by expanding nodes from $(F, 0°)$ towards node $(K, 0°)$ through $(G, 0°)$, $(H, 0°)$, and $(I, 0°)$. The edge from $(H, 0°)$ to $(I, 0°)$ crosses reference frame α, meaning a new open list is associated with homotopy class α, whereas the previously expanded nodes belonged to an open list associated with homotopy class $\varnothing$.

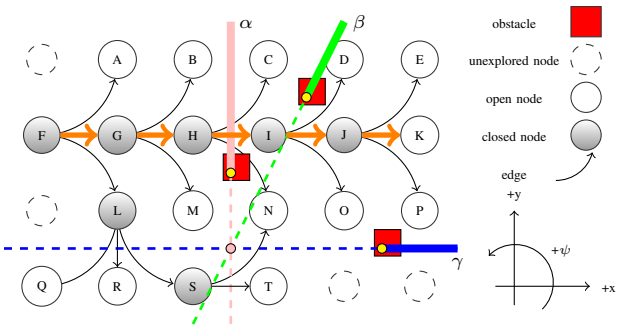

Fig. 4: HAEASL search continues in two directions by expanding two nodes in each homotopy class ($(L, 270°)$, and $(S, 0°)$ in $\varnothing$ and $(I, 0°)$, and $(J, 0°)$ in α) and finds a route from $(F, 0°)$ to node $(K, 0°)$ through $(G, 0°)$, $(H, 0°)$, $(I, 0°)$, and $(J, 0°)$. Since this route crossed the α boundary, we describe this solution as belonging to homotopy class α.

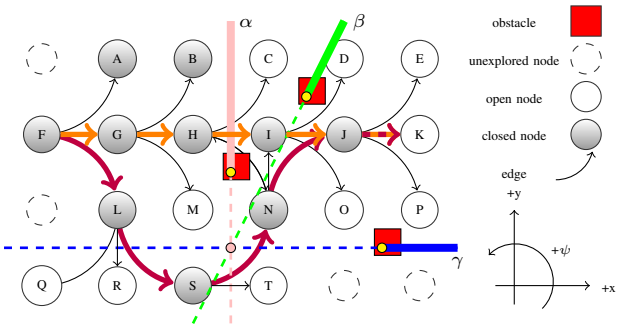

Fig. 5: HAEASL search continues by expanding nodes belonging to homotopy classes $\varnothing$ and α. After two additional expansions, a second route is found from $(F, 0°)$ to node $(K, 0°)$ through $(L, 270°)$, $(S, 0°)$, $(N, 90°)$, and $(J, 0°)$. Since this solution crossed no reference frames, we describe this solution as belonging to homotopy class $\varnothing$.

as S on line 4 of Algorithm 1, as the one which holds the first node that satisfies the goal constraints from each open list. We populate a vector of reference frames by calling the INITIALIZEREFERENCEFRAMES function. These frames are created using the same methodology specified in [20]. Each open list is iterated through to add the current node to the closed list, C (line 3). If the current node is the goal node, then it is added to the satisfied list. Inspired by anytime search techniques such as Anytime Repairing A* [14], the EXPAND function is repeatedly called until the search time parameter t_{max} is exhausted or every open list is processed. Finally, the satisfied list is iterated through to reconstruct each path HAEASL was able to find, through the BACKTRACK function call. The paths are returned, each one belonging to a separate homotopy class.

Algorithm 2 details the homotopy-aware node expansion process. Given a current node and the edges emanating from that node in the state lattice control set, the new node's x and y values are stored. The process for locally adapting nodes from [5] may also be interleaved with the step at line 3. If the new node belongs to the closed list, the procedure continues. The $path$ of the new node is formed from the edge connecting the current and new node, where $e.n_f$ represents the node offset. The g cost of the new node adds the connecting edge cost to the current node's g cost, while the f cost [4] adds the heuristic. $n_n.hc$ represents the homotopy class identifier of the new node, which is updated from the current node's hc if and only if the CROSSINGS function returns a new line crossing between the new node's path and a frame in the reference frames vector. $n_n.hc$ is $\varnothing$ if no frames are crossed. From here, if there exists an open list which has a node with the same hc as that of the new node, the new node is added to that existing open list. If not, the new node is added to a newly instantiated open list, O_n. If there exists a node, n_o, which has the same indices as the new node, the node with the lower f cost stays on the open list. If such an open list with the same hc as the new node does not exist, then a new open list is created and inserted into the map of open lists, termed $\mathcal{O}$.

Algorithm 1: HOMOTOPY-AWARE EFFICIENTLY ADAPTIVE STATE LATTICE SEARCH

Input: Start node $\mathbf{n}_s$, goal node $\mathbf{n}_g$, state lattice $\mathbf{L}$, obstacle map $\mathbf{M}$, maximum search time t_{max}
Output: Paths $\mathcal{P}$

1 $O \leftarrow \{\mathbf{n}_s\}$
2 $\mathcal{O}$.INSERT$(\varnothing, O)$
3 $C \leftarrow \emptyset$
4 $S \leftarrow \emptyset$
5 $\mathbf{RF} \leftarrow$ INITIALIZEREFERENCEFRAMES$(\mathbf{M})$
6 **while** $t_{curr} < t_{max}$ **do**
7 **for** $O \in \mathcal{O}$ **do**
8 **if** $O \neq \emptyset$ **then**
9 $\mathbf{n}_{curr} \leftarrow O$.POP$()$
10 C.PUSH$(\mathbf{n}_{curr})$
11 **if** $\mathbf{n}_{curr} = \mathbf{n}_g$ **then**
12 S.PUSH$(\mathbf{n}_{curr})$
13 EXPAND$(\mathbf{n}_{curr}, \mathbf{n}_g, \mathcal{O}, C, \mathbf{L}, \mathbf{M}, \mathbf{RF})$
14 $\mathcal{P} \leftarrow \emptyset$
15 **for** $s \in S$ **do**
16 P.PUSH(BACKTRACK(s))
17 **return** $\mathcal{P}$

Algorithm 2: HOMOTOPY-AWARE NODE EXPANSION

Input: Current node $\mathbf{n}_c$, goal node $\mathbf{n}_g$, open lists $\mathcal{O}$, closed list C, state lattice $\mathbf{L}$, obstacle map $\mathbf{M}$, reference frames $\mathbf{RF}$

1 **Function** EXPAND($\mathbf{n}_c$, $\mathbf{n}_g$, $\mathcal{O}$, C, $\mathbf{L}$, $\mathbf{M}$, $\mathbf{RF}$):
2 **for** $\mathbf{e} \in \mathbf{L}$.CONTROLSET$(\mathbf{n}_c)$ **do**
3 $\mathbf{n}_n \leftarrow$NODE$(\mathbf{n}_c[x] + \mathbf{e}.\mathbf{n}_f[x], \mathbf{n}_c[y] + \mathbf{e}.\mathbf{n}_f[y])$
4 **if** $\mathbf{n}_n \in C$ **then**
5 **continue**
6 $\mathbf{n}_n.path \leftarrow \mathbf{L}$.CONNECT$(\mathbf{n}_c, \mathbf{n}_n)$
7 $\mathbf{n}_n.g \leftarrow \mathbf{n}_c.g +$ COST$(\mathbf{n}_n.path, \mathbf{M})$
8 $\mathbf{n}_n.f \leftarrow \mathbf{n}_n.g +$ HEURISTIC$(\mathbf{n}_n, \mathbf{n}_g)$
9 $\mathbf{n}_n.hc \leftarrow \mathbf{n}_c.hc +$ CROSSINGS$(\mathbf{n}_n.path, \mathbf{RF})$
10 **if** $\mathcal{O}$.FIND$(\mathbf{n}_n.hc)$ **then**
11 $O \leftarrow \mathcal{O}[\mathbf{n}_n.hc]$
12 **if** $\exists \mathbf{n}_o \in O.$[EQUALINDICES$(\mathbf{n}_o, \mathbf{n}_n)$] **then**
13 **if** $\mathbf{n}_n.f < \mathbf{n}_o.f$ **then**
14 O.REPLACE$(\mathbf{n}_o, \mathbf{n}_n)$
15 **else**
16 O.PUSH$(\mathbf{n}_n)$
17 **else**
18 $O_n \leftarrow \{\mathbf{n}_n\}$
19 $\mathcal{O}$.INSERT$(\mathbf{n}_n.hc, O_n)$

IV. EXPERIMENTAL DESIGN

The performance of HAEASL was studied through several simulation-based experiments and its performance was benchmarked against our own implementations of A*HC [1] and HARRT* [20]. These custom implementations enabled us to ensure that obstacle checking was performed in a consistent manner and any optimization of how open lists are sorted was consistent for HAEASL and A*HC. The first experiment investigated the number of paths HAEASL, A*HC, and HARRT* generated as a function of the reference frame radius (RFR). The RFR casts out a circle from the robot's initial state in which all obstacles within the given radius have an associated reference frame that is drawn in the cost map. If an obstacle is outside this radius, it does not have a reference frame. The three algorithms planned motions in 80 randomly generated obstacle maps, each with a fixed start state and eight goal states sampled at a distance of 50 meters, for a total of 640 planning problems. To simulate

natural environments, Perlin noise was used to generate the obstacle maps [15]. The maps were varied to encompass a range of obstacle distributions by modulating the frequency of the noise, simulating multiple noise octaves, and varying the minimum noise value to be considered an obstacle in the cost map representation. Obstacles less than five meters or more than 45 meters from the start state were removed from these maps to ensure that sampled planning problems would not be infeasible because of invalid start and goal states. Each of the sampled boundary states used a different initial heading and a goal position that was 50 meters offset in the direction of the starting state's heading. This experiment was performed five times: at 10, 20, 30, 40, and 50-meter reference frame radii. Thus, the total number of unique planning problems amounted to 3,200. Since the goal states were always set to 50-meters away from the robot's initial state at varying headings, the 50 meter reference frame radius indicated that every obstacle in the map had an associated reference frame. In each of the 3,200 unique planning problems the runtime was fixed at 2.0 seconds. Experiments also explored the average minimum solution cost and the average and maximum path diversity of HAEASL, A*HC, and HARRT*, also a function RFR. In these experiments path cost is described as the length of the generated path. We compute the average path cost by averaging only those planning problems that return at least one solution. We record this as $|Solutions| > 0$ in Table I. Path diversity was measured as modified Hausdorff distance (MHD) [3], inspired by its usage in a previous robot navigation paper [17]:

$$h(A, B) = \frac{1}{|A|} \sum_{a_i \in A} (\min_{b_j \in B} d(a_i, b_j)), \tag{1}$$

where A is the lowest-cost path for each planning problem and B is every other path that each algorithm returned for that given problem. Lower MHD indicates less diversity of solutions. For each of the three algorithms across each planning problem, the average MHD and maximum MHD were tracked. The MHD calculated the minimum distance between the reference path and current path on a state-by-state basis and then normalized by the number of states in the reference path. For each planning problem, the MHD for each solution was summed and divided by the number of solutions to yield the average MHD. The maximum MHD is simply the highest MHD returned per planning problem. Computing MHD requires at least two solutions, so the number of planning problems where more than one solution is found is recorded as $|Solutions| > 1$ in Table II.

For all experiments, the maps had a square size of 512 by 512 cells at a 0.2-meter resolution. Thus, the maps represented 102.4-meter by 102.4-meter square grids. Obstacles were only drawn within the circle inscribed in the grid. All simulated experiments were implemented using C++ on an Intel Xeon 16-core CPU with a clock speed of 2.6GHz and 64GB of RAM. In some additional experiments, we varied the maximum search time and distance to the goal state to explore the influence on the number of solutions generated. In all of these experiments, we used an eight-connected control set sampled at a 0.6m resolution for HAEASL and A*HC with a heuristic inflation factor of 2.0. Finally, HAEASL was tested on a real-world, off-road environment while held to a search time constraint of one second. It was integrated with a kinodynamic search algorithm [2] on a Clearpath Warthog UGV. The robot's autonomy system was driven by two Neousys Nuvo-7166GC computing platforms running i7-8700T CPUs and NVIDIA Tesla T4 GPUs. This experiment was done to validate HAEASL's performance on a real robot and was not compared to any baseline.

V. Results

Figure 6 shows comparisons of HAEASL and the two benchmarks we tested against, A*HC and HARRT*, across three metrics. The left plot represents the first metric: the number of solutions that were generated across the five reference frame radii (10m, 20m, 30m, 40m, and 50m). The maximum search runtime permitted for all three algorithms was 2.0 seconds per planning problem. Points that are plotted underneath the black dotted line (slope = 1) with 'x' markers represent planning problems in which HAEASL generated more solutions than A*HC. Points plotted above this dotted line represent the opposite. This also applies for the number of solutions from HAEASL vs HARRT*, but with '+' markers. The average number of solutions generated across the three algorithms are shown in Table I. As seen in this table, our approach initially generated more solutions than A*HC in the most planning problems. However, as the RFR value increased, the A*HC surpassed HAEASL.

The reason for why A*HC starts to overtake HAEASL in the number of solutions generated is due to the different open list maintenance methods that each algorithm uses. As mentioned in Section II, A*HC maintains a single open list and performs node expansions in order of the nodes' associated f-costs [4]. HAEASL creates a new open list every time a new reference frame is crossed, indicating that the path has entered a new homotopy class. In this way, HAEASL forces greater exploration of the graph at the cost of the compute time that is needed to process many open lists. As the RFR increases, so too does the number of reference frames themselves, resulting in more open lists for HAEASL to process, thus reducing the speed at which it can find solutions. The original aim of this contribution was to provide an algorithm which reduces the oscillatory behavior of global plan guidance for path-following controllers. Therefore, we believe that homotopy class variance closer to the robot is more critical for field performance than variance farther away. For unmanned ground vehicles such as the Clearpath Warthog, options across different homotopy classes that are offered to the UGV's controller 50 meters from the robot aren't particularly useful, but such options within 10 meters of the robot are very pertinent for the controller to use.

For HAEASL vs HARRT*, most planning problems across all five RFR values yielded a larger number of solutions with HAEASL than HARRT*, as indicated by most of the data points marked with '+' being below the dividing line. This is further reinforced by the average number of HAEASL solutions in Table I being higher than the average number of HARRT* solutions at every RFR value.

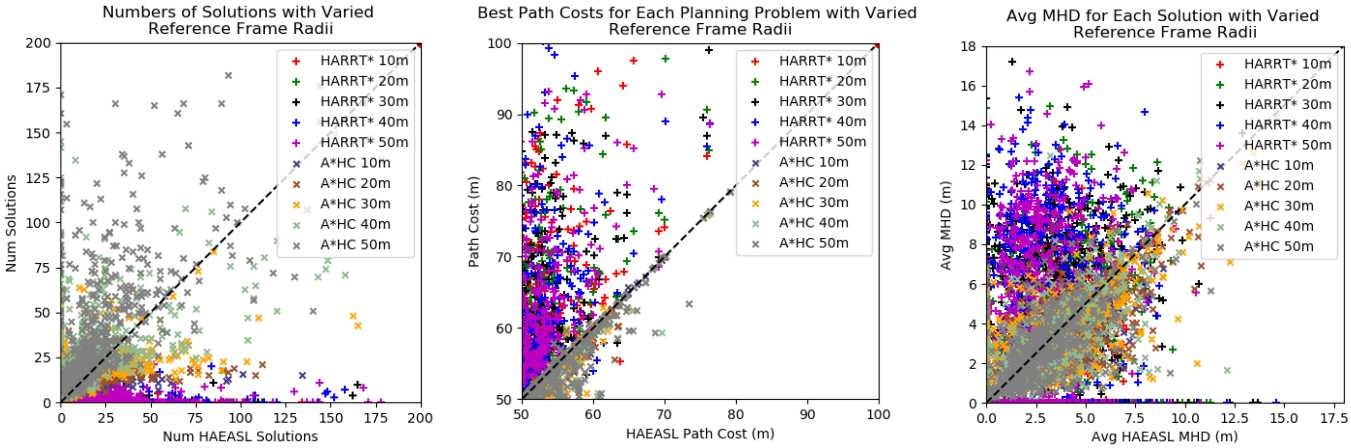

Fig. 6: Points marked with an 'x' represent HAEASL/A*HC comparisons while those marked with '+' represent HAEASL/HARRT* comparisons. Left: A scatterplot of the number of solutions, across all planning problems. Points under the line indicate more solutions from HAEASL, while points above the line indicate more solutions from A*HC or HARRT*. Middle: A scatterplot of the path costs of the best solutions, across all planning problems which returned at least one solution. Points above the line indicate better HAEASL performance, while points below the line indicate better A*HC or HARRT* performance. Right: A scatterplot of the average modified Hausdorff distance, showing path diversity. Every solution that belonged to a planning problem where each algorithm returned at least two solutions is displayed. Points under the line indicate greater path diversity from HAEASL; points above the line indicate greater path diversity from A*HC or HARRT*.

TABLE I: Comparison of HAEASL, A*HC, and HARRT* Cost Metrics

Reference Frame Radius (m)	Avg Number Solutions			$\|\text{Solutions}\| > 0$			Avg Minimum Solution Cost (m)		
	HAEASL	A*HC	HARRT*	HAEASL	A*HC	HARRT*	HAEASL	A*HC	HARRT*
10	7.9844	5.4469	1.3641	610	613	475	52.4192	52.4125	61.5607
20	10.9875	9.6594	2.4750	596	612	476	52.3679	52.2643	61.2541
30	14.0672	14.9297	3.2516	588	611	473	52.3947	52.1851	61.0065
40	16.9281	24.6844	3.6438	585	611	472	52.5933	52.1884	61.2135
50	20.8391	37.1219	3.5813	586	611	478	52.6737	52.1907	61.6359

TABLE II: Comparison of HAEASL, A*HC, and HARRT* Hausdorff Distance Metrics

Reference Frame Radius (m)	$\|\text{Solutions}\| > 1$			Avg Modified Hausdorff Distance (m)			Max Modified Hausdorff Distance (m)		
	HAEASL	A*HC	HARRT*	HAEASL	A*HC	HARRT*	HAEASL	A*HC	HARRT*
10	479	508	230	3.5689	3.3163	5.0356	7.0456	6.2509	10.3062
20	542	604	368	4.1075	3.6777	6.0885	7.7639	6.8846	12.1963
30	550	611	421	3.7780	3.4241	6.4794	6.9537	6.5249	12.9463
40	568	611	423	3.3004	3.0515	6.9649	6.1947	5.8744	13.7130
50	581	611	424	2.9157	2.6932	6.9062	5.4207	5.1911	13.5391

The next experiment compared the average path cost ratios of each algorithm's lowest-cost solution it returned. While the number of solutions that a homotopy-aware search algorithm returns is important, the best-performing solutions should also be relatively optimal to present the robot with efficient guidance. This metric was recorded to understand how those best-performing solutions compared against each other. Table I also reports on the average path costs of the optimal solutions returned across the 640 planning problems at each RFR. If an algorithm failed to return at least one solution, the associated planning problem was not included in the calculation. The average costs indicate that in the worst case, with a 50m RFR, the optimal solutions of HAEASL had 0.900% higher f-costs than those of A*HC. In the best case, at an RFR of 10m, HAEASL's optimal solutions had 0.013% higher f-costs. Given the size of the map in which these experiments were performed, this range of 0.01% to 0.9% equates to roughly a 0.007m to 0.450m difference in path distance.

We also report on the average path costs of HARRT*'s optimal solutions. HARRT*'s optimal solution costs were approximately 17% higher than HAEASL's optimal solution costs. This percentage difference is about a 8.5m difference in path distance – far more significant than the difference between HAEASL's and A*HC's optimal paths.

Table II shows the average path diversity across all valid planning problems when comparing HAEASL, A*HC, and HARRT*. Unlike Table I, for a planning problem to be considered valid for this metric, each algorithm had to return at least two solutions, so that the optimal solution could be compared against at least one other solution to generate the modified Hausdorff distance (MHD) value, as shown in Equation 1. Out of the three approaches, HARRT* yields the highest path diversity. This is due to the random trees expanding independently from the position of the goal. However, when

comparing the two lattice-based approaches, HAEASL outperforms A*HC by yielding a 9.16% average increase in path diversity. This was an expected result due to the differences in how our approach versus A*HC explores the graph. Since HAEASL creates a new open list every time an edge crosses a previously unencountered reference frame, its node expansion process encourages larger amounts of exploration by design. Contrasted with A*HC, which maintains a single open list and performs node expansions in the order of their f-costs, our approach tended to generate more diverse solutions.

Figure 7 illustrates a comparison of HAEASL, A*HC and HARRT*, whose solutions are shown in green, blue, and red, respectively. The left image shows the resulting paths when each algorithm is given two seconds to plan, and the right image corresponds to one second. In the two second case, HARRT* only generates one solution, but is unable to generate any solutions when given one second. With two seconds to plan, A*HC is able to generate solutions that plan around both sides of the obstacle marked with a pink star. Once the search runtime drops to one second, the routes to the goal that diverge to the right of the marked obstacle disappear. However, HAEASL still generates solutions in these homotopy classes, in addition to providing a route that diverges to the far left in the obstacle field. This highlights how prioritizing early exploration of the routes with distinct homotopy classes can result in greater diversity of plans, especially when obstacles are closer to the initial state.

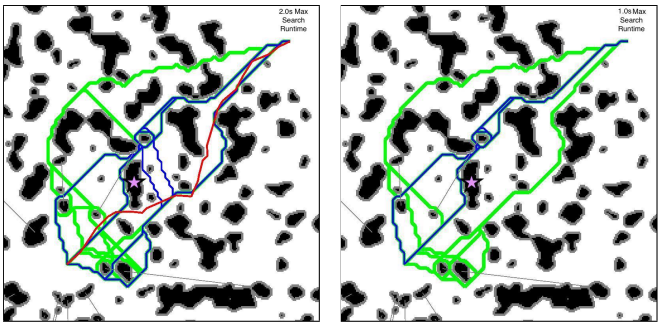

Fig. 7: Search with 2 seconds (left) and 1 second (right). Green is HAEASL, blue is A*HC, and red is HARRT*. As the maximum search runtime is reduced, HAEASL's number of solutions decreases from 20 to 12, A*HC's decreases from 10 to 6, and HARRT*'s decreases from 1 to 0.

Figure 8 illustrates the solutions and the search spaces of a unit test where we tasked both HAEASL and A*HC with finding multiple solutions to a goal state 100 meters from the start state. This was done to simulate the effect of a mobile robot navigating to the goal with a limited perceptual horizon. Both algorithms were given one second to generate plans, resulting in 13 from HAEASL and 2 from A*HC. The reason our approach was able to find so many more solutions in distinct homotopy classes is due its handling of multiple open lists. Early in the search process, edges cross a reference frame, which then creates a new open list for HAEASL to process. The creation of this new open list prompts node exploration in the area around that reference frame. This point is further highlighted by the strips of black expansions in the bottom left image in Figure 8. Once those reference frames are crossed, HAEASL spends time searching in those areas. Compared to the image on the bottom right, A*HC spends most of its time performing node expansions in free space after finding the first solution, as illustrated by the large ellipse beyond the obstacle region. This was confirmed by A*HC generating its first solution in 0.0125 seconds, but only generating the second solution in 0.9839 seconds, which was nearly the maximum allowed search runtime. In contrast, HAEASL generated three of the 13 solutions in 0.1376 seconds.

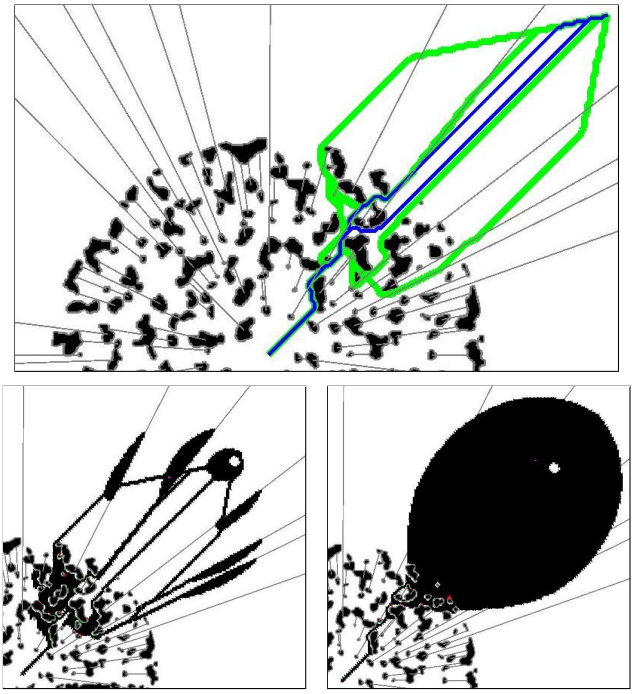

Fig. 8: Top: 13 solutions generated by HAEASL (green) and 2 solutions generated by A*HC (blue) where the goal is 100 meters from the start state. Search time was one second for both algorithms. Bottom left: HAEASL's node expansions. Bottom right: A*HC's node expansions.

Figure 9 shows two (left) and three (right) resulting trajectories from HAEASL during physical, on-robot field testing experiments. HAEASL and KEASL [2] were integrated to enable homotopically aware, kinodynamic motion planning through an unstructured, off-road environment. The multicolored lines emanating from the center of the maps are the reference frames. The maximum search runtime for these examples was one second, meaning our approach ran at 1Hz.

VI. Conclusions and Future Work

In this paper, we presented a novel search algorithm called Homotopy-Aware Efficiently Adaptive State Lattices that can generate multiple homotopically distinct paths for mobile robot planning. By introducing additional open lists as novel homotopy classes are encountered, the algorithm biases search directions of the state space that encourages HAEASL to generate multiple distinct solutions. The use of heuristic search in

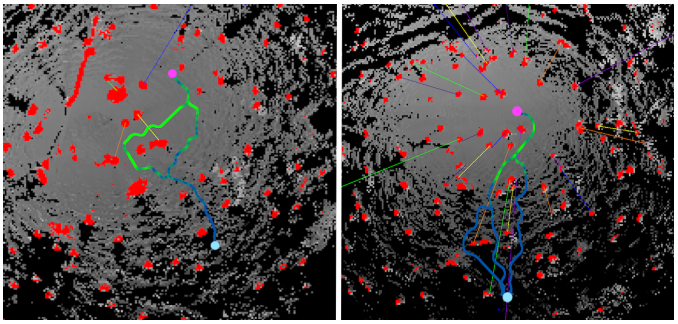

Fig. 9: A birds-eye view of the Clearpath Warthog's environment during off-road traversal. Red cells are obstacles, gray cells are traversable space, and black cells are unobserved space. Left, HAEASL generates two kinodynamic solutions from the start state (magenta) to the goal state (cyan). Right, HAEASL generates three solutions. The green parts of the paths indicate higher robot velocities and the blue parts indicate lower robot velocities. This speed reduction is due to the robot moving towards an area with more uncertainty. These constraints are calculated via roll and pitch values based on the map's height information [2].

each of these open lists ensures that a near-optimal path within a given homotopy class is generated. This has been done for the primary goal of easing the burden of a path-following controller in an autonomous navigation stack for negotiating through cluttered environments. The results of this contribution indicate that HAEASL can generate multiple high-quality, sufficiently diverse solutions, even when limited to a realistic time constraint of 2.0 seconds. On a per-planning-problem-basis, HAEASL is overtaken by A*HC [1] with respect to the number of solutions generated as the reference frame radius is increased. However, the average path diversity of HAEASL's solutions is better than those of A*HC's solutions by 9.16% across 2,720 valid planning problems. Across 2,965 valid planning problems, the average path cost of HAEASL's optimal solutions is about 0.7% higher than the average path cost of A*HC's optimal solutions. HAEASL outperformed HARRT* with respect to the average number of solutions generated and the optimality of those solutions. Our approach generated more solutions per problem in the most cases and its best solutions were approximately 17% better in terms of path cost. Finally, HAEASL was tested on a real-world environment and was successful in returning multiple solutions, even when incorporating kinodynamic constraints.

Ongoing work consists of integrating HAEASL into a navigation architecture with a path-following controller that considers multiple global plans. These experiments demonstrate how HAEASL may be able to generate multiple solutions quickly with a low reference frame radius in environments with limited perceptual information. By supplying a controller with a diverse set of solutions across different homotopy classes, we aim to demonstrate an quantifiable improvement in the safety and efficiency of mobile robot navigation in cluttered environments.

REFERENCES

[1] S. Bhattacharya. Search-based path planning with homotopy class constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1230–1237, 2010.

[2] E. R. Damm, J. M. Gregory, E. Lancaster, F. Sanchez, D. Sahu, and T. M. Howard. Terrain-aware kinodynamic planning with efficiently adaptive state lattices for mobile robot navigation in off-road environments. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2023. forthcoming.

[3] M. Dubuisson and A. K. Jain. A modified hausdorff distance for object matching. *Proceedings of 12th International Conference on Pattern Recognition*, 1:566–568 vol.1, 1994.

[4] 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*, SSC-4(2):100–107, 1968.

[5] B. Hedegaard, E. Fahnestock, J. Arkin, A. S. Menon, and T. M. Howard. Discrete optimization of adaptive state lattices for iterative motion planning on unmanned ground vehicles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5764–5771. IEEE, 2021.

[6] T. M. Howard et al. *Adaptive model-predictive motion planning for navigation in complex environments*. PhD thesis, Carnegie Mellon University, The Robotics Institute, 2009.

[7] T. M. Howard, C. J. Green, and A. Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Field and Service Robotics: Results of the 7th International Conference*, pages 69–78. Springer, 2010.

[8] T. M. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.

[9] K. D. Jenkins and J. R. Thornton. The shortest path problem in the plane with obstacles: A graph modeling approach to producing finite search lists of homotopy classes. Master's thesis, Naval Postgraduate School, 1991.

[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[11] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

[12] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[13] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[14] M. Likhachev, G. J. Gordon, and S. Thrun. ARA* : Anytime A* with provable bounds on sub-optimality. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 767–774. MIT Press, 2004.

[15] K. Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.

[16] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[17] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment. Robot navigation from human demonstration: Learning control behaviors. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1150–1157. IEEE, 2018.

[18] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[19] T. Yang, L. Huang, Y. Wang, and R. Xiong. Tree-based representation of locally shortest paths for 2d k-shortest non-homotopic path planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16553–16559. IEEE, 2024.

[20] D. Yi, M. A. Goodrich, and K. D. Seppi. Homotopy-aware rrt*: Toward human-robot topological path-planning. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 279–286. IEEE, 2016.