2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Generating Sparse Probabilistic Graphs for Efficient Planning in Uncertain Environments

Yasmin Veys, Martina Stadler Kurtz, and Nicholas Roy

***Abstract*— Environments with regions of uncertain traversability can be modeled as roadmaps with probabilistic edges for efficient planning under uncertainty. We would like to generate roadmaps that enable planners to efficiently find paths with expected low costs through uncertain environments. The roadmap must be sparse so that the planning problem is tractable, but still contain edges that are likely to contribute to low-cost plans under various realizations of the environmental uncertainty. Determining the optimal set of edges to add to the roadmap without considering an exponential number of traversability scenarios is challenging. We propose the use of a heuristic that bounds the ratio between the expected path cost in our graph and the expected path cost in an optimal graph to determine whether a given edge should be added to the roadmap. We test our approach in several environments, demonstrating that our uncertainty-aware roadmaps effectively trade off between plan quality and planning efficiency for uncertainty-aware agents navigating in the graph.**

I. Introduction

In this work, we consider navigation in environments where the traversabilities of many regions are unknown prior to planning. Recent works [1], [2] have demonstrated the effectiveness of representing uncertain environments as weighted graphs with probabilistic edges, because they enable long horizon reasoning about the environmental uncertainty and allow us to leverage existing algorithms, like those developed for the Canadian Traveler's Problem (CTP) [3], [4], to produce navigation policies that minimize expected costs. To guarantee the quality of the navigation solution, however, we must guarantee the quality of our graph representation. Existing approaches that use probabilistic graphs to find high-quality plans in realistic environments often rely on hand-designed graphs [2]. We would like to develop an algorithm for constructing roadmaps that enable uncertainty-aware planners to efficiently generate low-cost plans under various realizations of the environmental uncertainty.

Generating roadmaps that enable efficient planning under uncertainty requires trading off between competing objectives. The graph must be sufficiently sparse to ensure planning is computationally feasible, but must still represent a potentially exponential number of paths for the different traversability scenarios. We must be selective when deciding which edges to include in the graph, but it is unclear how to determine the optimal set of edges without explicitly considering every possible traversability scenario. For example, consider a robot tasked with navigating in a park that has an unreliable bridge (Fig. 1). The paths that should be included in the roadmap depend on the probability that the bridge is traversable. In small environments, we can reason about all of the different traversability scenarios, but in large environments, doing so becomes computationally intractable.

Fig. 1: The robot navigates from one side of the lake to the other. If the bridge is likely to be traversable, the roadmap should include the pink path so that the robot can inspect the bridge's traversability, as well as the two purple paths to allow the robot to go across the bridge if it is traversable and around the lake if it is not. If the bridge is likely untraversable, including those paths in the roadmap only increases planning complexity without significantly improving planning performance. A sparser graph that only has the blue path would still result in plans with expected near-optimal costs. Since the green and blue paths are both known to be traversable, including the longer green path in the roadmap is never useful.

While there is a rich literature of roadmap generation algorithms for planning in known environments, existing methods are insufficient for environments with uncertain regions. Existing techniques largely focus on generating graphs that cover the agent's configuration space, and many rely on the ability to partition the state space into free and obstacle regions [5]–[7]. For example, some algorithms use comparisons with the best known path in the environment to determine whether to add a new edge to the roadmap [8], [9]. Though these approaches work well in known environments, they are not well-suited to our problem, since they do not enable reasoning about traversability probabilities.

The contribution of this work is a method for efficiently generating high-quality roadmaps for navigation in environments with uncertain regions. We begin by constructing a graph that lies entirely in freespace. We then iteratively augment it by adding probabilistic edges that are likely to improve expected plan costs. The value of adding an edge to the graph is determined by computing a metric that bounds the ratio between the expected path cost in our graph and the expected path cost in an optimal graph, which considers all traversability scenarios. We provide analysis that shows we can bound the expected suboptimality of local paths in our graph, and test our approach in several environments, demonstrating that our uncertainty-aware roadmaps effectively trade off between plan quality and planning efficiency.

All authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology in Cambridge, USA. {yveys,mstadler,nickroy}@mit.edu

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-17-2-0181. Y. Veys acknowledges support from the Joseph T. Corso and Lily Corso graduate fellowship. Their support is gratefully acknowledged.

II. Problem Formulation

A. Modeling Uncertain Environments

We would like to generate a roadmap for a 2D agent navigating from start v_0 to goal $v_\star$ in an uncertain environment. We assume that the environment can be decomposed into regions with similar navigational properties (e.g., roads, forests, fields, etc.) and that the traversabilities of the regions are defined by a set of probability distributions that characterize the environmental uncertainty.

We assume the agent's environment is a polygon B that can be decomposed into a set of nonintersecting, simple 2D polygons $S = \{P_1, \ldots, P_b\}$, where each polygon is defined by an ordered list of points $p \in \mathbb{R}^2$, $P_i = \{p_1, \ldots, p_n\}$ (Fig. 2). We also assume access to a function f that maps each region to a traversability probability, $f : S \to [0, 1]$; for example, this function could be the result of assigning a traversability probability to each class in an overhead semantic segmentation of the environment. The regions that are known to be traversable make up *known freespace*: $\mathcal{X}_{free} = \cup\{P_i | f(P_i) = 1\}$. We assume that freespace is connected and that $v_0, v_\star \in \mathcal{X}_{free}$ such that the planning problem is always feasible, as in the CTP solvers of [4].

At the start of a planning instance, the (hidden) ground truth traversability $w_i \in \{0, 1\}$ of each region P_i is drawn from an independent Bernoulli distribution with parameter $f_i = f(P_i)$, and remains constant throughout the duration of the trial. Borrowing from the CTP literature, which introduces a similar concept, we name the assignment of regions to traversabilities a *weather*, $w = \{w_1, ..., w_b\}$ where $w \in W$, the set of all weathers. Then, the probability of a weather w can be written as the joint probability of the individual Bernoulli assignments, $\Phi = \mathbb{P}[w] = \prod_{i=0}^{|w|}(1 - f_i)^{(1-w_i)} f_i^{w_i}$.

B. Planning in Uncertain Environments

Given the environment abstraction, we model the planning problem as a modified version of the CTP, a formalization for navigation in a graph with probabilistic edges. In the traditional formulation, the ground truth traversability of each edge is drawn from an independent Bernoulli distribution, and a weather refers to an assignment of the edges to their traversabilities. An agent navigating in the graph observes the ground truth traversabilities of the edges incident to its current vertex and maintains a belief over the set of all weathers. The objective of the CTP is to find an optimal policy (i.e., a function that maps beliefs to actions) that minimizes the expected cost of traveling from v_0 to $v_\star$.

We adapt the CTP to consider weathers over regions, rather than edges. We let the traversability of an edge be the traversability of the region it intersects[1], and we let $\tau : E \to (0, 1]$[2] be the function that returns the probability of an edge. When a region is untraversable, all edges that intersect it are untraversable and vice versa when a region is traversable. Thus, when an agent observes the traversability of an edge, it learns the traversability of the region it intersects and of all other edges that intersect that region. The objective of the modified CTP remains the same, though the agent's belief is now over the set of region weathers. In this work, we assign edge costs based on Euclidean distance.

[1]If an edge intersects more than one region, we can segment it such that each segment only intersects one region.

[2]We exclude edges through untraversable regions (i.e., where $\tau(e) = 0$).

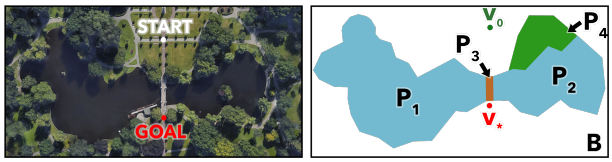

Fig. 2: The robot is tasked with navigating through a park. The bridge over the lake may or may not be traversable. We abstract the key regions in the environment as polygons (using e.g., a semantically segmented overhead image): P_1 and P_2 represent the lake, which is known to be untraversable. P_3 represents the bridge, which has a 50% chance of being traversable. P_4 represents the grove of trees, which has a 30% chance of being traversable. In the weather $w = \{0, 0, 1, 1\}$, the bridge and forest are traversable, while the lake is untraversable.

C. Problem Formulation

The roadmap $G = (V, E)$ we generate should enable an agent to find low-cost plans from v_0 to $v_\star$ in different weathers (i.e., traversability scenarios), while remaining efficient to build and plan in. This trade-off between the computational efficiency of planning and the quality of the plan can be formalized as the following optimization problem, where $t_g(G)$, $t_p(G, w)$, and $c_p(G, w)$ are the computation time for building the graph, the computation time for planning using the graph, and the cost of the plan in the graph, given a weather w, respectively.

$$G^* = \arg\min_G t_g(G) + \mathbb{E}_{w\sim\mathbb{P}[w]}\left[t_p(G, w) + c_p(G, w)\right] \quad (1)$$

III. Approach

Unfortunately, directly solving (1) is intractable, as evaluating c_p requires finding the optimal policy for the CTP, which is known to be #P-hard [4], [10], and solving the CTP to generate the optimal plan incurs t_p, further complicating the optimization. While we cannot explicitly solve (1), we can optimize the structure of the graph such that t_p and c_p are likely to be small. For example, removing all uncertain edges from the graph would ensure that planning is efficient, but the resulting plan may be relatively high cost. On the other hand, constructing a dense graph that represents all possible paths in the environment would likely result in expected low-cost plans, but doing so would be expensive. We aim to achieve a balance between these extremes by generating graphs that are sparse, yet still capture high-quality paths in likely weathers.

A. Overview

We propose to leverage environmental structure to efficiently select high-quality edges to add to the roadmap. In our approach (Fig. 3), we construct a graph connecting v_0 and $v_\star$ through freespace and then augment it by adding shortcut edges through uncertain regions. To determine whether a shortcut edge should be added, we compare its cost and traversability probability to the existing path through freespace. The resulting graph represents high-quality paths in the environment, but is not optimized for an agent planning under uncertainty. To enable an agent to recover from

mistakes when planning, we also add recovery edges. We also sparsify the roadmap by pruning vertices and edges that do not significantly impact plan quality.

B. Graph Connecting Known Freespace

To ensure that our roadmap contains high-quality paths through $\mathcal{X}_{free}$, we use a traditional roadmap method as a foundation for our uncertainty-aware roadmap. This roadmap can also be thought of as being risk-averse, as it does not include paths that go through any uncertain regions. Formally, we generate $G_{free} = (V_{free}, E_{free})$ such that $V_{free} \in \mathcal{X}_{free}$, and $\tau(e) = 1 \ \forall e \in E_{free}$. In practice, we select our freespace graph to be a reduced visibility graph [11], [12]. To generate the reduced visibility graph, we consider adding edges between all pairs of region vertices, v_0, and $v_\star$. If an edge lies entirely in freespace and is a common tangent of the regions[3], it is added to E_{free} and its vertices are added to V_{free}. The resulting graph G_{free} is known to contain the shortest Euclidean distance path from start to goal in freespace, which can be found by running a standard shortest path algorithm (e.g., A* [13]).

To sparsify G_{free}, we remove vertices and edges that are guaranteed not to lie on an optimal path using an admissible ellipsoid heuristic [8]. More specifically, we remove vertices and edges that lie outside of $\mathcal{E}(v_0, v_\star)$, where $\mathcal{E}(u, v) = \{y \in \mathbb{R}^2 | d(u, y) + d(y, v) \leq c^{free}_{u \to v}\}$ and $c^{free}_{u \to v}$ is the length of the shortest path between u and v in G_{free}. Intuitively, the length of the shortest path from u to v in the weather where all of the regions are untraversable (i.e., $c^{free}_{u \to v}$) is an upper bound on the shortest path in any weather, since making a region traversable cannot increase path costs. Thus, removing vertices and edges that lie outside of $\mathcal{E}(v_0, v_\star)$ has no impact on expected path costs from v_0 to $v_\star$.

C. Paths Through Uncertain Regions

We now augment G_{free} by adding edges through uncertain regions (i.e., shortcuts). The resulting augmented graph is $G = (V, E)$ where $V = V_{free}$ and $E \supseteq E_{free}$. A shortcut through an uncertain region may be useful for planning if the region is likely to be traversable or if the shortcut is significantly shorter than the existing path through freespace. Ideally, we would like to add edges to G_{free} that maximally decrease the ratio between the expected path cost from v_0 to $v_\star$ in the resulting graph G versus an optimal graph G^* that contains the shortest paths from v_0 to $v_\star$ in all weathers. Formally, we define $\delta : V \times V \times G \to \mathbb{R}$ to be the expected suboptimality of the paths between u and v in a given graph G versus the optimal graph G^*,

$$\delta(u, v, G) = \frac{\mathbb{E}_{w \sim \mathbb{P}[w]}[c_{u \to v}; G]}{\mathbb{E}_{w \sim \mathbb{P}[w]}[c_{u \to v}; G^*]}, \tag{2}$$

where G^* is the *stacked visibility graph* (SVG), constructed by generating $2^{|S|}$ reduced visibility graphs, one for each weather[4], and finding their union. We seek to minimize $\delta(v_0, v_\star, G)$ by adding high-quality shortcut edges to G_{free}.

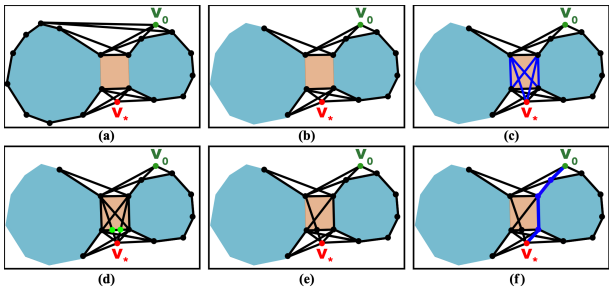

Fig. 3: (a) The reduced visibility graph connecting v_0 and $v_\star$ through freespace is generated. (b) Vertices that are known not to improve paths between v_0 and $v_\star$ are pruned, along with their incident edges. (c) Shortcut edges are added to the graph when the existing freespace paths are overly long. Many shortcut edges are added across the likely traversable bridge. (d) There are two edges that must be segmented according to their intersections with the bridge. The added vertices are shown in green. (e) Three redundant edges through the bridge are pruned. (f) In the weather where the bridge is traversable, the graph contains a near optimal path from v_0 to $v_\star$.

Determining the minimum set of edges that reduces $\delta(v_0, v_\star, G)$ is nontrivial. However, if while building the graph, we realize that the expected cost of traveling between a pair of vertices u and v is relatively high, then we can add the edge $e_{u,v}$ to the graph in hopes of improving paths between v_0 and $v_\star$ that contain u and v. Specifically, for all pairs of vertices $u, v \in V$, we propose to add the edge $e_{u,v}$ to graph G' during an intermediate step of the algorithm if the expected suboptimality between u and v in G' exceeds some threshold $k \geq 1$. Unfortunately, calculating $\delta(u, v, G')$ is generally expensive, so we instead calculate an upper bound.

We first consider an upper bound for the numerator, or the cost of navigating in the graph G' without the edge $e_{u,v}$. A conservative estimate of this cost is the length of the shortest path from u to v in the freespace graph G_{free}, $h := c^{free}_{u \to v}$. Thus, we have $\mathbb{E}_{w \sim \mathbb{P}[w]}[c_{u \to v}; G'] \leq h$. Next, we consider a lower bound for the denominator, or the cost of navigating in the optimal SVG. Employing the ellipsoid heuristic mentioned previously, we know that all shortest paths between u and v lie in an ellipse defined by h. When the regions inside of the ellipse are all untraversable (with probability ρ_b), the length of the shortest path from u to v in the SVG is h. When all regions are traversable, the optimal path is the straight-line edge $e_{u,v}$, which has length ℓ. To generate a lower bound, we assign the correct probability to h and the rest of the probability to ℓ, Thus, we have $\mathbb{E}_{w \sim \mathbb{P}[w]}[c_{u \to v}; G^*] \geq \rho_b h + (1 - \rho_b)\ell$. Applying the bounds, we recover an upper bound $\gamma(u, v)$[5] for $\delta(u, v, G')$,

$$\delta(u, v, G') \leq \frac{h}{\rho_b h + (1 - \rho_b)\ell} = \gamma(u, v). \tag{3}$$

When the existing path through freespace is likely to be useful (i.e., $\rho_b \approx 1$) or when the straight-line edge is similar in length to the freespace path (i.e., $\ell \approx h$), the upper

[3]Traditional visibility graphs include edges that are not common tangents; however these are known not to lie on optimal paths, so they can be removed.

[4]The reduced visibility graph is built using the set of untraversable regions defined by weather w.

[5]Note that h and ℓ are implicitly functions of u and v and that $\gamma(u, v)$ is not a function of a graph G'. $\gamma(u, v) \geq \delta(u, v, G')$ for all G' where $G' \supseteq G_{free}$, which is true by our construction.

bound tends to 1, implying that the shortcut edge should not be added to the graph. We define a maximum acceptable expected suboptimality, k, and iteratively add all shortcut edges $e_{u,v}$ for which the upper bound $\gamma(u,v) > k$ to generate the final augmented graph G.

D. Discussion

While our algorithm adds the edge $e_{u,v}$ to the intermediate graph G' whenever $\gamma(u,v) > k$, doing so does not generally ensure that the expected suboptimality between u and v in the final graph G is upper bounded by k. We provide analysis for when adding the edge may be overly conservative and discuss the special cases for which $\delta(u,v,G) \leq k$.

For all pairs of vertices u and v, we decide whether to add the edge $e_{u,v}$ to G' based on the relationship between the upper bound $\gamma(u,v)$ and k. If $\gamma(u,v) \leq k$, then we do not add the edge and trivially, $\delta(u,v,G) \leq \gamma(u,v) \leq k$. If $\gamma(u,v) > k$, then we add the edge to G', so the final graph G contains $e_{u,v}$. However, these properties do not allow us to establish a relationship between $\delta(u,v,G)$ and k. Adding the edge from u and v alone may not reduce $\mathbb{E}_{w\sim\mathbb{P}[w]}[c_{u\to v}; G]$ enough to ensure that $\delta(u,v,G) \leq k$ if there are alternate paths from u to v that are shorter than the freespace path, but more likely than the straight-line path. Thus, adding the edge may be overly conservative if doing so does not significantly reduce $\delta(u,v,G)$. However, conservatively adding edges does not increase expected plan costs, although it may densify G.

While we cannot ensure $\delta(u,v,G) \leq k$ in general, there are some pairs of vertices for which our algorithm guarantees $\delta(u,v,G) \leq k$. Given a pair of vertices u and v, consider the set of unique shortest paths from u to v in all weathers. If the only two paths in the set are the freespace path from u to v and the straight-line edge, then our algorithm guarantees that $\delta(u,v,G) \leq k$ even when $\gamma(u,v) > k$ and $e_{u,v}$ is added. Intuitively, this situation arises when the optimal paths between u and v are a function of the traversability of a single region. Since there are only two optimal paths from u to v and G contains both of them, we know that $\delta(u,v,G) = 1 \leq k$, even though $\gamma(u,v) > k$.

In practice, we can find the minimum k such that no shortcut edges are added to G_{free}, which we call k_{max}, and then iteratively plan in graphs with decreasing values of k, starting with k_{max}, until the allotted time to plan runs out.

E. Adding Recovery Paths

While the augmented graph G contains many high-quality paths through the environment, it is not yet optimized for planning under uncertainty and adaptation to sensed information. If an agent discovers that a region is untraversable, it should follow the boundary around the region to make progress to the goal, rather than backtracking. To enable failure recovery during online planning, we segment the shortcut and region boundary edges according to their intersections.

F. Removing Redundant Edges

When constructing G, the number of shortcut edges we consider adding is on the order of $|V|^2$, since we reason about all pairs of vertices. For very large graphs, adding every edge that satisfies $\gamma(u,v) > k$ may result in dense graphs that are impractical to plan with. We could increase k such that fewer edges are added, but the resulting graph may continue to have clusters of edges with similar $\gamma(u,v)$. Consider the example in Fig. 4. $\gamma(u,v)$ and $\gamma(w,v)$ will be almost identical in value, since edges $e_{u,v}$ and $e_{w,v}$ intersect the same region. Removing one of them will not have a significant impact on the expected path costs.

Fig. 4: Edge Reduction Step

Therefore, we refine G by considering the set of edges that intersect a given region and deleting redundant ones. By construction, the endpoints of these edges lie on the boundary of the region. We randomly sample one of the edges $e_{u,v}$ in the set and delete all edges $e_{w,v}$ where rerouting the path from w to v through $e_{u,v}$ does not increase the path cost from w to v by too large a factor $\alpha > 1$, which we choose. We repeat this until there are no new edges to sample. This step effectively increases our suboptimality bound from k to αk, since the length of all paths in the graph are guaranteed not to increase by more than α.

IV. Experimental Results

A. Toy Environment

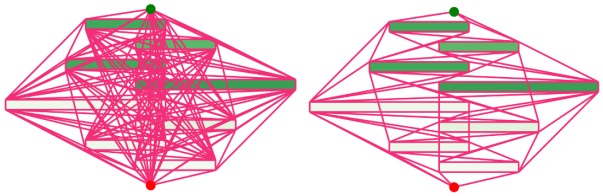
Fig. 5: Toy environment with long, thin obstacles. Darker colored regions have low traversability probabilities, lighter colored regions have high traversability probabilities, and white space is freespace. On the left is the dense, optimal stacked visbility graph (SVG). On the right is our graph with $k = 2.5$. As expected, there are more shortcut edges that go through likely traversable regions than likely untraversable regions.

We first tested our approach in a hand-designed, toy environment with long, thin regions (Fig. 5). In this environment, edges that cross likely traversable regions provide significant improvements over the much longer paths through freespace. We generated four graphs for comparison. We first modified the k-PRM algorithm [5] to allow edges to cross uncertain regions, which we segmented along obstacle boundaries (see Section III-E). Our sampling strategy was proportional to the region probabilities (e.g., a vertex sampled in a 40% likely traversable region was rejected with 60% probability), and we sampled 20 vertices for the toy environment and 50 vertices for the other environments. Each vertex was then connected to its 5 nearest neighbors[6]. We then generated the stacked visibility graph (i.e., the SVG defined in Section III-C) that contains the shortest path between v_0 and $v_\star$ for every weather. Finally, we generated two graphs using our

[6]These parameters were chosen such that the resulting graphs were of comparable size to ones produced by our approach.

approach: one where the expected suboptimality threshold is $k = \infty$ (i.e., the freespace graph that has no shortcuts) and one where $k = 2.5$. We let the edge reduction factor be $\alpha = 1.5$ for all of our experiments.

To evaluate graph quality, we calculated the average *oracle path cost* and *online path cost* across 50 randomly sampled weathers. The *oracle path cost* is the length of the shortest path between v_0 and $v_\star$, assuming that the traversabilities of the regions are known. The *online path cost* is the total path cost accumulated by an uncertainty-aware agent that makes observations online and replans according to a CTP policy generated using the optimistic rollout algorithm from [4]. If no path was found between v_0 and $v_\star$ (i.e., if the graph became disconnected), we considered that a failure weather.

In Table I, we report the time T_{gen} required to generate each graph in seconds, along with $|V|$ and $|E|$. We also report the percentage of failed weathers for each graph, along with the ratio between the average oracle path cost and the average optimal path cost, R_{ORA}, which approximates the expected suboptimality between v_0 and $v_\star$ in G. We similarly report the ratio between the average online path cost and the average optimal path cost[7], R_{CTP}. Finally, we report the time T_{plan} in seconds it took the planner to find a solution, averaged over the number of weathers.

TABLE I: Toy Environment Results

Graphs	T_{gen}	$\|V\|$	$\|E\|$	Failure Rate	R_{ORA}	R_{CTP}	T_{plan}
$k = 2.5$	0.068s	44	161	0%	1.07	1.09	1.72s
$k = \infty$	0.066s	34	118	0%	1.43	1.43	0.85s
SVG	13.04s	375	1224	0%	1.00	1.05	76.65s
PRM	0.030s	57	133	8%	1.36	1.68	0.79s

We demonstrate that classical motion planning techniques are unable to achieve a good balance between computational efficiency and plan quality. Of the three approaches, the optimal SVG had the lowest R_{ORA} and R_{CTP} but the highest T_{gen} and T_{plan}. The SVG's planning performance is near-optimal, because it explicitly considers all traversability scenarios, but at the cost of computational efficiency. The PRM graph, on the other hand, had the lowest T_{gen} and T_{plan} but the highest R_{ORA} and R_{CTP}. Since the algorithm does not consider any environmental uncertainty, its planning performance is poor, even though it is efficient. Our graph with $k = 2.5$ was over 190 times faster to generate than the SVG, 44 times faster to plan with, and was less than 10% worse in terms of path quality. In comparison to PRM, we performed 29% better for the oracle path costs and 59% better for the CTP path costs. Our graph took only twice as long to generate and plan with. We also show improvement over the freespace graph, with a 36% difference in R_{ORA} and a 34% difference in R_{CTP}.

B. Synthetic Polygonal Environments

We also tested our approach in 100 synthetic polygonal environments (Fig 6). We randomly sampled points inside of the environment polygon B to generate a Voronoi decomposition, and then merged and shrank the resulting Voronoi polygons to create irregular, concave regions and narrow paths of freespace. The traversability probabilities of the regions were uniformly sampled from $[0, 1]$. A total of 1000 trials were generated by randomly sampling 10 start-goal pairs per environment, discarding and re-sampling if the start or the goal was not in freespace.

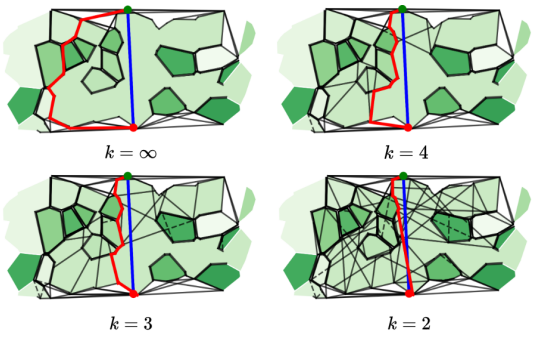

Fig. 6: An example of a randomly generated environment with concave and convex regions. The region in the middle is traversable, so the shortest path from the start to the goal is a straight line (blue). As k decreases, more shortcuts are added to the graph, and the length of the oracle path (red) approaches the optimal length.

We report metrics for the modified PRM graph and our approach with $k = 2, 3, 4, 5, 6,$ and ∞. We chose not to include the SVG as a baseline for computational reasons, since the synthetic environments can have up to 25 regions. The metrics are the same as for the toy environment, but averaged over all trials. More specifically, for a given graph, T_{gen}, $|V|$, and $|E|$ are averaged over 1000 trials and R_{ORA}, R_{CTP}, and T_{plan} are averaged over all weathers in all trials (i.e., 50,000 total planning instances).

TABLE II: Polygonal Environments Results

Graphs	T_{gen}	$\|V\|$	$\|E\|$	Failure Rate	R_{ORA}	R_{CTP}	T_{plan}
$k = 2$	54.68s	188	525	0%	1.12	1.12	16.35s
$k = 3$	14.18s	145	396	0%	1.13	1.14	7.56s
$k = 4$	5.70s	124	330	0%	1.16	1.16	3.70s
$k = 5$	3.56s	114	303	0%	1.18	1.19	2.34s
$k = 6$	2.88s	110	289	0%	1.20	1.20	1.73s
$k = \infty$	2.34s	104	268	0%	1.33	1.35	0.81s
PRM	0.29s	242	586	52.2%	1.47	1.57	12.18s

In these experiments, we compare the efficiency and planning performance with varying k (see Table II). As we decrease k, our graphs become increasingly dense, since more edges satisfy $\gamma(u, v) > k$. Though the criteria is calculated for all pairs of vertices, regardless of k, it becomes more expensive to generate graphs with lower k due to the edge reduction step, which reasons over the set of added edges. Our densest graph with $k = 2$, takes 27 times longer to generate than the freespace graph with $k = \infty$. In terms of planning performance, we notice that both R_{ORA} and R_{CTP} monotonically decrease as we decrease k. Furthermore, though our algorithm cannot guarantee that $\delta(v_0, v_\star, G) \leq k$ (as shown in Section III-D), both R_{ORA} and R_{CTP} are less than k for all graphs generated by our approach. Our densest graph, with $k = 2$, shows a 20% improvement in R_{ORA} and R_{CTP} over the freespace graph with $k = \infty$. Finally, we

[7]We compared the online path cost to the optimal path cost, instead of the optimal online path cost, because finding optimal CTP policies is computationally expensive. The ratio we report is an upper bound.

note the trends in planning efficiency. Unsurprisingly, as k decreases and the graph contains more uncertain edges, the time required to plan increases. Because the time required to plan is loosely a function of the number of edges in the graph, we notice that the PRM takes almost as long to plan in as the graph with $k = 2$, likely because they have a similar number of edges. However, the PRM is over 180 times faster to generate than the graph with $k = 2$.

C. MIT Environment

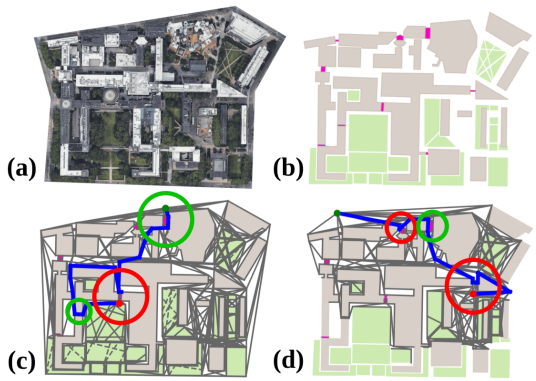

Fig. 7: (a) Overhead satellite image of MIT, taken from Google Earth. (b) Region polygons derived from OpenStreetMap [14]. (c) The online path that the robot takes as it traverses through campus (blue). The robot takes the shorter path through the open passageway circled in green to avoid going around all of the buildings, and then checks the traversability of the passageway circled in red, before taking the shortest freespace path from the passageway to the goal. (d) Three passageways provide potential shortcuts through buildings. The first and third are untraversable, while the second is traversable. When the robot observes an untraversable shortcut, it takes the shortest possible path to its next location using freespace and recovery edges.

Finally, we tested our approach in a real-world campus environment with 60 total regions. The buildings and fields were derived from OpenStreetMap [14], and walkways were hand-annotated. Buildings are assumed untraversable, while fields and walkways are traversable with 70% and 40% probability[8], respectively. Five trial starts and goals were hand-selected to capture semantically meaningful environmental traversals. We generated the same set of graphs and metrics as for the polygonal environments.

TABLE III: MIT Environment Results

Graphs	T_{gen}	$\|V\|$	$\|E\|$	Failure Rate	R_{ORA}	R_{CTP}	T_{plan}
$k = 2$	142.38s	2090	7356	0%	1.03	1.28	112.31s
$k = 3$	103.52s	1868	6701	0%	1.04	1.31	75.16s
$k = 4$	96.25s	1727	6255	0%	1.04	1.35	55.07s
$k = 5$	93.81s	1649	6001	0%	1.04	1.36	42.10s
$k = 6$	92.70s	1611	5878	0%	1.04	1.38	36.94s
$k = \infty$	90.85s	1513	5396	0%	1.56	1.56	3.92s
PRM	0.99s	1267	3050	80%	1.48	1.59	8.88s

We demonstrate that our approach enables an agent planning under uncertainty to generate high-quality navigation plans with good recovery paths in a real-world urban environment. The trends we see in the previous sections hold, so for this experiment, we highlight two intuitive examples that demonstrate the agent's plan through the campus (Fig. 7).

[8]These probabilities were hand-selected using expert knowledge of the environment.

V. Related Works

There exists a rich body of work that addresses the problem of roadmap generation in known environments. Some roadmaps, such as visibility graphs [11], [12] and generalized Voronoi diagrams [15], are built deterministically, and have optimality guarantees under specific distance metrics. Sampling-based algorithms, which generate random graphs through the environment, have been shown to be probabilistically complete [5], [6], and can be modified to be asymptotically optimal [7]. Additionally, many heuristics have been developed to practically improve the computational costs and planning outcomes of sampling-based motion planners, from intelligent sampling methods [16], [17] to graph sparsification heuristics [8]. Other methods leverage results from graph theory to generate high-quality roadmaps, such as adding useful cycles [18], using graph spanners [9], and pruning edges [19], [20]. While these techniques produce sparse, high-quality roadmaps in known environments, they are not well-suited to planning in uncertain environments, since they are not designed to model environmental uncertainty.

Additionally, the problem of planning under uncertainty has been addressed in various ways. Some approaches use discretized representations of uncertainty for planning (e.g., [21], [22]). However, these representations can become computationally inefficient as the size of the environment increases. Belief-space planners (e.g., [23]–[25]) construct belief-space graphs that represent both the agent configuration space and different realizations of the environmental uncertainty. However, because these methods are not designed to minimally represent environmental uncertainty, they can become computationally intractable as the number of realizations increases. In our work, rather than generating a graph that jointly represents configuration and belief space, we focus on generating a representation of configuration space that accurately represents the environment, including environmental uncertainty, and is efficient for uncertainty-aware planning (i.e., using Monte Carlo methods [4]). Finally, other approaches have used overhead views (e.g., images [1], [21], semantic segmentations [26]) to guide planning under uncertainty. However, these approaches do not leverage environmental structure during roadmap construction.

VI. Conclusions

In this work, we presented our approach for generating sparse, high-quality probabilistic graphs in uncertain environments. Using a metric that bounds the ratio between the expected path cost in our graph and the expected path cost in the optimal graph, we can determine what shortcuts are valuable to add to the underlying graph through known freespace. We tested our approach in several environments, demonstrating that our uncertainty-aware roadmaps effectively trade off between plan quality and planning efficiency. In future work, we would be interested in generating these roadmaps using polygons derived from real-world, semantically segmented satellite images. It would also be interesting to study an extension of the algorithm to higher dimensions (e.g., for a quadrotor navigating in an uncertain 3D environment).

References

[1] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 182–195, 2019.

[2] M. Stadler, J. Banfi, and N. Roy, "Approximating the value of collaborative team actions for efficient multi-agent navigation in uncertain graphs," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, pp. 677–685, 2023.

[3] C. H. Papadimitriou and M. Yannakakis, "Shortest paths without a map," *Theoretical Computer Science*, vol. 84, no. 1, pp. 127–150, 1991.

[4] P. Eyerich, T. Keller, and M. Helmert, "High-quality policies for the canadian traveler's problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010, pp. 51–58.

[5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

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

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2997–3004.

[9] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

[10] E. Nikolova and D. R. Karger, "Route planning under uncertainty: The canadian traveller problem.," in *AAAI*, 2008, pp. 969–974.

[11] M. De Berg, *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[12] H. Rohnert, "Shortest paths in the plane with convex polygonal obstacles," *Information Processing Letters*, vol. 23, no. 2, pp. 71–76, 1986.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[14] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, `https://www.openstreetmap.org`, 2017.

[15] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Transactions on robotics and automation*, vol. 5, no. 2, pp. 143–150, 1989.

[16] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, IEEE, vol. 3, 2003, pp. 4420–4426.

[17] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7087–7094.

[18] D. Nieuwenhuisen and M. Overmars, "Useful cycles in probabilistic roadmap graphs," vol. 1, May 2004, 446 –452 Vol.1, ISBN: 0-7803-8232-3.

[19] O. Salzman, D. Shaharabani, P. K. Agarwal, and D. Halperin, "Sparsification of motion-planning roadmaps by edge contraction," *The International Journal of Robotics Research*, vol. 33, no. 14, pp. 1711–1725, 2014.

[20] F. Zhou, S. Mahler, and H. Toivonen, "Simplification of networks by edge pruning," vol. 7250, Jan. 2012, pp. 179–198.

[21] L. Murphy and P. Newman, "Risky planning on probabilistic costmaps for path planning in outdoor environments," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 445–457, 2012.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[23] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[24] C. Phiquepal, A. Orthey, N. Viennot, and M. Toussaint, "Path-tree optimization in discrete partially observable environments using rapidly-exploring belief-space graphs," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 160–10 167, 2022.

[25] D. Zheng and P. Tsiotras, "Ibbt: Informed batch belief trees for motion planning under uncertainty," *arXiv preprint arXiv:2304.10984*, 2023.

[26] M. Everett, J. Miller, and J. P. How, "Planning beyond the sensing horizon using a learned context," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1064–1071.