2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

A Heuristic Approach to Finding Diverse Short Paths

Caleb Voss, Mark Moll, and Lydia E. Kavraki

Abstract— **We present an algorithm that seeks to find a set of diverse, short paths through a roadmap graph. The usefulness of a such a set is illustrated in robotic motion planning and routing applications wherein a precomputed roadmap of the environment is partially invalidated by some change, for example, relocation of obstacles or reconfiguration of the robot. Our algorithm employs the heuristic that nearby configurations are likely to be invalidated by the same change. To find diverse short paths, the algorithm finds the shortest detour avoiding a collection of balls imposed on the graph as simulated obstacles. Different collections yield different short paths. Paths may then be checked for validity as a cheap alternative to checking or reconstructing the entire roadmap. We describe a formal definition of path set diversity and several measures on which to evaluate our algorithm. We compare the speed and quality of our heuristic algorithm's results against an exact algorithm that computes the optimally shortest set of paths on the roadmap having a minimum diversity. We show that, with tolerable loss in shortness, we produce equally diverse path sets orders of magnitude more quickly.**

I. Introduction

The problem of robotic motion planning, how to move a robot between two configurations along a collision free path through an environment, is often approached through the techniques of sampling-based planners. These methods rely on building a data structure of valid motion segments through the configuration space, thereby reducing the problem to a graph search [1]. It is often the case that a robot needs to plan many successive queries through the same environment, but with different start and goal configurations. The probabilistic roadmap (PRM) [2] and algorithms based on it, like SPARS2 [3], are suited to this problem as they build a graph of valid segments throughout the entire environment. This is done offline in preparation for future queries. Online, the start and goal configurations are connected to the graph. Motion planning problems can then be solved using a graph search between the start and goal [4]. Because of the nature of such a roadmap, there can be many paths satisfying the same query.

Consider a problem in which the robot plans multiple times in the same environment using a roadmap, but in which the free space is subject to modifications between plans. As such, some edges and vertices in the roadmap graph may not always be valid. As a motivating example, a robotic arm on a mobile base is asked to move objects of various sizes through a cluttered space. It will find itself in a more constricted free space when the object is large, and relocation of objects may occlude some paths from plan to plan. If such a robot plans its motion using a precomputed roadmap, it is expensive to update the roadmap by collision-checking every edge to determine which of them are invalidated by the changes. Instead, we anticipate that much of the space remains unchanged and propose to find a number of alternative and diverse routes that achieve the goal according to the original roadmap. Finding a valid path among this limited set is less expensive than fixing the entire roadmap graph because there are many fewer edges to check. Moreover, the set of alternative paths needs only to be computed once, without depending on information about the various objects to be carried and possible changes in clutter. In contrast, a replan-from-scratch approach cannot begin until the object is known, and must be run again for each one. Finally, these alternatives can even be used as online contingencies in a scenario where the precise effect of a change is not known until the robot encounters it.

We present an algorithm that heuristically seeks to generate a set of diverse short paths through a graph. Our work considers a roadmap embedded in the configuration space of a robot; however, we will show that our algorithm does not actually require an embedding if the metric used is based on graph distance. An algorithm for finding diverse short paths should find paths that explore different regions of the configuration space, not paths that differ by just a few edges. A change in the environment, like the introduction of a new obstacle, is likely to invalidate not just a single edge, but multiple edges located near to one another. Another kind of change is to the robot itself, like having it carry a package. This may dilate obstacles and occlude narrow passages. In this case, all paths through the newly blocked passage are invalidated. Our heuristic is that similar paths are likely to fail together as a result of a change in the set of valid configurations. Not only do paths that share too many edges in common suffer this weakness, but also paths that otherwise remain near each other in the configuration space. This summarizes our intuitive concept of diverse paths through a space. Later we will formally define diversity. It is also an objective of our work to consider short paths. It is easy to find very different paths by making them very long, but unnecessarily long paths are undesirable for typical applications in motion planning.

Our contribution is more general than this motivating problem: diverse, short paths through an existing graph are useful in solving a broader range of problems. For example, as discussed in [5], there are algorithms that seek to deform a given path to optimize some property, but are limited to finding the local optimum because a continuous deformation

Work by Caleb Voss has been supported in part by NSF 1317849. Work by Mark Moll and Lydia Kavraki has been supported in part by NSF 1317849, 1139011, and ARL/ARO W911NF-09-1-0383.

The authors are with the Department of Computer Science at Rice University, Houston, TX, USA. {cav2, mmoll, kavraki}@rice.edu

can only produce a homotopically equivalent path. However, we are able to provide many paths which often represent different homotopies, and, starting from this set, such an approach can then find multiple optima and select the best.

Routing a vehicle through a busy city is another application for our algorithm. This problem is fundamentally different from the typical planning problem because a change in traffic does not often change the set of valid paths, but merely alters the cost of the paths. When congestion drastically alters travel time on nearby roads, we do not need to determine the updated weight for every edge in the roadmap if we can find just a small selection of paths to check.

In the next section we examine existing works on finding multiple paths through a graph and motion planning in a dynamic environment, as well as works on the diversity of paths. After formalizing our terminology, we present the algorithm and techniques we have developed to solve the diverse short paths problem. We proceed to make theoretical guarantees for our algorithm, then evaluate its performance and illustrate some useful applications. Finally, we conclude with a discussion of areas of future study.

II. Related Work

There is much research toward the development of algorithms that solve several variations of the multiple-path graph search problem, beginning with one of the most fundamental: Eppstein's k shortest paths algorithm [6]. For any query from a start to a goal vertex in a graph, it finds exactly the k shortest paths satisfying this query. The resultant paths are not suitable for our problem as they are often too similar to one another. However, we can filter these paths for diversity to form a baseline for our experiments. A heuristic improvement on Eppstein's algorithm is K* [7]. Whereas Eppstein prepares a data structure from which the paths may be extracted in order, K* delays the processing on areas of the graph which are far from the shortest path until it is necessary, so that computation is avoided for paths which are certainly longer than the k shortest. One can even begin K* before the graph is fully known. Since we are interested in diverse paths, it is necessary to examine large areas of the graph, away from the shortest path. This means we need a very large k and will not benefit from K*.

If one already has a large explicit set of paths, the work in [8] provides an algorithm to pass to a subset which is robust to local failure, i.e., for which an obstacle occluding one path is unlikely to occlude many others. Another algorithm [9] finds a subset of n paths optimized for dispersion, thereby covering the space of all paths. Achieving diversity in these ways is attractive, but our motivation starts from a graph, which implicitly defines a prohibitively large set of paths, and, moreover, we are also interested in short or, more generally, low cost paths.

In network routing, it is important to find multiple paths not relying on the same edges of the network graph [10]. However, there is a key difference between that problem and ours. Our graphs are embedded in a configuration space, so we have a concept of proximity between vertices in this space regardless of how they are connected in the graph. This is important because local changes in the environment can affect multiple vertices that are near each other in the configuration space. Network routing problems are thus typically concerned with paths that share the fewest number of edges or nodes, for example, by using algorithms based on Suurballe's k shortest disjoint paths [11], and not with paths that traverse different areas of the space.

Several different definitions of path set diversity that take the configuration space into account have been employed, including simple measures like the minimum of the distance between any two paths [12], or more involved measures like the average probability that at least one path remains valid over all possible environments [13]. Moreover, there are many ways to define the distance between two paths, including the Hausdorff, Levenshtein, and Fréchet measures [14]–[16]. We will examine these three, but focus our attention on one of them in our experiments.

One approach to motion planning in a changing environment is the dynamic roadmap (DRM) [17], which discretizes the space into cells. The objective of that algorithm is to find a path through a precomputed roadmap, given a set of cells that are obstructed by a change in the environment. The implementation in [18] can quickly determine the obstructed cells because its obstacle sensor returns a point cloud that can be mapped to invalid configurations. This is a reactive, local approach. We are instead proposing an offline solution to maximize use of previously computed information in preparation for the changing environment.

III. Definitions

Consider a metric space $(\mathcal{C}, \delta)$, a weighted undirected graph $G = (V, E)$ embedded in $\mathcal{C}$ via the mapping $\psi : V \to \mathcal{C}$, and edge weight function $w : E \to [0, \infty]$ with

$$w(e) = \delta(\psi(u), \psi(v)), \text{ where } e = \{u, v\}.$$

That is, the weight of an edge is the distance between its endpoints in the embedding. There is a standard concept of shortest path length between two vertices, but we would like to extend the idea to "virtual vertices" along an edge. For an edge $e \in E$ and $t \in [0, 1]$, arbitrarily call one endpoint u and the other v, and we imagine the virtual vertex $e(t)$ to be the point along e at time t. We also suppose the existence of corresponding virtual edges $(u, e(t))$ and $(e(t), v)$, weighted as $t \cdot w(e)$, $(1 - t) \cdot w(e)$, respectively. Now let

$$\hat{V} = \{e(t) : e \in E,\, t \in [0, 1]\},$$
$$\hat{E} = \{(u, e(t)) : e = \{u, v\} \in E,\, t \in [0, 1]\}$$

define a graph $\hat{G}$ which includes all the virtual vertices and edges. The extended graph distance function $d_g : \hat{V} \times \hat{V} \to [0, \infty]$ is defined as the length of the shortest path in $\hat{G}$.

We also define an extended mapping $\hat{\psi} : \hat{V} \to \mathcal{C}$ so that $\hat{\psi}(e(t))$ is a linear interpolation between $\psi(e(0))$ and $\psi(e(1))$ at time t for $t \in [0, 1]$ and $e \in E$. Now we may define a second distance function $d_c : \hat{V} \times \hat{V} \to [0, \infty]$ with respect to $\mathcal{C}$ as $d_c(u, v) = \delta(\hat{\psi}(u), \hat{\psi}(v))$ for all $u, v \in \hat{V}$. We refer to d_g as graph distance and d_c as $\mathcal{C}$-space distance.

In this paper, we will consider as a path distance metric the discrete Fréchet distance evaluated at the vertices [16]. For brevity, we refer to it simply as Fréchet distance or d_f. Let p, q be paths with vertex sequences $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$, respectively. The Fréchet distance between p, q is given as $d_f(p, q) = fre_{p,q}(n, m)$ where

$$fre_{p,q}(-1,-1) = 0,$$
$$fre_{p,q}(i,-1) = fre_{p,q}(-1,j) = \infty \text{ for } i,j \geq 0, \text{ and}$$
$$fre_{p,q}(i,j) = \max \begin{cases} d_c(p_i, q_j) \\ \min \begin{cases} fre_{p,q}(i, j-1) \\ fre_{p,q}(i-1, j) \\ fre_{p,q}(i-1, j-1) \end{cases} \end{cases}$$

for all $0 \leq i \leq n, 0 \leq j \leq m$. The computation lends itself to a dynamic programming algorithm.

For a set of paths, P, we define the "diversity" of P as

$$\min_{p_1, p_2 \in P, p_1 \neq p_2} d_f(p_1, p_2).$$

The "robust diversity" of P with respect to d_f is defined as

$$\frac{1}{|P|} \sum_{p_1 \in P} \min_{p_2 \in P, p_1 \neq p_2} d_f(p_1, p_2).$$

Both measurements are useful, as the first gives a worst-case indication of the nearness of paths, while the second is not sensitive to a close pair of paths in an otherwise diverse set.

We also considered two other path distance metrics. The first is Levenshtein edit distance [15], with edits weighted by the distance between corresponding vertices. We empirically determined it to be too generous in a test discussed later. The second is a discrete version of the Hausdorff [14] metric, which is given by

$$d_h(p, q) = \max\{\max_i \min_j d_c(p_i, q_j), \max_j \min_i d_c(p_i, q_j)\}.$$

It may not be apparent from the definition, but the rationale of Fréchet distance is to consider all possible continuous, monotonic parameterizations of the two paths and to take the max-min of the distance between corresponding pairs of points. Hausdorff can be viewed as doing the same, except it does not require that the parameterizations be continuous and monotonic. Thus Fréchet is stricter than Hausdorff. Ultimately, we chose to use the Fréchet-based diversity measure in our experiments, since it is more discriminating.

IV. Algorithms

We present an algorithm to heuristically search for a collection of paths all satisfying the same query in a graph, while seeking to ensure the robustness of the path set against failures in the graph. It does so by simulating obstacles in order to explore diverse regions of the space while still favoring shorter paths. A shortest path search is performed on variants of the original graph designed to encourage deviation from known paths by removing groups of edges near the paths. We are not concerned with finding precisely the next best path to add to our set. This freedom allows us to quickly yield paths that are much more diverse. The pseudocode is given in Algorithm 1. The first four inputs, G, s, g, k, specify the graph, the start and goal vertices, and the requested number of paths for the query, respectively. The final two parameters, b and ρ, called the branching factor and ball radius, tune the performance of the algorithm by specifying (informally) its thoroughness and the size of the simulated obstacles.

In this section we also discuss an existing algorithm in the literature that, with slight alterations, produces a baseline output for the same problem. It is important that we can show to what degree our algorithm outperforms this solution.

Algorithm 1 Diverse Short Paths

KDiverseShort(G, s, g, k, b, ρ)
Input: A graph $G = (V, E)$ having virtual vertex set $\hat{V}$ with distance function $d : \hat{V} \times \hat{V} \to [0, \infty]$; start and goal vertices $s, g \in V$; number of paths requested, $k \geq 1$; branching factor $b \geq 1$; ball radius $\rho > 0$.
Output: A set S of at most k diverse, short paths in G from s to g.

```
1:  U ← EmptyQueue
2:  S ← ∅
3:  p ← ShortestPath(G, s, g)
4:  if p not empty then
5:      enqueue(U, (p, G))
6:      S ← {p}
7:  while U not empty do
8:      (p, G) ← dequeue(U)
9:      for i ← 1, b do
10:         x ← SampleUniform(p)
11:         E' ← {e ∈ E : d(x, e(t)) ≥ ρ ∀t ∈ [0, 1]}
12:         G' ← (V, E')
13:         p' ← ShortestPath(G', s, g)
14:         if p' not empty then
15:             enqueue(U, (p', G'))
16:         if Acceptable(p') then
17:             S ← S ∪ {p'}
18:         if |S| = k then
19:             return S
20: return S
```

A. Graph Modification

Since we want short paths that satisfy the same query from vertex s to vertex g, but that are different from one another, we develop a method to modify the graph, $G = (V, E)$, on which the query is performed to create a new graph $G' = (V, E') \subset G$ such that the shortest path, p', from s to g in G' is different from the shortest path, p, from s to g in G. Then, both p and p' are valid paths from s to g in G. Furthermore, though p' is not the shortest in G, it is the shortest subject to a constraint, namely that it not traverse any edge in $E \backslash E'$.

We look first at the core of our algorithm, in which we create such a G' and p' (lines 10–13). First, we sample $x = e(t) \in \hat{V}$ uniformly from p (line 10). This is done by selecting an edge e from p with probability proportional to $w(e)/w(p)$ where $w(p)$ is the sum of the weights of the

edges comprising p, and by sampling t uniformly from $[0, 1]$. We then select a subset E' from E consisting only of those edges that do not intersect the open ball of radius ρ centered at x, with respect to a distance function d (line 11). Later we will see what happens if we choose $d = d_g$ or $d = d_c$. This open ball can be thought of as a simulated obstacle. We then find the shortest path through the modified graph using an A* search and the triangle inequality with respect to the $\mathcal{C}$-space embedding as the heuristic (lines 12–13). In our implementation, the modified graph G' is represented by the same data structure instance as G and simply uses a modified edge weight function, rather than copy G.

B. Random Avoidance

The process of finding multiple paths using these modified graphs operates on a queue of known path-graph pairs. We begin by initializing this queue and the set of result paths with the shortest path through the unmodified graph G from s to g (lines 1–6). The algorithm enters a loop that repeatedly retrieves the next path and graph from the queue into p and G (line 8). We use them in the graph modification method described above b times to independently produce new path-graph pairs (line 9). Empty paths, which indicate failure of the shortest path query, are thrown out, but the rest are added to the queue (lines 14–15). Each of the remaining paths is considered for addition to the result set subject to some filtering criteria (lines 16–17). This filtration is application specific: It may trivially accept any path, it may reject a path that is too close to a previously accepted path, it may reject a path that is too long, or reject a path failing some other quality check. The loop repeats until the queue is empty or the result set is large enough (lines 7, 18–20).

C. Notes on Parameters

For the sake of scalability, rather than use a constant ρ, which may be difficult to select, we will use the function $\rho(r) = r \cdot w(p)$ for a constant r in $(0, 1]$ and p the shortest path. Intuitively, the larger the value of r, the farther p' must deviate from its ancestor because a larger collection of edges along and near the ancestor path are removed.

When selecting a value for the branching factor b, it is important to understand the trade-off. If the algorithm terminates with fewer than k paths due to exhaustion of the queue, increasing b may increase the number of paths ultimately returned, but only after a much longer execution time. Likewise, decreasing b may decrease the number of paths found before the queue is exhausted. This is why we say b controls the thoroughness of the algorithm. In all our experiments, we set $b = 2$, which we found to give a reasonable balance for our test graphs.

D. Eppstein's k Shortest Paths

For comparison with our algorithm, we examine the k shortest paths algorithm due to Eppstein, with two small modifications. We introduce a filter on the paths that are ultimately returned—the same one our algorithm uses—so that we may compare the quality of outputs subject to some constraint. The second modification is one that allows us not to specify k upfront. Rather, we repeatedly invoke one iteration of the algorithm at a time. This is necessary because we do not know *a priori* how many we need to analyze to find the desired number after filtering. This refactoring makes no effective difference to the execution of the algorithm. We will utilize the filtering to constrain the result set to have a diversity above some threshold. In that case, Eppstein's algorithm will greedily select the shortest paths satisfying the diversity requirement.

V. Theoretical Guarantees

For the following proofs, consider $G = (V, E)$ a finite, undirected, connected graph embedded in a metric space $(\mathcal{C}, \delta)$ via the mapping $\psi : V \to \mathcal{C}$, with edge weight function $w(\{u, v\}) = \delta(\psi(u), \psi(v))$. Suppose that no edge has weight 0, and that every pair of edges intersects in $\mathcal{C}$-space only finitely many times. In an abuse of notation, let

$$e(T) = \{e(t) : t \in T\} \text{ and } \hat{\psi}(X) = \{\hat{\psi}(x) : x \in X\}.$$

Theorem 1. *There exists a $\rho > 0$ such that for each $e_1 \in E$, if t is sampled uniformly from $[0, 1]$ we have $\mathcal{B}(\hat{\psi}(e_1(t)), \rho) \cap \hat{\psi}(e_2) = \varnothing$ for all $e_2 \neq e_1$, with nonzero probability.*

Proof. Let $I \subset \mathcal{C}$ be the set of intersections of all edges, and let $e \in E$ be given. Since I is finite, we may associate with e an interval $(a, b) \subset [0, 1]$ such that $e((a, b)) \cap I = \varnothing$. Let $N(e)$ be the set of the shortest distance between $e((a, b))$ and each of the other edges:

$$N(e) = \{\inf\{d_c(e(s), e'([0, 1])) : s \in (a, b)\} : e' \in E \setminus \{e\}\}.$$

Since E is finite, $N(e)$ is finite, and moreover the set $\mathcal{N} = \bigcup_{e \in E} N(e)$ is finite. Thus $\min \mathcal{N}$ exists and is strictly positive, since otherwise $\hat{\psi}(e((a, b))) \cap I \neq \varnothing$ for some e. Put $0 < \rho < \min \mathcal{N}$. The reader may now verify that for each $e_1 \in E$, there exists an associated subinterval (a, b) of $[0, 1]$ such that $\mathcal{B}(\hat{\psi}(e_1(s)), \rho) \cap \hat{\psi}(e_2([0, 1])) = \varnothing$ for all $s \in (a, b)$ and $e_2 \in E$. Since (a, b) has finite measure, the statement of the theorem holds. □

Theorem 2. *Take $s, g \in V$ to be start and goal vertices, and let $b \geq 1$. There exists a $\rho > 0$ such that for any simple path q in G from s to g, the probability that $q \in \text{KDIVERSESHORT}(G, s, g, k, b, \rho)$ approaches 1 as k approaches the total number of simple paths.*

Proof. Let q, s, g be given. Though $b \geq 1$, we will trace the evaluation of only one branch in the loop at line 9, and when the generated graph G' is enqueued, we will await its reappearance at the dequeue step in line 8. Choose a ρ according to Theorem 1. By the result of that theorem, lines 10-12 generate a new G' that is equal to the old G minus exactly one edge with nonzero probability. Let p be the shortest path satisfying the query in the old G, as in line 8. Note that p must be simple. While $p \neq q$, there exists an edge e in p and not in q, for otherwise p could not satisfy the query.

TABLE I: Graph parameters and statistics.

Name	t	δ	Δ	$\lvert V\rvert$	$\lvert E\rvert$	Mean Degree
grid-like	1.4	.0004	.06	485	2994	6.17320
cubicles1	3.0	.0010	.25	103	414	4.01942
cubicles2	2.0	.0004	.10	242	1180	4.87603
cubicles3	1.1	.0002	.04	1533	7120	4.64449
houston	—	—	—	5848	10393	1.77719

Suppose e is the sole edge removed in each such iteration. After at most $|E| - |q|$ of these iterations, the only edges remaining are those in q, and therefore $p = q$, which is added to the result set S. Since each step occurs with nonzero probability, the final result occurs with nonzero probability. As k approaches the number of simple paths from s to g, the probability of $q \in \text{KDIVERSESHORT}(G, s, g, k, b, \rho)$ approaches 1. □

VI. Evaluation and Application

A. Experiments

We measure the performance of the algorithm in three ways: 1) diversity with respect to the ball radius, 2) diversity with respect to graph density, and 3) execution speed and path length with respect to diversity. We generated the graphs using the SPARS2 algorithm on a 3D environment. We chose SPARS2 because it gives good coverage of the space and allows us to tune the density of the graph. It also has the spanner property, such that between two vertices there exists a short path approximating the shortest possible path within some stretch factor t. The space is SE(3), with a distance metric that sums the Euclidean distance and the quaternion distance. The stretch factor and the dense and sparse delta parameters for SPARS2 that yielded each graph, t, δ, Δ, are given in Table I. In all experiments, we request a set of 10 paths and average the results over 200 runs of our algorithm. In experiments 1 and 2, the filtration step of the algorithm accepts all paths, while in experiment 3 it maintains a minimum diversity by only accepting paths sufficiently far from previously accepted ones.

1) Our first performance measure is to compute the diversity of the path sets for varying values of the radius factor r. We explore two variants of our algorithm, one using graph distance and the other using $\mathcal{C}$-space distance to measure ball radius. We use the "grid-like" environment, a 5×5 array of block obstacles (see Figure 1). By varying the radius factor, we will be able to see how well the algorithm responds to it and whether there is a significant difference in quality of output depending on which distance measure is used.

2) In the second experiment we use three different graphs in the "cubicles" environment of varying density (Figure 3). We run our algorithm and Eppstein's k shortest paths to compare the diversity and robust diversity of the result sets and to determine if the algorithm performs poorly on sparse graphs, which have fewer total paths.

3) Finally, we compare the relative speed of our algorithm and the length of paths it returns with Eppstein's k shortest paths, subject to a minimum diversity constraint. We use the "grid-like" environment as in the first experiment. Since we are interested in length, the start and goal vertices in this environment have been chosen near one another, rather than at opposite ends of the graph, to open the opportunity of finding paths much longer than the shortest.

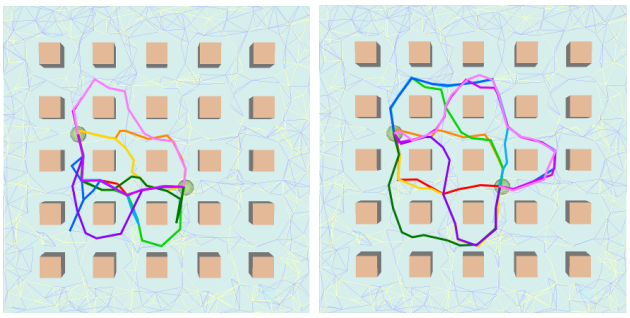


Fig. 1: Comparison of two path sets with the same robust diversity. Left: Eppstein; robust diversity 5.37296; 6 homotopies. Right: Voss; robust diversity 5.37087; 8 homotopies.

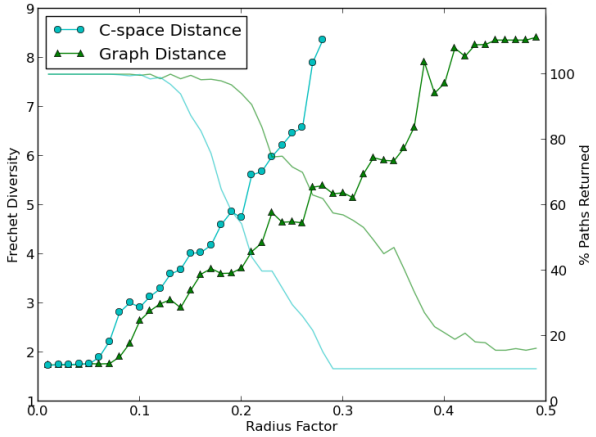


Fig. 2: Comparison of the two distance measures for ball radius. The bold lines show diversity, and the faint lines show the fraction of requested paths that were returned.

B. Results

Before analyzing the results of the three experiments, we will inspect the quality of path sets returned by our algorithm and Eppstein's. Figure 1 shows two such sets, with filtering applied to Eppstein so that the two sets have the same robust diversity. It is apparent that the Eppstein paths wastefully backtrack along edges already traversed, which is not robust against edge failure. Each of our paths is a simple path because it is the shortest on some graph and thus cannot exhibit this deficiency. Despite the two sets having the same robust diversity, our path set represents more homotopy classes. Recall that our diversity is based on Fréchet distance. We performed this same test using weighted Levenshtein edit distance and found Levenshtein to be too generous, allowing the filtered Eppstein paths to exhibit even fewer homotopies. It is difficult to programmatically count the homotopies, and this concept loses meaning in higher dimensions. As such, we do not rely on it as a diversity measure; however, it indicates that our robust diversity definition could be improved.

For experiment 1, Figure 2 shows the parameter sweep of the radius factor using both graph distance and $\mathcal{C}$-space

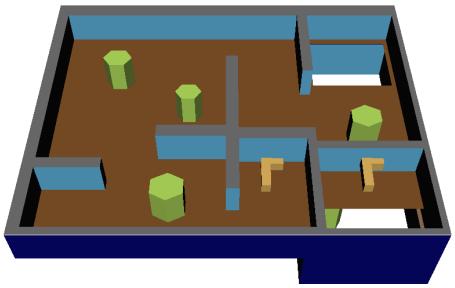


Fig. 3: "Cubicles" environment in which the 'L'-shaped robot must navigate among hexagonal obstacles and through a passage beneath the floor. We generate roadmaps of varying density by tweaking the parameters of SPARS2.

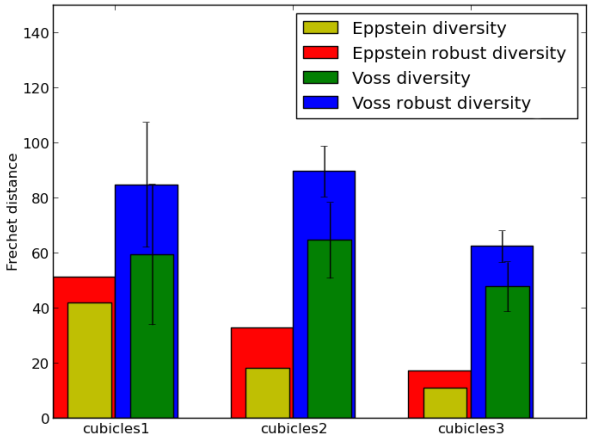


Fig. 4: Diversity of paths for various graph densities.

distance. Observe that the diversity in both cases responds well to an increase in the radius factor. When the algorithm avoids larger regions, the returned paths are naturally farther apart. However, we expect using a larger radius risks covering too much of the graph so that fewer paths than requested can be returned; we see a corresponding increase in the failure rate on the plot for higher values of the radius factor. Note that the graph distance variant consistently requires a larger radius to achieve the same diversity as the $\mathcal{C}$-space variant. This is expected because $d_c \leq d_g$ due to the triangle inequality.

Now we move to experiment 2 and the "cubicles" environment (Figure 3). Of the three graphs in Table I that are embedded in this environment, observe that each one is more dense than the last, with more nodes mapping into the same area, but the average degree remains approximately the same. We choose a fixed radius factor for each graph to be the highest value that still yields an average of 8 out of the 10 paths requested. Looking at the bars for Eppstein's algorithm in Figure 4 we see very predictable behavior: the k shortest paths in a denser graph are much more similar to each other than those in a sparser graph, so diversity decreases with an increase in density. Compare this to our algorithm, which does not exhibit the same consistent decrease in diversity. In fact, the values are higher for the second sparsest graph than for the sparsest graph. We note that the diversity is near to the robust diversity for the k shortest paths, but much lower than the robust diversity for the paths returned by our algorithm. This suggests that paths from our algorithm are typically far apart, though a few may be close. We also observe that the standard deviation tightens as the density increases. Thus, our algorithm performs more consistently on denser graphs.

Finally in experiment 3, we compare the execution time of our algorithm with that of Eppstein's. On the problem of k shortest paths, Eppstein's algorithm executes extremely quickly, but it produces poor results if we are looking for a diverse path set. We apply greedy filtering to keep a path only if it would not bring the diversity of the result set below a minimum threshold. Returning to the "grid-like" environment, we fix our radius factor at 0.1 to yield a moderate diversity naturally. Shown in Figure 5 are the execution times for our algorithm and Eppstein's, both using filtering to meet the diversity requirement. There is an exponential increase

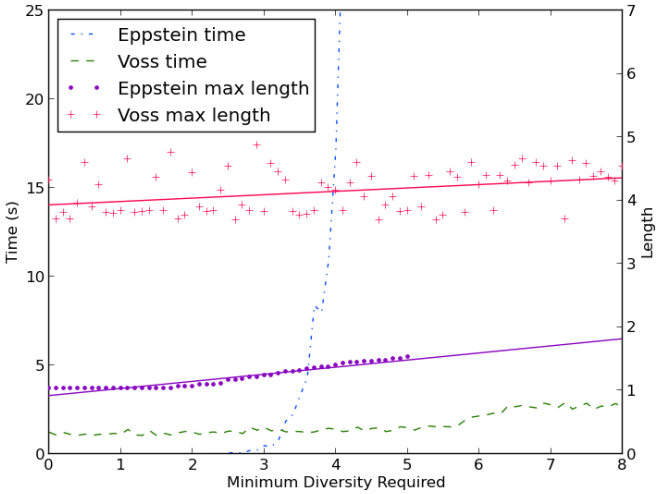


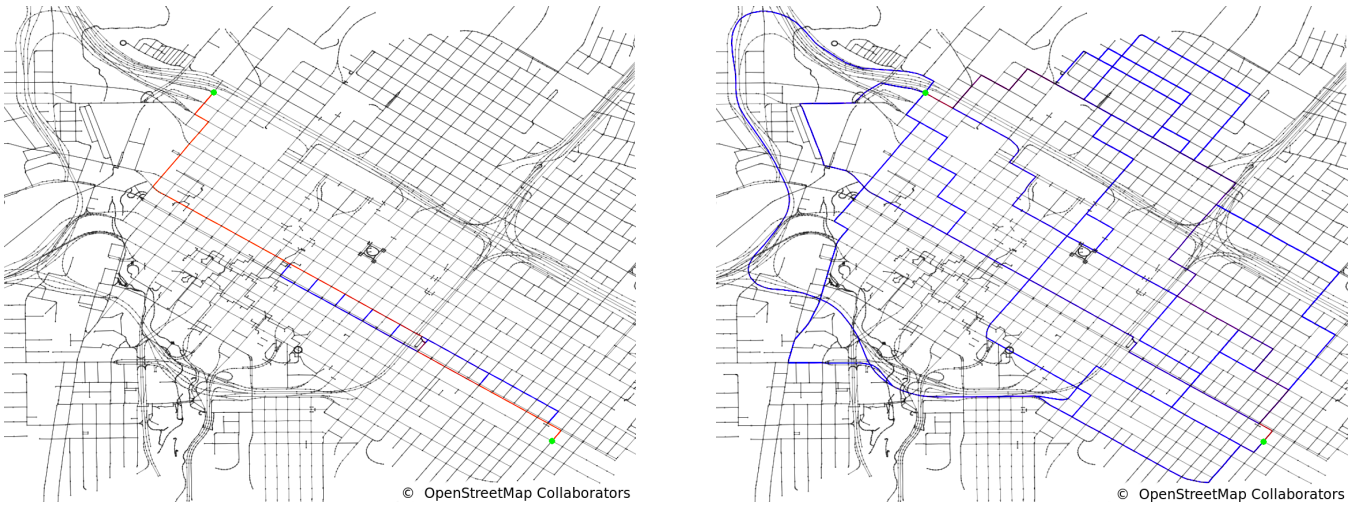
Fig. 5: Speed of algorithms and path length when filtering for minimum diversity. We set $r = 0.1$. Eppstein's algorithm is prohibitively expensive at diversities greater than 5.

in the time required for Eppstein's algorithm to yield the requisite paths, which becomes unmanageable for even modest diversities. Our algorithm, on the other hand, only exhibits a small increase in execution time and not until much higher diversities. Our algorithm is advantageous in that selecting a low branching factor causes it to return early with a partial result in a situation where an unacceptable amount of time would be required to find the full result.

Also depicted in Figure 5 is the length of the longest path returned on any run. We do not have values for Eppstein's lengths at high diversities due to the prohibitive execution time, but linear regression suggests that the lengths from both algorithms respond to diversity at similar rates, though our algorithm begins with longer paths.

C. Scope of Applications

We have seen our algorithm applied to graphs in some simple motion planning problems, like navigating the "grid-like" environment. To illustrate another application, we acquired a graph of the streets in downtown Houston from openstreetmap.org. Included is information about one-way streets and speed limits. We were able to extend our algorithm



(a) Eppstein's 20 shortest paths. Many use the same roads for most of the route.

(b) Voss' 20 diverse, short paths ($r = 0.2$). Very few roads are shared.

Fig. 6: Comparison of paths in downtown Houston. Orange indicates many paths sharing the road; blue indicates few.

to operate on directed graphs. We must be careful now with graph distance over virtual vertices, since a path may only exit an edge through its target vertex. The edge weights are set to the $\mathcal{C}$-space length of the edge divided by its speed limit, so that "short" in this application means "fast." When Eppstein's algorithm is used (Figure 6a), the amount of time multiple paths spend utilizing the same roads is high, so unforeseen traffic can cause wide areas of congestion, affecting too many paths at once. Our algorithm (Figure 6b) finds alternate routes through very different parts of the city, with none slower than 1.5 times the fastest path.

VII. Conclusion and Future Work

Our approach to finding a diverse set of short paths is highly modular. We discussed using graph distance or $\mathcal{C}$-space distance to measure the ball radius. Other distance metrics can be easily substituted here. We looked at three path distance functions, ultimately choosing discrete Fréchet. However, distance between paths is still a debated subject, and other can be appropriate. We hope to find a diversity measure that is better still than the one we used. The filtering criteria are also application dependent, as some problems may have specific requirements for path quality. Future work can explore the benefits of various choices for these configurable parts of the algorithm. We extended the algorithm to use directed graphs with different weights. It remains to try other extensions like varying the ball radius or the branching factor during evaluation. How to choose appropriate values for these two parameters in advance or on the fly are important questions to answer. Future work can also apply the algorithm to higher dimensional problems and systems with differential constraints to determine the advantage of maximizing the use of information from one roadmap which may be very expensive to compute.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, England: Cambridge University Press, 2006.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[3] A. Dobson and K. Bekris, "Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning," *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 432–444, 2013.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005.

[5] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Multiscale Topological Trajectory Classification with Persistent Homology," in *Robot. Sci. Syst.*, 2014.

[6] D. Eppstein, "Finding the k shortest paths," in *35th Annu. Symp. Found. Comput. Sci.*, pp. 154–165, 1994.

[7] H. Aljazzar and S. Leue, "K*: A heuristic search algorithm for finding the k shortest paths," *Artif. Intell.*, vol. 175, no. 18, pp. 2129–2154, 2011.

[8] L. H. Erickson and S. M. Lavalle, "Survivability: Measuring and ensuring path diversity," in *IEEE Int. Conf. Robot. Autom.*, pp. 2068–2073, 2009.

[9] C. J. Green and A. Kelly, "Toward optimal sampling in the space of paths," in *Int. Symp. Robot. Res.*, pp. 171–180, 2007.

[10] J. P. Rohrer, A. Jabbar, and J. P. G. Sterbenz, "Path diversification: A multipath resilience mechanism," in *7th Int. Work. Des. Reliab. Commun. Networks*, pp. 343–351, 2009.

[11] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974.

[12] R. A. Knepper, S. S. Siddhartha, and M. T. Mason, "Toward a deeper understanding of motion alternatives via an equivalence relation on local paths," *Int. J. Rob. Res.*, vol. 31, no. 2, pp. 167–186, 2012.

[13] M. S. Branicky, R. A. Knepper, and J. J. Kuffner, "Path and trajectory diversity: Theory and algorithms," in *IEEE Int. Conf. Robot. Autom.*, pp. 1359–1364, IEEE, 2008.

[14] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Springer, 3rd ed., 2009.

[15] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[16] T. Eiter and H. Mannila, "Computing Discrete Fréchet Distance," tech. rep., Information Systems Dept., Technical University of Vienna, 1994.

[17] M. Kallmann and M. Mataric, "Motion planning using dynamic roadmaps," in *IEEE Int. Conf. Robot. Autom.*, pp. 4399–4404, 2004.

[18] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 5906–5911, 2010.