Finding a Diverse Set of Short Paths through a Roadmap

Caleb Voss

Background

Motion planning for robots is expensive, and there are well-known strategies for constructing a high-level roadmap of an environment which can be used to quickly find paths in areas that have already been explored. An unresolved question is how to effectively use such a roadmap if the environment is dynamic. A well-known algorithm [1] exists to find paths through a graph in order of increasing length; however, these paths typically differ only slightly, so a change in the environment affecting one path would likely invalidate many.

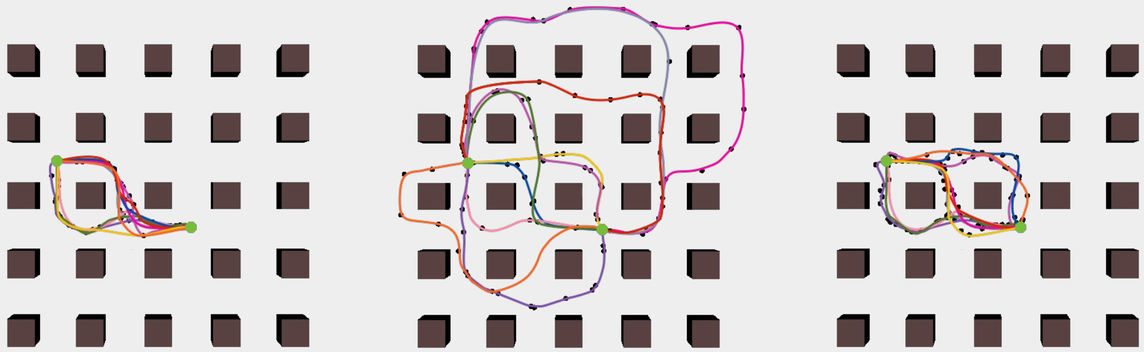
Problem Statement

Given a roadmap of valid motions, (a graph embedded in a state space), and two nodes on the roadmap labeled *start* and *goal*, find k paths from *start* to *goal* such that:

- Each path is short
- Set of paths is diverse (has a large minimum distance between paths)

Algorithm

I propose an algorithm which proceeds by finding the shortest path through each of several subgraphs of the roadmap, formed by successively sampling random regions of the space along known paths to create simulated obstacles so that further paths avoid the area. A pictorial explanation is given below.

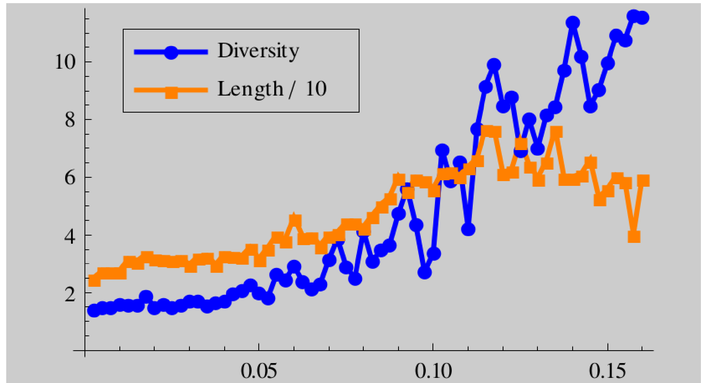


(a) Eppstein's shortest paths (2 homotopy classes)

(b) My Algorithm, radius 0.1075 (10 homotopy classes)

(c) Eppstein, filtered for distance (3 homotopy classes)

Figure 1: Grid environment with the paths highlighted. **a)** Eppstein's paths are short but too similar to each other. **b)** Paths returned by my algorithm are longer, but very different; this set has diversity 10.026. **c)** First 10 shortest paths such that each one added is a distance of at least 10.026 from the others.



Plot 1: Parameter sweep on the radius of simulated obstacles in the grid environment. Diversity of the path set (blue), and length of the set's longest path (orange) are averaged over 10 runs; length scaled to fit on plot.

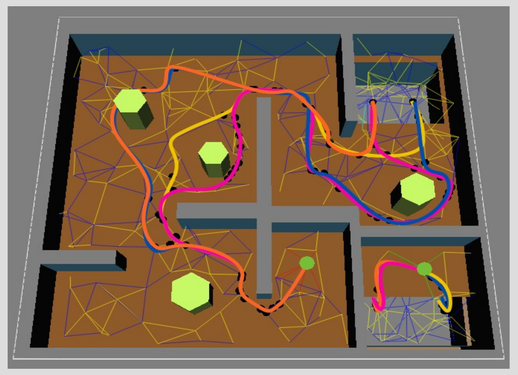
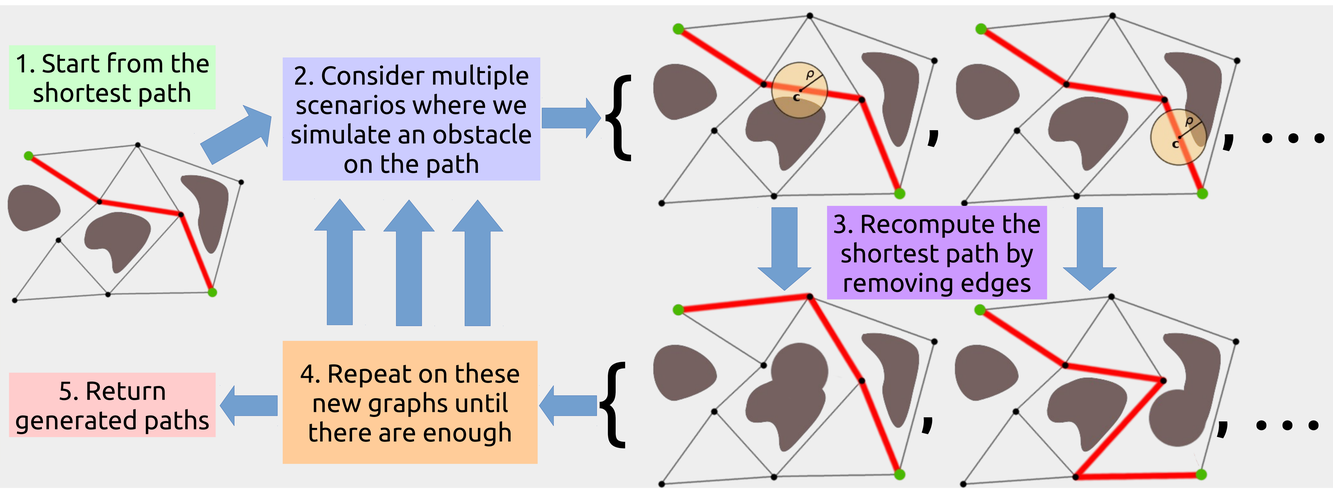


Figure 2: My algorithm run on a more irregular environment than the grid. The paths differ in the direction each takes to pass around the obstacles.

Algorithm 1 Diverse Short Paths

KDiverseShort(G, s, g, k, b, ρ)
input: A graph $G = (V, E)$ embedded in a space $\mathcal{C}$ with distance metric $d_1 : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$; start and goal vertices $s, g \in V$; number of paths requested, $k \geq 0$; branching factor $b \geq 1$; ball radius $\rho > 0$.
output: At most k diverse, short paths in G from s to g.

$U \leftarrow$ *EmptyQueue*
$p \leftarrow$ ShortestPath(G, s, g) // Returns a path p in $\mathcal{C}$
if p **not empty then**
  $U.enqueue((p, \varnothing))$
$S \leftarrow \varnothing$
while $|S| < k$ **and** U **not empty do**
  $(p, A) \leftarrow U.dequeue()$
  $S \leftarrow S \cup \{p\}$
  for $i \leftarrow 1, b$ **do**
    $\mathbf{c} \leftarrow$ SampleUniform(p) // Returns a point $\mathbf{c}$ along p
    $A' \leftarrow A \cup \{\mathbf{x} \in \mathcal{C} : d_1(\mathbf{x}, \mathbf{c}) < \rho\}$
    $E' \leftarrow \{e \in E : e \cap A' = \varnothing\}$
    $p' \leftarrow$ ShortestPath$(G' = (V, E'), s, g)$
    if p' **not empty then**
      $U.enqueue((p', A'))$
return S



Method and Results

I test my algorithm by comparing its performance to the k-shortest paths algorithm due to Eppstein [1], which returns the k shortest paths for a given query.

- Performance measurements:
  - number of distinct homotopy classes
  - diversity of the path set
  - maximum length of paths
- Test environment:
  - uniform grid of obstacles
  - many path homotopy classes possible

An important parameter in our algorithm is the radius of simulated obstacles. I define the radius to be the length of the shortest path times a factor, and perform a parameter sweep on this factor, shown in Plot 1. Observe that:

- Large radii yield diverse paths
- Paths are moderately long for large radii

Figure 1 compares the paths returned by the two algorithms. The 10 shortest paths (a) only represent 2 homotopy classes, while my algorithm (b) found 10 paths, each in a different homotopy class. I then filtered the shortest paths (c) to only include a path if its distance to the others was at least the diversity of set (b), yielding 3 homotopy classes. By construction, sets (b) and (c) have the same diversity, yet the number of homotopy classes suggests that my algorithm naturally produces a much richer set.

Conclusion

The algorithm presented finds a diverse set of paths that are not too long, all satisfying the same query. Because the paths are very diverse, this algorithm has applications in problems where the environment changes after the roadmap is created. If such a change invalidates the shortest path, we hope that some of these alternate paths will not be affected, allowing the continued use of the existing roadmap without the need to recompute it because of a small change in the environment.

This algorithm can be adapted easily to other needs because it is very modular. Possible modifications include:

- Filtering paths according to some criteria
- Using a different path distance metric
- Altering the simulated obstacles

References

[1] D. Eppstein, Finding the k shortest paths, *35th IEEE Symp. Foundations of Comp. Sci.*, Santa Fe, 1994, pp. 154-165.