Schedule

Dates	Objective	Status
3/26 - 4/2	Create path-planning problem with maps. Plan out how we will implement which algorithms; do deeper research on them, our options, opportunities for parallelism, approach that can be taken, etc.	Done
4/2 - 4/9	Complete a prototype of Dijkstra's Algorithm that can be easily modified to try different approaches of exploiting parallelism / different policies that might yield differing results.	Done
4/9 - 4/16	Begin working on getting the best speedup possible for Dijkstra's, and continue working on edge contraction.	In progress
4/16 - 4/19	Finalize edge contraction and begin star contraction, in terms of maximum parallelization speedup.	
4/20 - 4/23	Finalize star contraction, begin contraction hierarchies to assist in speedup for Dijkstra's algorithm. If time allows, begin implementing PRMs.	
4/24 - 4/27	Finalize these algorithms (most importantly, contraction hierarchies), and combine with Dijkstra's. Record results.	
4/28 - 4/29	Final report and poster-board preparation	

Work Completed So Far

So far, we've completed a decent amount of work and, for the most part, we remain on track with our schedule. We finished implementing a sequential version of Dijkstra's algorithm, and we also found an interesting path-planning problem for it to solve by downloading elevation data of the Washington, DC area from the USGS. Because the data is extremely high-resolution, we wrote a Python script to downscale it to a resolution of our choosing (essentially controlling the difficulty of the problem), and we also created a visualizer tool that shows the map and an overlay of the path that was found. Currently, we're working on parallelizing Dijkstra's through a delta stepping approach.

We have also begun implementing star-contraction, and implemented edge-contraction (two graph contraction techniques learned in 15-210), and parallelizing them using OpenMP. While we believe that we may be able to use these algorithms to assist in running Dijkstra's algorithm faster, research on the topic of graph contraction in path-planning indicates that implementing a different contraction technique - contraction hierarchies - should yield the best results.

Our overall plan is to first process the graph and simplify it using parallelized contractions, then run it through a parallelized Dijkstra's algorithm to output plans in significantly less time.

Goals and Deliverables

Based on the feedback that we received on our project proposal, we decided to modify the scope of the project. We're focusing mostly on parallelizing Dijkstra's algorithm, and parallelizing techniques (such as graph contraction) that are used to assist with Dijkstra's algorithm, rather than focusing on other path-planning algorithms that better lend themselves to parallelization that we originally mentioned in our proposal (PRMs, RRTs, etc.). Our goal remains to see what kind of speedup we can achieve from parallelizing Dijkstra's algorithm, considering also speedups to subroutines that can be used to assist in Dijkstra's algorithm, and what tradeoffs emerge in terms of performance vs. quality of path found. With extra time, we may explore PRMs, but we'll be removing RRTs from our project and will only explore PRMs if we feel that we've explored parallelization of Dijkstra's algorithm using several techniques to a deep level.

Poster Session Presentation

During our poster session, we'll do a live demo that shows our planning algorithm using the sequential implementation and our best parallelized implementation.

Preliminary Results

We don't have preliminary results to show because we've mostly completed the sequential versions and the tooling required for our problem. We're currently working through parallelization, including parallelizing the edge contractions and parallelizing Dijkstra's algorithm through delta stepping.

Issues of Concern

In terms of parallelizing Dijkstra's algorithm, there are a few different approaches. We've decided to move forward with a delta-stepping approach, but there could potentially be better options that offer a better speedup. We are hoping that implementing contraction hierarchies for improving Dijkstra's algorithm will not be too much more difficult than the graph contraction techniques that we have already done; it seems that this graph contraction technique is significantly more involved.

PDF

Milestone Report PDF