Pathfinding Algorithms for Mutating Weight Graphs: Research Paper

Haitao Mao

Computer Systems Lab 2007-2008

1 Abstract

Consider a map of an unknown place represented as a graph, where vertices represent landmarks and edges represent connections between landmarks. You have current information on the time it will take to travel between landmarks, as well as an archive about how the travel times changed through the past. You have a preset destination that you want to reach as fast as possible. Pathfinding algorithms for static graphs involve computing the whole path from start to destination, but if the weights are rapidly changing due to some extreme condition of the place, then calculating the whole path in the beginning will not be feasible. The purpose of this project is to design and compare different pathfinding algorithms for a graph whose edge weights mutate randomly to a significant extent. Algorithms may involve probabilistic theory, dynamic programming, heuristics, genetic programming, and variations of standard shortest-path algorithms such as Dijkstra's algorithm.

2 Purpose and Scope

The plan is to create a sturdy algorithm for the general case of the problem, as well as variations for specific cases where the main algorithm would not be as effective. Algorithms will be compared and analyzed to determine the circumstances for which each one is best. This project will involve more theory and design than actual programming.

3 Background Literature

There are little to no studies available concerning graphs with randomly mutating weights, so research has been focused on graph theory in general as well as the more specific topic of dynamic graphs, which may change in structure. General shortest path and flow algorithms have been reviewed. Dynamic graph algorithms and query/update algorithms have been reviewed lightly. The results from this project are expected to be completely new and original.

4 Theory and Algorithms

Define randomized distance as the distance to destination node taking graph structure into account. For example, a vertex with two unit length paths leading to the destination will be closer in this sense than a vertex with only one. We use steady-state convergence and methods from numerical analysis to set up a system of equations we want the randomized distances to satisfy, and solve the system. We use dynamic programming to approximate distance to heuristically closer points first, then base calculations for farther vertices on these approximations. We use the previous states of the graph: we can use this data to develop a hashmap to approximate future mutations. We use genetic programming to find optimal values for algorithm-specific variables. We focus on sparse graphs, graphs where the number of edges is significantly less than the square of the number of vertices. The edge weights are limited to positive doubles so mutation will be somewhat controlled; edge weights that are too large will never be traversed anyway. Complexity will be limited to $O(E^2 \log(E) + V^2 \log(V))$.

5 Testing

The testing interface as well as the algorithms themselves will be written in Java. Since graphs are difficult to develop graphics for, output will be limited to textual lists and charts. Testing will be done by generating graph structures and initial weights and devising a system for the random edge weight mutation. Then, repeated simulations will be run and the algorithms will be scored based on their performance for various types of starting parameters.

First, the algorithms will be tested for functionality and stability by examining its pathfinding in the interface and seeing if it behaves as expected. Then, algorithms will be tested for efficiency through random and user-specified initial states. Algorithms will be compared based on how fast they can find their destination, runtime complexity, and memory usage. If the algorithms take parameters, then genetic algorithms can be used to find optimal values for the parameters.

Second quarter will be devoted solely to building algorithms and developing theory for solving the problem. Testing and variating will not begin until third quarter. The results should be ready by the end of third quarter and fourth quarter will be focused on analyzing and finishing off the project and writing it up.

6 Expected Results

Results will consist of the efficiency, complexity, and stability of the algorithms tested. Results will be presented in charts, graphs, data tables, and qualitative statements. Applications of the results are undetermined as this point, since this is not a commonly trod subfield of graph theory. Robots may be able to apply the algorithms in natural or man-modeled environments. The graph may be able to simulate a transportation network in order to find paths for pioneers. The randomly mutating edge weights may represent an unknown cause of change in an environment, even if there is a systematic pattern to the change. The factor may simply be too complex to model exactly and would be better approximated by a random variable. The project may be useful for applications further into the future, or may spark further development in the area which will lead to results that may be put into practice.