Map Path Finding with Realistic Conditions Using Efficient Search Techniques

Computer Systems Tech Lab 2006-2007
by Olex Ponomarenko

Abstract

An abstract representation of a map will be used to incorporate realistic conditions not currently in place in commercial path finding programs. Along with a visual display, my project could also be used as a tool for educating students on different types of search algorithms, memory efficiency, and runtimes.

The plan is to incorporate more realistic aspects of traffic movement such as traffic light and stop sign delays in order to provide better paths through a map. In the process, a random graph generator will be created, incorporating locations, intersections, and different size roads (ranging from a small residential road to an interstate highway). An efficient heuristic will incorporate these aspects into an A* algorithm or a different graph traversal technique.

The entire project is also designed to be easily testable, flexible, and scalable. All of the parts of the project, from the shell to the final heuristic will be designed and created with a large-scale problem in mind.

Methods and Results

The python language is being used to create the problem and solve it. Java is used to display the graph, in a similar manner to the figure on the right.

A* search is used to determine the fastest path between locations, which means the answer is always the optimal path. The heuristic used will be highly sophisticated and thoroughly tested with the use of a random graph generator.

I'm expecting the program to output nicer, simpler paths than I would expect from conventional speed-limit-only map path finding programs. The program will work for any input, and the random graph generator will produce maps where each point connects to each other point. This will be key to randomizing and optimizing my heuristic and outer program shell.

Background

Navigating a map has been a common Artificial Intelligence problem for a number of years. Sites such as Google Maps and MapQuest are common examples of map traversal. But they often disregard all aspects of traffic besides speed limits, and come up with very complicated paths that would in reality be slower than a simpler path.

Usually such commercial programs simply use speed limits multiplied by the distance of the road to approximate travel time in their heuristics. On smaller roads, this is often wildly different than the actual travel time on those roads. The main cause of this is the failure to consider traffic lights and stop signs (not to mention traffic congestion) by said commercial graph traversing programs. With added factors such as traffic lights and stop signs, the program will avoid most of the flaws of Google Maps and similar programs – complicated series of smaller roads leading to the destination.

Figure

The figure below shows a simple graph of six locations, two intersections, and a highway (thicker line) in between some of the points. This is the type of abstract map representation that I will use in my program.