TJHSST Senior Research Project
Map Path Finding in Realistic Situations
2006-2007

Olex Ponomarenko

January 23, 2007

1 Abstract

The project will deal with maps and path finding. The primary goal of the project would be to use an extension of the A* search algorithm to find the fastest path through a randomly generated graph - an abstract representation of a real life map. The program will include realistic artificial intelligence concepts such as speed limits and street lights. A random graph generator was also be created, allowing a user to generate a graph to select two locations and have the program determine the fastest paths through the map in a smart manner, without using a preset of data. The overall goal of this program is to determine the error that map path finding programs incur if they do not account for

2 Background

This topic has obviously been covered in commercial programs such as google maps and mapquest. Most such programs, however, only consider the maximum speed limit when searching for the fastest possible path. Some of the other important factors of travel speed, such as delays caused by traffic lights and stop signs, are ignored. There isn't a whole lot of research out there on complex graph traversal and path finding - few consider the real world application in maps and travel and instead focus on applications in problem-solving programs and genetic algorithms.

3 Introduction

So the problem we are facing with the commercial programs is that they often ignore delays caused by stop signs and traffic lights on roads. For short distances, this often results in overly-complicated paths that are slower than a longer but simpler path. This project is aimed at creating a primarily back-end program which will incorporate such real-life features and provide funtionality to randomly create and search through maps with such attributes.

The graph generator and the backend of the program were created using the Python computer language. This language is very easy to use, very feature-rich, and fast because it is interpreted and not compiled like Java. The program started out from very basic maps, without speed limits and path costs, and advanced on to more complex ones with added features such as intersections, traffic lights, and stop signs. The solutions were checked manually in the early stages, and later using slower but guaranteed-correct searching techniques, ensuring that the program worked perfectly. In addition, a Java program was also created to display the randomly generated graph and solutions in order to avoid obvious errors and provide visual results. However, the Python code includes all of the features of the program besides display, and the display is not needed to run and interact with the important parts of the code. This results in a program that can be incorporated into bigger projects easily.

4 Structure of the Program

4.1 Random Graph Generator

While it is important that the program considers realistic obstacles when searching for the quickest path through a map, it is just as important that the map that the program is using is believable and realistic. The random graph generator created for the program is fairly effective at achieving both of these requirements. While it does not account for population density and highway structure is not concentric like in real life, the structure makes sense: there are no abrupt ends to larger roads, and there are plenty of smaller roads connecting to the highways.

The graph generator creates a map, comprised of the two auxillary classes

together in a structure of dictionaries (also known as maps). This allows O(1) access time to both connecting locations and the roads that connect to them, allowing the heuristic to perform its function effectively.

4.2 Auxillary Classes

In terms of structure, there are two main component classes to the random graph generator: Location and Road. The Location class is used to represent both locations such as certain buildings and intersections between two or more roads. This allows for flexibility both in terms of searching and optimizing the heuristic. A user can select to go from an intersection to another intersection, rather than simply from one building to another. The heuristic can exploit this similarity between buildings and intersections and provide faster results.

The Road class represents all kinds of roads that are out there. Whether it is a small residential road or a state highway, the Road class is used. Intersections of roads are realistically represented. Highways do not have exits onto small roads, traffic lights are favored to the larger road, and the amount of smaller roads is greater than the number of highways.

4.3 Display

Since the subject matter relies heavily on realism, it is important to have a display to make sure that the end result is indeed a realistic representation of real-life maps and travel on a small scale. Java is used to create this interface, simply because the graphics are somewhat simpler. The most important requirement for the display was that it wasn't mandatory - the Python code can be used by another program, thus avoiding a middleman.

5 Results and Discussion

NOTE: As I do not have my display working, I cannot predict or judge how well my random graph generator will perform on a larger scale. The current functionality is fine, but if I write about small-scale testing, I will 100% guaranteed take it out when I go on to write further drafts and the final paper. There isn't a huge difference at this time between the stereotypical (road weights only) heuristic and my special heuristic that will incorporate

traffic obstacles, so they run at similar time spans, and evaluating something that is not near its final form is a waste of time. Some other projects, those that have already finished their program and started with the interface, may benefit from preliminary evaluation. My project, however, will not have concrete indisputable results until the display is finished, which is why I decided to avoid inputting results into this section.

The addition of traffic obstacles to path finding techniques in maps has a greater effect on small-scale applications such as cities. When it comes to cross-country road trips where one simply drives on the highway for a great majority of the trip, the program will not be very valuable. It is designed to study different road patterns for commuters with several choices for roads. It is valuable in figuring out whether to take the extra couple miles and use a highway versus using a smaller, direct road with a lot of intersections. Further research into real-life traffic patterns and delay caused by different kinds of traffic obstacles would be the obvious next step for the program. The better the approximations for such delays, the better the program will be at pointing the user in the right direction. Another extension could be incorporating this structure into a traffic simulation with a dynamic search, perhaps even showing real-time fastest paths as traffic load changes.