Route Optimization and Google Maps

Will Cranford

Abstract:

The focus of this presentation was the Shortest Path problem as well as the Travelling Salesman problem. I considered suitable algorithms to solve these problems, computational limits on these algorithms, and data-driven approaches to solving these problems. Outside applications were also examined.

Intro:

The inspiration for this presentation was my lack of knowledge of how Google Maps generates the optimal route between point A and point B. This is a topic that is interesting, relatable and contains plenty of math, so it seemed like a good topic to present on.

Theoretical Underpinnings:

Edsger Dijkstra created an algorithm in 1959 which guaranteed a solution to any shortest path problem, which is a problem where you are trying to get from point a to point b in minimal time. Dijkstra's algorithm divided the map into nodes (intersections) and branches (roads). The branches have an associated length, which is a measure of how long it takes to traverse the branch. His algorithm consisted of two steps, repeated until the optimal path had been identified. The algorithm instructions are clear and concise and is one of the most popular shortest path algorithms for those reasons. However, Dijkstra's algorithm does not contain any strategy: it considers each branch as equally likely to lead to the destination in minimal time. For large maps with lots of nodes and branches, Dijkstra's algorithm can be improved upon.

One such algorithm that improves upon Dijkstra's performance is the A* algorithm. This algorithm uses heuristics to solve the problem in a more efficient manner. An example of a heuristic is taking into consideration the vector between the current node and the destination node. So, if your current node is at (x,y) in the coordinate plane, and the destination node is at (a,b), then the vector between the two points would be (a-x, b-y). This vector has a magnitude and a direction. The A* algorithm first considers branches that run close to parallel to this vector, which will help to find the solution in less time.

Real World Complexities:

For the Dijkstra and A* algorithms, we assume that the lengths assigned to each branch are constant. For example, a branch connecting Williamsburg and Richmond would be assigned a length of an hour. But this length is only an estimate of the time it takes to traverse the branch from Williamsburg to Richmond. In reality, each length has an element of stochasticity. Rather than always taking an hour to go from Williamsburg to Richmond, perhaps the length is normally distributed, with a mean of one hour and a variance of 15 minutes. When you factor in randomness, using algorithms such as Dijkstra's seems a lot less feasible.

There are many factors to consider when determining how long a route will take. Along with the distance, there is the traffic, the weather, road conditions, and luck (how many red lights do you face?). Google Maps has to factor all these variables into its methodology. It is for this reason, along with the wealth of data Google has, that I believe Google employs a data driven approach. Google Maps is installed on millions of smartphones, which gives Google millions of data points. Each data point contains the driver's position and velocity. This information tells Google locations of traffic in real time. It also allows Google to make predictions on when and where there will be traffic. Google has such a large database of past queries that it can make an

estimation for your trip length without having to run through a potentially computationally expensive algorithm. I'm almost certain that Google uses machine learning to make predictions on the fastest available route, due to the mountain of data and the feedback mechanism (predicted trip length vs actual trip length).

The Travelling Salesman Problem:

The Shortest Path Problem is useful for when you are looking to get from an origin to a destination as quickly as possible. But what if you are running errands, and you need to figure out the shortest path between multiple destinations? This is where solutions to the Travelling Salesman (TSP) problem are useful. Unfortunately, the solutions to the Travelling Salesman problem are way too computationally expensive to be practical, so approximations have to be made. The best algorithms can get to within 1% of the optimal solution for very large datasets. What I found interesting was the various applications of the TSP. For instance, let's say that you need to drill holes in a circuit board, used in an electronics appliance. It is important to produce these circuit boards as quickly as possible. Thus, the fastest possible route between all the hole positions needs to be known. The TSP can be used, treating each hole position as a node.

There are many other applications of the TSP, including 3D printers, warehouse management, and finding the optimal route when shopping. A whole presentation could be done on the TSP.

Reflection and Self-Evaluation:

I enjoyed the research part of the project. The progression of my research was representative of science and industry, I thought. I first looked at the theory behind the problem, then peeled away assumptions and introduced layers of complexity, then looked at how the

problem was tackled in the real world. The biggest challenge in my research was figuring out what to do with the Dijkstra algorithm. I had to include it, as it is one of the most important algorithms in this area of optimization. I believed that including a short example of the algorithm in my presentation would get the class involved in the presentation. However, Dijkstra's implementation would take up too much of the allotted thirty minutes, so I decided to simplify the algorithm for the class, concerned that I would lose my audience's attention.

When it came time to present, I ended up losing the class. In hindsight, I did not do a good job simplifying the algorithm. Many in the class were confused about how I concluded what the shortest path was, which I did not think would be an issue. An effective remedy to this problem would have been to present this algorithm to a non-math person, to see if they understood it. If I had done this, I could have made changes to my presentation of the algorithm and been more clear. This is definitely a mistake I have learned from.

The rest of my presentation went well. I thought the class enjoyed my explanation of the various applications of the Shortest Path problem and the Travelling Salesman problem, and including a recap of the presentation was a good tool to jog the classes' memory before questions were asked.

When I looked at the classes' comments, they were about what I expected. Many in the class commented on the algorithm and how it was hard to follow. Others liked my recap, and a few liked my choice of topic. I know what I need to do better for next time.