
Project explained: Delivery routing solution

April 19th, 2024

If you owned a delivery business and you needed to determine the best and most efficient routing for the delivery trucks in your fleet, where would you start? The exciting prospect of developing a creative solution to this problem is what led me to develop this very basic JavaScript application. This short blog post explains the app, its logic, and my thinking behind the way I built it.

A quick note about the content on my portfolio site and within this blog: You will notice that I do not host ads, sell anything, or in any way attempt to monetize my content. Unlike many blogs, I do not participate in affiliate marketing, and I get no compensation for anything that I discuss within my blog posts. The things that I mention within this and subsequent blog posts come purely from my personal experience and are mentioned only in the interest of sharing my journey with others. Let me know if you have any questions!

So, what does the program actually do?

Built as a simple web-based JavaScript application, the program functions as a very basic logistics routing optimizer. Users input the location of various points by clicking on a map interface (powered by Leaflet JS with OpenStreetMap tiles) and the application attempts to calculate the most efficient routing solution between the points, given the number of delivery trucks available. A visual depiction of the most efficient solution is then drawn on the map for the user to review.

The logic

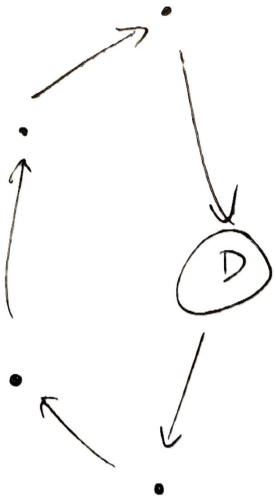
Using simple indexing operations in combination with matrices that are built within arrays, the logic behind this program considers the

great circle distance between the user-defined points and it attempts to sequence them according to both bearing and distance from a defined point of origin. It then attempts to find the most efficent route between the points by comparing the total distance traveled of several different solutions, selecting the solution with the shortest distance traveled as the most efficient choice.

All of the data processing and calculation logic is written in JavaScript (no external libraries or frameworks) and it uses Leaflet JS for map interaction, with OpenStreetMap map graphics.

My thinking behind the application's development

When I sat down to begin building this project, I initially built it to handle the routing of just one truck, routing the truck based only on the distance between the stops. Always beginning at the site of the user-selected delivery depot, the logic would initially route the truck to the stop that happened to be closest in distance to the depot. The truck would then proceed through the rest of the stops, always routed to the next closest location. After the truck was sequenced through each individual stop, it would then proceed directly back to the delivery depot. This was a great place to start, but this simplistic logic would generate solutions that included backtracking along the truck's route, resulting in solutions that were obviously very inefficient. As I thought about the problem, I realized that there was a easy fix to this issue. I sat down and drew the following diagram on a piece of paper:



By taking a step back and thinking about the problem in a more simplistic way, I realized that I would need to write logic that considered both the location of the stops and the distance between each one. To do this, I started writing the routing logic for the routing of two trucks, writing code that accomplished the following, in the following order:

1. Stops are divided by their location relative to the delivery depot.

- Two geographic sectors are defined (either N/S or E/W)
- The sectors are defined such that the stops are divided as equally as possible between them

2. For the stops within the first sector (for N/S sectors, the North sector; for E/W sectors, the West sector):

1. The most directionally-extreme stop is determined (for N/S sectors we use the westernmost point; for E/W sectors we use the northernmost point)
2. The stops within the sector are then sequenced solely based on their distance to each other (as we did in the original design)
3. From the output of the distance-only sequencing, the index (position of) of the most directionally-extreme point is noted
4. Multiple solutions are then generated, positioning the most directionally-extreme point from index zero (the first stop in the sequence) through its natural index (the position it held in the distance-only sequence), using distance-only sequencing both before and after it
5. The total distance traveled within each solution is calculated
6. The solution that results in the shortest distance travelled is output as the most efficient solution for that sector and it is drawn to the map in the user interface

3. Step two is accomplished for the stops within the other sector

Although still a very simplistic approach to what can be a complex problem, the improved logic largely avoids backtracking and it results in much more efficient solutions.

What's next?

I built this program while I was still learning the fundamentals of JavaScript, mainly as a way of applying some of the things I had learned. This is version zero of the app, as the logic has not been developed to calculate solutions for more than two available trucks. In its current state, there is no option for the user to select the number of trucks available and, as such, the logic assumes two trucks. If I decide to continue development of this app in the future, the code will need to be refactored and the user interface will need to be fully developed and styled. I have many ideas for further development of this application, however, I am most interested in exploring machine learning as it could apply this application's route optimization. I am, at the moment, focusing on JavaScript as my learning priority, but I hope to come back this in the future.

Although this project is not hosted on my portfolio site (due to data processing and privacy considerations associated with the use of the OpenStreetMap tiles), the project titled "Delivery Routing Solution" in the "projects" section of this portfolio site links directly to the project's GitHub repository. The repo can also be found at this address:

https://github.com/MikeSchober/delivery-routing-solution

Thank you for reading! Reach out to me with any feedback!

mike@mikeschober.dev