Tesla Mobile Routing Modifications

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Luke Anglin

Spring, 2022

Technical Project Team Members

Aaron Bloomfield, Department of Computer Science

ABSTRACT

Electric vehicle manufacturer Tesla does mobile servicing, sending technicians to customers in need of maintenance. However, the app that the technicians use to get their routes was unable to handle events when the technician did not know where they would be starting or ending their days. Investigating other repositories on GitHub and other companies that have faced this routing logic challenge provided some insight, though it was not perfectly compatible with Tesla's code base. Many companies had solved either the start or the end location being missing, but not cases in which both are missing. To solve this, I modified the cost grid for the routing logic along with changing the route construction logic. Through cost heuristic changes, technicians are now able to schedule their appointments without a start or end location (or both neither). One small issue is that, when both a start and end location are missing, the vehicles routed by the app increase. This means that more vehicles are needed to get the technicians to the designated vehicles about 0.01% of the time, so future work will involve fixing this is the route construction logic.

1. INTRODUCTION

Tesla's mobile servicing starting during COVID. The company leveraged the unique conditions of the pandemic to launch a service that differentiates itself from other car companies. While most car consumers take their car to a shop when something breaks, Tesla offers to come to the consumer at a moment's notice. With the press of a button on an app, the consumer can be serviced. Technicians schedule their routes based on the locations of the consumers in need and the efficiency of the route design.

Occam’s razor says “more things should not be used than are necessary.” Essentially, the simplest approach is the best approach. Forcing technicians to schedule their start and end locations when scheduling their daily routes is unnecessary and confusing. Tesla wanted to circumvent this, and that is where this project came in. The project required a lot of research, comparing various others approaches to Tesla’s, and many changes to Tesla’s underlying route construction logic.

2. RELATED WORKS

1. Google’s OR-Tools for vehicle routing problem (VRP) – this online tool was foundational for building a new algorithm for Tesla without start and end location
2. JSPRIT is a library on GitHub for VRP which gave me inspiration
3. VROOM is another GitHub library which gave me inspiration.

[This section should be in sentence (text) form. For each item : 1) introduce the reference—typically with author surname or company name and year of publication or posting; 2) a brief explanation of the main points of interest in the source; and 3) how it informed your project or process. Re the third source, for instance, what is the source of the reference and year of publication? You state it gave you inspiration but you don’t explain WHAT specifically about it gave you inspiration or how your project reflects that inspiration. EACH of these sources need this kind of presentation and explanation.]

3. CODEBASE DESIGN

This section details the changes I made to the codebase to get this modified algorithm to work.

3.1. PathNode Modifications

The first change necessitated by this assignment was a modification of the representation of the way a Path object was represented in the codebase.

When building routes for the requests sent by technicians, a *Path* object would be created. A Path, in simple terms, is a set of stops, which are called nodes. So, a Path of the route of a UVA student might look like:

Node 1: Apartment - **START**

Node 2: Rice Hall

Node 3: Olsonn Hall

Node 4: Apartment - **END**

Notice node one and two are the start and end nodes. In the original implementation these were required inputs; now, they are not.

Internally, we will still represent the nodes if they are not provided. So if we build a path without a start node, we will represent it as a "dummy" node, as shown below, in order to prevent making extensive changes to the codebase:

Node1 - DUMMY – START

Node 2: Rice Hall

Node 3: Olsonn Hall

Node 4: Apartment - **END**

3.2. CostGrid Modifications

The *CostGrid* is the object that held the costs of going from one place to another. This was largely based on time traveled and the ability to get from that place to the other places in the route.

The CostGrid had to be modified to make any costs from or to dummy nodes equal to zero. By doing this, the route could be properly built.

3.3. Construction Challenges

The biggest change that had to be made was covering the case when neither a start nor an end location was given. In this case, the route construction – one in which the path is built based on the inputs provided by the technician—had to be modified so it would work properly with two dummy nodes. I added conditional logic to check for this case and handle it by randomly initializing the route with two potential location nodes. By doing this, the AI eventually improved the construction and solved the issue.

4. RESULTS

The app now is able to handle technicians who make route requests without a start and end location. Travel time has been reduced in accordance with the lack of a start or end location, as has distance traveled. The organization gains credibility as the app is now significantly easier to use, and technicians are putting in fewer [Less refers to volume; fewer refers to number or quantity.] complaints about a lack of flexibility.

5. CONCLUSION

This project was instrumental in simplifying the process for technicians receiving their routes. After this was implemented, technicians no longer needed to specify their start and end locations. Because this was unnecessary and often unknown, the project successfully simplified the process for them and helped Tesla reduce confusion.

6. FUTURE WORK

Future work will involve speeding up the algorithm. While the algorithm is fast enough, it could be better. There is some overhead and extra checks necessary to be done in order to make this work. Reducing these would be helpful.

REFERENCES

[1] Google. (n.d.). Vehicle routing problem OR-tools google developers. Google. Retrieved September 5, 2022, from https://developers.google.com/optimization/routing/vrp

[2] Graphhopper. (n.d.). Graphhopper/JSPRIT: Jsprit is a Java based, open source toolkit for solving rich vehicle routing problems. GitHub. Retrieved September 5, 2022, from https://github.com/graphhopper/jsprit

[3] VROOM-Project. (n.d.). Vroom-project/vroom: Vehicle routing open-source optimization machine. GitHub. Retrieved September 5, 2022, from https://github.com/VROOM-Project/vroom

[4] Kosiur, D. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.