A close up of a map

Description automatically generated

Data Update and Manipulation on a Web Application for finding the shortest feasible route

for Alternative Fueling Vehicles

by

Jingchao Zhou

A Capstone Proposal Presented in Partial Fulfillment

of the Requirements for the Degree

MAS-GIS

ARIZONA STATE UNIVERSITY

April 2020

Table of Contents

Page

1. SCOPE OF WORK ……………………………………………………………………. 1

Background 1

Purpose 2

Challenges………………………………………………………………………2

1. METHODOLOGY …………………………………………………………………….. 4

Data Collection and Analysis 4

Data Manipulation……………………………………………………………...4

Optional Work…………………………………………………………………6

1. TIMELINE……………………………………………………………………………...7
2. DELIVERABLES………………………………………………………………………8

5 REFERENCES ………………………………………………………………………….9

**Scope of Work**

**Background**

The fuel stations for alternative fueling vehicles are not as adequate as those for the regular vehicles, in term of both the quantity and the support of online mapping tools that help the drive to find feasible routes without running out of the fuels. Finding available stations and planning an efficient route is crucial for the drivers who own the alternative fueling vehicles. To help with the issues addressed, Dr. Michael Kuby and his group created an online mapping tool for the alternative fueling vehicle routing with range and refueling stations in 2014. This tool will help the drivers to find the shortest feasible path from the origin to the destination. The users can put in start and end location, driving range of the vehicle, initial fuel level and travel type in the interface, then the tool will general the feasible shortest path with the stations for refueling. However, the data of the stations and pre-processed distance between all pairs of the station points are not updated since 2014.

A close up of a map

Description automatically generated

Figure 1. An example of the shortest feasible path from Cleveland to Tempe generated by the mapping tool.

**Purpose**

Although the change of the alternative fueling station since 2014, including the hydrogen and GNC (compressed natural gas) station, may not be significant in terms of its quantity, the discrepancy of data will cause inaccuracy of the generated path and stations, which can affect the user experience of the tool. For example, the user might be led to a nonexistent station by the tool because the data of the point may be removed in 2020. Due to the requirement of large computational power, the distance between all pairs of nodes is pre-generated by python programming and google map matrix API. I will also need to regenerate the distance of the nodes because of the update of the station-location data.

Fact: GNC 672-921 Stations

H2 54-47

The capstone project will help to update the station data from 2014 to the current 2020 version. It will help me to apply the data management and programming knowledge I have learned in the MAS-GIS program, consolidate and review the skills of Python and SQL. I will also help to deepen my understanding of the relational database management system by using PostgreSQL and PostGIS extension that applies the SQL query language in a real-world project.

**Challenges**

Google API is not a totally free service; it requires certain credits to run the API tools. A certain number of requests are free, but the project needs us to generate thousands of requests. Building the distance matrix is not a one-time job with a larger number of requests, and it needs to be generated little by little due to the daily limitations for the users. Obviously, we could reduce the number of requests if we filter out the station pairs with Euclidean distance > 400 miles first, rather than requesting driving distances for all pairs first and then filtering out those > 400 miles. With this being considered, a pre-generated dataset of the distance of the nodes must exclude those which have more than 400-mile distance between them.

Understanding the previous project and explore new methods of data anylasis are also very challenging. It is possible to use other services such as Bing map or other mapping services, but the risk is that the route or distance will be slightly different than Google’s. It could cause potential problems if the artificial feasible network would be built from distances from 2 different sources. For instance, if Bing calculated a route as 249 miles, and that segment and its distance is used in an optimal route by our tool, it is possible that when our tool generates the final output and map, it uses Google to calculate that segment, and if Google computes a different shortest-time route it could be 251 miles long for instance. If the user entered a driving range of 250, the input from the distance matrix would show 249 miles and the route would be feasible but the user would only see the output on the screen which could show 251 miles as the length of that leg of the trip and the user would think that our tool is not working properly. Using a consistent routing program throughout should help minimize these kinds of discrepancies that are caused by different sources of computing.

**Methodology**

**Data Collection and Analysis**

The station data of the hydrogen and CNG are from the website of Alternative Fuel Data Center: afdc. energy.gov. After downloading the data of 2020, I will compare the fields of the data and change of values of the relational database, so I can decide what I can change for the previous scripts. The number of points will also affect the performance of the tool because it will require more computing power to run the tools. A potential significant change of the points will also change the data size of the pre-generated distance between all pairs of nodes.

**Data Manipulation**

This step needs to combine the use of python programming, SOL, and google map matrix API. The preprocessed networks generated by google API is the core part of my work this project. The advantage of the Google Maps API approach is that it does not need the preparation of a network dataset compared to other tools. For example, an important step in modeling the OD cost matrix in ArcGIS is to prepare the network dataset including the extraction of data for the study area and defining network settings and attributes. (Wang, Xu, 2011)

The imitations are that an ordinary user without a paid license to Google Maps API Premier is subject to a daily query limit of 2,500 geolocation requests and is likely to experience some ‘hiccups’ in executing the tool. It only gives 300 free trial credits for a regular user, and I need to figure out how much credits are need for generating the distance between the nodes of all stations.

A close up of a map

Description automatically generated

Figure 2. Flowchart of Solution Algorithm. (Kuby, 2014)

**Optional Work**

If there is extra time, I will make some front-end improvement using HTML CSS and JavaScript. Neo4j can be used in database development in the backend as well. In some cases, Neo4j should be considered as an alternative for specific tasks. The two technologies, relational databases and non- relational database, will remain in usage side by side, each with the perfect fit for its own capabilities. Compared to RDBMS, Neo4J as a graph data base management system which is considered as NoSQL, is able to query in real time and may perform faster in some occasions. Because of its data structure based on key, value, nodes, and property, it is much easier to traverse the data than the RDBMS. Neo4J is widely used in real time fraud detection and building social network websites. It is used by companies such as Walmart, Ebay, Airbnb, HP, and Sisco.

Other NoSQL database might be proper for the project, such as CouchDB and MongoDB, but their spatial features are limited. For example, CouDB is relatively slow in performance and MongoDB only support points. On the other hand, Neo4j-Spatial has more spatial advantages that makes it a potential competitor of PostGIS (Wang, Xu, 2011).

Dr. Kuby and Ayan also suggested that I could Tweak the web user interface to be more responsive and mobile friendly using Bootstrap.

A close up of text on a white background

Description automatically generated

**Deliverables:**

|  |  |
| --- | --- |
| **Content** | **Memo** |
| Collection of current Updated data of the CNG and Hydrogen Station in 2020 | From the website <https://afdc.energy.gov> |
| Using google map API to pre-generate the shortest distance between all pairs of the nodes of the stations. | .csv file |
| Python code and SQL query documents. | .py and .txt files |
| A repository on GitHub. | Potential future collaborative work |
| Screen shot of test results. | Two tests for long and short distances. |
| Web Application. | * Prototype for finding shortest feasible route. www.afvrouting.com * Tweak the web user interface to be responsive and mobile friendly using Bootstrap. |
| Report | In MLA format, with instructions and test results for each distance tool. |
| Presentation | August before graduation |

Hi Jingchao,

Ayan and I have discussed it and think the following are a reasonable set of deliverables for a MAS-GIS capstone:

1. Set up a development environment and get the application working on your workstation. Ayan can get you the original script and let you know what version of PHP and Postgres to use.
2. Update the current CNG and H2 stations with the newest set of data and then update the distances with Mike Palmer's Script.
3. Change the distcalc.py file to use ESRI's AGOL geocoder for updating distances. Since ASU already has a license for this we don't have to spend extra cash and this will greatly increase the speed of the distance calculations as we can now batch geocode.
4. Alter the web tool to also use the AGOL geocoder to make sure calculated network distances match up to the geocoder used for the start and end addresses, the user types in.
5. Research other routing services that could be used for #3 and #4, such as neo4j and openroute service. Test one or more of them for speed and consistency.
6. Write instructions and test results for each distance tool.
7. Tweak the web user interface to be responsive and mobile friendly using Bootstrap.

Let us know if you have any questions.

Thanks!

Mike

Use PHP instead of Python.

References

Baas, B. L. P. *Nosql spatial–neo4j versus postgis*. MS thesis. 2012.

Kuby, Michael, et al. "An efficient online mapping tool for finding the shortest feasible path for alternative-fuel vehicles." *International journal of hydrogen energy* 39.32 (2014): 18433-18439.

Melendez M. Transitioning to a hydrogen future: learning from the alternative fuels experience. Golden, CO: National Renewable Energy Laboratory; 2006. Technical Report No. NREL/TP-540e39423.

Wang, Fahui, and Yanqing Xu. "Estimating O–D travel time matrix by Google Maps API: implementation, advantages, and implications." *Annals of GIS* 17.4 (2011): 199-209.

Zandbergen, Paul A. *Python scripting for ArcGIS*. Esri press, 2015.

Load CSV files

Let us see some of the ways Neo4j can read and import CSV files.

1. **LOAD CSV** Cypher command: this command is a great starting point and handles small- to medium-sized data sets (up to 10 million records).
2. **neo4j-admin** bulk import tool: command line tool useful for straightforward loading of large data sets.
3. Kettle import tool: maps and executes steps for the data process flow and works well for very large data sets, especially if developers are already familiar with using this tool.

<https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/all-pairs-shortest-path/>

### **9.4.4. The All Pairs Shortest Path algorithm**

<https://neo4j.com/developer/kb/all-shortest-paths-between-set-of-nodes/>

All shortest paths between a set of nodes

<https://neo4j.com/docs/cypher-manual/current/execution-plans/shortestpath-planning/>

Shortest path planning

Spatial Library

### **4.11.1. distance()**

distance() returns a floating point number representing the geodesic distance between two points in the same Coordinate Reference System (CRS).

* If the points are in the Cartesian CRS (2D or 3D), then the units of the returned distance will be the same as the units of the points, calculated using Pythagoras' theorem.
* If the points are in the WGS-84 CRS (2D), then the units of the returned distance will be meters, based on the haversine formula over a spherical earth approximation.
* If the points are in the WGS-84 CRS (3D), then the units of the returned distance will be meters.
  + The distance is calculated in two steps.
    - First, a haversine formula over a spherical earth is used, at the average height of the two points.
    - To account for the difference in height, Pythagoras' theorem is used, combining the previously calculated spherical distance with the height difference.
  + This formula works well for points close to the earth’s surface; for instance, it is well-suited for calculating the distance of an airplane flight. It is less suitable for greater heights, however, such as when calculating the distance between two satellites.

**Syntax:** distance(point1, point2)

**Returns:**

|  |
| --- |
| A Float. |