INITIAL PLAN – COMPUTING REACHABILITY GRAPH

Winston Ellis
Supervisor: Padraig Corcoran

PROJECT DESCRIPTION

As electric vehicles become more affordable, they will one day be as popular as fossil fuel powered vehicles on the road. Petrol stations already exist but there is no real widespread equivalent for electric vehicles yet. One shortcoming of electric vehicles is that one battery charge will not last as long as a full tank of petrol, therefore charging stations need to be reachable from more locations so found more often on road networks.

Therefore the placement of the charging stations needs to be frequent but have some intelligent placement in order to lower the number of stations due to cost of construction, impractical locations and planning permission implications etc. The reachability of these stations from any location is important so that an individual will always have a station to visit for their long journey trip planning purposes or in case of a battery emergency.

GOAL OF PROJECT

The aim of the project is to process openly available street map data to create a highly connected graph network, representing which road intersections and dead ends can reach other road intersections and dead ends of up to 3Km using a modified Dijkstra's algorithm. This graph, which is the reachability graph will then have a greedy algorithm applied which will select the nodes with degrees of highest connectivity, that will be used to suggest the optimal locations for electric charging stations. I will cover each step and processes to be used below in more detail.

DATA EXTRACTION

The data that I need is provided by the Open Street Maps, this is an openly available resource of detailed map data[1]. The only pieces of data from the street network data that I need are the intersections, dead ends and the length of the streets. This process will be handled by a python library called OSMnx, which can pull the data from the OSM servers and strip the unnecessary attributes that I don't need and save in a graphML format which is widely used for storing graph representations. OSMnx is also useful for visualising the graphs that are produced from the OSM database and the reachability graphs that I will produce for testing.

COMPUTING REACHABILITY GRAPH

To compute the reachability graph, I will need to modify Dijkstras algorithm, which is similar to the uniform cost search algorithm where it will visit neighbouring nodes updating the path cost to each node depending on the edge weights. For example, after assigning each unvisited node a cost of positive infinity, It will visit the initial node A and look at its neighbours to find the distance from A to each neighbouring node by edge weighting. It will update each nodes distance value usually from positive infinity and pick the node with the shortest distance from A to travel to next. Some nodes will be neighbours to multiple nodes so may have its value lowered further, hence the update being usually from positive infinity. As each node is visited, it is removed from the unvisited set and will never be visited again. This will continue until the goal node is found or until all nodes are visited, depending on how you want to use the algorithm. [2]

For the purposes of this project, I do not have one goal node. Instead I theoretically have many goal nodes, every node that is reachable from the initial node up to the given distance threshold is a goal node. That means as the algorithm goes through all the unvisited nodes, it adds each one to a set of reachable nodes which can all be called goal nodes. The algorithm should then stop when all the unvisited nodes are above the given distance threshold. The algorithm does not need to explore the rest of the nodes as they will not be reachable, unlike the traditional Dijkstra's algorithm.

The set of reachable nodes can then be added to the original graph as new edges from the initial node to all the reachable nodes. This must then be calculated for all the nodes in the graph to compute the full reachability graph.

COMPUTING OPTIMAL CHARGING STATION LOCATIONS

Once the reachability graph has been computed, the nodes with the highest degree of connectivity are said to be the most reachable from other nodes. These are the most important nodes as they are more likely to be a good spot for a charging station. However just choosing the most connected nodes will not give a satisfactory solution as they aren't likely to be distributed evenly across a street network.

This problem is solved by finding the dominating set of a graph. This is simply stating that all nodes are either part of the subset of this dominating subset or is a neighbour to a node in this subset[3]. That means that no nodes can be a neighbour of a neighbour or even more disconnected than that. (Definition 7.1 in reference 3).

This problem is considered to be an NP-hard, due to the large amounts of possible combinations of nodes in a dominating set growing exponentially as you add more nodes and edges to the original graph. To solve this kind of problem, you need approximation algorithms. The definitive algorithm to be used will still be researched during the project in order to find which is the most suitable for the type of graph that I produce.

Once I have a dominating set calculated, I can then suggest the optimal location for charging stations for a given street network.

TIMELINE

For the 12 weeks that I have been given to complete this project, I have a rough timeline of milestones that I wish to achieve. This has been planned out to give myself reasonable time to research the tools that I will be using, to then test out implementations on a simplified task and reduced data sets, to finally implementing the functional implementation on the real data set. Depending on the success of the functional implementation, I will hopefully even have time to optimise the implementation. Due to the nature of this project and the large number of individual pieces of data that must be processed in a worthwhile street network size, optimisations will go a long way in reducing the times for testing on the full size street networks.

I will give myself a buffer at the end of the implementation timeline in order to be used as extra time to complete work that I have struggled with or hopefully to implement further ideas and functionality to the project. I have listed the potential further work that I have thought of so far at the end of the plan.

Work Plan

Week 1 – 23rd Jan

- Complete Initial Plan
- Consider potential modifications to Dijkstras algorithm

Week 2 – 30th Jan

- Obtain Open Street Map data
- Familiarise with OSMnx library
- Use OSMnx library to process data and to understand the data structure
    - Think about ways I can modify the data structure to ease use and visualisation for later in the project
- Supervisor meeting to discuss libraries and research start points

Week 3 – 6th Feb

- Familiarise with graph handling libraries in python, choose suitable library
- Research Dijkstras algorithm
- Formalise suitable modifications to fit project criteria
- Document findings

Week 4 – 13th Feb

- Research into dominating sets and respective algorithms
- Small scale testing on proposed algorithm using test data
- Familiarise with using more complex graphs with code
- Supervisor meeting to discuss if research is going in right direction

Week 5 – 20th Feb

- Small scale testing on simplified Dijkstra's problem with test data
- Implement fully modified algorithm using proposed changes on full dataset to produce reachability graph
- Document work

Week 6 – 27th Feb

- Find suitable way to visualise the reachability graph

- Start creating program to contain all the separate modules of project into one program for Viva demo
- Supervisor meeting to discuss progress and visualisation

Week 7 – 6th March

- Implement full scale algorithm on reachability graph to produce dominating set of nodes to represent electric charging stations
- Document work

Week 8 – 13th March

- Find suitable way to visualise the dominating set graph
- Merge algorithm and visualisation into the single program

Week 9 – 20th March

- Assess if the data output is suitable
- Supervisor Meeting to discuss validity of results
- Start analysis and report writing

Week 10 – 27th March

- Focus on report writing
- Some attention on making sure all code implementation works under the single encapsulating program if needed

Week 11 – 3rd April

- Supervisor meeting to discuss direction of report
- Work on report
- Assess if there is time to implement further work

Easter Break 10th April – 30th April

- Buffer period used to either catch up, optimise implementations or start adding extra functionality
- Work on report

Week 12 – 1st May

- Finalise report for submission

References

1. Open Street Map data info https://www.openstreetmap.org/about
2. Dijkstras algorithm basis https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html
3. Dominating set http://www.inf.usi.ch/faculty/kuhn/teaching/netalg/lectures/chapter7.pdf