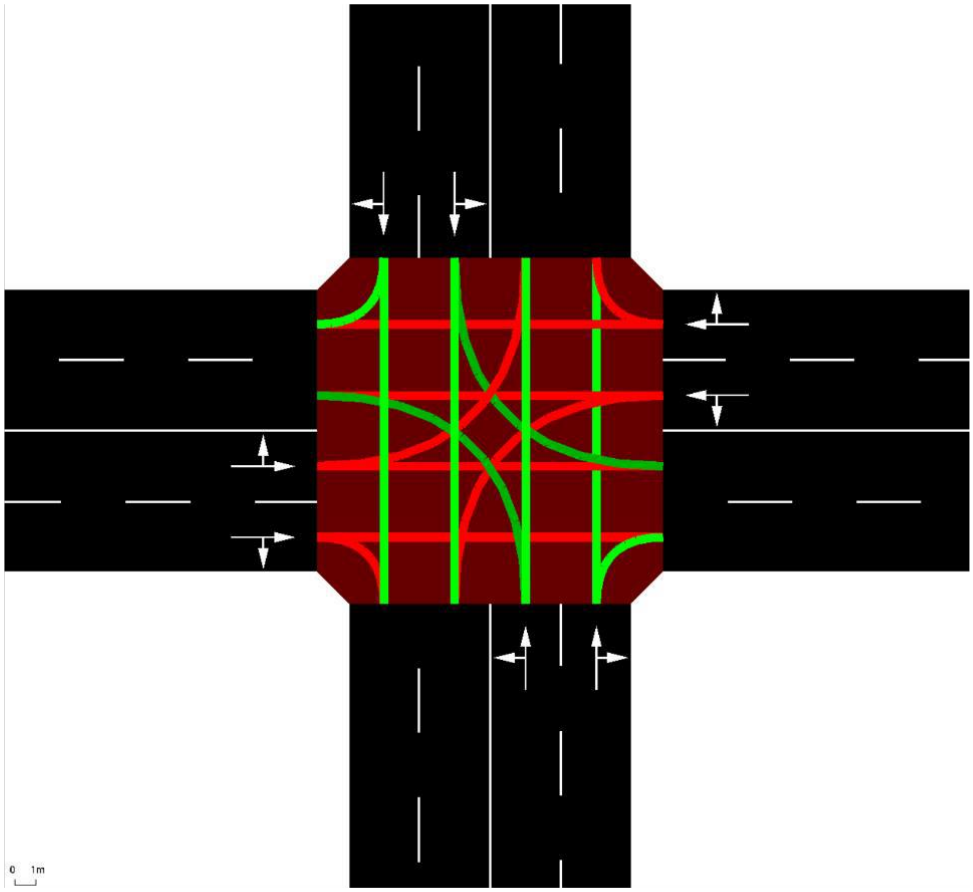


Efficient Detour Computation Scheme for Electric Vehicles

PRODUCED BY

FAISAL KABEER

SUPERVISED BY

SOUFINE DJAHEL

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed F.Kabeer

Abstract:

This report is going to research the various methods being used to control an electric vehicle on the roads of smart cities.

Vehicles on the road are constantly facing new challenges in major cities across the world as urban areas become more populated resulting in an increased number of vehicles on the road. The current solution of on-board navigation systems for individual vehicles provides the user with the ability to pre-plan their route, finding the most convenient route to take to achieve reaching their destination. This method however does not accommodate for future traffic changes as it relies on the current traffic state and sometimes paired with previous traffic trends. Thus, resulting in being unable to provide a more efficient alternative route for an electric vehicle quickly enough if an unexpected congestion occurs. This is especially challenging for electrical vehicles as their routing decisions can suffer from further constraints. As a solution to the above problem, this project will be focused upon designing a heuristic algorithm that can provide an efficient detour route for electric vehicle.

Contents

1. Introduction

The aim of this project is to design a heuristic algorithm that provides electric vehicles with an efficient detour route, that at minimum is similar to timings set by other variations of the Dijkstra and Dynamic A* algorithm. And To evaluate the performance of the algorithm using SUMO by importing a model of a grid as the testing base.

2. Literature Review

In this chapter I will be introducing the topic of smart cities and how electric vehicles can use charging times efficiently with routing and detour computation algorithms to produce an efficient navigation system.

2.1. Smart city

Figure 1. An image of a Smart City [23]

The basis of a smart city can be seen as developing an Information systems infrastructure to consistently improve the quality of life for the citizens. Examples can be seen in areas such as transport, parking systems, air quality monitoring, traffic congestion and city energy consumption [1]. In traffic related principles of a smart city, the data collected from the vehicles can be fed back to the driver [2].

One source being used to gather real time traffic data are static sensors. e.g. video cameras and induction loops. This equipment provides an estimation into the number of vehicles on the road, however this equipment can be costly. Smart cities have instead implemented crowd sensing technology to gather accurate and vast amounts of reliable information on traffic affairs. This information is produced by smart phones and tablets (e.g. Android devices) of electric vehicle owners. This data is used to produce a large amount of data over a mass amount of traffic [3].

This data helps producing accurate representations of current traffic and allows electric vehicle owners to gain significantly faster planned routes and en-route trips, than if the data had been limited to static sensors.

Along with collection of data, smart cities are also able to facilitate electric vehicle owners charging requirements. Public spaces in an urban environment will alleviate some of the problems faced by electric vehicle owners, such as range anxiety, since charging points can be deployed in areas where electric vehicle charging station criteria are met.

Electric Vehicle Charging Station Criteria referred to as the environment of where the EV Charging station will be located. Along with also making sure that the economic and social criteria are also met when deciding an area to place an electric vehicle charging station [4]. Smart cities will be able to fully optimise the unique locations for electric vehicle charging stations to create an effective incorporated ecosystem, allowing electric vehicle owners to charge their vehicles and continue with their trip. Furthermore, providing EV with the ability to plan their routes to take into consideration the number of charging stations en-route.

EV are the future to traffic in many countries going forward, this is to aid efforts in producing more environmentally friendly traffic, so that greenhouse gases production is reduced. Many major countries have confirmed that there are plans to ban petrol and diesel based vehicles as soon as 2040 (e.g. France, UK) with other countries announcing a ban will be put in place yet not providing a firm date to which they will implement the ban. This shift from combustion engines are being made to aid reaching climate change obligations [5,6]. However, EV is still a relatively new market, with issues such as range and charging times limiting the potential of EV being the main power drives of the near future.

2.2. Electric Vehicles

Figure 2. An electric vehicle (Tesla model 60d) [24]

This project will only be focused on Pure Electric Vehicles (PEV), as the software being used (SUMO DLR) can only incorporate PEV or the alternative fossil fuelled vehicles. EV in comparison to fossil fuelled vehicles have several setbacks in becoming a prevalent alternative, notably electric vehicles are yet to produce batteries that are equivalent in charging time as refuelling is within an internal combustion engine [7].

However, since electric vehicles are powered through electric motors, rather than carbon dioxide being produced from an internal combustion engine, EV's can be deemed a credible alternative. Thus, further research and development investments have been put in place by several vehicle manufacturers such as Volvo who recently announced that following 2019, each vehicle will have an electric motor [8], thus indicating that electric vehicles are being considered as a dependable mode of transport over fossil fuelled vehicles.

EV have several different charging types and modes, as can be seen from the table below. From Figure 3 you can see that level 1 charging is the slowest method of charging and is what will be used to trickle charge a vehicle when stopped over long durations e.g. home.

Level 2 is a charging method that may regularly be seen in private as well as public outlets, as it can be used to charge an EV in under 6 hours. Making it ideal for businesses.

Level 3 charging is best suited for motorway rest stops or public areas within a city, where the charging station can be readily accessible. However, this charging level is rarely available in residential areas, due to the high cost associated with the infrastructure required to build and sustain the charging station [9].

Power Level Types	Typical Use	Charging Time
Level 1 (Opportunity) 120 Vac (US) 230 Vac (EU)	Charging at home or office	4–11 hours 11–36 hours
Level 2 (Primary) 240 Vac (US) 400 Vac (EU)	Charging at private or public outlets	1–4 hours 2–6 hours 2–3 hours
Level 3 (Fast) (208-600 Vac)	Commercial, analogous to a filling station	0.4–1 hour 0.2–0.5 hour

Figure 3. Description of charging times for electric vehicle [9]

2.3. Routing and Detour computation in Electric Vehicles

Routing algorithms for trip planning requires a strain of a dynamic algorithm.

Electric vehicles are susceptible to various issues with common routing algorithms. Current routing algorithms are focused on finding the shortest path to the destination, whether that equates to the shortest amount of time to reach a destination or the least amount of distance travelled. However electrical vehicles require further constraints to make the route optimised towards the ability of the electrical vehicle. Factors such as regenerative braking and the ability for an EV reaching a charging station should be considered when implementing a rerouting algorithm [10].

Shortest path algorithms can be used within routing planning. e.g. Dijkstra algorithm and A*. These provide quick solutions from starting point to the destination point. The shortest path algorithm is a static type of algorithm, where the total value of weights, between two nodes will always be the same. This makes using the algorithm within real life traffic scenarios an issue, due to the algorithms inability to dynamically react to future events within traffic [11]. e.g. rush hour, thus traffic scenarios require an algorithm that can dynamically update edge weights along nodes. Therefore, traffic routing is prominently made up of strains of dynamic routing algorithms.

Dynamic routing algorithms are optimal for use in vehicle routing. They provide vehicles with the ability to have an efficient route, since rerouting can occur in real time with dynamic routing algorithms [11].

This can be combined with crowd sourcing techniques to produce a real-time image of current traffic; thus, dynamic rerouting algorithms can make efficient calculations for trips in real world situations [2]. Within this project, the routing algorithm will have to factor various factors of current traffic information alongside having to factor the closest EV charging station and the vehicles current charge and mileage.

2.4. Related Works

In this section, I will be evaluating studies that have proposed detour computation solutions for electric vehicles (EV).

2.4.1. Comprehensive Performance Analysis and Comparison of Vehicles Routing Algorithms in Smart Cities [11]

This research study was published by the IEEE, the IEEE is the largest technical professional organisation, aiming to be at the forefront of technology that benefits humanity [12]. The study conducted a performance analysis and comparison of typical routing algorithms [11].

The paper produced a general classification into vehicle routing algorithms, seen in Figure 4. This classifies the vehicle routing algorithms as either being static or dynamic. Static being an algorithm where the total sum of weights between two nodes will always be the same, as static algorithms fail to update edge weights in real time. The paper summarised the best vehicle routing algorithms available at the time the paper was written, providing reasoning to the four algorithms being used in their experiments.

The static algorithm that is used within the research study is the A* [14] algorithm, as it is the improved version of Dijkstras' [13] algorithm. Due to A* distinguishing the Euclidean Distance as the lower bound ensuring it would never overestimate the real travel distance between the origin and the destination nodes.

In terms of dynamic vehicle routing algorithms, there are a variety of significant dynamic routing algorithms that this paper describes. One dynamic routing algorithm that this paper describes is Dynamic Adaptation A* with mixed lower bounds (DAA*_M), proposed in 2002 by I. Chabini and S. Lan [15]. DAA*_M is a pinnacle of dynamic vehicular routing algorithms since it is the first theoretical and experimental contribution of this research field. DAA*_M essentially improves on the lower bound within the current time interval, dependant of the shortest path choice in the previous time interval.

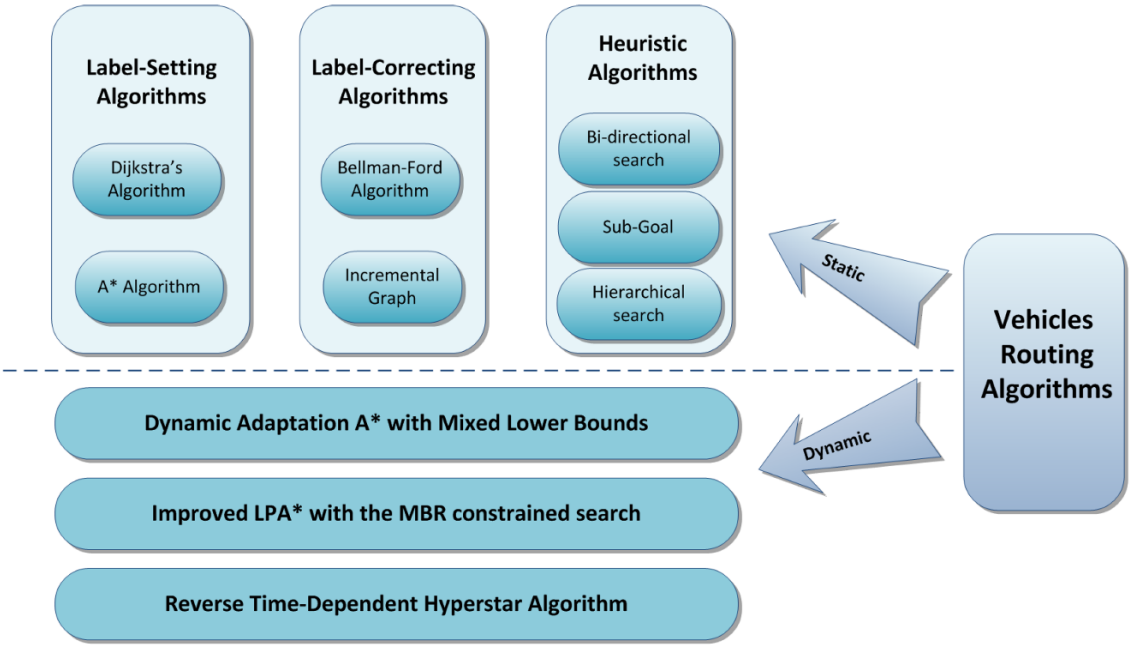


Figure 4. The classification of vehicles routing algorithms [11]

The latest significant dynamic vehicle routing algorithm at the time the paper was written is Hyperstar, proposed by M.G.H. Bell et al. [16]. The algorithm is based upon a combination of the static A* algorithm [14] and an established transit planning algorithm put forward by Spiess and Florian [17]. Hyperstar describes a solution to calculating an efficient and dependable route. It performs through connecting pair of origin-destination nodes, avoiding routes that contain repeated congestion. Thus, calculating several origin-destination routes in comparison to a single efficient route, this provide several reliable alternative routes if one becomes highly congested with unpredicted traffic occurrences. This result is favourable as being able to adjust to random occurrences while en-route provides drivers with the most efficient and reliable route available.

The comparative study chose to use two static and two dynamic algorithms, which consisted of the static algorithms, Dijkstras' algorithm and static A*. The dynamic algorithms being used were Dynamic A* alongside dynamic Dijkstras' Algorithm. The comparative study used an open source microscopic urban transportation simulator, SUMO as a form of simulating a realistic real-life traffic simulation.

The comparative study concluded that static Dijkstras' algorithms can be recommended for the shortest route, for use within centralised intelligent transportation systems within isolated areas. This was due to the simplicity and time taken to compute a relatively good performance in terms of computation time compared to the other algorithms. Also finding that within centre and suburban

areas that static A* can be useful for trips equating to more than 5 kilometres, but if the trips are less than 5 kilometres Dijkstras' algorithm is a better alternative.

In terms of dynamic algorithms, the researcher concluded that it would be highly recommended for the fastest route enquires, because of the quality of the produced route along with a low computation time when having to calculate longer trips.

This study produced valuable data on how the four algorithms react to different scenarios in terms of computation time to a trip length, providing a thorough demonstration into which algorithm would best work for a trip will be dependent upon the area and the complexity of the routes the vehicle will be in. However, this paper only provides 4 proficient algorithms, failing to include other more recent algorithms at the time the paper was written such as Hyperstar [16] algorithm, this is not a major disadvantage though since the algorithms that were used are tried and tested methods used within commercial navigation systems and will be used within this project.

2.4.2. Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion [18]

This paper is another, that was published by the IEEE, the researchers of this study proposed an original vehicle rerouting system, they named the system Next Road Rerouting (NRR). It aimed to aid drivers navigate through traffic by creating a heuristic rerouting decision that would avoid any congestion.

The research states that Vehicle Navigation Systems (VNS) (i.e. google maps, Waze and TomTom [21]) all produce the quickest route to a destination, with some of the services being able to calculate trips with historical data of traffic and even being able to implement prediction models. Although these features are implemented within a VNS, there are several drawbacks to it use within a smart city. It infrequently updates traffic (every 2 minutes or greater than) making the data slightly offset to what would be currently occurring on the road. Another drawback is that VNS fails to consider other users that are using the same method of routing. This can cause secondary congestion as the vehicles within the congested areas will take the same route out of the congestion to avoid the disruption of traffic with the first disturbance.

The research paper defined various fundamental concepts that is used for the description of NRR. One such fundamental concept is defining a road segment and road. They defined a road segment as what is used to join two neighbouring intersections. Following this a road is defined as a unidirectional part of the neighbouring intersection, roads may be divided into multiple lanes for traffic.

Travel time index is used to measure urban traffic congestion level. Being calculated by the ratio of summing travel time to the sum to all the free-flow travel times of all vehicles. Thus, providing a measure to the increase to travel time from what is the ideal.

NRR is used to detour vehicles affected by several en-route factors. The research paper used SUMO to simulate a realistic real-life traffic simulation in which they could evaluate NRR's performance. The researchers found that NRR is able to improve average trip times of vehicles by up to 38.2%. Signifying that VNS when overused can produce unwanted results in terms of travel time.

The study provides a strong insight into implementing a smart altruistic rerouting strategies over a static vehicle detour routing system since findings indicate that trip time can increase for vehicles when their routing strategies are purely based off a VNS. These findings indicate that each vehicle will require dynamic vehicle navigation routing to produce an efficient navigation system.

2.4.3. Electric Vehicle Route Selection and Charging Navigation Strategy based on Crowd Sensing [3]

This paper proposed a novel EV route selection algorithm that aimed at increasing the EV's travel efficiency, making factors such as time travelled more efficient for the EV.

The study describes how crowd sensing technology can be used to gather large amounts of data within traffic through smart phones (i.e. Android, iPhone) used as sensors. This method produces a large amount of data within a vast area in comparison to static sensors which are limited to the area they were set up in. A visual representation of this can be seen in Figure 5.

The research states that traffic data, such as the values of traffic volume, vehicle speed and traffic density can play a crucial role with navigation traffic routing systems. But they also argue that having outdated information in terms of traffic data will be unable to reflect upon real time events. Therefore, generating inaccurate routes for drivers, making it strenuous for the driver as the routes will be inefficient in comparison to a dynamic navigation system [19].

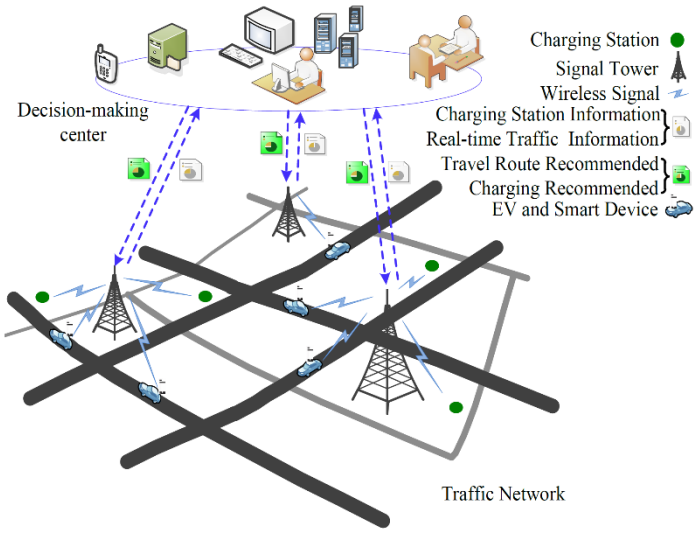


Figure 5. A diagram illustrating how crowd sensing could be implemented [3]

Their testing simulation is a road network within Shenzhen, Guangdong province, China, along with implementing their simulation within MATLAB software. This allowed the researchers to produce a comparison into EV routing, with and without the use of crowd sensing.

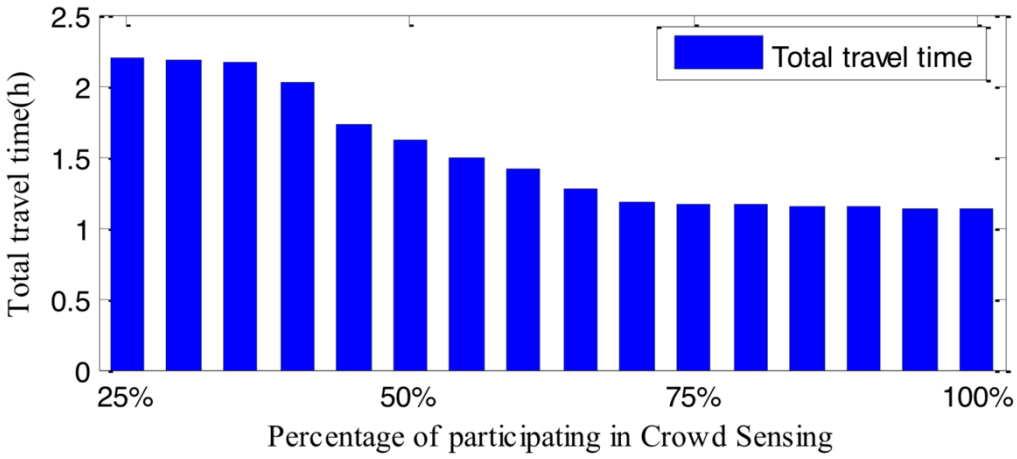


Figure 6. Percentage of Total travel time (hours) / Crowd Sensing [3]:

The study found that by uploading a greater amount of traffic information collected from crowd sensing higher reduction in travel time would be achieved. However, the study highlighted also that factors such as the number of EV's on the road and the length of the route being taken will affect the significance of crowd sensing's impact on EVs' routing. With 70% of vehicles using crowd sensing being the optimal amount as the difference diminishes over this value, as can be seen in figure 6. These results signify the importance of how crowd sensing should be implemented in a city, indicating that under their testing environment that there has to be a strong percentage of drivers participating in crowd sensing to reach an ideal efficiency in travel time.

2.4.4. A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches [20]

This research paper is published through the IEEE, in the first quarter of 2015. The research survey describes a summary of various technologies involved in a Traffic Management System (TMS), along with discussing how social media could be used for accurate detection of traffic congestion and reduction of traffic. TMS is a term that is used to describe the technology that will be deployed to aid combating traffic congestion from occurring and relieving any congestion that may occur.

Although Cities have implemented various solutions to attempt to reduce traffic congestion, they are still suffering from two major types of traffic congestion. The research distinguishes the two separate major types of traffic congestion as recurrent and non-recurrent. Recurrent congestion occurs while a large number of vehicles attempt to simultaneously use a limited amount of space on a road network (e.g. weekday mornings peak hours). Non-Recurrent congestion is defined as occurring through random events such as traffic incidents (e.g. car crash), road works and sporting events.

The most significant phase of a TMS is the data gathering stage. This is where traffic events can be gathered in real time and processed through the data fusion, processing and aggregation phase so that the data collected from the first phase can be extracted into useful traffic information within the second phase. The third phase uses the extracted information to construct efficient routes that fit the scenario of the end user. Service delivery is the phase where the constructed optimised route is delivered this to the end user.

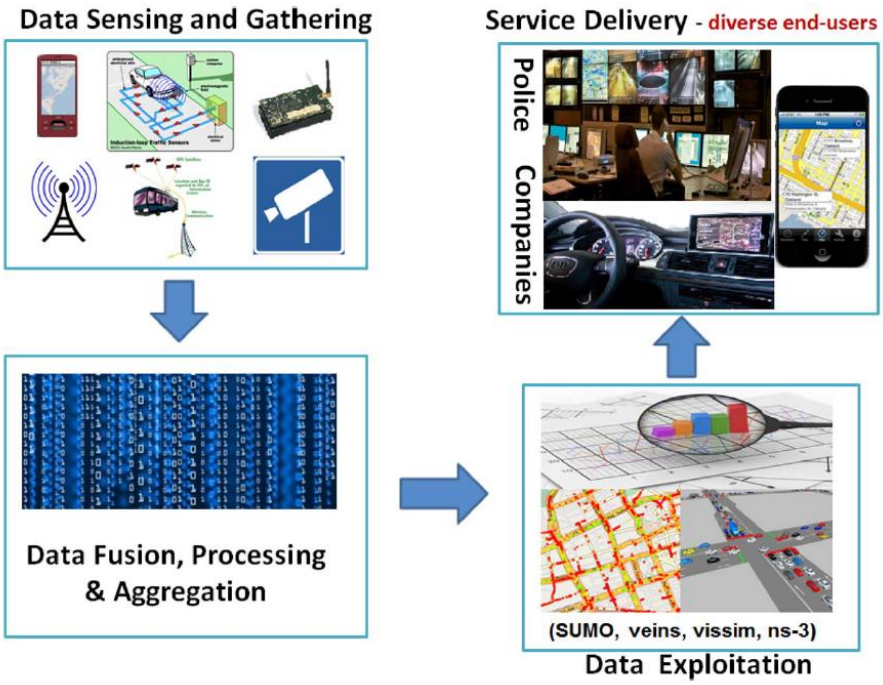


Figure 7. Traffic Management System phases [20]:

This concept can only be completely realised if all phases are fully optimised to their role. A TMS will only be able to decrease traffic congestion if implemented correctly, each phase should be implemented with sufficient hardware and software to form a functional TMS. For example, having data gathering methods that are real time but suffer from only covering a small amount of area (e.g. static sensors) will produce routes that are less efficient than routes that have all information of a network available, thus producing the possible by-product of further traffic congestion.

Route planning has seen an unprecedented expansion as cities further expand existing complex road networks. Route planning systems, such as Garmin [21] and TomTom [22] provide an easy to use method for drivers to navigate road networks. With features such as GPS, road network map and the ability to produce a static route to navigate. However, this fails to take into account various factors, such as other users within a road network. Therefore, producing a route which fails to adapt to real time on road events and can suffer from major bottlenecks en-route. Smart vehicles in the future should follow a similar dynamic routing algorithm as seen in figure 8. Using a Dynamic routing algorithm, a vehicle is able to benefit from an optimised route that decreases chances of being affected by bottlenecks and is able to produce an optimisable route that matches the driver's criteria (e.g. least distance travelled).

Figure 6: Example of a Dynamic route planning algorithm [20]:

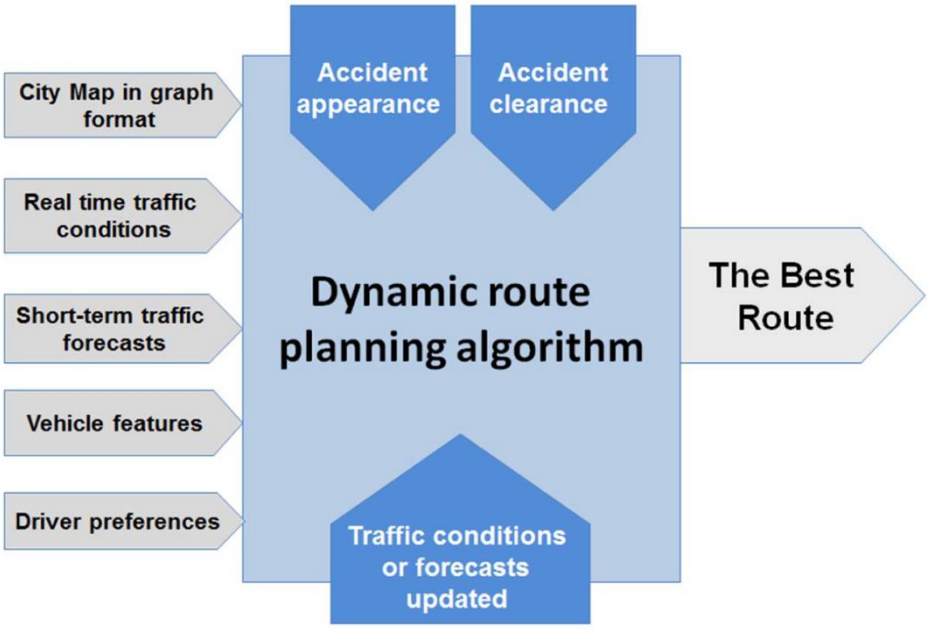


Figure 8. Example of a Dynamic route planning algorithm [20]:

This research survey provided valuable information into how a TMS can be fully optimised for a city to produce an efficient method to combat traffic congestion. It also provided a valuable foresight into the evolution of TMSs and how they can use dynamic planning algorithms to further enhance the efficiency of routing in comparison to simple route planning systems that produce static routes.

2.5. Conclusion

This review indicates that for traffic congestion to decrease, there must be further advancements within crowd sensing technology and the collection of traffic data. Without further advancements and research in these areas, traffic congestion will remain an issue for populated cities. The studies also suggest that using VNS can be counter intuitive for a driver, since all drivers would be advised by common routing methods to choose the most efficient route out of traffic congestion, there is a possibility of the vehicles causing a secondary congestion. Therefore, additional research is required into this area as dynamic routing alone will be unable to solve the difficulties noted above. Nevertheless, this research paper will attempt to produce an efficient detouring scheme for electric vehicles to attempt to produce a simulation which would be considerably more efficient than that of static algorithms and hopefully that of previous dynamic vehicle routing algorithms.

3 Product design:

In this section I will be discussing the different methodologies that could be used to produce the proposed algorithm solution for this project. Then concluding, with the model I have chosen to use to develop the solution. Then, the following section is used to demonstrate what a detour computation scheme is and the proposed algorithm solution for this project. This solution will later be implemented using Python and the open source microscopic urban transportation simulator (SUMO). The proposed algorithm will aim to make detouring for electric vehicles more efficient by accounting for the various requirements needed to make detour routing more efficient.

3.1. Agile Model:

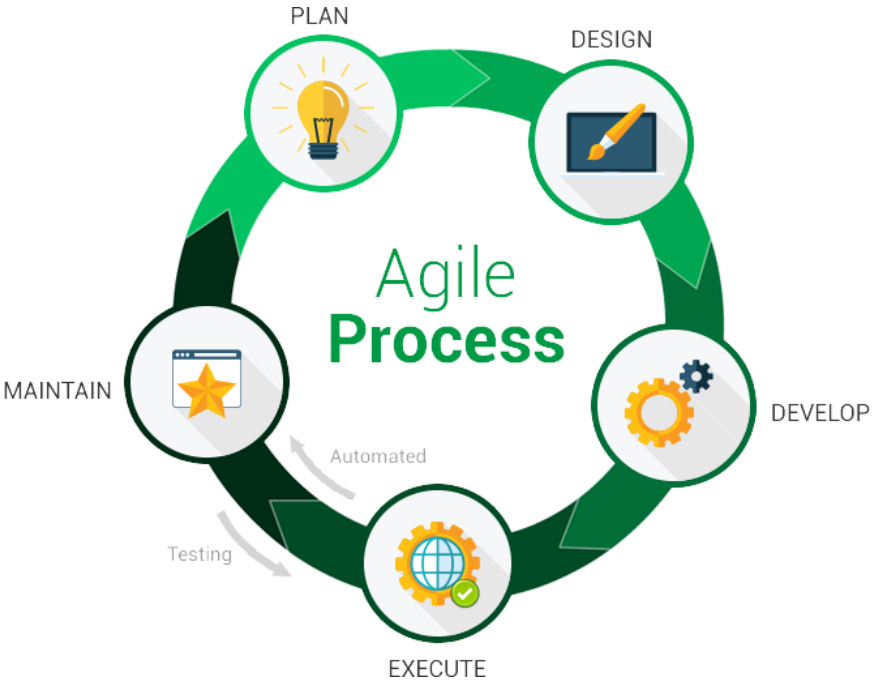


Figure 9. Example of an agile model [25].

The agile methodology is an iterative process towards development [26]. The methodology requires rapid delivery at each development stage. Focusing on what features need to be developed at that stage rather than having a concrete plan from the beginning of the process. Using this continuous approach requires constant testing after the software has been developed to incorporate any missing features the product may have. After testing is completed the process restarts with the feedback from the previous iteration.

Advantages:

- One advantage of the agile model is that the customer can see rapid solutions of the software and can readily amend any changes to the software if it does not match the required deliverable.
- There is a regular focus on the technical design of the software and focuses on producing a design specific to the customers' needs.

Disadvantages:

- One disadvantage of agile is that there is a difficulty to measure progress as progress of the software will only occur after several iterations.
- It also fails to address the necessary time that is required for documentation. This makes it difficult for any new developer working on the software as the documents will not be clearly defined for the functions that are implemented in the software.

3.2. Waterfall Models:

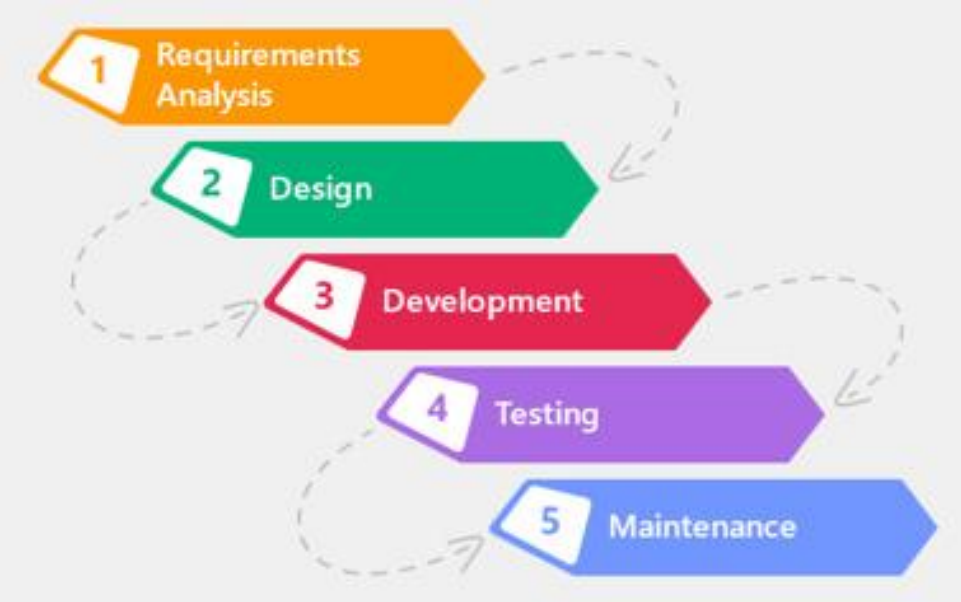


Figure 10. Example of a Waterfall Model [27].

The waterfall model (also known as Software Development Life Cycle [27]) is a sequential development process. The requirement stage of the waterfall model is the most significant [28] since it requires the gathering of information so that it can make predictive forecasts for the following phases [29]. This methodology is a linear process, after the previous phase has been completed they will not revisit the previous phase moving onto the next phase.

The maintenance phase involves adjusting the end product to the changed request of the customer, or if there were any defects that were uncovered in the live version of the product.

Advantages:

- One advantage of the waterfall model is that it provides a software team with a defined beginning and ending to each phase. This allows progress to be conclusively objectified, from the start of the project to the end.
- The heavy emphasis at the requirement and design stage will improve the quality of the final product. This is due to detecting and correcting flaws at these phases compared to the testing phase can lead to less improvisations having to be made since components of the software will have already been implemented in the testing phase.

Disadvantages:

- The waterfall method is criticised due to the models' assumption that the requirements are completely known in the requirement phase. One example is that if a customer only has a vague idea to what is required of the software, the end product will fail to meet their demands.
- Another disadvantage is that the model fails to include the customer in phases 3 and 4 (refer to Figure 10). This is a weakness of the model as not involving the customer can lead to having to amend crucial code in the end product.

For this project, I have decided to follow the agile methodology to propose the solution for this project. I have chosen to follow this methodology as agile better accounts for problems that may occur while conducting this project. In comparison to the waterfall method which would require knowledge of all future tasks before the development phase which is not possible as the outputted data of when the algorithm is ran can affect the development of the solution.

3.3. Detour computation scheme

To propose a specific detouring scheme for electric vehicles, this report will cover the basic knowledge of shortest path algorithms. A vehicle detour computation scheme focuses on producing a route from point A, that will aim to reach point B in the shortest path possible. With various implementations of shortest path routing proposing specific constraints, to achieve desirable routes that are optimised for the specified constraints.

An article within the Computers and Operations Research journal [30], presented a review of several heuristic shortest path algorithms developed for transport applications. The research indicated that shortest path problems over the past 4 decades have been studied extensively in various fields including computer science (e.g. Internet) and transportation (e.g. vehicle rerouting schemes). Research in these fields focused on developing increasingly efficient optimal algorithms to solve the problem.

The article also outlined that the majority of optimal shortest path algorithms are essentially applied variants of dynamic programming theory, implemented to find the shortest paths within a graph. The shortest path is established through recursive decision-making procedure from an origin node to the destination node or vice versa. Label of the node is defined as the routes cost from an origin node to the destination node. The scan eligible node set is the method that is used to manage the nodes that are going to be examined during the search procedure [30]. The study indicated that the significant differences between the various algorithms can be found in the data structure that is used to form the scan eligible node set.

This project will be using a variation of a label-setting algorithm. A label-setting algorithm will scan an eligible node set in an order that will be based on current path costs from origin to destination node. The node with the least labels are chosen for examination whilst concurrently identifying the shortest path to the destination node [30].

3.4 Electric Vehicle detour computation requirements

EVs' require an altered detour computation scheme in comparison to fossil fuelled vehicles. Thus, when producing an electric vehicle detour computation scheme, these various constraints are essential in proposing an efficient and improved scheme intending to make the EV to be efficient as possible. One constraint that requires acknowledgment in EV detour computation schemes, is the battery level of the EV. In comparison to fossil fuelled vehicles, an EV cannot recharge batteries at the same rate a fossil fuelled vehicle can refuel. Therefore, there needs to be extra consideration on the range remaining within the battery. Using the range, it is possible to locate all the possible routes for the EV of valid charging stations.

Another constraint that needs to be acknowledged is the availability of the CS. This needs to be taken under consideration as charging times for an EV will take several hours to have the range to complete their specified journey. Thus, the proposed algorithm needs to have a calculation where the viable charging stations are queried on their availability for use of the specified EV.
The final constraint that is required is the rerouting algorithm that will be used when there is a traffic obstruction for the route of the EV. To alleviate the effect of traffic, the rerouting algorithm that will be used is Dijkstras' algorithm, to find a shorter path to the EVs' destination.
For this project to be successful, the detouring scheme will require essential data from the EV itself along with current traffic conditions. The algorithm will use this data to define, and optimise a new route dependent on whether the conditions have been met, as seen in Figure 11.

N.B. In Figure 11 AVCS is All Viable Charging Stations.

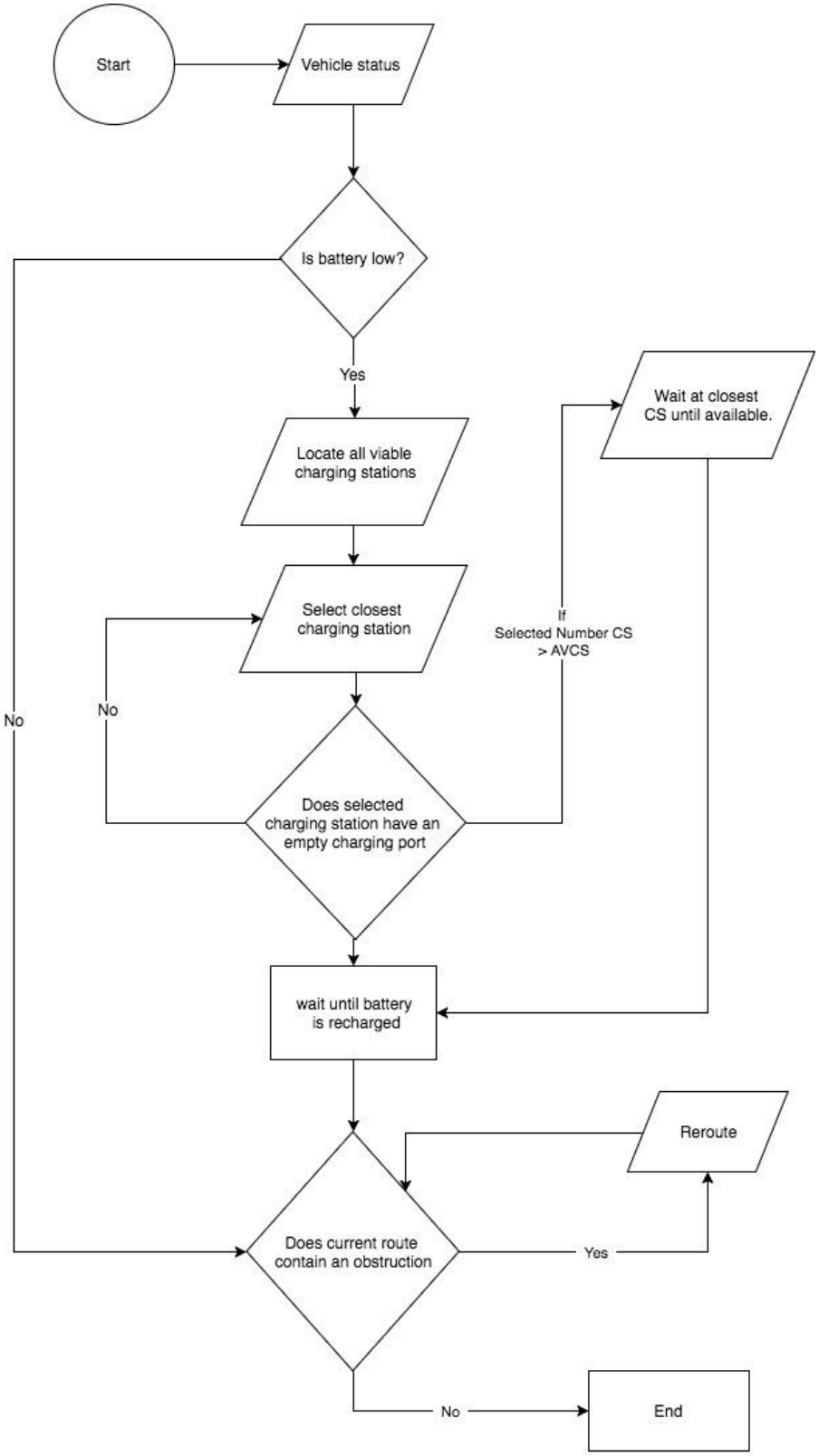


Figure 11. Flow chart of the algorithm.

3.5. Electric Vehicle computation solution

The detouring scheme will consist of two main parts for the solution, the first part will aim to ensure that the battery capacity will never fall below a threshold. The threshold will be the remaining kilometres the vehicle is able to complete with the current, remaining battery capacity of the EV, this is calculated using the range of the EV. To calculate the range, I first must calculate the amount of charge each EV will consume per kilometre at the average speed of the vehicle after every kilometre within the simulation. Using this value, the threshold will be set to the remaining battery left in the electric vehicle versus the kilometres remaining. If the battery is low the algorithm will locate all charging stations in the vicinity of the EV. Selecting the closest charging station and checking if it is available. If not, it will go through the list of charging stations to find a viable charging station. If it has gone through the list of all the charging stations, and there are no viable charging stations due to the high number of vehicles at the site, then the EV will wait at the closest CS until the vehicle is recharged. This algorithm assumes that there will be at least one charging station that is viable for the EV when the battery reaches its threshold. This assumption is being made since the detour computation scheme is intended to be used within a populated city / urban environment.

The second section of the flowchart aims to continually check if the current route, contains a lane that will increase journey times. To minimise the effect of the obstructed street, the EVs' will feature simulated rerouting devices. These rerouting devices will activate once it reaches a selected edge, that has been obstructed. These edges will be simulated randomly across the simulation. The rerouting will be done by using Dijkstras algorithm to calculate the route with the shortest path. Dijkstras' algorithm will be used as findings [11] show that Dijkstras algorithm is useful for trips that equate to less than 5 kilometres. As the testing sandbox will be in a relatively small environment, this should suffice as a fast rerouting algorithm.

3.6. Conclusion

To conclude, to produce an effective solution for detouring an EV requires various information of the EV. Such as the actual battery capacity while the vehicle is running. Using this information, it is possible to calculate the distances it can travel with the remaining charge therefore able to locate charging stations that are in the vicinity of the vehicle. Using this product design and following the agile model, I am able to move onto implementing and testing the proposed algorithm.

4. Software Evaluation

In the previous section (Product design section), the design of the algorithm had been outlined, in this section the implementation of the proposed design will be implemented and evaluated against various testing that are explained below.

4.1. Network Configuration

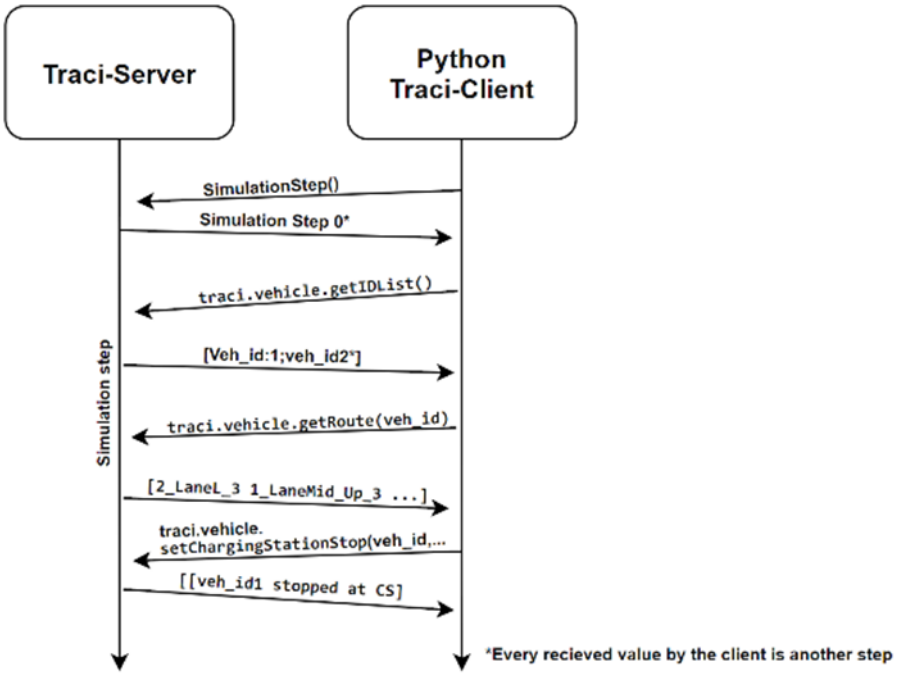


Figure 12.. EV detour scheme implemented using Traci

The diagram above demonstrations the communication that is happening between the server and client within SUMO using the Traci server, exchanging through a python script.

4.1.1. Route Implementation

To produce a rudimentary road network for SUMO I used several tools that are available within the open source framework. The NetEdit tool provides me with a visual interface thus permitting me with the ability to produce road networks of various complexities. It outputs the created road network as a xml format, since this is the format that is required to be used with the Sumo simulation.

The implementation of the network, using the NetEdit tool, produces an xml file that is used within the SUMO configuration file.

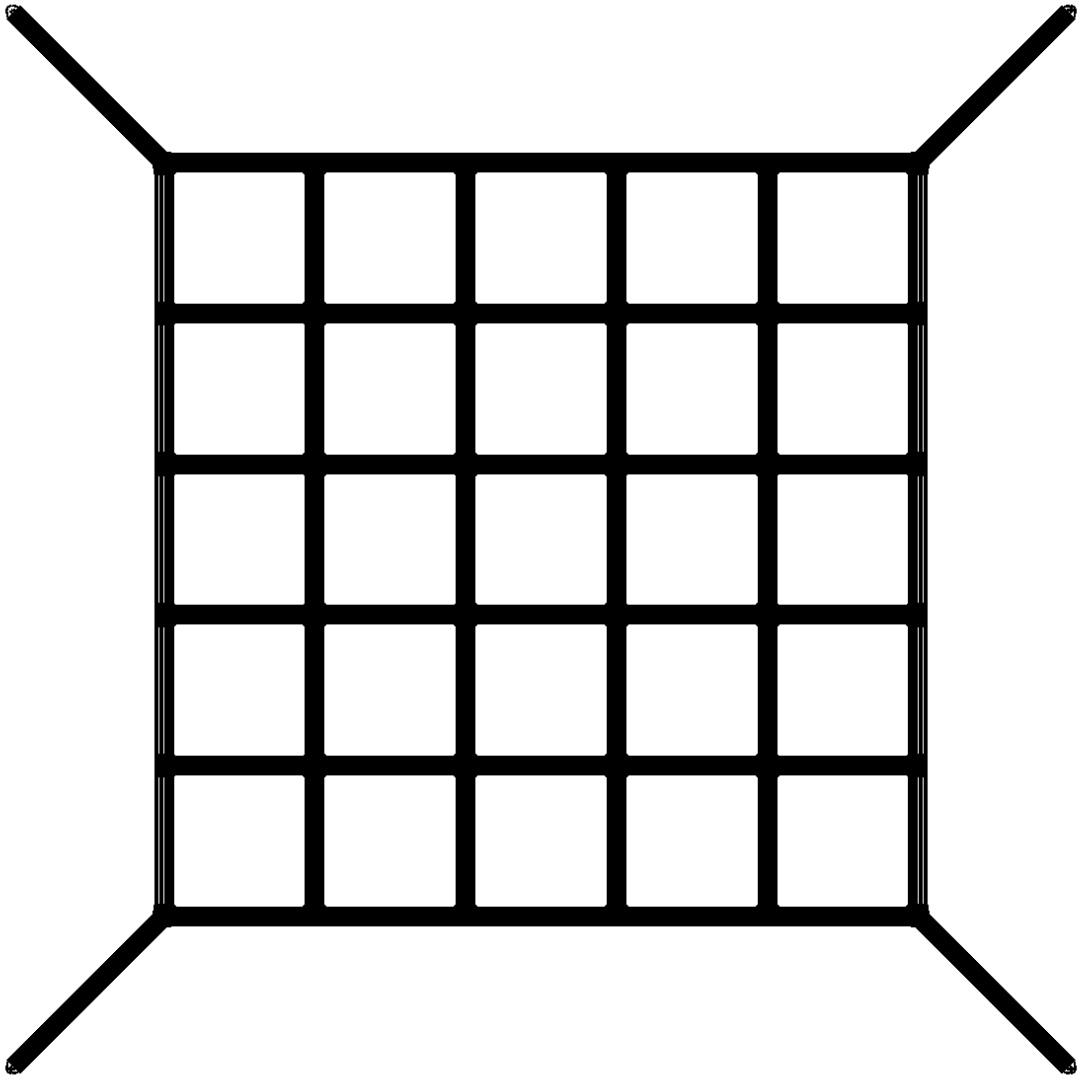


Figure 13. Image of the network produced with NetEdit

```
<edge id=":gneJ141_0" function="internal">
<lane id=":gneJ141_0_0" index="0" speed="13.89" length="8.06" shape="-391.95,204.95 -394.36,204.81 -396.14,204.70 -397.70,205.09 -399.48,206.48"/>
</edge>
<edge id=":gneJ141_1" function="internal">
    <lane id=":gneJ141_1_0" index="0" speed="13.89" length="10.39" shape="-391.95,201.65 -395.33,201.81 -398.10,202.27 -400.26,203.05 -401.81,204.15"/>
</edge>
<edge id=":gneJ141_2" function="internal">
    <lane id=":gneJ141_2_0" index="0" speed="13.89" length="5.48" shape="-391.95,201.65 -396.19,201.04 -397.21,200.43"/>
</edge>
```

Figure 14. . xml extract of the junctions in the network created

Figure 14 is an extract of the junctions within the network. These junctions act as nodes where the edges will connect between the junctions, as can be seen in figure 15.

```
<edge id="0_LaneL_0" from="gneJ142" to="gneJ141" priority="1">
    <lane id="0_LaneL_0_0" index="0" speed="13.89" length="83.90" shape="-308.05,204.95 -391.95,204.95"/>
    <lane id="0_LaneL_0_1" index="1" speed="13.89" length="83.90" shape="-308.05,201.65 -391.95,201.65"/>
</edge>
<edge id="0_LaneL_1" from="gneJ143" to="gneJ142" priority="1">
    <lane id="0_LaneL_1_0" index="0" speed="13.89" length="83.90" shape="-208.05,204.95 -291.95,204.95"/>
<lane id="0_LaneL_1_1" index="1" speed="13.89" length="83.90" shape="-208.05,201.65 -291.95,201.65"/>
</edge>
```

Figure 15. .xml extract of the edges in the network created

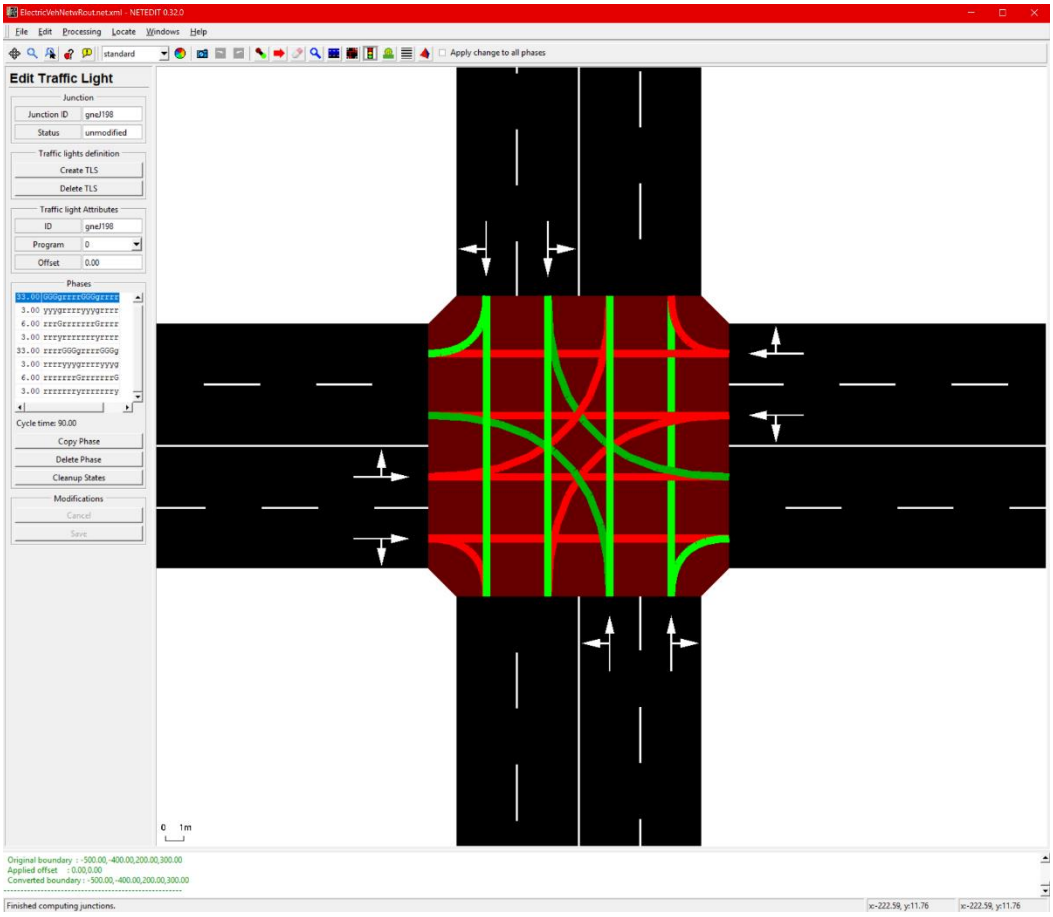


Figure 16. Image of the traffic light phases implemented

The resulting network produced from connecting the nodes and the edges can be seen in figure 13. Each junction within the simulation will have a default traffic light simulation put in place. This default setting is provided through the NetEdit software. This is implemented into the network, to make the grid simulation as realistic as possible under the constraints of the size and topology of the grid environment being used.

4.1.2. Charging Stations

The charging stations being used have the following attributes that can be seen in figure 17. The id is the identification value of the charging station and is used within the code to determine which stop the EV requires. The start position is the beginning position on the specified lane, with end position the final position. Power is the maximum amount of charge available to distribute to an electric vehicle, measured in watt hours. The efficiency value is the charging efficiency of the charging station to an EV.

```
<chargingStation id="chargingStation_csLane_0_0_0" lane="csLane_0_0" startPos="00.90" ↵
endPos="115.90" friendlyPos="0" power="100000.00" efficiency="0.95" chargeInTransit="0" ↵
chargeDelay="0.00"/>
<chargingStation id="chargingStation_csLane_2_0_1" lane="csLane_2_0" startPos="00.90" ↵
endPos="115.90" friendlyPos="0" power="100000.00" efficiency="0.95" chargeInTransit="0" ↵
chargeDelay="0.00"/>
<chargingStation id="chargingStation_csLane_4_0_2" lane="csLane_4_0" startPos="00.90" ↵
endPos="115.90" friendlyPos="0" power="100000.00" efficiency="0.95" chargeInTransit="0" ↵
chargeDelay="0.00"/>
<chargingStation id="chargingStation_csLane_6_0_3" lane="csLane_6_0" startPos="00.90" ↵
endPos="115.90" friendlyPos="0" power="100000.00" efficiency="0.95" chargeInTransit="0" ↵
chargeDelay="0.00"/>
```

Figure 17. .xml extract of charging station attributes

4.1.3. Electric Vehicle class

The EV class has the following parameters. This code is implemented within the additional file along with the charging station attributes for the sumo configuration file to implement these features into the simulation environment.

```
<vType id="evehicle" vClass="evehicle" accel="2.6" decel="4.5" maxSpeed="160.0" length="4.5"↵
          minGap="2.0" emissionClass="Energy/unknown">
          <param key="has.battery.device" value="true"/>
          <param key="actualBatteryCapacity" value="10000.0"/>
          <param key="constantPowerIntake" value="100"/>
          <param key="arrivalLane" value="99"/>
          <param key="arrivalSpeed" value="0.9"/>
          <param key="maximumBatteryCapacity" value="11000"/>
          <param key="maximumPower" value="100"/>
          <param key="vehicleMass" value="2090"/>
          <param key="frontSurfaceArea" value="6.2"/>
          <param key="airDragCoefficient" value="0.24"/>
          <param key="recuperationEfficiency" value="0.9"/>
          <param key="stoppingThreshold" value="1"/>
</vType>
```

Figure 18. .xml EV parameters

The most notable parameters are the maximum battery capacity and the actual battery capacity. This is because these parameters define the amount of charge the vehicle can hold along with starting the simulation with, respectively. Other parameters are attributes of the vehicle and are used by SUMO to calculate the various definitions of the vehicle for within the simulation. Such as the position of the edge the vehicle will stop at is defined by arrival lane.

4.1.4. Routes

To make use of the topologies described above, there must be simulated routes for the EV to follow. These routes will be produced by using the RandomTrips function that is provided by SUMO. This function allows for control of the number of vehicles that will be in the sandbox environment at every step interval. The length of the simulation can also be controlled along with the ability to control the number of trips. Thus, producing a trip file for the Sumo environment that allows for numerous levels of complexity.

This complexity can be seen in the number of EVs' being in the sandbox environment at any one time and the length of time of the produced random trips. This function will output an .xml file that is made use of by SUMO. figure 19 is an extract of said xml file.

```
<vehicle id="0" type="evehicle" depart="0.00">
    <route edges="4_LaneR_2 3_LaneMid_Up_3 2_LaneMid_Up_3 2_LaneL_2 2_LaneL_1 2_LaneL_0↵
2_LaneMid_Dn_0"/>
</vehicle>
<vehicle id="1" type="evehicle" depart="1.00">
    <route edges="2_LaneMid_Up_1 1_LaneMid_Up_1 1_LaneL_0 0_LaneMid_Up_0"/>
</vehicle>
<vehicle id="2" type="evehicle" depart="2.00">
    <route edges="4_LaneMid_Dn_3 5_LaneL_2 4_LaneMid_Up_2 3_LaneMid_Up_2 2_LaneMid_Up_2↵
2_LaneL_1 2_LaneL_0 1_LaneMid_Up_0"/>
</vehicle>
<vehicle id="3" type="evehicle" depart="3.00">
    <route edges="3_LaneMid_Dn_0 4_LaneR_0 4_LaneR_1 4_LaneMid_Dn_2 5_LaneR_2↵
4_LaneMid_Up_3"/>
</vehicle>
<vehicle id="4" type="evehicle" depart="4.00">
    <route edges="2_LaneMid_Up_2 1_LaneMid_Up_2 1_LaneL_1"/>
</vehicle>
<vehicle id="5" type="evehicle" depart="5.00">
    <route edges="2_LaneL_2 2_LaneL_1 2_LaneL_0 1_LaneMid_Up_0"/>
</vehicle>
```

Figure 19. .xml extract of 6 random routes from the 100-vehicle scenario

4.2. Electric Vehicle computation solution

Once the configuration for SUMO has been implemented, the configuration being the files that are required by the simulation for the Traci API to interact with, I moved onto the next step. This step was to produce the Traci script that will be interacting with the SUMO simulation. Each step within SUMO is a second in real time. Using the Traci API provides a method of controlling the simulation between time steps. This control enables the possibility of producing various calculations for each EV, such as determining the distance to a destination amid time steps. Visualisatiuon of the simulation can be seen in figure 20. Each EV is being constantly updated through the Traci client. The data produced by the simulation will then be stored in an array within the Traci client and will be outputted into a csv file for further analysis later on in this report.

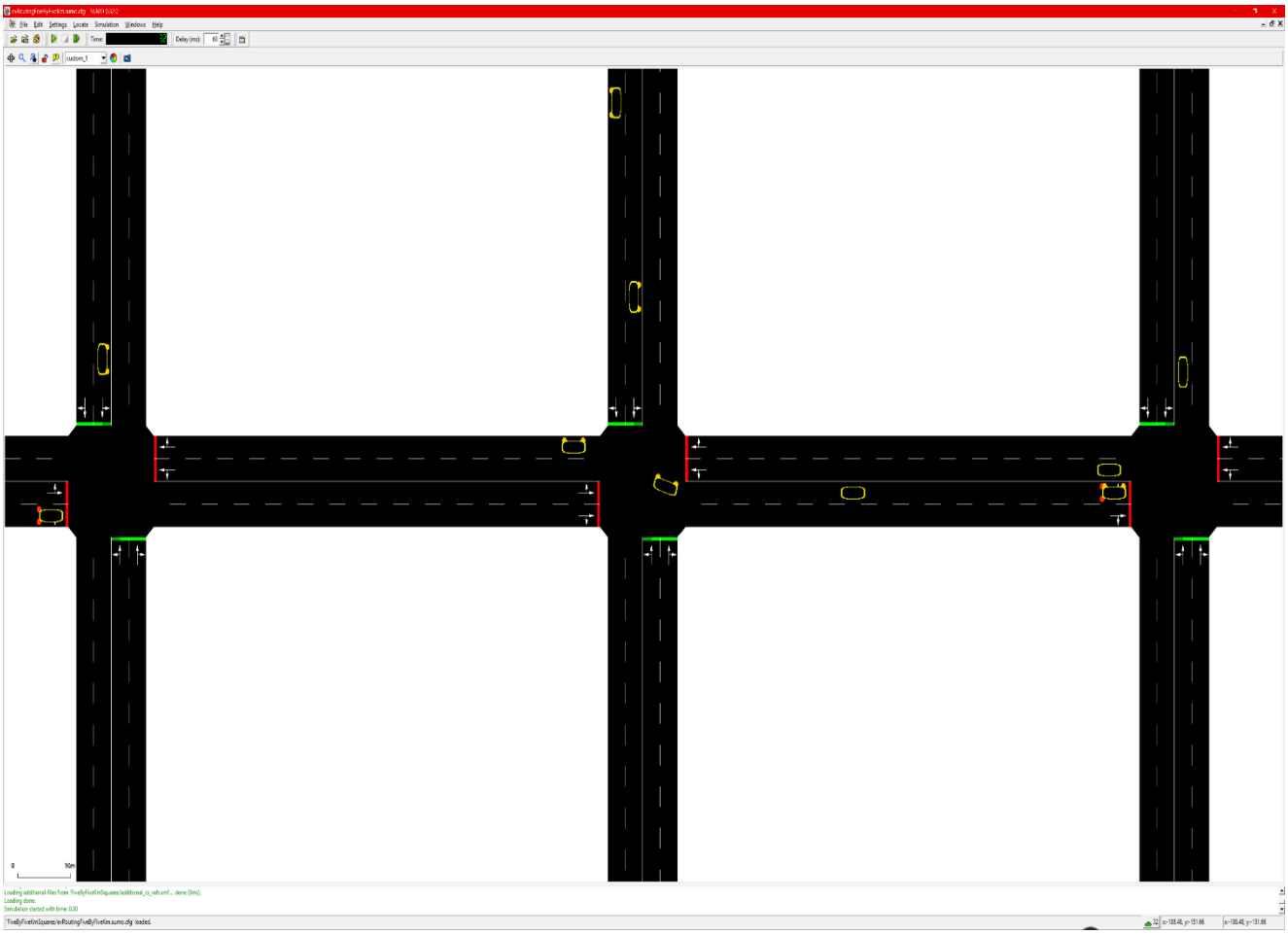


Figure 20. SUMO GUI visualisation of the simulation running

4.2.1. Connection to simulation:

```
263 sumoCmd = [sumoBinary, "-c", "FiveByFiveSquares/evRoutingFiveByFiveKm.sumo.cfg"]
264 traci.start(sumoCmd)
265 run()
```

Figure 21. python script extract

The connection to the simulation is implemented using the three lines of code seen in the above figure. This starts the simulation in SUMO, the Traci server and runs the main portion of the script, where the calculations will be implemented for each EV.

4.2.2. Updating the simulation:

The main section of the script is entered as seen in figure 22. The data collection from the script has been ommited from this figure.

The Run function is the main function for the script, where the data collection along with calculations will be taking place. Lines 134:135 is the step counter for the simulation. This while loop will continue updating until there is no more vehicles within the simulation.

Lines 139:156 are implemented to gather the data from the EV at that timestep. The data being collected are as follows; the edge the current EV is at that timestep, the final position of the EV, the distance to the final position, the current battery capacity, the vehicle speed and the consumption of electricity within the current timestep. This data is then used in 'if' and 'elif' to set constraints for the electric vehicle. Line 158 tests wheter or not the EV is moving in the simulation. If the EV is moving, further constraints can be set onto the EV. Line 164 tests whether the EV has enough charge to remain in the simulation. If there is enough charge the function will continue to the next constraint. If there is not enough charge the EV will be removed from the simulation and moved onto the next timestep.

Line 168 tests whether there is enough battery charge for the vehicle to reach the destination edge. If it does not have enough charge then a function will be called to determine the closest viable charging station available to the EV, from the vehicles current position. The code to implement this can be seen in figure 23. If it does have enough charge to reach the final edge then the edge will be set to the final edge that was defined in the xml route file.

One other function that requires attention, is the range calculation, figure 24. This calculation was provided by my tutor. The calculation is used as a method of receiving the current range of the vehicle at that time step and is used to set the threshold to the distance that can be travelled by the EV.

```
72  # calculates the range using the equation
73  # Range(m)= remaining Battery capacity(Kwh)/electricity consumed(wh/second)*speed(m/second)
74  # electricity consumed(wh/s) is per second due to it being retrieved from the last time step.
75  def range_calculation(veh_id):
76      range_calc = 0
77      remaining_battery_capacity_kwh = float(traci.vehicle.getParameter(veh_id,
78                                'device.battery.actualBatteryCapacity'))
79
80      electricity_consumption_wh = float(traci.vehicle.getElectricityConsumption(veh_id))
81
82      speed_mps = get_speed_mps(veh_id)
83      if electricity_consumption_wh > 0 and speed_mps > 1:
84          range_calc = (remaining_battery_capacity_kwh / electricity_consumption_wh) * speed_mps
85      return range_calc
```

Figure 22. Range Calculation

```
def run():
    vehicles = list(map(str, range(100 + 1)))
    get_route_edges = []

    while traci.simulation.getMinExpectedNumber() > 0:
        traci.simulationStep()
        id_list = traci.vehicle.getIDList()
        for veh_id in id_list:
            get_route_id = traci.vehicle.getRoute(veh_id)
            for i in vehicles:
                if i == veh_id:
                    rou_data.append([veh_id, get_route_id])
                    get_route_edges.append(traci.vehicle.getRoute(veh_id))
                    vehicles.remove(i)
            try:
                veh_last_edge = get_route_edges[int(veh_id)][-1]
            except IndexError:
                veh_last_edge = traci.vehicle.getRoute(veh_id)[-1]
            get_current_speed_mps = float(traci.vehicle.getSpeed(veh_id))
            electricity_consumption =
                                float(traci.vehicle.getElectricityConsumption(veh_id))
            current_battery_capacity = float(traci.vehicle.getParameter(veh_id,
                                'device.battery.actualBatteryCapacity'))

            range_calc = range_calculation(veh_id)
            range_calc_cons = range_calc

            if electricity_consumption > 0 and get_current_speed_mps > 1:
                veh_edge = traci.vehicle.getLaneID(veh_id)[:-2]
                veh_lane_position = traci.vehicle.getLanePosition(veh_id)
remaining_distance = traci.simulation.getDistanceRoad(veh_edge,int(veh_lane_position),
veh_last_edge, 83, isDriving=True)

                if current_battery_capacity == 0:
                    id_list.remove(veh_id)
                    remove_ev(veh_id)
                    break
                elif (range_calc_cons - (range_calc_cons * threshold))
                     < remaining_distance:
                    ev_charging_station_stop(veh_id, veh_last_edge)
                elif (range_calc_cons - (range_calc_cons * threshold))
                     >= remaining_distance:
                    if traci.vehicle.getRoute(veh_id)[-1] == veh_last_edge:
                        continue
                    else:
                        traci.vehicle.changeTarget(veh_id, veh_last_edge)
    traci.close()
    sys.stdout.flush()
```

Figure 23. Run method

```
def ev_charging_station_stop(veh_id, veh_last_edge):
    travel_distance_array = []
    veh_lane_index = traci.vehicle.getLaneID(veh_id)
    veh_lane_position = traci.vehicle.getLanePosition(veh_id)
    changed = False
    for cs in charging_stations:
        travel_distance = traci.simulation.getDistanceRoad(veh_lane_index[:-2],
                          veh_lane_position, cs[1][:-2], float(cs[2]), True)
travel_distance_array.append([int(travel_distance), veh_id, cs[0], cs[1],
                          float(cs[2])])
  travel_distance_array.sort()
  for min_travel_distance in travel_distance_array:
          if min_travel_distance[0] == 1.7976931348623157e+308 and veh_lane_index ==
                  min_travel_distance[3] and 1 <=  veh_lane_position <= 80:
              print(veh_lane_position)
              print(veh_id, veh_lane_index, min_travel_distance[3], ', stop at cs2')
              traci.vehicle.setChargingStationStop(veh_id, min_travel_distance[2],
                          duration=1800000)
              traci.vehicle.changeTarget(veh_id, veh_last_edge)
              changed = True
              break
          elif 1 <= min_travel_distance[0] <= 80 and veh_lane_index ==
                          min_travel_distance[3]:
              print(veh_id, veh_lane_index, min_travel_distance[0],
                          min_travel_distance[2], ', stop at cs3')
              traci.vehicle.setChargingStationStop(veh_id, min_travel_distance[2],
                          duration=1800000)
              traci.vehicle.changeTarget(veh_id, veh_last_edge)
              changed = True
              break
  if changed is False:
      traci.vehicle.changeTarget(veh_id, travel_distance_array[0][3][:-2])
```

Figure 24.Locate Charging Station Extract

4.3. Results:

Producing an evaluation of the implemented, proposed electric vehicle computation scheme required a baseline to compare the effectiveness of the charging station computation scheme. Along with a baseline, various thresholds will be tested with 101 vehicles in a simulation. The threshold values will range from 0 - 0.9. Whichever value is found as optimum will be applied to the larger routing simulation.

```
# The following lines of code, are to find the first and last value of each attribute of the
vehicle
if veh_id not in veh_id_arr:
    veh_id_arr.append(veh_id)
    curr_bat_capac_arr.append(current_battery_capacity)
    veh_last_edge_arr.append(traci.vehicle.getRoute(veh_id)[-1])
    range_arr.append(range_calc)
    remaining_distance_arr.append(remaining_distance)
elif veh_id_arr.count(veh_id) < 2:
    veh_id_arr.append(veh_id)
    curr_bat_capac_arr.append(current_battery_capacity)
    veh_last_edge_arr.append(traci.vehicle.getRoute(veh_id)[-1])
    range_arr.append(range_calc)
    remaining_distance_arr.append(remaining_distance)
else:
    last_ind = last_occurrence(veh_id_arr, veh_id)
    del veh_id_arr[last_ind]
    del curr_bat_capac_arr[last_ind]
    del range_arr[last_ind]
    del remaining_distance_arr[last_ind]
    veh_id_arr.append(veh_id)
    curr_bat_capac_arr.append(current_battery_capacity)
    veh_last_edge_arr.append(traci.vehicle.getRoute(veh_id)[-1])
    range_arr.append(range_calc)
    remaining_distance_arr.append(remaining_distance)
csv_out = open(output_filename+'.csv', 'w', newline='')
# create the csv writer object.
cvs_write = csv.writer(csv_out)
for row in zip(veh_id_arr, curr_bat_capac_arr, veh_last_edge_arr,
range_arr,remaining_distance_arr):
    cvs_write.writerow(row)
csv_out.close()
```

Figure 25. Data gathering script

The code above describes the method being used to produce a csv file to analyse the data produced from the simulation. The code outputs the first timestep of the simulation for each EV and the final step. These values are chosen since they will be the main attributes produced by the simulation which can be analysed in this section.

Testing of the various thresholds using the algorithm with 100 vehicles produced the resulting values which can be seen in the line graph below. The optimum value produced is between 0.4 and 0.5 as only 2 vehicles failed to reach their final destinations when the threshold is set to this value. Using this information 0.5 was chosen as the threshold value to implement on the remainder of the experiments. Since it is a more restrictive value, forcing the EV to locate a charging station earlier.

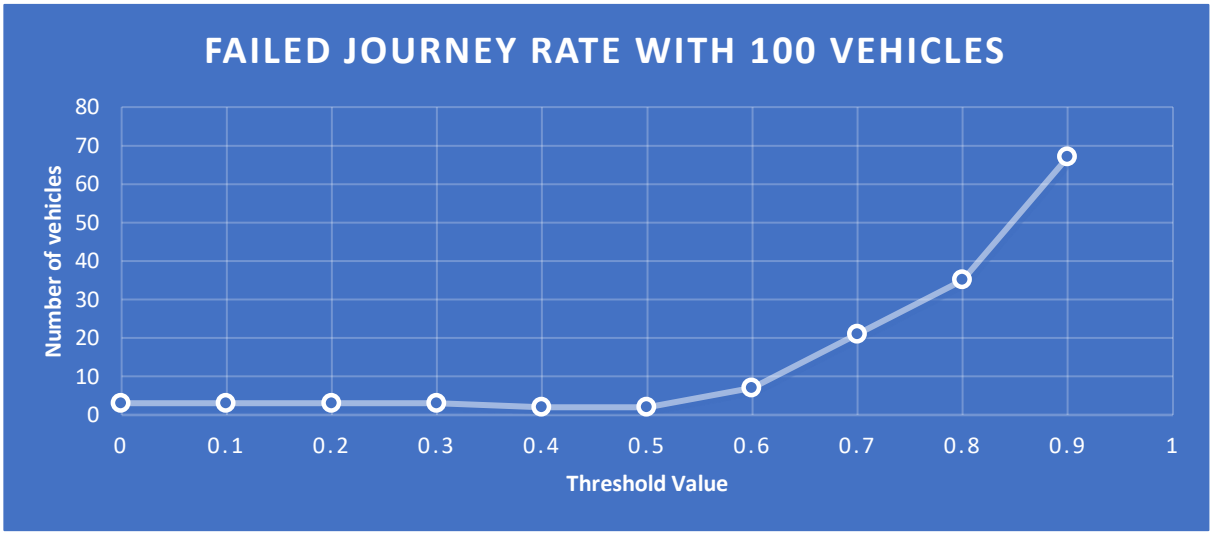


Figure 26. Failed Journey rate

From this, testing was then done to find the comparison of the baseline, which is a run of the simulation with no algorithm affecting the route of the simulation. And the proposed EV computation scheme (EVCS). The threshold of the EVCS is set at the value of 0.5.

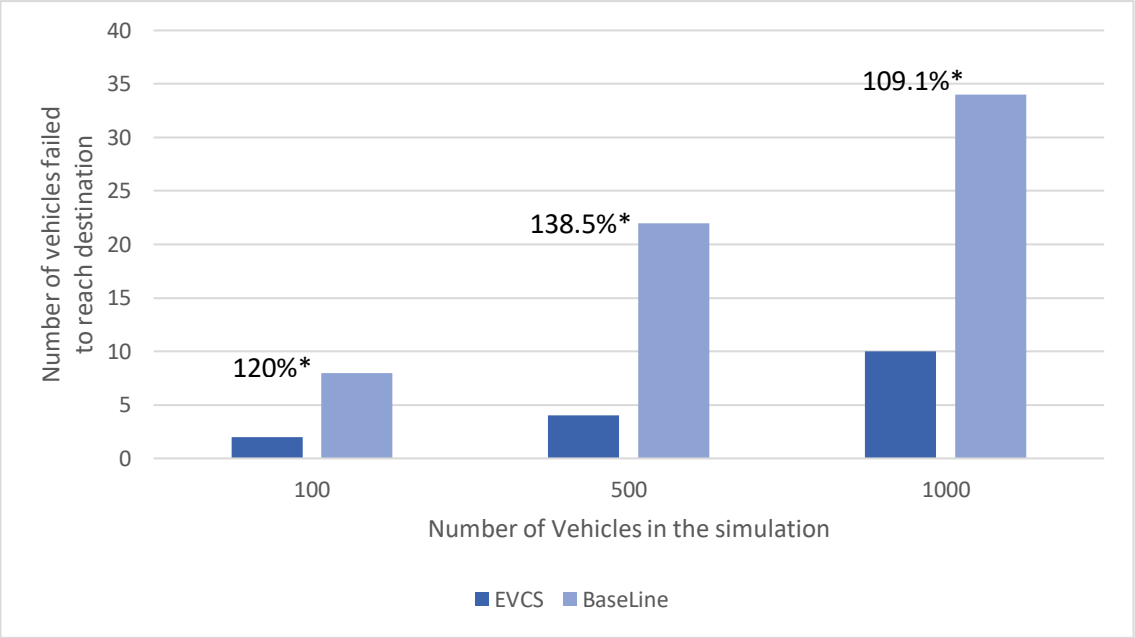


Figure 27. Completed routes

The results of this testing indicate that there is a significant difference between the number of vehicles completing the journey when using the proposed algorithm in comparison to the baseline where there is no routing for the EV. Therefore the proposed scheme is succesfull in producing the first part of the propsed product design.

However, the objective of this project had been to also take into account any obstructions (accidents, road closures), that may occur en-route to the final edge. Failure to implement this feature had been due to failure to understand the implementation of this feature into the Traci API in the time frame available for this project. Furthermore the results produced can only be generalised to the small scale simulation that is implemented within SUMO. To be able to generalise the results to real world scenarios, the simulation will require a network that has been produced using data of cities.

Citations.

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, (Feb. 2014.) "Internet of Things for Smart Cities," in *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22-32, (URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6740844&isnumber=6810798) (Accessed: 29 Oct. 2017.)

[2]M. Gerla, E. K. Lee, G. Pau and U. Lee, (2014), "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Seoul, South Korea , pp. 241-246 (URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6803166&isnumber=6803102)(Accessed: 30 Oct. 2017.)

[3]H. Yang, Y. Deng, J. Qiu, M. Li, M. Lai and Z. Y. Dong, (Oct. 2017.), "Electric Vehicle Route Selection and Charging Navigation Strategy Based on Crowd Sensing," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2214-2226, (URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7879839&isnumber=8059716) , (Accessed: 3 Nov. 2017.)

[4]Sen Guo, Huiru Zhao, (2015), "Optimal site selection of electric vehicle charging station by using fuzzy TOPSIS based on sustainability perspective", In *Applied Energy*, Volume 158, pp. 390-402, (URL: http://www.sciencedirect.com/science/article/pii/S0306261915010181)(Accessed: 3 Nov. 2017.)

[5] Dave Leggett, (26 July 2017), "UK gov to ban ICE vehicles from 2040.", In *just-auto.com, General OneFile* (URL:http://go.galegroup.com.ezproxy.mmu.ac.uk/ps/i.do?p=ITOF&sw=w&u=mmucal5&v=2.1&it=r&id=GALE|A499302199&sid=summon&asid=bb9ff0db9b29a8673eab1953549ce929.) (Accessed 8 Nov. 2017.)

[6] (14 Sept. 2017.), "China will ban petrol and diesel vehicles.", In *European Union News, Infotrac Newsstand* (URL: go.galegroup.com.ezproxy.mmu.ac.uk/ps/i.do?p=STND&sw=w&u=mmucal5&v=2.1&it=r&id=GALE%7CA504566156&sid=summon&asid=3f2c539f1cad4a0b09b9292bbc0a24b4.) (Accessed 8 Nov. 2017)

[7] Ghazale Haddadian, Mohammad Khodayar, Mohammad Shahidehpour, (2015) “Accelerating the Global Adoption of Electric Vehicles: Barriers and Drivers”, In *The Electricity Journal*, Volume 28, Issue 10, pp. 53-68 (URL: http://www.sciencedirect.com/science/article/pii/S104061901500250X) (Accessed 13 Nov. 2017)

[8] (6 July 2017), "Volvo Cars to go all electric.", In *ENP Newswire, Infotrac Newsstand* go.galegroup.com.ezproxy.mmu.ac.uk/ps/i.do?p=STND&sw=w&u=mmucal5&v=2.1&it=r&id=GALE%7CA497793772&sid=summon&asid=6770ea402ee7d78c6daad0005b6bb2ef. (Accessed 13 Nov. 2017.)

[9] M. Yilmaz and P. T. Krein, (2012) "Review of charging power levels and infrastructure for plug-in electric and hybrid vehicles," In *2012 IEEE International Electric Vehicle Conference*, Greenville, SC, pp. 1-8 (URL: http://ieeexplore.ieee.org.ezproxy.mmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=6183208&isnumber=6183155) (Accessed 15 Nov. 2017.)

[10] Martin Strehler, Sören Merting, Christian Schwan, (2017) “Energy-efficient shortest routes for electric and hybrid vehicles”, In *Transportation Research Part B: Methodological*, Volume 103, pp. 111-135 (URL: http://www.sciencedirect.com/science/article/pii/S0191261516304404) (Accessed 16 Nov. 2017.)

[11] S. Wang, S. Djahel, J. McManis, C. McKenna and L. Murphy, (2013), "Comprehensive performance analysis and comparison of vehicles routing algorithms in smart cities," *Global Information Infrastructure Symposium - GIIS 2013*, Trento, pp. 1-8 (URL: http://ieeexplore.ieee.org.ezproxy.mmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=6684365&isnumber=6684341) (Accessed 18 Nov. 2017.)

[12] IEEE website, about page[Online]. https://www.ieee.org/about/today/at_a_glance.html (Accessed 19 Nov. 2017.)

[13] Dijkstra, Edsger W. (1959),” A note on two problems in connexion with graphs.” Numerische mathematik 1.1, pp. 269-271.

[14] P. E. Hart, N. J. Nilsson and B. Raphael, (July 1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in *IEEE Transactions on Systems Science and*

Cybernetics, vol. 4, no. 2, pp. 100-107
(URL: http://ieeexplore.ieee.org.ezproxy.mmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=4082128&isnumber=4082123) (Accessed 21 Nov. 2017.)

[15] I. Chabini and Shan Lan, (Mar 2002), "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 60-74
(URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=994796&isnumber=21456) (Accessed 21 Nov. 2017.)

[16] Michael G.H. Bell, Valentina Trozzi, Solmaz Haji Hosseinloo, Guido Gentile, Achille Fonzone, (2012), "Time-dependent Hyperstar algorithm for robust vehicle navigation", *In Transportation Research Part A: Policy and Practice*, Volume 46, Issue 5, pp. 790-800
(URL: http://www.sciencedirect.com/science/article/pii/S0965856412000201) (Accessed 22 Nov. 2017.)

[17] Heinz Spiess, Michael Florian, (1989), "Optimal strategies: A new assignment model for transit networks", In *Transportation Research Part B: Methodological*, Volume 23, Issue 2, pp. 83-102 (URL:http://www.sciencedirect.com/science/article/pii/0191261589900349)
(Accessed 22 Nov. 2017.)

[18] S. Wang, S. Djahel, Z. Zhang and J. McManis, (Oct. 2016), "Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2888-2899
(URL: http://ieeexplore.ieee.org.ezproxy.mmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=7433412&isnumber=7580578) (Accessed 23 Nov. 2017.)

[19] B Guo, Z Wang, Z Yu, Y Wang and NY Yen, (Aug 2015), "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm." Acm Computing Surveys, vol. 48, no.1, pp. 1-31. (Accessed 23 Nov. 2017.)

[20] S. Djahel, R. Doolan, G. M. Muntean and J. Murphy,(First quarter 2015), "A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 125-151
(URL: http://ieeexplore.ieee.org.ezproxy.mmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=6857980&isnumber=7061782) (Accessed 24 Nov. 2017.)

[21] TomTom Website. [Online]. (URL: http://www.tomtom.com/) (Accessed 24 Nov.2017.)

[22] Garmin Website. [Online]. (URL: http://www.garmin.com/) (Accessed 24 Nov. 2017.)

[23] How smart cities are building the future [ONLINE].
(URL: https://fm.cnbc.com/applications/cnbc.com/resources/img/editorial/2015/12/15/103244239-GettyImages-499612372-1.1910x1000.jpg/) (Accessed 1 January 2018)

[24] Tesla.com, 2018. [Online]. Available:
https://www.tesla.com/tesla_theme/assets/img/modals/model-select-models.png?20160811.
[Accessed: 27- Apr- 2018].

[25] Taazaa, (2017.), "Agile Process". (image) Available at: https://taazaa.com/wp-content/uploads/2013/05/infographic.png (Accessed: 12 Dec. 2017).

[26] Segue Technologies, (Aug. 2015.), "What is Agile Software Development?" Segue Technologies. (online) (URL: https://www.seguetech.com/what-is-agile-software-development/) (Accessed: 12 Dec. 2017).

[27] Gordiyenko, S., (Nov. 2014.), "Waterfall Software Development Life Cycle (SDLC) Model: Steps, Stages, Case Studies." (online) XB Software. (URL: https://xbsoftware.com/blog/software-development-life-cycle-waterfall-model/) (Accessed: 12 Dec. 2017).

[28] TechRepublic, (Sept. 2016.), "Understanding the pros and cons of the Waterfall Model of software development." (online) (URL: https://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/) (Accessed: 14 Dec. 2017).

[29] Segue Technologies. , (Jul. 2013.), "Waterfall vs. Agile: Which Methodology is Right for Your Project?." (online) (URL: https://www.seguetech.com/waterfall-vs-agile-methodology/) (Accessed: 14 Dec. 2017).

[30] L. Fu, D. Sun, L.R. Rilett, (Heuristic shortest path algorithms for transportation applications: State of the art, In Computers & Operations Research, Volume 33, Issue 11, pp. 3324-3343, (URL: http://www.sciencedirect.com/science/article/pii/S030505480500122X) (Accessed: 14 Dec. 2017).

Appendices:

Appendix A:

Efficient Detour computation scheme for Electric Vehicles

Student: Faisal Kabeer; Student ID: 15063379

Supervisor: Dr Soufiene Djahel

Learning Outcomes:

- Analyse real world situations critically
- Find, evaluate, synthesise and use information
- Use systems and scenario thinking
- Analyse, compare, discuss and assess existing work, as well as your own work.
- Independently plan, manage and successfully complete a project of substantial size in an area that is relevant to the degree programme
- Identify, develop and critically reflect on own engagement with professional development, confidently apply for professional job roles in your subject area and develop your professional development plan (PDP).

Background:

Vehicles on the road are constantly facing new challenges in major cities across the world as urban areas become more populated resulting in an increased number of vehicles on the road. The current solution of on-board navigation systems for individual vehicles provides the user with the ability to pre-plan their route, finding the most convenient route to take to achieve reaching their destination. This method however does not accommodate for future traffic changes as it relies on the current traffic state and sometimes paired with previous traffic trends. Thus, resulting in being unable to provide a more efficient alternative route for an electric vehicle quickly enough if an unexpected congestion occurs. This is especially challenging for electrical vehicles as their routing decisions can suffer from further constraints. As a solution to the above problem, this project will be focused upon designing a heuristic algorithm that can provide an efficient detour route for electric vehicle.

Aim:

- The aim of this project is to design a heuristic algorithm that provides electric vehicles with an efficient detour route, that at minimum is similar to timings set by other variations of the Dijkstra and Dynamic A* algorithm.
- To evaluate the performance of the algorithm using SUMO by importing a model of a real city as the testing base.

Objectives:

- Investigate and reference any existing studies that have tested current routing algorithms for vehicles.
- Develop an efficient heuristic algorithm that is based upon the functional Dynamic A* routing algorithm.
- Proactively Implement and investigate the efficiency of the heuristic algorithm developed, experiment with the overall efficiency of the developed algorithm within sumo in comparison with other routing algorithms.
- Using SUMO to investigate the algorithms performances in comparison to the heuristic algorithm this project will be focused on developing.

- The comparison will be done by running the navigation from two separate points at varying distances, enclosed later in the project schedule, within SUMO using a realistic traffic dataset.

Problems:

A project of this intricacy can have various problems that could arise.

- One such problem is having to organise a feasible plan factoring in the multiple topics I will be partaking alongside the project.
- Another issue is understanding how the software implements and displays the python routing algorithm and its efficiency for the vehicles immediately, therefore causing me to need a prolonged length of time on learning how the software implements the routing algorithm.
- Other problems could occur due to my limited experience using the simulation tool along with my limited experience coding within the python language.

Resources:

To complete this project, the following application and tools are required:

- Simulation of Urban Mobility (SUMO Version 0.31.0) an open sourced readily obtainable software

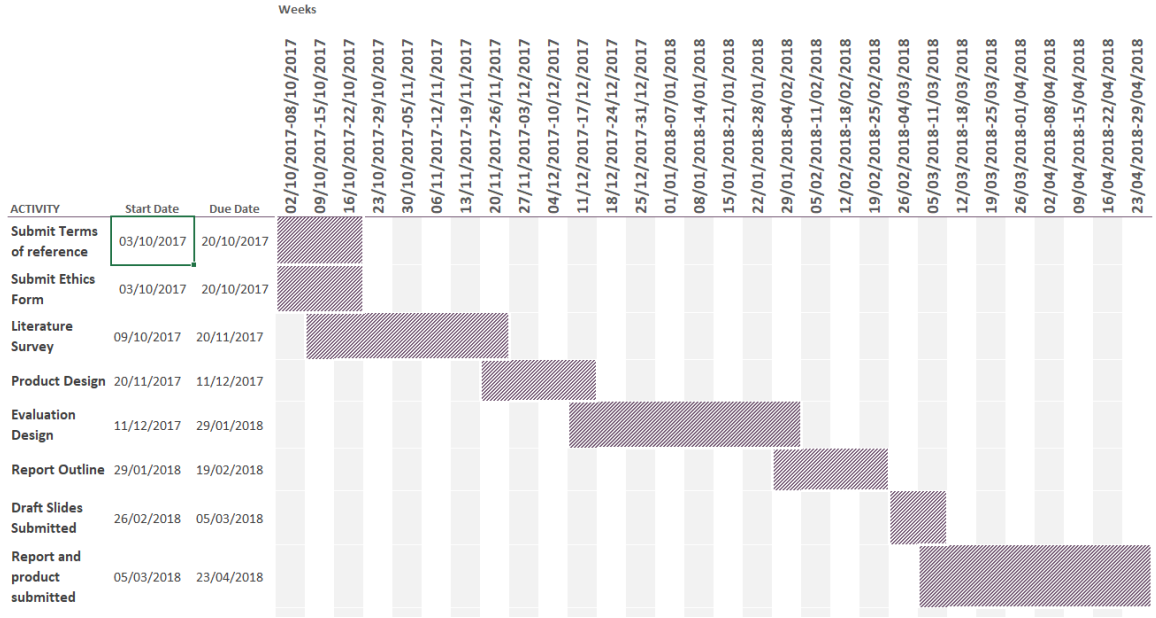
As a standard, all testing will be done on a PC with the following specification:

- Windows 10 Pro 64-bit
- Intel(R) Core i5-6500 CPU @ 3.2GHz
- 16.00 GB DDR4 RAM
- SSD hard drive.

Timetable and Deliverables:

To aid in completing this project, I will be proactively planning the activities which will help organise my time more efficiently. Dedicating my time to relevant activities ensures that I will be able to meet deadlines and develop the relevant documents when they are required.

Below is a table that displays deadlines for the project that I must complete within the 32-week timeframe. However, this is only a rough planning method as of now, since further tasks will be added when the project initiates.



Timetable:

Activity	Start Date	Completion Date	Due Date
Submit Terms of reference	03/10/2017		20/10/2017
Submit Ethics Form	03/10/2017		20/10/2017
- **Researching into the background of the project:**			
- **Research into current works (Research papers):**			
Literature Survey	9/10/2017		20/11/2017
- **Outline how decisions are being made in producing the code.**			
- **Describe the procedure being used to develop the final code.**			
Product Design	20/11/2017		11/12/2017
- **Define techniques being used to assess the quality of any developed code**			
Evaluation Design	11/12/2017		29/01/2018
- **Plan a hierarchal structure of chapters, sections and sub-sections for the main report**			
Report Outline	29/01/2018		19/02/2018
- **Using a max of 5 slides, demonstrate the developed project and the projects findings**			
Draft Slides Submitted	26/02/2018		05/03/2018
Report and product submitted	05/03/2018		23/04/2018

Appendix B:

https://stummuac-my.sharepoint.com/:b:/r/personal/15063379_stu_mmu_ac_uk/Documents/Faisal_Project/MMU-Ethics-Check-List-v9-10-Jan-2017%20(2)%20(2).pdf?csf=1&e=xGEWlu

https://stummuac-my.sharepoint.com/:w:/r/personal/15063379_stu_mmu_ac_uk/Documents/Faisal_Project/MMU-Research-Insurance-Checklist-v1-0-19-Sept-2016.docx?d=wff1ff656dadb40a8899794fdd25a14e7&csf=1&e=6yEAtU

https://stummuac-my.sharepoint.com/:b:/r/personal/15063379_stu_mmu_ac_uk/Documents/Faisal_Project/RA_Project_SoftwareDevelopment_171016%20.pdf?csf=1&e=OIGbgC

Appendix C:

https://stummuac-my.sharepoint.com/:u:/g/personal/15063379_stu_mmu_ac_uk/EdAve5G7kRVKkczA5KvNk1cB70viNEYjcuFV-b4EsSiVDQ