A Cognitive Radio based solution for travel time reduction in Smart Cities.

Produced by [redacted] [redacted] Supervised by Soufiene Djahel

Dedications:

I would firstly like to thank Soufiene for being so supportive during this project and helping me achieve such a milestone.

I would also like to thank [redacted] for being there for the ride.

[redacted] for providing the path to get here.

Finally my family and loving girlfriend for supporting me and believing in me, even when I didn't believe in myself.

Abstract:

This report will be researching and discussing the many different techniques on how to mitigate traffic congestion.

Traffic congestion, is a problem faced by many densely populated areas all around the world. The effects of traffic congestion are becoming of increasing concern, not only is there a detrimental effect to the economies of these areas, due to delayed or missed events. There is also an adverse effect on the inhabitants health and also the environmental health of the area.

The research will attempt to produce a Cognitive Radio inspired algorithm, Cognitive Radio is network based protocol that can dynamically allocate users to under utilised network bands. The algorithm aims to mimic the qualities of Cognitive Radio on a road traffic network, with the hope that the algorithm will produce a reduction in congestion levels, by controlling access to priority lanes found in many densely populated road networks.

Utilising open source traffic simulation software to produce a wide range of road topologies, to try and effectively show how Cognitive Radio inspired traffic mitigation can be effective in multiple different types of scenarios.

Contents

1 Introduction

Traffic congestion is one of the largest growing problems many cities have to face in the modern day. Due to the accelerated economic growth in city areas this leads to an increase in population, which leads to a densely populated area. In terms of the transport, this generally equates to a larger number of private vehicles operating on the roads [1]. Many cities do not have the capacity, logistics or funding to cope with this increase in demand [2], which leads to a degradation in the road infrastructure and then ultimately increased traffic congestion problems on the road networks.

The effects of traffic congestion are still under research, there has been many studies that have highlighted the negative impact on the local economies as well as the health of city inhabitants, this condition is not simple to analyse and predict due to being influenced by a large range of variables. The Victoria Transport Policy Institute, which is an independent think tank of traffic congestion research, provided some measurements that can be used to measure traffic-congestion, such as Level-Of-Service (LOS) and Travel-Time-Index (TTI). However, these are only useful for measuring the current traffic congestion and not how to predict traffic congestion in a future context [3].

Due to the nature of traffic congestion, it has been shown to have a direct effect on the average travel times of all users on a road network, which naturally leads to delayed or cancelled events, meetings and deliveries [4]. A study conducted by INRIX a leading traffic investigation service and backed by the Texas Transportation Institute's 2012 Urban Mobility Report [11], shown that between the years 2013 and 2030 the expected economic loss due to traffic congestion in UK alone is expected to be a staggering £307 billion [10], further reporting that this roughly equates to an increased congestion per-household cost from £1,426 in 2013 to £2,057 in 2030 [10].

Factors like these further lead to an increase in stress levels of individual drivers, which has been strongly correlated in a study on driver aggression to further escalate the stress and aggression of drivers [5], this can also lead to further chances of road traffic incidents due to lapse in concentration or poor judgements. In a study conducted by the University of California and backed by the Institute of Transportation studies [12], the effects of stress and aggression caused by traffic congestion on the health of city inhabitants were investigated. In particular, this study attempted to measure the effects of stress on the health of individual drivers using an impedance measurement novaco-stress-2, this tried to show how much specific tasks or events increased or decreased this measurement and shown that an increased impedance measurement leads to higher stress levels, which could contribute to poor mental and physical health.

Alongside physical health, vehicle congestion also contributes to the poor natural environmental conditions, such as poor air and water conditions due to increased pollution [1]. This is mainly because a very large percentage of private vehicles still using fossil fuels [13] as their primary power source. When a fossil fuel is burnt it releases toxic bi-products into the environment [14], which coupled with traffic congestion leads to a dense production of these toxins in a dense area such as city, this is a large contributor to poor environmental conditions in city areas.

As discussed afore, traffic congestion is a colossal problem in particular for cities, but it also has wider implications in every aspect of modern civilisation. Although there has been a lot of research into how we can mitigate this problem and its negative impact, we are still a long way from meeting the reality of a truly 'smart city' where road networks can dynamically cope with the ever increasing demand on the road infrastructure.

1.1 SUMO

SUMO is an open source microscopic traffic simulation tool; it is not only a traffic simulator but provides also a suite of tools to aid in the development of traffic simulations [7]. Traffic simulations are the artificial simulation of mathematical models based on a typical road network. Traffic simu-

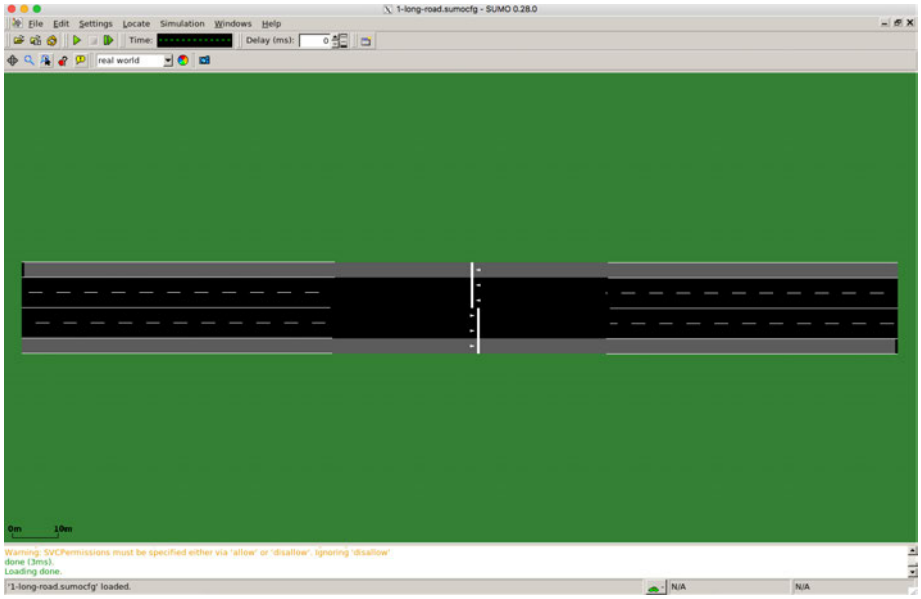

Figure 1: SUMO GUI

lations work based on an array of variables that can be altered to affect the state of the simulation [15]. SUMO was first developed by the German Aerospace Centre (DLR) institute in 2001 [8]. It was the first open source traffic simulator to be produced, the reasons why DLR made the software open source was due to two factors, firstly to ensure their simulation models could be supported and scrutinised in the future [7] and, secondly to also draw interest by the traffic simulation community [7] in order to gain popularity from other proprietary simulators.

In 2012, a study investigated a large sample of academic papers based around transport and traffic simulation to reveal the most used transport traffic simulator. This investigation showeds that in 2010 30% of papers published were using SUMO for their simulations and evidence [16], another 60% of papers did not specify what simulators they were using [16] and the other 6% and 4% were taken by VISSIM and Vanet Mobisim respectively [16]. Based on this results SUMO is now considered the most popular open source traffic simulator.

As reported in [17], Vanet Mobisim is essentially a less feature rich open source traffic simulator than SUMO, only supporting two types of transport modes and a very limited GUI. VISSIM on the other hand is the most popular proprietary option in comparison with SUMO. VISSIMs' target market is large-scale organisations who require an even more feature rich experience than SUMO, such as a 3D GUI and parking management simulation.

SUMO works with standardised file formats, consisting of a XML based representation of the road network topology you would like to simulate and a second XML file that defines the routes the vehicles should take. These files can then be loaded in via two interfaces in order to perform the simulation and produce statistics, the first method is shown in Figure 1, which is a graphical user interface (GUI), which connects the SUMO simulator and provides a graphical view of the road topology along with some interactive tools to aid the user with managing the simulation.

Figure 2: SUMO CLI

The second is shown in Figure 2, a Command Line Interface (CLI) that takes the files as a command line argument and then proceeds runs the simulation without any visual/user interaction.

Once a SUMO simulation is running, it begins to model and simulate a road traffic network based on the files that were provided at run time. SUMO then begins to execute each step of the simulation until it reaches an end point or the users stop the simulation. Each step in the simulation quotes to 1 second of real time [7], the concept of a step provides the ability to manipulate the simulation based on calculations extracted from previous steps.

1.2 TraCi

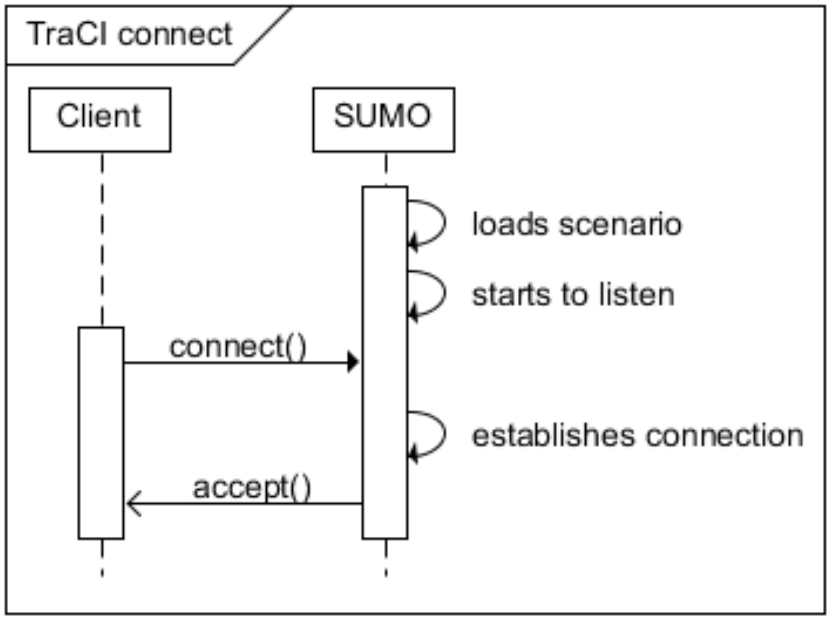

Figure 3: TraCi UML Diagram [48]

Traffic Control Interface (TraCi) is an open source application programming interface (API), implemented by Axel Wegener and his colleagues from the University of Lübeck [7]. The API is an open source extension of SUMO and is now part of the official release. As illustrated in figure 3, TraCi provides a way of communicating with an active SUMO simulation [9], this provides the ability to interact and manage the simulation whilst it is actually running and it also provides a way to retrieve live data from each simulation step. The power of TraCi is what makes SUMO so versatile in the traffic simulation world, as having live data and the ability to modify a simulation on a step-by-step basis provides users with a very powerful tool set in order to experiment with different ideas and techniques, whilst seeing live changes and results based on these changes.

1.3 Cognitive Radio

Cognitive Radio (CR) is a radio wave based communication protocol that is aware of the surrounding environment and can intelligently detect and utilise under-utilised radio frequencies (RF) [18]. CR was developed as a response to how congested current RF bands are becoming, the idea is to try and utilise the entire RF spectrum when a specific RF band becomes congested [19]. CR provides a way for network companies and providers to further utilise current bands more efficiently, meaning less expensive investments are needed into adding more network bands to the current spectrum, CR leads to a better quality of service (QoS) and overall a more reliable user experience, due to the network allowing a greater throughput of traffic.
The fundamental idea of CR will play a large role in the research, due to the principle of under utilised radio frequencies being akin to a similar problem faced by road users in the form of proprietary lanes. A proprietary lane is a lane that has been designated to a specific form of transport, similar to when this restriction is placed on RF bands there is a familiar effect on the road network, specific roads become further congested and over-utilised, whilst at certain intervals proprietary lanes will see very little or no traffic at all.

2 Cutting Edge Research

In this section I will be discussing the most cutting edge research currently being conducted to try and reduce traffic congestion, comparing and contrasting multiple different solutions.

2.0.1 A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches [17]

The IEEE organisation [20], which is the world's largest technical professional organisation dedicated to advancing technology for the benefit of humanity, published a study that attempted to scrutinise traffic management systems (TMS) with the end goal of trying to improve TMS systems by utilising innovative technologies and methodologies.

TMS is an umbrella term for a large complex system of hardware, software and users. A TMS uses all of these mediums to try and efficiently analyse, manage and predict traffic congestion. The primary goal for any TMS is to try and mitigate traffic congestion and create a good quality of service for all road users.

Figure 5 provides a very clear view of the process taken by a TMS to achieve its goal, each phase is equally as important, and all can be improved in some form with either new physical technology or new software methodologies.

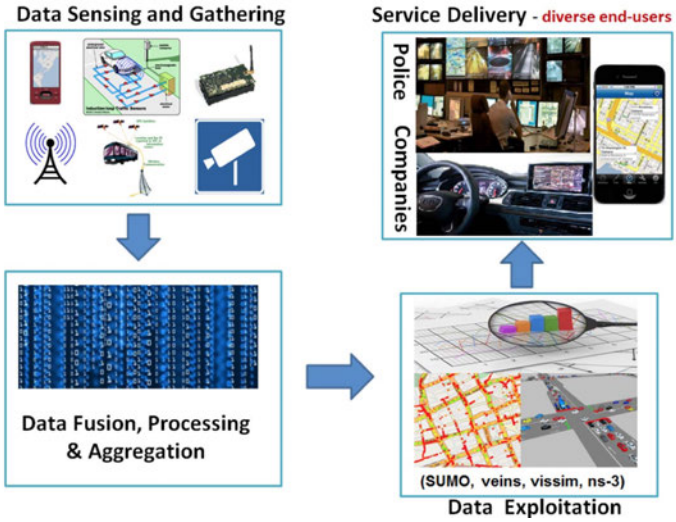

Figure 4: TMS diagram [17]

The Data sensing and Gathering (DSG) phase, is the first and most important of the whole TMS system. The main goal of the DSG to focus on the scalable collection of traffic flow information [17], the paper discusses that the main source of traffic information for many TMS systems is usually a mix of third party hardware and software systems that are difficult to maintain and integrate together. This is due to the fact that each system or interface used by a TMS has a specific set of standards or rules that do not integrate well with other systems, leading to a very costly TMS platform for organisations to maintain and extend, this is one area where improvement is needed to provide a future TMS. A solution suggested by the paper is that modern TMS should rely less on third party systems and more on existing or open source data aggregation services/hardware.

The rise in capabilities of Machine to Machine (M2M) technologies [17] such as the recent uptake of the Internet of Things (IoT) [21], which provides a platform to interface with an array of different physical sensing devices, and the rise of readily available mobile technologies such as smart phones, provides a perfect platform to make these improvements possible. The paper suggests that future mobile platforms like these could help improve the speed and reliability of how a TMS can

collect and process data due to the devices being compatible with more modern telecoms/communications methods such as GPS/3G/LTE, the only issue this poses is the costs associated with relying on these networks introduces.

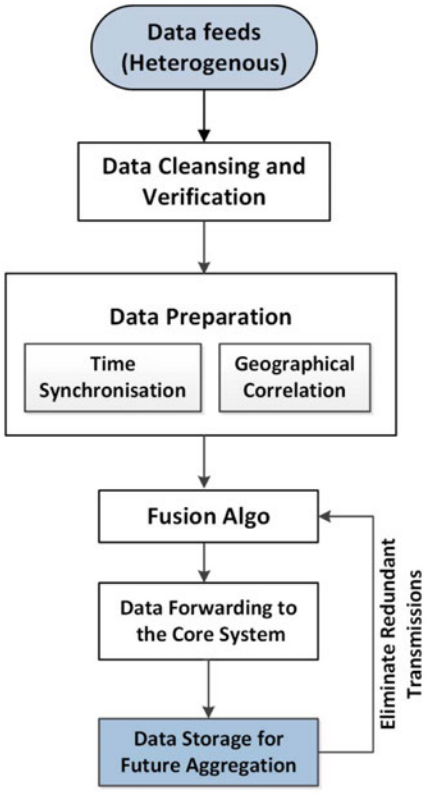

Figure 5: DFPA diagram [17]

Phase 2 of a TMS is the Data Fusion, Processing and Aggregation phase (DFPA) is responsible for managing and cleaning the data collected by the DSG phase. As mentioned previously, the DSG phase suffers from data integrity issues due to the sheer scale of heterogeneous data sources, this issue also persists in the DFPA stage. Figure 6 shows the steps taken by the DFPA in order to firstly try and sanitise the data and the secondly try to produce some usable statistics that can help in the prevention or mitigation of traffic congestion. Once the DFPA receives the data, it must first sanitise the data to ensure the data is valid and in a single usable form, it applies many methods to do this based on sources of the data types it has been provided.

The paper states that a modern TMS should have the ability to quickly aggregate data from these heterogeneous sources, process this data and store it in some format, which can be used as a reference for future predictions. The Traffic Management Data Dictionary (TMDD) developed by the Institute of Traffic Engineers [17], is a standard that describes the data concepts for traffic data, metadata, network devices and events [17]. The reason why TMDD is so important as its primary goal is to try and reduce the complexity introduced by multiple heterogeneous systems and ultimately speed up the computation and collection of statistics. This helps provide faster and more useful traffic predictions and results, the concepts of TMDD have already been applied to many new novel research ideas in the attempt of finding more useful ways to predict traffic congestion. Although steps have been taken to try and reduce complexity and computation time, DFPA can still benefit from more standardised data formats or quicker more real-time data aggregation devices to improve the speed and efficiency of providing useful statistics to the end users.

Data Exploitation and Service Delivery are part of the same phase in a TMS, once the data has been processed in the TMDD, it can then be further manipulated or delivered in many different ways. Some examples of these services are vehicle routing, traffic prediction, parking management,

traffic flow and infotainment systems [17]. The paper discusses many of these services in detail, but to give an example of how these services could be improved by a modern TMS, the paper discusses how a modern TMS could improve vehicle routing.

Systems such as GARMIN [22] and TOMTOM [23] navigation systems have provided modern drivers with very powerful new technology that can make journey planning a much easier task, these services offer automatic route planning based on a users GPS location using a satellite navigation (SATNAV) [24] device. The device then computes the optimal route based on the users preferences and location to their desired destination, the problem with these devices though is that they are not context aware, they cannot predict traffic flow, accidents or road closures on their own. The way these devices provide these technologies are by utilising modern TMS systems and , which can quickly and accurately provide the SATNAV with up to date information based on the route the device has decided to follow, this allows the device to optimise the route in order to provide a better QoS. This is only a very trivial example of how optimised smart TMS systems can aid the improvement of road traffic congestion in future smart cities, by utilising M2M technologies coupled with more efficient data aggregation and processing, this gives authorities much more control over managing traffic and reducing congestion.

This paper provided a great insight into the future of smart city transport systems and the processes that govern these systems. The concept of a TMS and how each process is clearly defined allows me to relate to where my research will impact these systems and how useful this impact could be. It also raised presented me with some useful questions to answer in terms of how secure future TMS systems will be and how robust new protocols and technologies must be.

2.0.2 Adaptive traffic management for secure and efficient emergency services in smart cities [25]

A paper by researchers from UCD in Ireland, investigates the effects of modern Traffic Management Systems (TMS) systems integrated into smart cities on the efficiency of emergency service response times and how these new technologies can help reduce the response times in order to improve the QoS and success rate of the emergency services.

The paper states that the effects of slow response times from any emergency service such as Fire, Police or Ambulance leads to human and financial loss for the local community. This could be due to slow response times to a serious accident by an ambulance service, or a slow response time from a fire department trying to save an important building. Along side the effects of slow response times there is also the dangers introduced by fast response vehicles trying to navigate congested roads, this introduces a very high risk of a collision. In the USA alone during 2009, 918 injuries and 60 deaths were a results if emergency vehicle accidents [25].

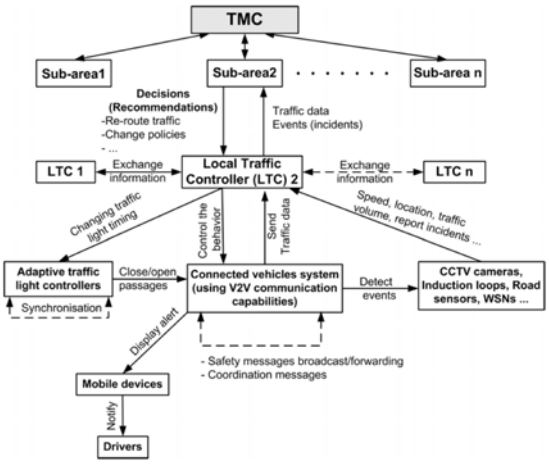

Figure 6: A UML representation of how the traffic prioritisation system works [26]

The solution the paper offers is to integrate emergency vehicles into the TMS systems utilising Traffic Management Controllers (TMC) and Local Traffic Controllers (LTC) along with the Vehicle to Vehicle and Vehicle to Infrastructure (V2V/V2I) communication [25]. As shown in figure 7, these systems will integrate into the entire TMS system and serve specific local areas to ensure a distributed and manageable system. Once an emergency vehicle is requested, the vehicle notifies the LTC of the incident, which can then decide based on the severity level of the incident and the location what the best options are to take.

To decide on the best route the LTC communicates with both the TMC and uses V2I communication to retrieve information about the state of the road network, such data like road congestion, traffic light status and whether any cars may be broken down. Once all of this information is collected it can then begin to take appropriate actions such as ensuring all other LTC on the route are notified of the incident in order to create a green light passage for the emergency vehicle and also notifying all other vehicles of the appropriate action they should take such as reducing speed or pulling into a lay-by.

The V2V and V2I communication protocols are an important technology discussed by this paper, as V2V communication is still under utilised in many modern TMS systems, the idea that vehicles can communicate between each other, without relying on any form of controller could be very powerful in future congested urban spaces. The paper discusses the IEEE802.11P MAC protocol [25], which is a networking protocol. The protocol could be used to implement V2V communication in the future, but it suggest some requirements will be needed in order to cope with the large number of data broadcasts and packets produced by large congested networks. If this protocol is implemented it could lead to a more real time TMS system with a lot less latency due to the direct communication between vehicles.

This paper was very useful for studying a TMS in more granular detail; the fact that the paper was also structured around a real world problem makes the research more meaningful and useful. The paper did not however discuss the costs or logistics associated with integrating these technologies.

2.0.3 Context-Awareness and Collaborative Driving for Intelligent Vehicles and Smart Roads [26]

Researchers at the University of Klagenfurt produced a paper studying intelligent vehicles and smart roads. The research investigated the most effective approaches for both Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I)communication techniques and how we could use these to make the roads safer in the future.

The paper states that in the year 2000, 40,000 people were killed and a further 1.7 million injured as a result of road accidents, with an estimated cost of around 160 billion euros to the EU economy [26]. The problem of road safety is clearly a large issue and the EU fully understands this and have set a Transport program to research and provide new solutions to mitigate the problem, with an emphasis on information and communication technologies (ICT) [26].

The researchers try to propose one solution, which is a context-aware Driver Assistant System (DAS). The idea behind context-aware DAS systems is to be able to provide the vehicle with informations regarding its current geography, utilising V2V communication in more dense urban areas and V2I communication in areas where congestion is not an issue.

With regard to the V2V communication, an example of this technology would be crowd-sourced road conditions. When a vehicle is travelling on a road and it suddenly experiences traction loss due to ice forming on the road, that vehicle can then use a V2V protocol to broadcast a warning to vehicles nearby that can then adjust their speed or route accordingly, preventing any accidents occurring.

When V2V is not a viable solution, for example in rural areas the other solution is V2I. This

solution is more costly than V2V due to the nature of deploying a large array of sensing devices across a large scale road network, but is more reliable in areas with less congestion or population density. One example of how V2I would work is an example where a car is approaching a blind bend, the V2V/DAS systems cannot see around the corner and therefore will continue to drive following the roads speed limits and rules, but around this corner is a tree that has fallen over blocking the route. The V2I sensing devices such as cameras, traffic lights and congestion sensors could analyse this road segment, notice that an obstruction has occurred and then broadcast an alert to local drivers and authorities to prevent any accidents from occurring and to also try and get the blockage removed.

This research further backs the power that can be achieved using V2V and V2I communication techniques inside of a modern traffic management system. The technology will only continue to improve and innovate leading to further more efficient and powerful tools.

2.0.4 A Distributed Algorithm for Adaptive Traffic Lights Control in Wireless Sensor Networks [27]

A paper produced by a group of researches from TELECOM-ParisTech [28], studied the current cutting edge technology available in traffic light management. They proposed to try and produce a more effective algorithm to control traffic lights at individual intersections, using cheaper and more distributed technologies.

Traffic Light Controllers (TLC) are devices that manage the intersections of roads, by controlling each set of traffic lights that manage each intersecting road [27]. The researchers wanted to integrate these controllers using a Wireless Sensor Network (WSN), which is essentially a network of small sensing devices communicating together over a multi-hop network. The benefits of implementing a WSN is that it relies less on costly infrastructure, it is easier to maintain and the TLC can dynamically respond to new events with much less latency, due to not necessarily relying on a central server or controller.

A traditional TLC network, uses expensive induction loop technology [29], to detect when vehicles leave and enter a specific area. These devices have to be installed by the road maintenance department and connected to a central base station, which is then pre-programmed with a TLC algorithm. This means that the algorithm is static and cannot dynamically react to different traffic or congestion issues, the algorithm only knows when vehicles have been detected by the induction loop system.

The research paper discusses the idea of performing traffic analysis by forming a WSN from a multitude of different sensing devices installed at the intersection. These devices are generally much cheaper to install due to relying on wireless protocols for communicating and the devices are also cheaper to manufacture because they are usually smaller in size. Once a selection of sensors have been installed at an intersection, they can then form a WSN and begin to communicate between themselves, or with a central base station.

The devices on the WSN can begin to try and predict and analyse the current traffic flow and utilise measurements from a range of different sensors on the network to try and better predict how to mitigate the traffic congestion. The WSN also allows the network to avoid an overloaded network, due to all devices being able to communicate through the multi-hop network, rather than overloading a specific radio band and causing network dropouts for other devices in the area. If the local TLC does need to connect to the Wide Area Network (WAN), this is still possible as the devices can still communicate with their local base station that can then contact other base stations the area in order or a central service, in order to share information. But utilising the WSN over these base stations, means the TLC can rely less on slow responses from the WAN and more on the quick responses provided from the sensors in order to try and mirage the traffic congestion.

To investigate their results, the researchers used the SUMO simulator to try and model multiple different TLC algorithms and they found that their custom algorithm, which relied greatly on

WSN performed faster and more efficiently in terms of response time on the network. The study also raised some key issue with WSNs in terms of legal and political issue in terms of the data collected by the sensing devices employed on the intersections, such data collected by these devices could be used by malicious parties, so strict security policies must be in place.

2.1 Conclusion

The research shows that traffic congestion is a large problem that every country is going to have to face at some point in the near future, with some countries already beginning to see the damages it can cause. Not only is traffic congestion a problem that must be solved, it is also a problem that must be widely researched with careful consideration into what steps must be taken to try mitigate it. As the previously mentioned research has shown, we do have ways that can improve the traffic situation in major cities, but the costs and ethical factors are not researched widely enough to give these cities the confidence to invest. Further research must be undertaken to try and consider new and more cutting edge technologies, which can be deployed to reduce traffic congestion in the future. The research I want to undertake will hopefully add to this future solution, applying already proven radio frequency network principles directly to the road networks to see if the principles can also help with the reduction of traffic congestion. Whilst further studying the most appropriate solution to how these principles could be applied in future real world situations.

3 Solution Design

In this chapter I will be discussing the design details of the proposed **C**ognitive **R**adio **I**nspired **T**raff**i**c **C**ongestion algorithm (CRITIC), referring to what tools and techniques I will be using, furthermore what features I will be implementing with the these tools.

After researching and analysing previous cutting edge research into the problem of traffic congestion, I am proposing to implement a Cognitive Radio Inspired Traffic Congestion algorithm(CRITIC) in order to try and better manage traffic on the road network. The approach will be to implement an algorithm that allows for a central traffic manager such as a Traffic Light Controller (TLC) [27], to manage the road network utilising the CRITIC algorithm. The TLC type interface is needed to communicate with vehicles on the road network and decide on most effective lane guidance strategy for each vehicle to use within a certain set of rules or heuristics. The ultimate goal is to try and utilise the current road network more efficiently without affecting any local laws, such as bus lane priority laws or cycling laws.

3.1 Development Process

To begin implementing the proposed solution, an appropriate development cycle must be decided upon in order to effectively produce, test and improve the solution over time. Two common methodologies used today to aid in the production of new products and services are The Waterfall Model [30] and Agile software development [31]. Agile and Waterfall are widely used within the software development industry with Agile becoming the more used of the two in recent years [32], both have positive and negative points that I will try to highlight below.

3.1.1 Waterfall

Figure 8 shows the sequential structure of the waterfall model, where each step flows into the next step in a linear direction. The fundamental idea is the entire process can be split up into individual steps [30], with a predefined order to the steps. In software terms, this means the requirements would first be finalised before any development could take place.

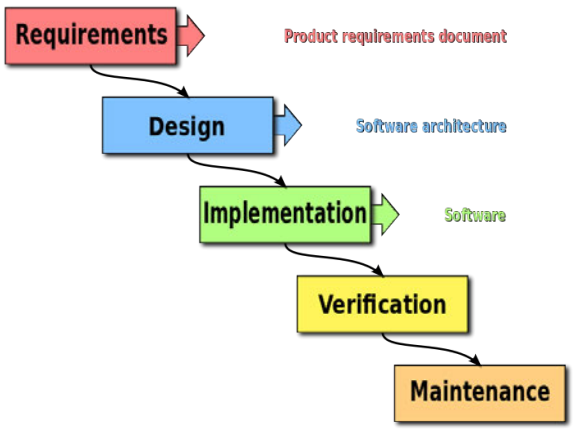

Figure 7: Waterfall model [30]

Advantages:

- In terms of business; the waterfall model provides a key benefit to the development cycle, as each step in standalone, this allows for the business to departmentalise [30] the entire development process.
- Another advantage of the waterfall method is that by design it requires very rigid requirements to be in place before the next step can begin. To a business this can be very appealing as it means they can more efficiently predict and manage the development process in the future.

Disadvantages:

- Once product requirements have been finalised, the waterfall cycle demands that these do not change in order for the next process to proceed. This means that if a client or an unexpected problem arises down the line, the entire process in theory must be restarted in order to re-evaluate the requirements.
- In relation to the previous disadvantage, if the requirements keep changing, then the product could easily run out of time and budget even before anything is built.

3.1.2 Agile

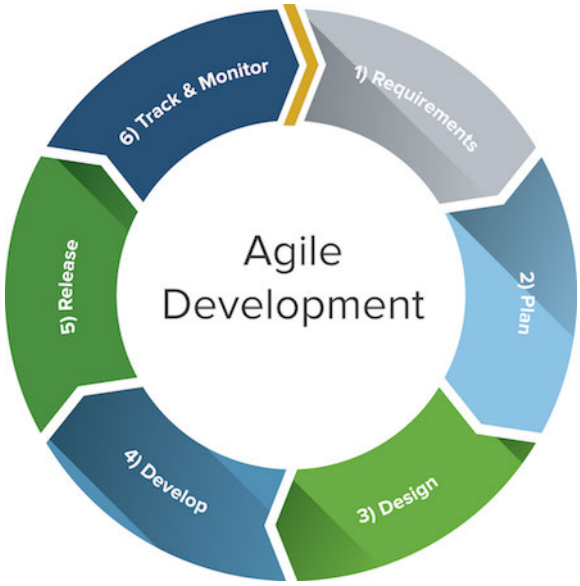

Figure 8: Agile model [31]

Figure 9, shows the visual representation of how the Agile methodology works. Agile is iterative[31] and each stage of the development cycle feeds another, this promotes rapid feedback and adaptation of the requirements within the development cycle. If the requirements change, this can then flow back down the development cycle as it's generally much quicker than the waterfall model. Agile itself is very rarely used within organisations, it is usually taught and used in other formats such as SCRUM[31].

Advantages:

- Iterative requirements; unlike the waterfall model, requirements can change throughout the project and these requirements are then quickly passed into the other stages, providing customers with a much more flexible way of providing requirements as the project grows.
- Reduced costs, because Agile methodologies are so quick and iterative, if a business detects something isn't working as expected or something arises that blocks progress, the requirements can change and therefore less time and money are wasted on the problem.

Disadvantages:

- Agile if applied poorly can struggle to work in large scale organisations, due to less of a departmentalised approach. Teams can become less focused on the end goal and could lead the risk of wasting resources on unnecessary changes or requirements.

For the research an AGILE approach will be applied, this is because although the research has a defined goal, the goal is still not fully realised due to it being a novel idea. This means that the product will have to be constantly refined and improved, also possibly entirely re-factored as

the research progresses and more data is produced. AGILE provides the most appropriate design model for this type of problem, with quick design and improvement cycles and a greater tolerance to change.

3.2 Understanding Cognitive Radio

In order to implement an effective algorithm that inherits from the ideas behind Cognitive Radio(CR). We must first understand how CR functions, and how it's basic principals can be applied to the traffic management domain. In an article wrote by Telecom ParisTech [35], which discussed the basic principles of CR, the article discussed these three key points:

- To obtain knowledge of its operational and geographical environment, established policies, and its internal state (cognitive capability). [35]
- To dynamically and autonomously adjust its operational parameters and protocols according to its obtained knowledge in order to achieve predefined objectives (reconfigurable capability). [35]
- To learn from the results obtained (learning capability). [35]

The first point, discusses how CR attempts to analyse the current network state to ensure it can understand key information needed in order to monitor the performance of the network and also what features it can manipulate to try and improve the network performance. An example of this would be to firstly try and calculate how busy the current network band is, in order to gauge whether on not it is becoming congested. This is directly applicable to a road network simulation, the same concept could be applied to the *lanes* of each *road*, working out if any lanes are congested and taking the appropriate action to reduce congestion.

The second point further discusses how once the network manager has analysed the network and discovered the necessary information, how it can then proceed to make optimisations to the network, in order to try and improve performance or reduce congestion. In the terms of radio networks, this could be the adaptation to another radio frequency that is currently under-utilised, in order to more efficiently utilise the whole radio network, improving the throughput. Again, the same types of optimisations can be applied to a road network, a traffic analysers could try to gauge the congestion in each lane on a road segment and then try utilise other lanes on this road segment more efficiently, for example allowing non-priority vehicles to use a priority lane if this priority lane is under-utilised.

The final point mentioned attempts to provide a way for CR to be able to learn from the information it has gathered from the two previous steps, in order to try and better service the network in future circumstances. An example of this in terms of a road network, would be to keep track of times when priority vehicles were affected by standard vehicles using priority lanes, trying to better understand what variables caused this to happen. In future circumstances the CR algorithm can then try to avoid these scenarios in the future, making the algorithm more efficient and causing less delay to priority vehicles.

3.3 Cognitive radio like road network management

For CRITIC the research will be attempting to implement the first two key points presented by the CR specification, which are to gain knowledge of the geographical environment and to dynamically adjust operational parameters. Implementing the final stage of CR would require much research into the appropriate data aggregation and classification techniques, such as artificial intelligence (AI) or Big Data(BD) techniques, if CRITIC were to be used in the real world, the final stage would need to be implemented in order to effectively assess real world performance. The reason the final stage is not needed in this research is due to the project running in a simulated/sand-boxed environment, which means the input and outputs can be controlled and analysed manually, this is where the data will be assessed to improved future simulations.

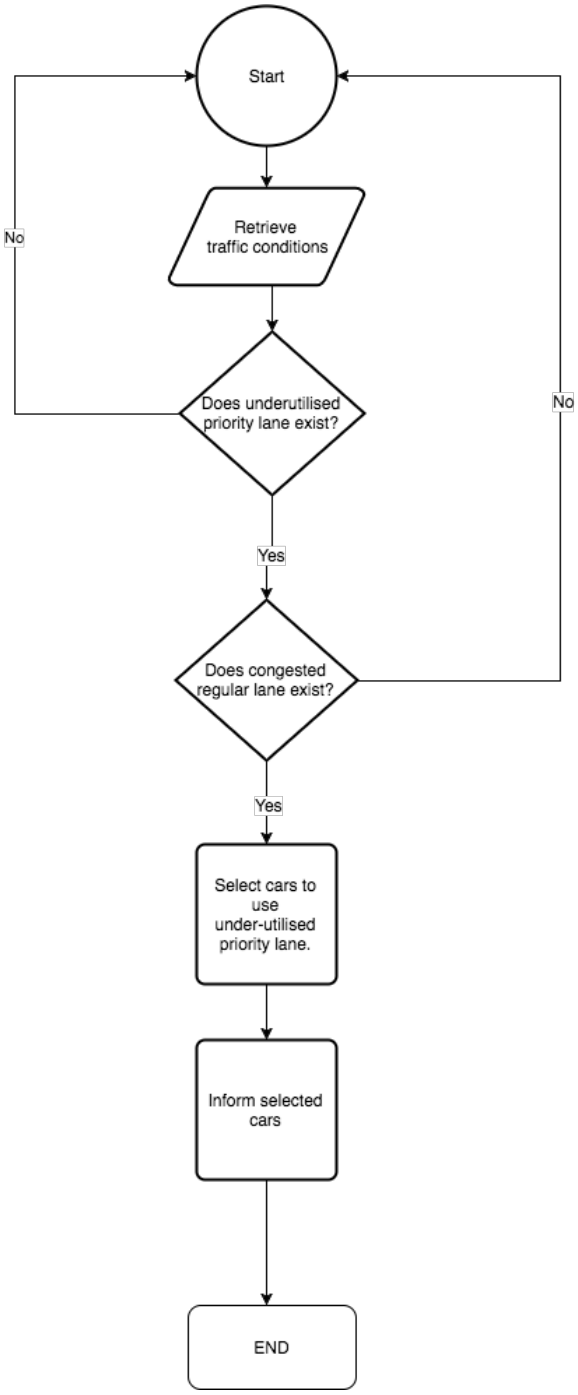

Figure 9: A flow chart modelling TMS action within CRITIC

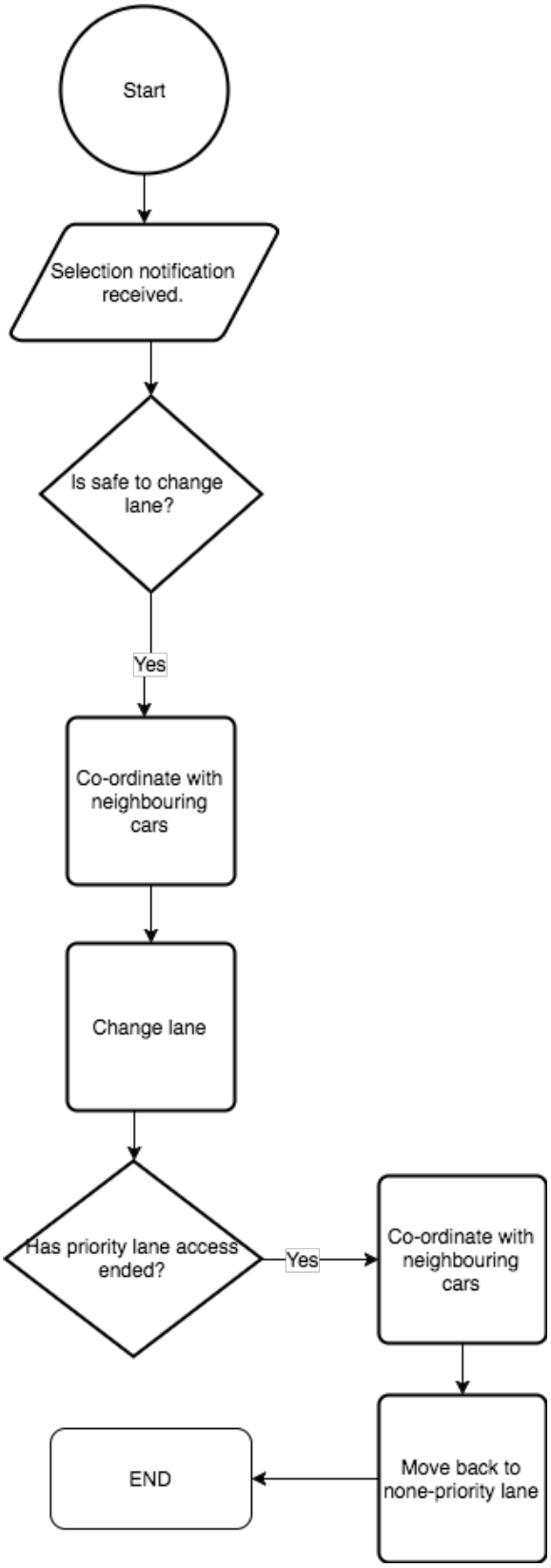

Figure 10: A flow chart modelling Vehicle action within CRITIC

In order to provide a visual representation of the solution, with a clear definition of the proposed logic needed to simulate CR on a road network, the logic has been displayed in the form of a flow chart. Figure 9 is the first section of the algorithm, which represents the logic needed to be

performed by the TLC. The TLC must first assess the road network to retrieve the current road conditions, once the network has been assessed, both an underutilised priority lane and a congested non-priority lane must exists, if both of these conditions are met, then the algorithm should begin to mimic CR. In order to mimic CR, the algorithm should notify the users of the road network that they now have permission to use the priority lane, this could be for a specific period or until further notified.

Figure 10 shows a flow chart depicting the actions needed to be taken by a user of the network once a TLC request is received, the main feature a user needs to ensure is that no collision occurs during the time taken to move to a new lane, this is an additional feature that is required compared with the traditional CR algorithm, as in terms of virtual networks, collisions do not cause as much of an issue than on a physical road network. Once a vehicle receives a request from the TLC it must first ensure it is safe to change lanes, once the lane change has been received, it must then further monitor that it still has the necessary rights to utilise the priority lane, if those rights are revoked then the user must then ensure a safe lane change can be made back to a none-priority lane.

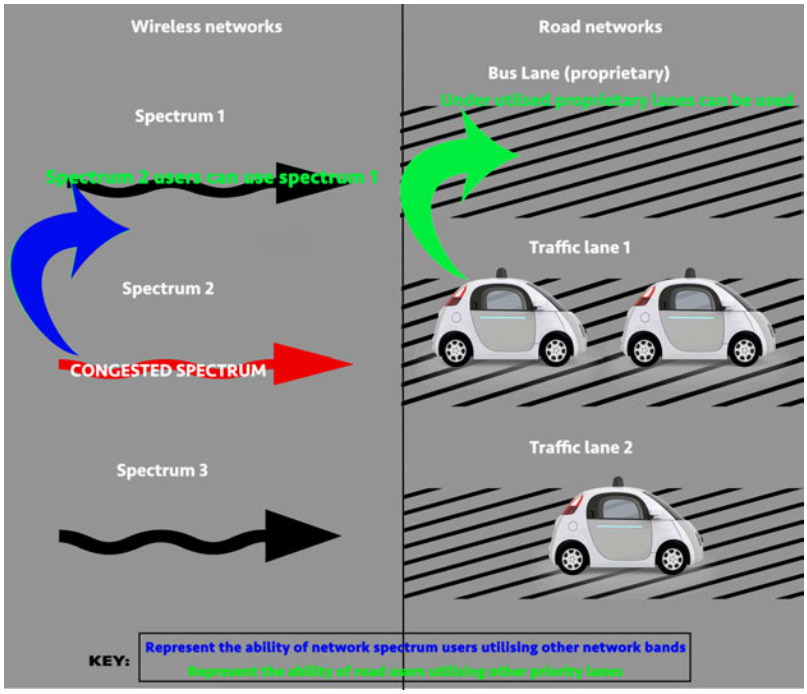

Figure 11: Illustration of CR functionality on the roads

The figure pictured above shows a more in depth illustration of the similarities between CR and CRITIC, on the left hand side of the diagram the picture depicts what a CR network would look like and the actions taken by CR to further utilise the network bands, by allowing users of the congested spectrum 2, to utilise spectrum 1. The right hand side of the illustration shows how well this concept can be applied to a road network with priority lanes, here traffic lane 1 is congested and the CRITIC algorithm allows users to utilise the bus lane in order to reduce congestion in other lanes.

3.4 Design tools

3.4.1 Python

Python [34] is a dynamic, high-level multi-purpose programming language that has been widely used since the 1990s since it's first release. Many people consider it to be a simple scripting language, but with such a vibrant developer community, it has become much more than that, with lots of powerful extensions and packages being added to the language on a daily basis. One large advantage Python has over other languages is the Python interpreter, which allows for very fast-paced development and debugging, due to the language being interpreted and not compiled. Python is also widely supported by the SUMO community, with many of their tools being built using the Python language, this further adds support for utilising the language within in this project.

3.4.2 SUMO

As previously discussed, the research will rely heavily on the open source traffic simulation software called SUMO. This software will provide the necessary tools to generate and execute many different network scenarios in order to evaluate the proposed algorithm.

3.4.3 Traci

SUMO supports many Python libraries and one of the most popular and an essential part of this research is the Traci API [9]. Due to the features Traci provides, it will play a key role in the development of the research. Without Traci there would be no way to manipulate a SUMO simulation dynamically, therefore the algorithm would be extremely difficult to implement as the SUMO source code would have to be modified, which presents further more complex issues and challenges.

3.4.4 Minitab

Minitab is a widely used piece of data analysis software, the software provides a wide range of statistical tools help quantify, manage and visuals data sets [38]. This piece of software will aid greatly during the testing of the CRITIC algorithm, ensuring clear and accurate results can be produced for each simulation.

3.5 Conclusion

Overall this section discusses the proposed logic of the CRITIC algorithm, discussing how to implement each section and the tools necessary to achieve this. With this clear design process it should enable quick development to an early prototype.

4 Performance Evaluation

During the evaluation section the steps required to convert the previously discussed algorithm into a working product will be discussed, along with any challenges or modifications that had to be overcome in order to achieve this. Along side the design process the efficiency of the solution will also be analysed, to provide evidence on how the algorithm performs under different scenarios and demands, both the implementation and testing will require the discussion of many concepts and tools, which are further discussed throughout.

To evaluate the efficiency of the proposed Cognitive Radio(CR) based algorithm a plan to simulate an array of different test environments using SUMO os proposed and to later process the data produced by SUMO using Python and Minitabs. Using this aggregated data an attempt to produce a range of statistics and metrics will be made, that will try to provide evidence that the algorithm is having a positive effect on the simulation.

As mentioned afore the algorithm is a CR inspired approach to traffic management, CR attempts to more efficiently utilise a radio network by using all of the available radio wave bands on a given network, increasing throughput and overall quality of service. The initial test approach is to try and produce a bespoke simulation, which can appropriately show how the logic utilised by CR can be applied to a road network. Once the initial test case has been implemented and the algorithm is proved to be working as expected, It will then be possible to produce more larger and complex networks, to produce more reliable and interesting results.

4.1 Road network implementation in SUMO

4.1.1 Network Configuration

As the simulations are implemented using the open source SUMO framework, a vast array of open source tools provided by SUMO were utilised, one of which is called NETEDIT (36). NETEDIT provides a graphical user interface(GUI), with the main purpose of aiding in the productions of bespoke simulation environments. It also provides the ability to view and manipulate existing simulations, Figure-12 shows an example of the interface presented to the user when building a network.

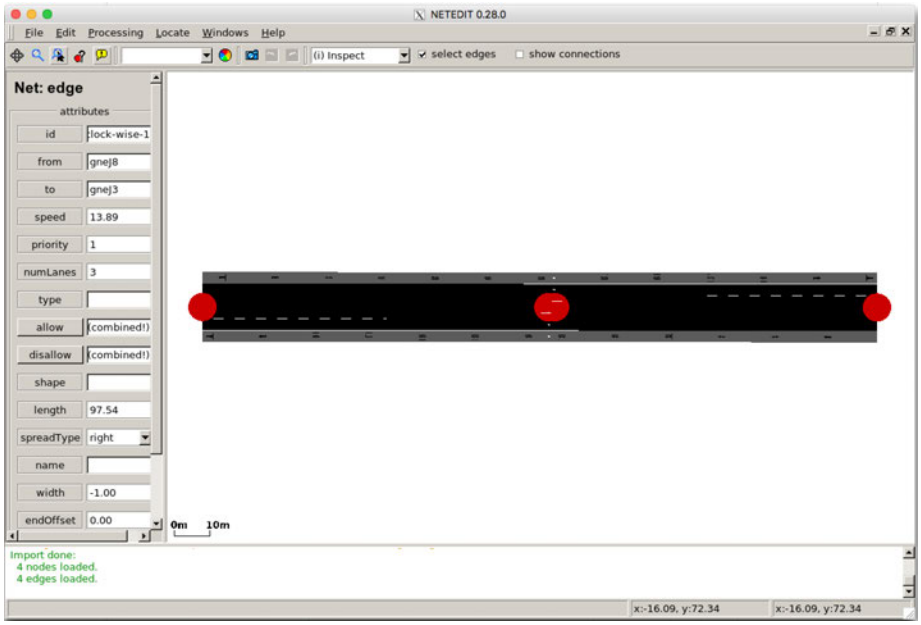

Figure 12: Image showing the NETEDIT GUI

```
    <edge id="anti-clock-1" from="gneJ3" to="gneJ9" priority="1">
        <lane id="anti-clock-1_0" index="0" disallow="bus" speed="13.89"
↪ length="99.44" shape="-63.75,54.28 35.69,54.22"/>
        <lane id="anti-clock-1_1" index="1" disallow="bus" speed="13.89"
↪ length="99.44" shape="-63.75,57.58 35.69,57.52"/>
        <lane id="anti-clock-1_2" index="2" allow="bus" speed="13.89"
↪ length="99.44" shape="-63.75,60.88 35.69,60.82"/>
    </edge>
```

Listing 1: XML representation of a road network

SUMO primarily supports XML based files for configuration and statistics, XML files can become extremely large and difficult to maintain. One benefit to using NETEDIT is that the network files produced are also XML based files. Listing 1 shows an extract of the data produced from NETEDIT once a user saves a custom network, this data can then also be utilised by SUMO to build a simulation environment without any modifications. The files consist of information such as the number of roads, which SUMO refer to as *Edges*, the number of lanes on each road, what vehicles can drive in each lane, junctions and even speed limits.

The SUMO framework also provide an additional tool for generating map topologies without any GUI, this tool is called NETGENERATE [40]. NETGENERATE provides a CLI interface to aid in the production of SUMO network files, the interface takes an array of different options, which all have a different effect on the network produced. NETGENERATE provides 3 key types of networks that will be discussed later during the testing/evaluation stage, these three network types are Grid, Abstract and Spider networks. Each network type tries to imitate common road networks found in the real world, providing additional control to configure and expand these network topologies to larger and more complex adaptations.

```
    <vehicle id="0" type="privateType" depart="0.00">
        <route edges="3/0to2/0 2/0to1/0 1/0to2/0 2/0to2/1 2/1to2/2
↪ 2/2to1/2"/>
    </vehicle>
```

Listing 2: XML representation of a vehicle route

Along with XML data describing the physical topology of a road network, SUMO also requires another XML based file, which is a *route file* providing additional information regarding the traffic flows it should simulate on the provided network, consisting of data such as how many vehicles should be used, the routes these vehicles are allowed to take and also further information regarding the types of vehicles that should be simulated, this data is used to then produce moving traffic on the simulated network.

Traffic data for a simulation relies on an understanding of the generated network topology, such as the lanes and how lanes are interconnected, this makes producing traffic data very difficult to generate manually or in large quantities. SUMO provides another tool to help mitigate these issues called randomTrips [41], this tool is built for generating custom traffic flow data on a large scale. RandomTrips is a command line tool that takes the generated network file as an input along with several user defined parameters, once randomTrips has parsed the network and parameters, it then begins to generate vehicles for each time step of the simulation. The time step is one of the parameters the user must provide. The final product produced is an XML file consisting of the same data depicted in Listing 2, with multiple different trips defined across the simulated network.

The ability for all of these tools to work together provides a very efficient and reliable tool set to work with, allowing many simulations to be built and tested in a very AGILE manner. Without

these tools It would not have been possible to test the solution as rigorously, meaning the evidence I present later on in this chapter would have been less accurate and reliable.

4.2 CRITIC implementation

Once the network and route files have been generated, the next phase of the process was to implement the CR logic into the simulations. Once SUMO constructed a network utilising the previously generated files, the next step was to prototype the first novel solution utilising the Traci API to connect to the simulation environment and begin to manipulate the data. During a simulation SUMO utilises the concept of *steps*, each step equates to 1 second in real time (1000 ms) [36], in order to advance the next iteration of the simulation you must progress the current time step. If SUMO is run on it's own with no interaction from TarCi, the time step is advanced automatically, if TraCi is used however, the simulation must be manually advanced via the API. TraCI harnesses the concept of steps and provides an array of different methods in order to modify the current state of a simulation before the next step is called, this is referred to as the *control loop*. In order to simulate CR on a road network, the concepts of steps and the power of TraCI to perform calculations based on the current state of the road network were required to be used, the algorithm then performs the necessary changes to the network and the simulation is advanced to the next step.

Figure 13: SUMO GUI visualisation of the XML data for a single edge(road)

Figure 13 shows the visualisation of the first network I decided to implement, consisting of two roads each with three lanes and one of those lanes being a priority lane. This basic simulation allowed me to build and iterate on the logic proposed in the design section, in order to fine tune and improve the CRITIC algorithm. Each road in this example would be passed to the algorithm and processed in order to deduce the required data previously mentioned. Once the data has been processed the control loop will then continue to check each priority lane during each step, In order to understand when it is best to disable or enable this lane, better improving overall traffic flow for both non-priority vehicles and priority vehicles.

4.2.1 Connecting to a simulation

```
175             traci.start(["sumo", "-c", "test-edge.sumocfg",
    ↪  "--tripinfo-output", output_file, "--seed", str(i)])
176             run('test-edge.net.xml', options.algorithm)
```

Listing 3: TraCi command to instantiate a SUMO simulation

Listing 3 shows the entry point for the CRITIC solution, the code shows how the TraCi API instantiates a SUMO instance, whilst also passing in the necessary parameters for the simulation to run. Along side TraCi there is also the *run(0)* function, this is a custom function that gets executed after the TraCi command is issued. The *run* function is where the *control loop* is implemented within the python script, which is the main point of execution for a SUMO simulation, the run function takes the name of the network file currently being simulated, which is used for parsing and also a boolean flag to determine whether CRITIC should be run using this simulation.

4.2.2 Parsing the network data

Once the simulation had been loaded via Python and TraCi, the first step was to parse the road data to be utilised by the TMS, this data will be used to make decisions based on the logic presented in the flow charts above. Initially I decided to further utilise the TraCi API to try and retrieve the necessary information from the road network, although this was possible to do, due to TraCi being an API interface to SUMO there were inherent performance issues as the number of API calls increased exponentially. To try and mitigate the performance issues investigation into the sumo documentation was required, which found another Python module provided by SUMO called *SUMOLIB* [42]. This module provides a way to parse a network file in Python, rather than utilising TraCi to send multiple requests during the simulation, SUMOLIB parses the entire network file before the simulation begins reducing the amount of requests made by TraCi during the simulations.

```
33      NET = sumolib.net.readNet(NET_FILE)
34
35      #get the names of each road/"edge"
36      EDGE_IDS = NET.getEdges()
37
38      # need to work out priority lanes before simulation to ensure they don't
39      # get over-writen
40      global_edge_lanes = defaultdict(list)
41      global_priority_lanes = defaultdict(list)
42      global_edge_vehicles = defaultdict(list)
43
44      for edge in EDGE_IDS:
45          edge_id = edge.getID()
46          edge_lanes = [ln.getID() for ln in sumolib.net.Edge.getLanes(edge)]
47          global_edge_lanes[edge_id].append(edge_lanes)
48          global_priority_lanes[edge_id].append(get_priority_lanes(edge_lanes))
49          global_edge_vehicles[edge_id].append([])
```

Listing 4: SUMOLIB module, utilised for parsing the network file

The listing above is an extract from the main Python implementation file, this listing represents the steps taken to parse and cache the data before the simulation begins. The first step is to pass in the name of the current network file, in this case it is a global variable called *NET_FILE*, once this has been passed to SUMOLIB, an object is returned containing information regarding the parsed network. The object returned is usable throughout the Python script to retrieve any necessary information when needed, line 36 shows how the code utilised this to retrieve all of the known edges in the simulation, again greatly reducing the number of TraCi calls needed.

Once the network had been parsed, the next phase was to try and store data regarding what lanes in the network had been defined as priority access lanes. SUMO utilises the term of *edges* to define an entire *road*, in real terms a single road can contain multiple lanes, with each lane having different access and speed rules defined, this concept can be seen in Figure 13. The problem faced was determining which lanes belonging to each edge were priority access or not priority access, once this data was known, the algorithm could then begin to manage the priority lanes. SUMO utilises XML data on the defined lanes to specify what vehicles can or cannot access the lane during the simulation, I was able to utilise this data in order to detect if a lane only allowed priority vehicles, if it did then I could save this lane as a priority lane.

As a SUMO simulation changes during each step of the control loop, I had to ensure that the data retrieved at the start of the simulation was cached, in order to be able to correctly utilise the data again during future steps. An example of this was when I first ran a simulation, the known priority lanes were removed after the first iteration, the reason was due to not caching the known

priority lanes before the simulation began modifying the lane data. To fix this issue I added global variables, which were used to keep track of the priority and non-priority lanes before the simulation began, this ensured the priority lanes could be reverted back to the correct configuration in future steps.

4.2.3 Detecting priority lanes

As mentioned in the previous section, once the data for the simulation had been cached, the algorithm required knowledge of the location of priority lanes and none-priority lanes. TraCi or SUMOLIB do not provide any way to retrieve priority lanes automatically, however TraCi does provide a way to retrieve the know access rights of a single lane, utilising this I built a custom function to perform this action.

```
94  def get_priority_lanes(lanes):
95      """Takes a list of lanes and returns priority lanes"""
96      priority_lanes = []
97      priority_types = set(['bus', 'emergency', 'taxi'])
98      non_priority_types = set(['private', 'evehicle', 'passenger', 'truck'])
99      for lane in lanes:
100         lane_types = set(traci.lane.getAllowed(lane))
101         if lane_types & priority_types and not lane_types &
    ↪  non_priority_types:
102             priority_lanes.append(lane)
103     return priority_lanes
```

Listing 5: An example of a custom TraCi method to detect priority lanes

The code above is the custom function written utilising TraCi and Python, the function takes in a list of lanes retrieved by TraCi from a given edge and returns a new list of priority lanes. In order to assess what lanes are priority and what lanes are not, the function utilises two defined *sets*, sets in python are unique lists of values, the sets consists of pre-defined vehicle classes Vehicles classess are arbitrary names assigned to different vehicle types defined in the SUMO documentation [43], the vehicle classes generally map to real world vehicle types used in real road networks, such as 'bus', 'private vehicle', 'taxi'.

For the initial research the main focus was on detecting 'bus lanes', which are very common priority lanes found in the UK, these lanes are reserved for buses, taxis and motorcycles [44]. The initial logic for detecting a bus lane was to define a set containing all 3 classes allowed by a UK bus lane, in contrast to this another set of non-priority classes had to be defined to ensure non-priority lanes could also be detected, this set consists of common non-priority vehicle types, such as *private vehicle*, *electric vehicles* and *trucks*. Once these classes were defined, each lane passed to the function was checked against these types, the function utilises TraCi to retrieve the lanes current *allowed* classes and also converts these to a set, once these are retrieved the function then checks to the retrieved types against the two pre-defined sets. If there is a match found between the priority set and no matches found between the none-priority set, then this provides enough evidence to suggest that this lane is a priority lane. If both of these conditions are not met, then the lane is considered none-priority. Although this function currently only checks for common bus-lane traits, the method can easily be extended with further classes to detect many other forms of priority lanes.

4.2.4 Managing priority access

The next stage of development was to take action on the data retrieved from the road network, this section of the algorithm requires a large amount of custom logic to effectively manage and move traffic across multiple lanes without causing any incidents. As shown in the flow charts earlier, the CRITIC algorithm must be able to detect congestion on a none-priority lane and also

detect no congestion on a priority lane, before the algorithm performs any alteration to the network.

However TraCi does not provide any method to gauge whether a lane is currently congested, meaning there is no metric to accurately judge if a lane is busy or not. One solution would be to gauge the average travel speed for each lane and compare this to the speed limit of the lane, however this is not an accurate way of measuring congestion as many other factors can decrease the speed of a vehicle. For the simulations I decided the best approach was to remove the check for congested lanes as implementing custom logic added unnecessary computation time to the CRITIC algorithm and this is something that can be mocked during the simulation.

```
85  def detect_priority_vehicle(priority_lanes):
86      priority_types = set(['bus', 'emergency', 'taxi'])
87      for lane in priority_lanes:
88          vehicles = traci.lane.getLastStepVehicleIDs(lane)
89          for vehicle in vehicles:
90              if(traci.vehicle.getVehicleClass(vehicle) in priority_types):
91                  return True
92      return False
```

Listing 6: Custom logic used to detect priority vehicles

The above code abstract is the custom logic implemented to detect if a priority vehicle is present on the currently selected lane. When implementing the logic for the TLC(Traffic Light Controller) side of the CRITIC algorithm, much of the real world implementation details do not need to be implemented, as for the research CRITIC will be run in simulated environments, meaning details of the physical mediums utilised to transmit the data to the vehicles in the road network do not need to be assessed or implemented.

The function takes a single parameters, which is a list of know priority lanes on a given edge/road in the network similar to a frequency band in a radio network. The function then loops over each lane and utilises TraCi to retrieve all of the currently active vehicles for the given lane, the function again utilises TraCi to retrieve the class of each know vehicle. This function also utilises Pythons *set* data-type to store a list of known priority classes, if any of the know classes match the class returned for the vehicle, then this vehicle is considered to be a priority vehicle.

In the flow chart implementation of CRITIC the logic suggest that the priority lane should be under-utilised, meaning the lane has free capacity to facilitate future vehicles. Using this approach in the simulation was not possible as again the congestion of a given lane could not be accurately calculated, this meant that when cars were moved into an under-utilised bus lane, the bus was becoming affected by the extra traffic now in the priority lane. In order to circumvent this issue, I decided to simplify the logic and simply check to see if a priority lane was currently being used by a priority vehicle. If the lane was being used then the lane remained a priority lane, if the lane was not then it was switched to a none-priority lane.

```
105 def enable_priority_access(priority_lanes):
106     """Enables standard vehicles defined in 'priority access' to drive in
    ↪  priority lanes"""
107     priority_access = set(['passenger', 'private', 'evehicle'])
108     for lane in priority_lanes:
109         currently_allowed = set(traci.lane.getAllowed(lane))
110         traci.lane.setAllowed(lane, list(currently_allowed |
    ↪  priority_access))
```

Listing 7: Custom function for disabling priority lanes

The code above is the custom logic required by CRITIC to effectively manipulate priority lanes once the appropriate logic is triggered. This function accepts a list of priority lanes that have been selected to be utilised by the CRITIC algorithm, utilising this list the function then modifies the lane to accept additional SUMO car classes, which are actually none-priority vehicle types.

SUMO or TraCi do not provide any methods to override a priority lane in terms of routing, meaning the TLC can not physically move a vehicle into a priority lane, this is due to limitations within the way vehicle classes are implemented in the SUMO source code. This means in order to implement the ability of none-priority users to access priority lanes, the algorithm must modify the physical road network instead of the vehicles. This is the reason for caching the initial road data discussed in chapter **4.2.2**, once the road network is modified by CRITIC, the original configuration must be maintained in case the modifications need to be reverted.

Once the function receives the list of priory lanes to be modified, further API calls are made to TraCi, in order to retrieve the lanes currently accepted vehicle classes After the classes are received the pre-defined classes set on line 107 as merged with the lanes current classes, then a further call to TraCi is made to set the new list of classes back to the current lane, this means on the next iteration of the simulation the lane will now accept all the newly defined vehicle classes

As shown in the flow charts for CRITIC, vehicles need to be able to move over safely and appropriately to lanes, without causing any issue for other network users. SUMO provides in-built functionality for lane changing, that attempts to realistically respect other drivers on the road, by maintaining safe driving distances, speed and ensuring a lane change will not effect any of these conditions. Once a priority vehicle re-enters a priority lane, that is currently being utilised by none-priority vehicles, the CRITIC algorithm must assess this event and issue the appropriate commands to the vehicles in order to restore the original conditions expected from the priority lanes.

Figure 14: None-priority vehicle stuck in priority lane

In order to reverse the commands present in listing 7, the same logic is applied in reverse, which simply removes the none-priority vehicle classes from the priority lane reverting the lane back to a standard priority lane. This step is required to ensure the CRITIC algorithm respects the users of the priority lanes and minimises the impact on the users of those lanes. One issues that was discovered with this simulation is shown in Figure 14, which is some none-priority vehicles got stuck in a priority lane, this was caused by the proceeding priority lane becoming active and therefore blocking the none-priority vehicle from progressing, this caused major delays on some bus routes and effectively jammed the priority lane.

```
125 def clean_priority_lanes(priority_lanes):
126     priority_types = set(['bus', 'emergency', 'taxi'])
127     for lane in priority_lanes:
128         lane_id = get_lane_id(lane)
129         current_vehicles = traci.lane.getLastStepVehicleIDs(lane)
130         for vehicle in current_vehicles:
131             if traci.vehicle.getVehicleClass not in priority_types:
132                 traci.vehicle.changeLane(vehicle, lane_id+1, 0)
```

Listing 8: Custom logic to remove none-priority vehicles from a priority lane

The function above is the solution implemented into CRITIC in order to effectively mitigate this problem in future circumstances. The function takes the priority lanes that are about to be re-activated on the next simulation step and proceeds to manually re-route vehicles out of the priority lanes into adjacent none-priority lanes. The function utilises the TraCi command on line 132 to

issue the lane change to the selected vehicle, this command is designed to respect the safety laws defined in the simulation and further improves the safety of the network as the vehicles change lane, avoiding any collisions.

4.2.5 Routing vehicles

```
138 def update_vehicle_travel_time(vehicles):
139     """Updates the travel time on each route"""
140     for vehicle in vehicles:
141         traci.vehicle.rerouteTraveltime(vehicle)
```

Listing 9: SUMO simulated re-route

Finally, the last step in the CRITIC algorithm was to simulate how the vehicles would manage receiving a transmission to change lane. Although vehicles can be forced to change lane utilising the TraCi API, the computation time along with the complexity of re-routing multiple vehicles at once increases the number of collisions in the road network and also the time taken for a simulation to complete. The code listing above shows the method chosen in order to efficiently simulate the vehicle re-routing. The function takes the list of vehicles currently present on a given edge and issue a re-routing command to each vehicle, this then triggers SUMO to re-calculate the route for the vehicle, this then encourages the vehicles to utilise any new found routes, which also means the vehicles will now utilise the now accessible priority lanes. Although this method is still computationally intensive, the solution is much safer as less manual re-routing is performed, so the number of collisions are decreased.

4.3 Evaluation Scenarios & Metrics

4.3.1 Producing test data

To evaluate the effectiveness of the proposed CRITIC algorithm, multiple test scenarios were required, along with a proof of concept simulation that had been specifically designed to show how the algorithm worked on a road network. As shown above each simulation is ran via a TraCi command, which passes the required files to SUMO, which is turn begins the simulation.

```
167     for i in range(3):
168         output_file = options.filename
169         if options.algorithm:
170             output_file += '-algorithm'
171         else:
172             output_file += '-no-algorithm'
173         output_file += '-{0}.xml'.format(str(i))
174         if options.nogui:
175             traci.start(["sumo", "-c", "test-edge.sumocfg",
    ↪ "--tripinfo-output", output_file, "--seed", str(i)])
176             run('test-edge.net.xml', options.algorithm)
177         else:
178             traci.start(["sumo-gui", "-c", "test-edge.sumocfg",
    ↪ "--tripinfo-output", output_file, "--seed", str(i)])
179             run('test-edge.net.xml', options.algorithm)
```

Listing 10: Python code showing how the SUMO test simulations are run

The code above is the Python code utilised for running a simulation to produce test data, in order to produce reliable results the simulation must be run multiple times with different SUMO seed values, this provides a large selection of data to be analysed, providing more broad results. As mentioned earlier, SUMO utilises seed values in order to ensure simulations can be run again with reproducible results, this means the results produced from a standard simulation that does not utilise CRITIC, can be compared with a simulation that did utilise the CRITIC algorithm. As each simulation is run under two different conditions, one which is a standard simulation called the *baseline* and the other, which is run with the algorithm called CRITIC, this ensures the simulators performance was consistent throughout each simulation, meaning that this variable is controlled and will not have an effect on the comparable results.

Each simulation is run **three** times, this is due to the computation time taken to run larger simulations, for smaller simulations the computation time for nominal. In initial testing the smaller simulations did not cause any concern for speed taken to produce results, but as the simulations increased in complexity the computation time began to increase exponentially. Coupled with multiple runs and some simulation did not complete within 24 hours of computation time, in order to mitigate this problem, optimisations were applied to the algorithm as discussed in the earlier implementation section and finally the decision to run each simulation three times was made, as this seemed to be the best compromise.

When a SUMO simulation is complete, another XML file is produced for analysis. The file consists of detailed information regarding each step taken during the simulation and what actions were taken on each vehicle, such as the route taken, any accidents and how much time was incurred by the vehicle during non-routine stops. In the code above, the Python scripts appropriately manages these output file by dynamically naming each output file for each simulation, with the simulation name provided at run-time and whether or not the algorithm was utilised for this data. Once all of these files have been processed and saved for each simulation, the three repetitions produced by SUMO can be concatenated together in order to provide a reliable dataset to perform statistical analysis on.

4.3.2 Evaluation Metrics

With the data produced by the SUMO simulations, analysis must be performed in order to effectively compare and contrast different simulations in order to reliably prove that CRITIC has a positive effect on the traffic simulation. In terms of this research, a positive effect on a road network would be a sign of a decreased travel time for vehicles in the network, along side this as CRITIC is designed to manage priority lanes, the travel times for the priority vehicles or lanes should not be greatly impacted.

Average Travel Time (ATT):

The first metric that will be calculated from the SUMO data is the ATT (Average Travel Time), although the travel time for vehicles can vary due to the size of the route taken by the vehicle, there is still a strong correlation between the time taken for a vehicle to complete it's journey and the average travel time of a network.

$$ATT = \frac{TotalTravelTime}{NumberOfVehcles} \tag{1}$$

The equation above shows how this metric is calculated from the sumo data, showing how the overall total travel time of a simulation measured in seconds, is summed and then divided by the total number of vehicles simulated. For CRITIC to show a positive result, the ATT must be lower than the baseline simulation, which does not utilise the algorithm.

Percentage Change:

ATT results obtained from the simulations are meaningless when presented in the raw form, in order to provide some value to the metric, when the values are calculated for the baseline simulation and CRITIC simulations, the ATT percentage change will be calculated. In this paper the percentages are shown as reductions or gains, if the percentage is negative, this means the metric has been reduced, showing positive results, if the percentage is positive, this means the metric has increased, showing a negative result. This metric should clearly show the positive or negative impact the algorithm had on the overall travel times

$$\%Change = \frac{BaselineATT - CRITICATT}{BaselineATT} * 100 \tag{2}$$

Travel Time index (TTI):

The second metric that will be utilised during the comparison stage of the research will be TTI(Travel Time Index), this metric attempts to provide a more accurate representation of the travel time on a road network, by taking into account the free-flow travel time of a vehicle. Free flow travel time is the time taken for a vehicle to reach it's destination under optimal conditions with no interruptions or delays, this metric requires further data to be calculated, but should provide a more appropriate result to compare multiple different simulations at once.

$$TTI = \frac{ATT}{ATT - AverageTimeLoss} \tag{3}$$

The equation above are the metrics required to calculate the TTI of a simulation, the calculation utilises the results produced by ATT along with an additional metric named ATL(Average Time Loss). ATL is calculated from the SUMO results file, SUMO saves the amount of time lost by each vehicle in seconds, calculated by a number of factors such as traffic, accidents or waiting at traffic lights. Once an average of the ATL is calculated, it can then be subtracted from the ATT to gain the absolute travel time, then the overall ATT can be divided by the absolute travel time to produce the TTI. The closer the TTI is to 1.0 provides evidence that the travel times are close to optimal, so in order to provide evidence CRITIC is working, the TTI produced should be lower than the baseline TTI.

4.3.3 Bespoke Simulation

The first simulation produced was a bespoke simulation that attempts to show how the algorithm works on a road network, the simulation does not accurately represent a real road network, but does show evidence of how the concepts behind CR when applied to road network, can show positive improvement on both ATT and TTI.

Figure 15: SUMO visualisation of the bespoke network

Figure 15 shows how the bespoke simulation looks when loaded into SUMO-GUI, the bespoke simulation is a single road with 6 lanes, 3 going clockwise and 3 going anti-clockwise and on both sides the outer most lane is a priority lane. To assess the performance of the bespoke simulation, 5000 random vehicles were generated, which consisted of buses and standard private vehicles, buses entered the network after every 50 simulation steps, in order to provide opportunities for the vehicles to utilise these lanes. One additional step required to produce usable statistics was to introduce some form of congestion, as this network has no bottle necks, the traffic maintains a very consistent speed with very little delays even on high density traffic flows, in order to simulate congestion a manual reduction of the speed of the none-priority lanes had to be performed, to ensure that once vehicles utilised the none-congested priority lanes, the speed increase could be measured. This allowed me to then assess the travel times between both baseline and CRITIC simulation and see if CRITIC was performing as expected.

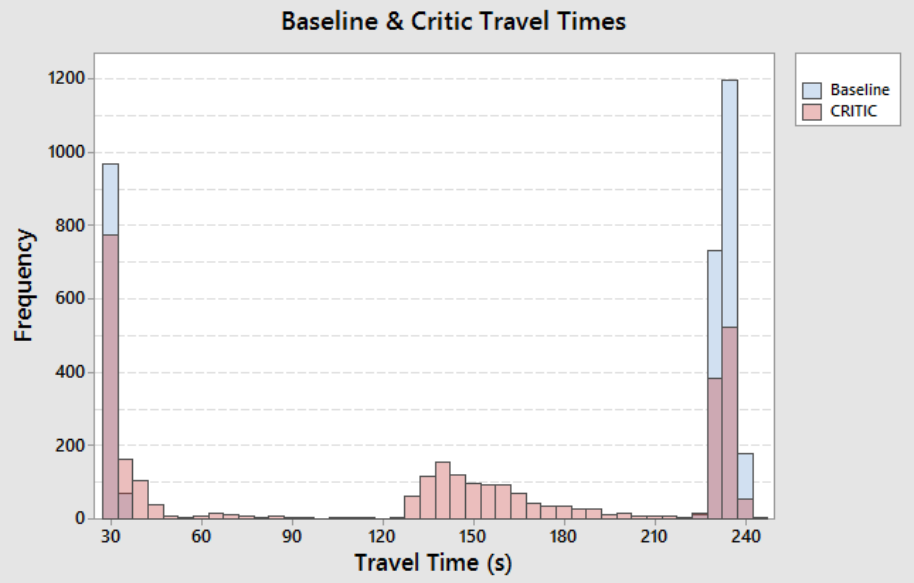

Figure 16: Obtained Travel Time in bespoke simulation: Baseline vs CRITIC

The first graph shown above shows the results obtained from the baseline simulation and the CRITIC enabled simulation, the results show the overall travel times achieved for all vehicles in the network. The baseline results show how the congestion affected all none-priority vehicles as their travel times almost entirely lie over the 200 seconds mark, where as CRITIC provides a much more varied travel time, ranging from 30 up to 240 seconds. Baseline does also produce travel time data at the 30 seconds point, but as the graph shows data for all vehicle types, these results will be priority vehicles, whereas CRITIC achieves much more positive results for all vehicles across many different time points, with almost a 50% reduction in travel times at 230 seconds.

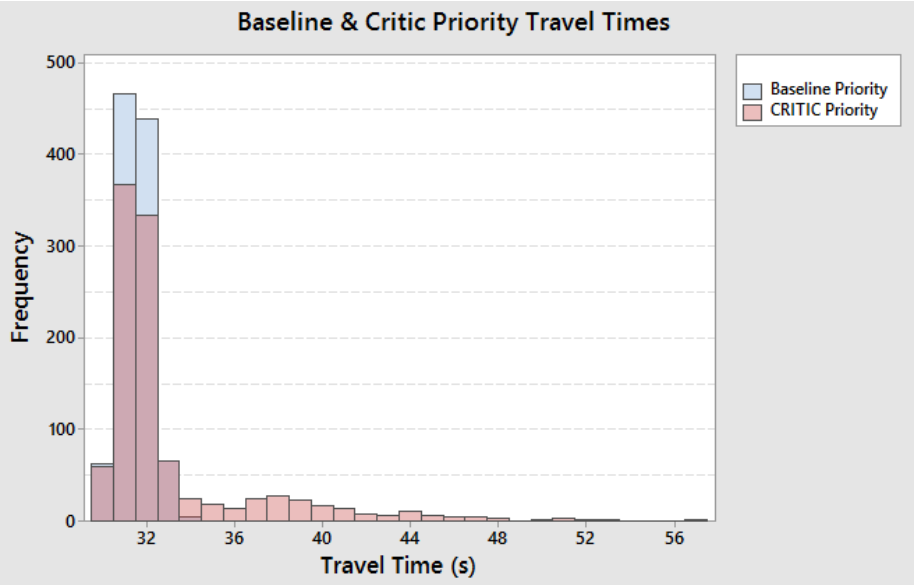

Figure 17: Obtained Priority Travel Times in bespoke simulation: Baseline vs CRITIC

The second graph is this series, shows the travel time results exclusively for priority vehicles, which in the case of the simulations are buses. The results shown by this graph, provide some negative results for the CRITIC algorithm, showing that in some cases the priority vehicles were effected by the algorithm. The results obtained do not seem to show drastic numbers of priority vehicles being affected, as the majority still lie within the same 0-34 second band as the baseline results, but this still shows evidence of priority vehicles being delayed.

$$ATT \quad Percentage \quad Change = -19.24\%$$

After analysing the data further and calculating the ATT values for all traffic data, the results are very positive. There is almost a 20% decrease in ATT for all vehicles in the network, which shows CRITIC does have a positive effect on traffic congestion, these results are calculated as a percentage difference from the baseline results, where only standard road rules apply.
Along-side the decrease in ATT for the overall network, there is also some negative results, showing that there was an 11% increase in over all travel time for the priority lanes. The increase can be justified due to more traffic utilising the priority lanes, meaning in some cases there could be high congested in all lanes in a network, meaning the travel time for specific lanes can vary. However this simulation was not designed to mimic real world conditions, therefore no reliable results can be drawn, as further more complex simulations are needed to show more reliable results, with much greater sample numbers and more complex road topologies.

$$Baseline \quad TTI = 12.32$$

$$CRITIC \quad TTI = 9.91$$

The final metric used to assess the bespoke simulation was the TTI, as TTI tries to show a more accurate travel time by accounting for the time loss of each vehicle, the TTI shows clearly how the network performance has changed over time, the closer the value is to 1.0, the less congested a network is. The results obtained from this simulation clearly show that the CRITIC algorithm is providing a more optimal driving experience to the road users, as the TTI is over 2 points closer to the optimal value of 1.0.

4.3.4 Grid simulation

Once the bespoke simulation had been built and the algorithm refined to provide the best computational performance, further test cases had to be researched in order to provide more realistic test data, data that is more closely related to real world road conditions. As mention earlier, to generate the networks used for testing the open source SUMO tool called NETEDIT was used, this tools provides 3 different style of networks that try to mimic common real world road topologies.

Figure 18: NYC Grid network [46]

For the first large scale simulation CRITIC was tested on a GRID style network, GRID networks can be found in many large cities, with the most notable example being shown in Figure 18, which an aerial view of the New York City road network. As this photograph shown, all of the roads connect in a grid form, forming a pattern of rectangles & squares, this topology can be very effective at reducing congestion as less complex joins occur between the road intersections, meaning cars can can travel across the junctions easier, forming less traffic build up. However due the GRID style network can still be over-congested, as the number of vehicles exceeds to physical capacity of the network, traffic congestion begins to build.

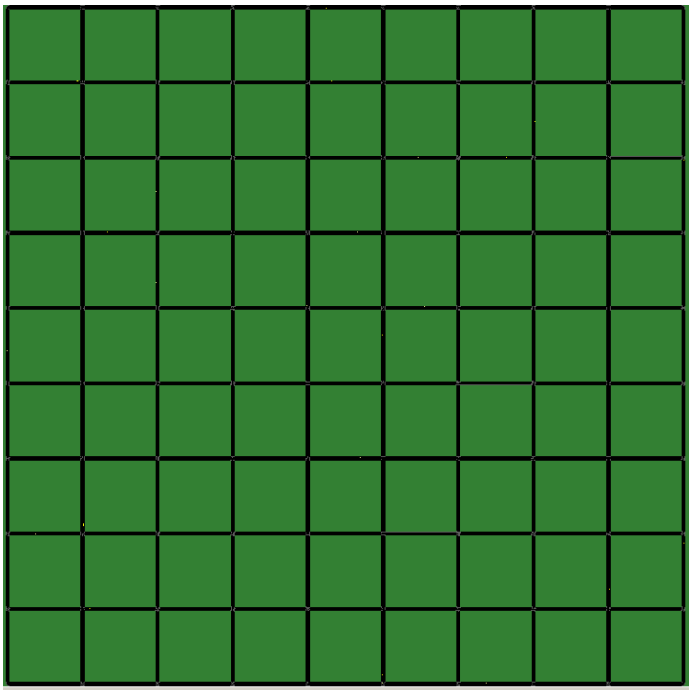

Figure 19: NYC Grid network [46]

Figure 19 shows the network generated by NETGENERATE, this is 10 x 10 GRID network, consisting of multiple 3 x 3 lanes, with the outer most lane being a priority lane. This simulation was built with the intention to see if CRITIC could help improve the performance of congested GRID

networks, with the hypothesis that the algorithm could help improve traffic throughput for heavily congested intersections, by opening up new lanes.

For the large scale networks, It was decided that it was appropriate to run multiple different test-cases, in order to assess how well CRITIC performed under different congestion loads. Due to the previously mentioned performance issues on large scale simulations, the decision to test three different traffic levels was taken, 500, 1000 and 2000 vehicles each with additional bus traffic generated at 20 step intervals. The delays between buses provide time for the CRITIC algorithm to optimise the road network, providing opportunities for vehicles to make use of this priority lanes.

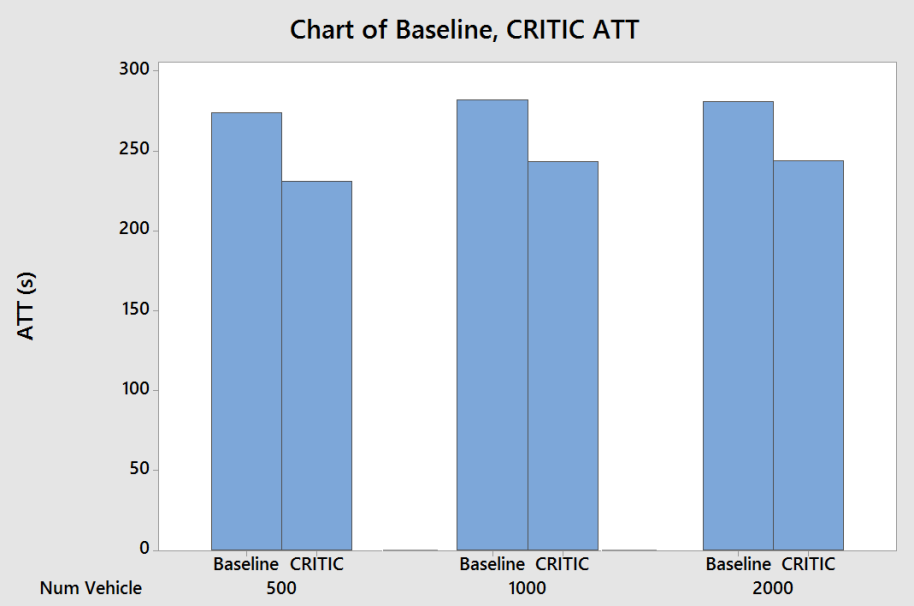

Figure 20: Average Travel Times in 10x10 grid simulation: Baseline vs CRITIC

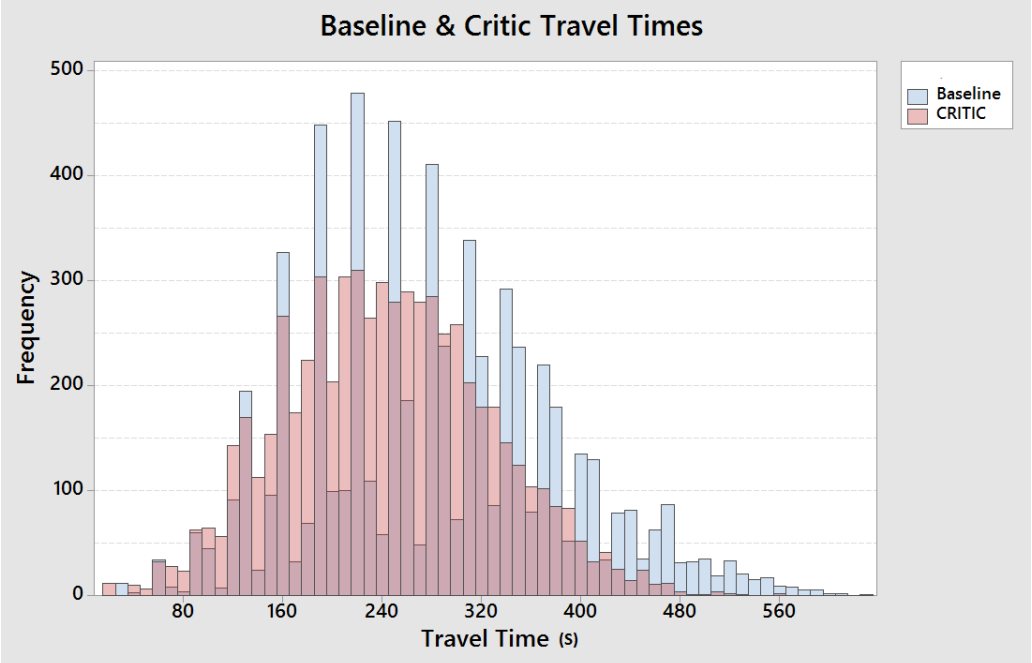

Figure 21: Obtained Travel Time in 10x10 grid simulation: Baseline vs CRITIC

The above bar chart shows the obtained ATT results from the three simulations, each simulation is run three times each with a different seed value. Once the statistics are produced from each individual run, the three files produced by SUMO are concatenated together, providing a larger more reliable dataset to analyse. As you can see, overall CRITIC provided a notable decrease in ATT in all three simulation environments, reducing the ATT by roughly 100 seconds in each case.

Along side the bar chart, the histogram depicting the distribution of all saved travel times in the GRID networks for both baseline simulations and CRITIC. Showing that the CRITIC algorithm produces a large sample of shorter travel times in comparison to the baseline simulation, with fewer extreme travel times.

Simulation	% Improvement of priority ATT
GRID-500	-25.80
GRID-1000	-25.85
GRID-2000	-19.50

Table 1: % Change for ATT in priority lanes: 10x10 GRID network

Along side producing positive results for overall ATT, the GRID simulation also provided more insightful data in terms of the effects CRITIC would have on the priority lanes and their intended purpose. Table 1 shows the ATT percentage change for each simulation, as you can see the largest percentage decrease in travel time was 28.85%, proving critic not only improves overall network performance, but also the performance of the priority lanes themselves. This data is the physical ATT of the priority lanes, not the ATT of the vehicles that used this lane, this statistic is more relevant to proving that CRITIC does not impact travel times in bus lanes, as SUMO implements real world bus policies, meaning buses do not always use bus lanes, they can also use none-priority lanes when required, in such cases as turning at a junction.

Simulation	% Improvement of TTI
GRID-500	-15.05
GRID-1000	-13.62
GRID-2000	-12.91

Table 2: % Change of TTI: 10x10 GRID network

Finally, along side calculating ATT values, TTI was also calculated for the GRID network, showing how the CRITIC algorithm reduced the overall TTI for each simulation. Table 2 shows the results achieved with the greatest reduction achieved at 500 vehicles, the reduction means that not only is travel time faster, but at the networks optimum free flow capacity, the journey time is actually closer to the optimum.

4.3.5 Abstract Simulation

The third simulation produced was an Abstract network simulation, this was the final topology simulated, as mentioned earlier NETGENERATE provides three different topologies, the final topology offered did not suite the research needs, so was therefore excluded. Abstract networks try to mimic a more complex, natural road networks, that unlike the GRID style networks have grown from multiple different stages.

The image above is aerial photo taken of Manchester, a northern UK city, as you can see the road network does not appear to have any uniformity to it, almost being completely random in the way it arranged. This is primary due to the road network being changed over the space of many years, with new roads being added and connected.

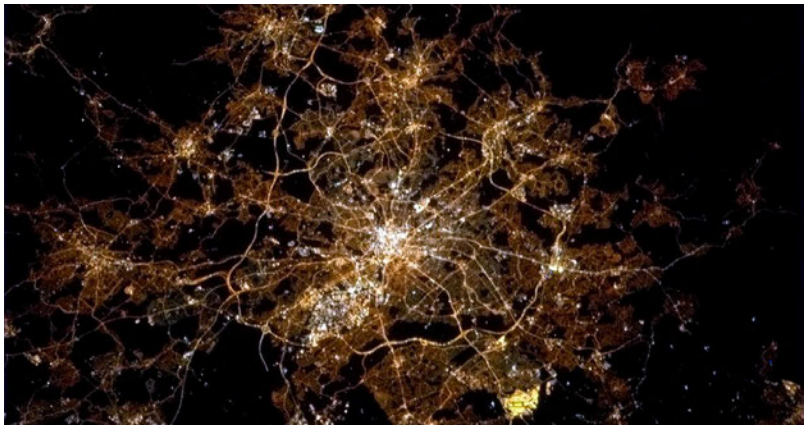

Figure 22: Aerial view of the Manchester road network [47]

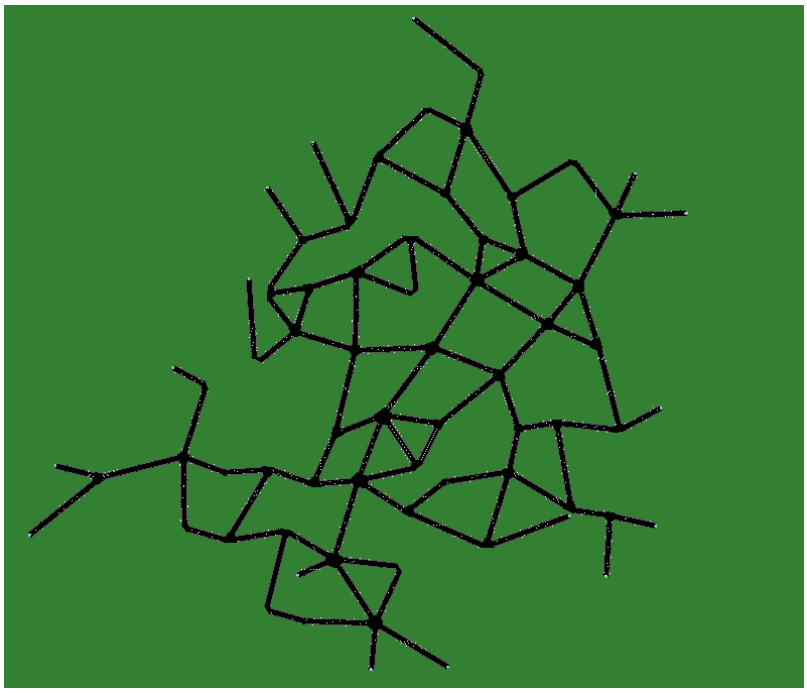

Figure 23: SUMO GUI showing an Abstract network

From the above image, which is a visualisation produced by SUMO of a custom Abstract topology, you can clearly see how these network attempt to mimic the natural road networks produced by many cities, with multiple different junctions and connecting lanes, with no uniformity between each edge.

In order to maintain the same test environment as the GRID network simulation, the same simulation sizes were reproduced, ranging from 500 vehicles to 2000. This meant the results could easily be compared at a later stage with other results produced by SUMO from multiple different topologies, such as the GRID network.

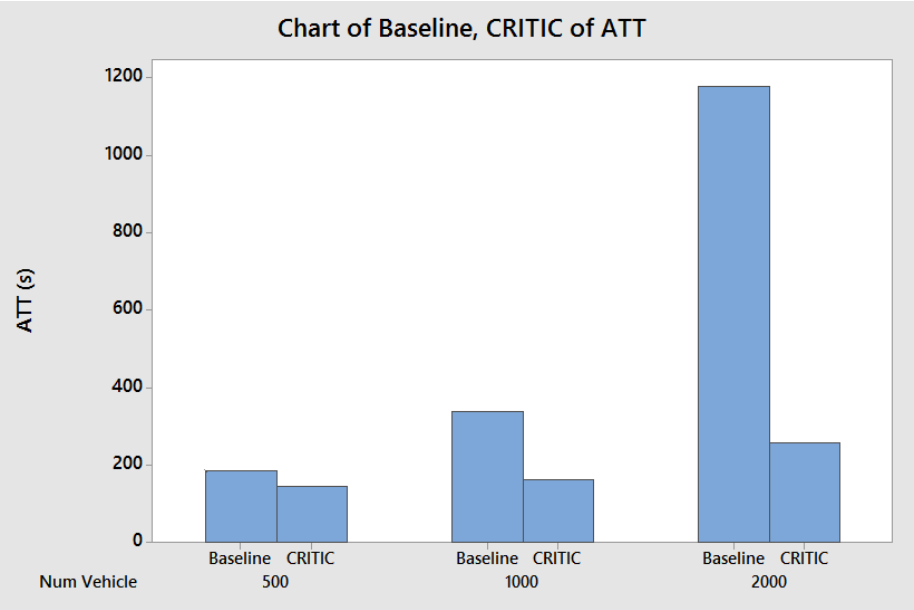

Figure 24: Average Travel Times in Abstract simulation: Baseline vs CRITIC

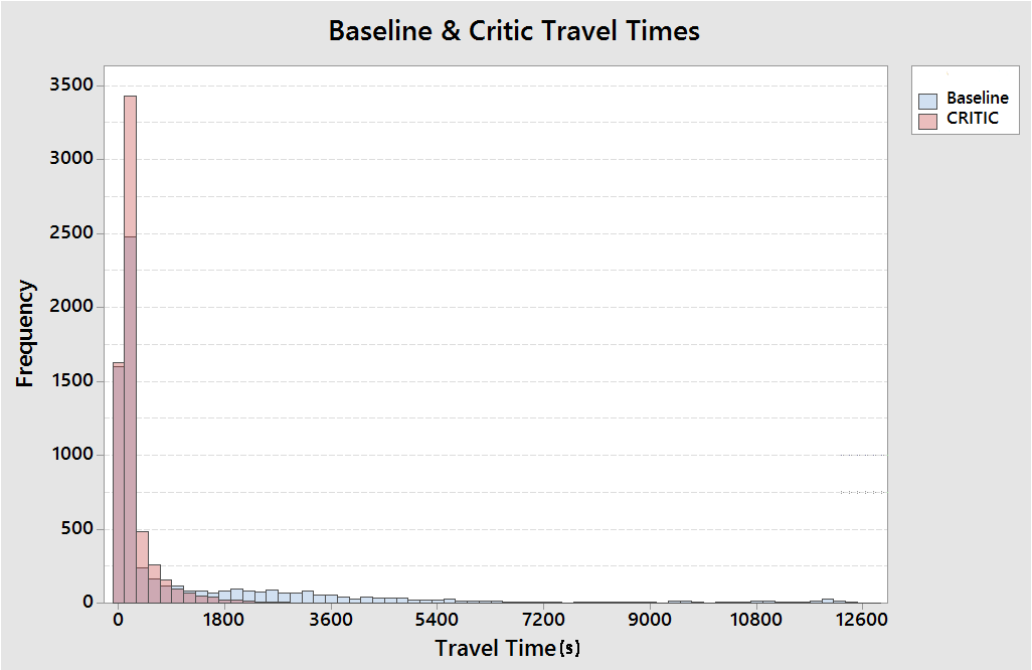

Figure 25: Obtained Travel Times in Abstract simulation: Baseline vs CRITIC

The above graphs show the statistics achieved from the Abstract network simulated, the bar chart shows how the ATT of the baseline simulations began to increase drastically as the number of vehicles in the simulation doubled, but CRITIC was only slightly affected by the same change. The histogram shows the results obtained from the most per-formant simulation, which was 2000 vehicles, the histogram compares all the achieved travel times of each vehicle in the network for baseline and CRITIC simulations. As you can see a greater majority of vehicles achieved much shorter travel times in comparison with the baseline results, whilst also producing less extreme results, which can be caused by extreme congestion or road incidents.

Along side very positive results for ATT on Abstract networks, the results also produced some very positive statistics showing how due to the ability for buses to utilise all lanes in a network, the CRITIC algorithm reduces overall congestion on the network and therefore improves the per-

formance of priority vehicles.

Simulation	% ATT of priority vehicles
ABSTRACT-500	-16.27
ABSTRACT-1000	-26.98
ABSTRACT-2000	-59.71

Table 3: % Change of ATT for priority vehicles: Abstract network

The table above shows the percentage improvement of the ATT for all priority vehicles in the abstract network, as you can see as the network becomes more congested, the CRITIC algorithm achieves significant reduction in travel, with an almost 60% reduction for the abstract-200 simulation. This is very strong data than due to the nature of CRITIC, once a network becomes extremely congested, all vehicles benefit to the reduced congestion of no priority lanes.

Simulation	% Improvement of TTI
ABSTRACT-500	-19.95
ABSTRACT-1000	-51.42
ABSTRACT-2000	-75.99

Table 4: % Change of TTI: Abstract network

The Abstract network also provides extremely positive TTI results for the CRITIC algorithm, as the table above shows in the most congested network of 2000 vehicles, the CRITIC algorithm provides a 75% reduction in TTI. This clearly shows how CRITIC has the ability to reduce road traffic congestion on highly congested road networks, not only reducing overall travel times but also increasing the overall quality of service of the road network.

4.4 Conclusion

This section has aimed to provide evidence that CRITIC has the ability to show promising reductions in traffic congestion, showing the algorithm performing in many different environments. The section also discussed what problems were faced during the implementation of the algorithm and these problems were overcome in order to achieve the necessary goal of producing simulations in SUMO with the CRITIC algorithm.

5 Reflections

This research has touched upon one of the most pressing issues of the 21st century, as cities grow and the human population increases to unforeseen levels, new challenges arise, which must be solved in order to maintain a productive and healthy society. Traffic congestion is one these issues, without an efficient road network, the economy and health of many cities will deteriorate, as business fail to grow and the environment begins to suffer.

The research has shown the cutting edge research being conducted by many organisations into many different types of solutions, solutions that may change the way we think about traffic and vehicle transport in the future. My research attempts to provide evidence that not all changes need to be in the physical network, but changes to laws and road policies can also see profound effects in the efficiency of current road traffic networks.

During the research many challenges were faced, ranging from understanding current cutting edge research and learning the current road laws and techniques used today inn order to manage traffic. Along side researching and understanding the theory of the research, I also had to overcome the practical side of things, such as learning SUMO and understanding how each tool is utilised in producing and analysing different networks.

SUMO is an open source piece of software, with no commercial licensing at all, meaning the level of support expected is of course at a lower standard, as many open source projects are maintained by people for free, as a hobby or as a small part of their working life. Due to these reasons, SUMO poses many issues to users who are unfamiliar with road traffic simulators and the functionality the offer.

The first issue SUMO has is the documentation, the project is poorly arranged and is not simple to navigate and retrieve vital information to run custom networks and simulations, the documentation is spread across three different sources, with varying versions and maintainers. Once a piece of documentation is found, it is also not uncommon for many implementation details to be missing or incomplete, this prevented multiple delays to the research, as some specific tools or configurations I needed to utilise were not present or hard to find.

The second issue faced is the fact that SUMO comes bundled with a large range of different tools, which many are required to be utilised in order to produce custom simulation environments. Many of these tools rely on specific software versions, with very little help of guidance on how to install or run these pieces of software. Once a piece of software is installed, there is also the task of understanding how to use it and what parameters this software accepts in order to produce output.

Finally the TraCi API used to interface with SUMO is extremely cumbersome to use, the API requires that a specific version of Python must be installed and linked to the script in order to execute. Along side this the documentation for each method is extremely poor, with no description of the databases methods accept or return and also sometimes the discovery of multiple methods that implement the same functionality.

Overall SUMO is still a very useful piece of software, with great overall performance in the core simulation software, but the tools provided by SUMO are very poor and also poorly maintained an documented. I feel that the software requires an extremely detailed analysis of all the software packages included, with each package being assesssed and documented accordingly, with specific developers maintaining specific tools in order to keep them up to do date and relevant. Along side this, SUMO could also benefit from a real user based forum, where questions and answers can be categorised and saved for future use, as currently this does not exists.

Along side the struggles faced by SUMO, I also faced new challenges TraCiself, in terms of understanding how to implement complex algorithms and understanding how to produce meaningful statistics to assesss the performance of TraCi algorithm, adopting an AGILE work flow in order to improve and iterate quickly.

Finally CRITIC could require some further improvements in terms of computational performance, with further professional input to the road laws and rules, along with additional development time, i'm sure CRITIC could become much more efficient I also feel CRITIC could benefit from more real world research, by being tested on historic traffic data, In order to assesss whether or not CRITIC can provide improvements without any changes to the current road network.

Overall this research project was a success, with some useful research into new and novel traffic management techniques being performed. Along with multiple different statistics, demonstrating between 15% to 75% reductions in overall travel time and shorter travel times overall. These statistics provide evidence to defend and display the benefits CRITIC has to offer.

References

[1] Cox, W. (2000) HOW URBAN DENSITY INTENSIFIES TRAFFIC CONGESTION AND AIR POLLUTION. Available at: http://www.americandreamcoalition.org/landuse/denseair.pdf (Accessed: 26 October 2016).

[2] Bradshaw, R. (2015) The impact of budget cuts on local road maintenance and road safety. Available at: http://www.lgiu.org.uk/wp-content/uploads/2015/11/The-impact-of-budget-cuts-on-local-road-maintenance-and-road-safety.pdf (Accessed: 27 October 2016).

[3] Litman, T. (2016) Comprehensive Evaluation Of Traffic Congestion Costs and Congestion Reduction Strategies. Available at: http://www.vtpi.org/cong_relief.pdf (Accessed: 27 October 2016).

[4] EDRG (2007), The Cost of Highway Limitations and Traffic Delay to Oregons EconoTraCi, Oregon Business Council and Portland Business Alliance (www.orbusinesscouncil.org); at https://www.portofportland.com/PDFPOP/Trade_Trans_Studies_CostHwy_Lmtns.pdf (Accessed: 30 October 2016)

[5] HENNESSY, D.A. and WIESENTHAL, D.L. (1999) Traffic Congestion, Driver Stress, and Driver Aggression. Available at: https://www.researchgate.net/profile/Dwight_Hennessy/publication/229863510 (Accessed: 5 November 2016).

[6] Novaco, R.W. and Stokols, D. (1979) Transportation, Stress, and Community Psychology. Available at: https://www.researchgate.net/publication/22645861 (Accessed: 5 November 2016).

[7] Behrisch, M., Bieker, L., Erdmann, J. and Krajzewicz, D. (no date) http://elib.dlr.de/71460/1/SUMO_survey_SIMUL2011.pdf. Available at: http://elib.dlr.de/71460/1/SUMO_survey_SIMUL2011.pdf (Accessed: 5 November 2016).

[8] Pattberg, B. (2016) Institute of transportation systems - SUMO – simulation of urban mObility. Available at: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/ (Accessed: 5 November 2016).

[9] TraCI (2008) Available at: http://www.sumo.dlr.de/wiki/TraCI (Accessed: 5 November 2016).

[10] INRIX (2016) Traffic congestion to cost the UK econoTraCi more than £300 Billion over the next 16 years. Available at: http://inrix.com/press/traffic-congestion-to-cost-the-uk-econoTraCi-more-than-300-billion-over-the-next-16-years/ (Accessed: 12 November 2016).

[11] Schrank, D., Eisele, B., Lomax, T. and Bak, J. (2015) 2015 URBAN MOBILITY SCORECARD. Available at: http://d2dtl5nnlpfr0r.cloudfront.net/tti.tamu.edu/documents/mobility-scorecard-2015.pdf (Accessed: 12 November 2016).

[12] 2013, L. (2013) Institute for transport studies: Institute for transport studies. Available at: https://www.its.leeds.ac.uk (Accessed: 12 November 2016).

[13] Fossil (no date) Available at: http://www.energy.gov/science-innovation/energy-sources/fossil (Accessed: 12 November 2016).

[14] Health effects of traffic-related air pollution (2016) Available at: http://healthycanadians.gc.ca/healthy-living-vie-saine/environment-environnement/air/vehicles-vehicules-eng.php (Accessed: 12 November 2016).

[15] Traffic simulation (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Traffic_simulation (Accessed: 12 November 2016).

[16] Joerer, S., Dressler, F. and Sommer, C. (2012) Comparing Apples and Oranges? Trends in IVC Simulations. Available at: http://www.ccs-

labs.org/bib/joerer2012comparing/joerer2012comparing.pdf (Accessed: 12 November 2016).

[17] Djahel, S., Doolan, R., Muntean, G.-M. and Murphy, J. (2017) A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches. Available at: http://ieeexplore.ieee.org/document/6857980/ (Accessed: 12 November 2016).

[18] Introduction to CR and DSA (no date) Available at: http://www.wirelessinnovation.org/assets/documents/Introduction%20to%20CR%20and%20DSA.pdf (Accessed: 13 November 2016).

[19] Mitola III, J. (2000) An Integrated Agent Architecture for Software Defined Radio. Available at: https://web.archive.org/web/20120917062752/http://web.it.kth.se/ maguire/jmitola/Mitola_Dissertation8_Integrated.pdf (Accessed: 13 November 2016).

[20] https://www.ieee.org/index.html

[21] Internet of Things (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Internet_of_things (Accessed: 19 November 2016).

[22] Ltd, G. (1996) GARMIN United Kingdom. Available at: http://www.garmin.com/en-GB/company/about (Accessed: 19 November 2016).

[23] BV, T.I. (2016) About TomTom - investor relations. Available at: http://corporate.tomtom.com (Accessed: 19 November 2016).

[24] Satellite navigation (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Satellite_navigation (Accessed: 19 November 2016).

[25] Djahel, S., Salehie, M., Tal, I. and Jamshidi, P. (2013) 'Adaptive traffic management for secure and efficient emergency services in smart cities', pp. 340–343. Available at: http//dx..org/10.1109/PerComW.2013.6529511.

[26] Fuchs, S., Rass, S., Lamprecht, B. and Kyamakya, K. (no date) 'Context-Awareness and Collaborative Driving for Intelligent Vehicles and Smart Roads', Available at: http://vi.uni-klu.ac.at/publications/papers/2007-14.pdf

[27] Faye, S., Chaudet, C. and Demeure, I. (no date) A Distributed Algorithm for Adaptive Traffic Lights Control in Wireless Sensor Networks. Available at: http://biblio.telecom-paristech.fr/cgi-bin/download.cgi?id=12588 (Accessed: 24 November 2016)

[28] http://www.telecom-paristech.fr

[29] http://www.nortechcontrol.com/products/vehicle-detection-and-parking/inductive-loop-detectors/

[30] Strategy, C.M., blog, A. and Tong, J. (2014) Waterfall software development model. Available at: http://www.oxagile.com/company/blog/the-waterfall-model/ (Accessed: 15 December 2016).

[31] http://agilemethodology.org

[32] Jeremiah, J. (2016) Agile vs. Waterfall: Survey shows agile is now the norm. Available at: http://techbeacon.com/survey-agile-new-norm (Accessed: 15 December 2016).

[33] https://en.wikipedia.org/wiki/Waterfall_model

[34] https://www.python.org

[35] Nguyen, Tam, V., Villain, F., Guillou, L. and Corporation, H.P. (2012) 'Cognitive radio RF: Overview and challenges', VLSI Design, 2012.

[36] http://sumo.dlr.de/wiki/SUMO#Time (Accessed: 20 January 2017).

[37] http://sumo.dlr.de/wiki/Simulation/Randomness#Random_number_generation_.28RNG.29 (Accessed: 20 January 2017).

[38] http://www.minitab.com/en-us/products/minitab/ (Accessed: 31 March 2017).

[39] http://sumo.dlr.de/wiki/NETEDIT N.p., 2017. Web. 13 Apr. 2017.

[40] http://sumo.dlr.de/wiki/Networks/Abstract_Network_Generation N.p., 2017. Web. 13 Apr. 2017.

[41] http://sumo.dlr.de/wiki/Tools/Trip N.p., 2017. Web. 13 Apr. 2017.

[42] http://sumo.dlr.de/wiki/Tools/Sumolib#import_a_network_and_retrieve_nodes_and_edges N.p., 2017. Web. 16 Apr. 2017.

[43] http://sumo.dlr.de/wiki/Definition_of_Vehicles_Vehicle_Types,_and_Routes "Definition Of Vehicles, Vehicle Types, And Routes - Sumo". Sumo.dlr.de. N.p., 2017. Web. 18 Apr. 2017.

[44] Council, Manchester. "Bus Lane Rules — Bus Lane Rules — Manchester City Council". Manchester.gov.uk. N.p., 2017. Web. 18 Apr. 2017.

[45] En.wikibooks.org. (2017). Travel Time Reliability Reference Manual/Travel Time Reliability Indices - Wikibooks, open books for an open world. [online] Available at: https://en.wikibooks.org/wiki/Travel_Time_Reliability_Reference_Manual/Travel_Time_Reliability_Indices [Accessed 19 Apr. 2017].

[46] Distl.co. (2017). [online] Available at: http://distl.co/wp-content/uploads/2013/09/Untitled-1-970x514.jpg [Accessed 20 Apr. 2017].

[47] Salford-gis.co.uk. (2017). Astronaut Views of Earth from Space - mapmad. [online] Available at: http://www.salford-gis.co.uk/wordpress/?p=837 [Accessed 22 Apr. 2017].

[48] http://sumo.dlr.de/wiki/File:TraciConnect_sequence.png [Accessed 20 Apr. 2017]