4G System Handover Optimisation

Iain Cuthbertson
201015895
Computer & Electronic Systems
University of Strathclyde

April 2014

I hereby declare that this work has not been submitted for any other degree/course at this University or any other institution and that, except where reference is made to the work of other authors, the material presented is original and entirely the result of my own work at the University of Strathclyde under the supervision of Dr. Robert C. Atkinson.

Signature _____

Abstract

With more and more customers using mobile communications it is important for the service providers to give their customers the best Quality-of-Service (QoS) they can. Many providers have taken to improve their networks and make them more appealing to customers. One such improvement that providers can give to their customers is to improve the reliability of their network meaning that customers calls are less likely to be dropped by the network.

This dissertation explores improving the reliability of a 4G network by optimising the parameters used in handovers. The process of handover within mobile communication networks is very important and allowing for users to move around freely while still staying connected to the network. The parameters used in the handover process are the Time-to-Trigger (TTT) and Hysteresis (hys). These parameters are used to determined where a base station better then the serving base station by enough to warrant a handover taking place. The challenge in optimising the handover parameters is that there is a fine balance that needs to be struck between calls being dropped due to a handover failing and the connection switching back and forth between two base stations, unnecessarily, wasting the networks resources. The approach taken is to use a machine learning technique known as Q-Learning to optimise the handover parameters by generating a policy that can be followed to adjust the parameters as needed.

Acknowledgements

Contents

Nomenclature

4G	Fourth Generation
AI	Artificial Intelligence
dB	Decibels
DES	Discrete Event Simulation
eNodeB	Evolved Node B
EPC	Evolved Packet Core
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
HSS	Home Subscriber Server
hys	Hysteresis
LTE	Long Term Evolution
MME	Mobility Management Entity
P-GW	Packet Data Network Gateway
PCell	Primary Cell
QoS	Quality-of-Service
RSRP	Reference Signal Received Power
RSS	Received Signal Strength
SCell	Secondary Cell
S-GW	Serving Gateway
TTT	Time-to-Trigger
UE	User Equipment

List of Figures

List of Tables

List of Algorithms

Listings

Chapter 1

Introduction

Handover is a key process in Fourth Generation (4G) cellular networks. It allows for customers to be able to continue there calls and using services while moving out of range of the base station they are currently connected to by transferring the resources for that customer to another base station. A reinforcement learning technique known as Q-Learning was used to carry out the optimisation of the handover parameters.

The performance of handovers within a cellular network is very important. Initiating a handover too late can cause an on-going call to be dropped by the network. This would be viewed as poor Quality-of-Service (QoS) by the customer. However, initiating a handover too early also comes with its own problems. It can cause handover ping-pong's where a mobiles' connection switches back and forth between base stations in a short period of time and this wastes resources within the network.

A simulation was created that would simulate mobile phones moving around a group of network base stations. This simulation would be used to determine the performance of the machine learning algorithm compared to if no optimisation was taking place. It would also supply the machine learning algorithm with rewards to tell it if it is making decision that are improving the performance or making it worse.

1.1 Project Objectives

The main objective of this project was to development a machine learning algorithm to fine tune the parameters used in the handover process of 4G networks.

Key Objectives:

- Research and understand the parameters used in 4G handovers.

- Create a basic simulation of a 4G network with mobiles moving around a group of base stations.
- Implement a machine learning algorithm to fine-tune the handover parameters to improve the performance of the network.
- Evaluate the success of the machine learning approach in 4G handovers and compare this to using static parameters.

1.2 Project Outcomes

During the course of the project it has been possible to meet all of the objectives. The parameters that are used in the handover process, the Time-to-Trigger (TTT) and Hysteresis (hys), were researched and the ways they are used were discovered. A basic simulation of a cellular network has been created and a Q-Learning algorithm was developed that optimised the handover parameters.

Experiments were run to find the worth of the machine learning system against using static values for the handover parameters. It was found that the machine learning system was able to improve the performance of the network. However, the system was found to be capable of having the potential to perform even better.

1.3 Dissertation Outline

Chapter 2 describes a basic account of a 4G network along with the handover process. Chapter 3 gives a brief discussion on the different types of machine learning and goes more in-depth into reinforcement learning and Q-Learning. Chapter 4 describes the different propagation and mobility models considered for use within the simulation, as well as giving accounts of how the simulation was designed and the testing done to make sure the simulation was functioning as required. Chapter 5 describes the approach taken to the machine learning problem and the experiments run to find the worth of the system. This chapter also gives the results of the experiments and these results are discussed. Chapter 6 discusses the possible future work that could take this project further. Finally, Chapter 7 gives an account of the conclusions drawn from the project.

Chapter 2

LTE

Mobile communications are a main stay of modern life, with Ericsson reporting that there were "6.7 Billion mobile subscriptions globally in Q4 2013" [1]. The infrastructure used in mobile telecommunications are now in their Fourth Generation (4G) with Long Term Evolution (LTE). This network infrastructure was developed by 3GPP and is an improvement upon Universal Mobile Telecommunications System (UMTS), which is a third generation network (3G). LTE has downlink (DL) speeds of up to $300Mbit/s$ and uplink (UL) speeds of $75Mbit/s$ with a flexible bandwidth that ranges from $1.4MHz$ to $20MHz$. Ericsson also reported that the number of LTE subscriptions had reached 200 Million in Q4 2013. The development of faster downloads speed was driven by the consumers want for better quality images, faster Internet browsing and smoother video streaming. [2]

2.1 Network Structure

The structure of the LTE network can be broken down into 3 main parts, the User Equipment (UE), the Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) and the Evolved Packet Core (EPC). The UE can simply be considered as a standard mobile phone or smartphone. The purpose of the E-UTRAN is to connect a UE to the EPC and is made up of just one component, the Evolved Node B (eNodeB) or simply put base station. Figure 2.1 shows an illustration of the E-UTRAN. An UE will only communicate with one eNodeB at any time and this eNodeB is known as the Serving eNodeB. An eNodeB proves two main functions within the network; the first is to send all the radio traffic for an UE on the DL as well as receiving any traffic sent from the UE on the UL. The second function of the eNodeB is to control low-level operations such as handovers as well as provide the signalling for such operations. Due to the eNodeBs having the added complexity of controlling operations, such as handovers, it moves more of the processing from the core network to the edge of the network, reducing the

latency of decisions being made. Every eNodeB is connected to the EPC using the S1 interface. They can also be connected to other eNodeBs by the X2 interface. This interface is mainly used for signalling and forwarding data from a serving eNodeB to a neighbouring eNodeB during a handover.

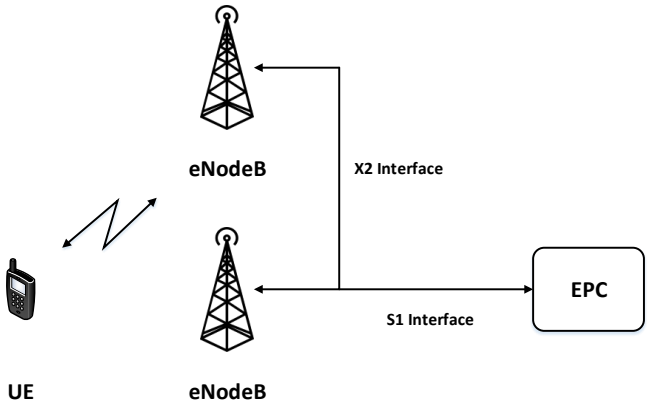


Figure 2.1: Illustration of E-UTRAN.

An illustration of an EPC can be seen in Figure 2.2. It can be seen that the EPC is made up of four main components, they are the: Mobility Management Entity (MME), Home Subscriber Server (HSS), Packet Data Network Gateway (P-GW) and Serving Gateway (S-GW). The MME provide the high-level operations for a UE such as security and managing non-radio communication data streams as well as controlling other elements within the EPC. There are very few MME's within the LTE network, generally assigned to a certain geographical region. A UE will be assigned to a single MME known as the Serving MME.

The HSS is a central database that holds all the information about the network subscribers such as authentication and billing information.

The P-GW is the bridge between the EPC and other packet data networks such as the Internet. The P-GW uses the SGi interface to exchange data with outside networks, such as the network operator's servers. The S-GW's function is to forward data to and from the eNodeBs to the P-GW; this means that the S-GW effectively acts like a router. Much like the MME, a UE will be assigned to one S-GW and there will be very few within the network as a whole. [2, 3]

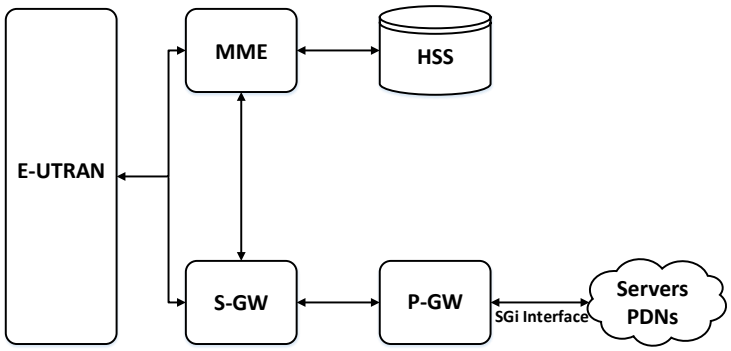


Figure 2.2: Illustration of EPC.

2.2 Self-Organising Network

LTE, just like any other mobile network, needs to be managed. Due to LTE being more complex than its predecessors the management of the network has also become more complex. The use of automation can simplify management of a network greatly. A technology that can be employed in LTE is that of a Self-Organising Network. This system provides three main functions: Self-configuration, Self-optimisation and Self-healing. [4, 5]

- **Self-configuration** allows for a newly installed eNodeB to be configured automatically with the basic parameters for operation.
- **Self-optimisation** is a process where measurements from the UE and eNodeB along with performance measurements are used to optimise the network to make it perform better.
- **Self-healing** is designed to detect and identify failures within the network. After the self-healing system detects a failure it aims to recover from the failure without the need for human interaction with the system.

2.3 Handover

The process of handover is very important in mobile telecommunications. It involves moving the resource allocation for a mobile phone or a piece of UE from

one base station to another. This process is used to provide better Quality-of-Service (QoS) to customers by allowing them to continue to use provided services even after moving out of range of the original serving base station. It is important that handovers are performed quickly, cause little-to-no disruption to the user's experience and are completed with a very high success rate. If a handover is unsuccessful it is likely that an on-going call will be dropped due to there not being enough resources available on a base station (known as an eNodeB in LTE) or the if Received Signal Strength (RSS) to the UE drops below a certain threshold needed to maintain the call. This threshold, in LTE, is known as the noise floor and has a value of $-97.5dB$ [6]. Handovers are stated to take roughly 0.25 seconds to complete after the decision has been made for a handover to take place [7].

2.3.1 Parameters

In LTE there are two main parameters that are used in the handover process. These parameters are the Time-to-Trigger (TTT) and Hysteresis (hys). The hys is used to define how much better the RSS of a neighbouring base station must be than the serving base station for a handover to be considered. The values of hys are defined in Decibels (dB) and range from 0 to $10dB$ in $0.5dB$ increments, this results in there being 21 different values of hys. The full range of hys values can be seen in Table 2.1.

Index	0	1	2	3	4	5	6	7	8	9	10
hys (dB)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Index	11	12	13	14	15	16	17	18	19	20	
hys (dB)	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10	

Table 2.1: Table of the different LTE hys values.

The TTT is a length of time, defined in seconds, that is used to define how long a neighbouring base station must be considered better than the serving base station for. There are 16 different values of TTT ranging from 0 to 5.12 seconds. Unlike with hys, the TTT values do not increase linearly; instead they increase exponential with smaller increases at the lower values and bigger increases at the larger values. The full list of TTT values can be seen in Table 2.2 and a graph of how the TTT values increase can be seen in Figure 2.3.

Index	0	1	2	3	4	5	6	7	8
TTT (s)	0.0	0.04	0.064	0.08	0.1	0.128	0.16	0.256	0.32
Index	9	10	11	12	13	14	15		
TTT (s)	0.48	0.512	0.64	1.024	1.280	2.56	5.12		

Table 2.2: Table of the different LTE TTT values.

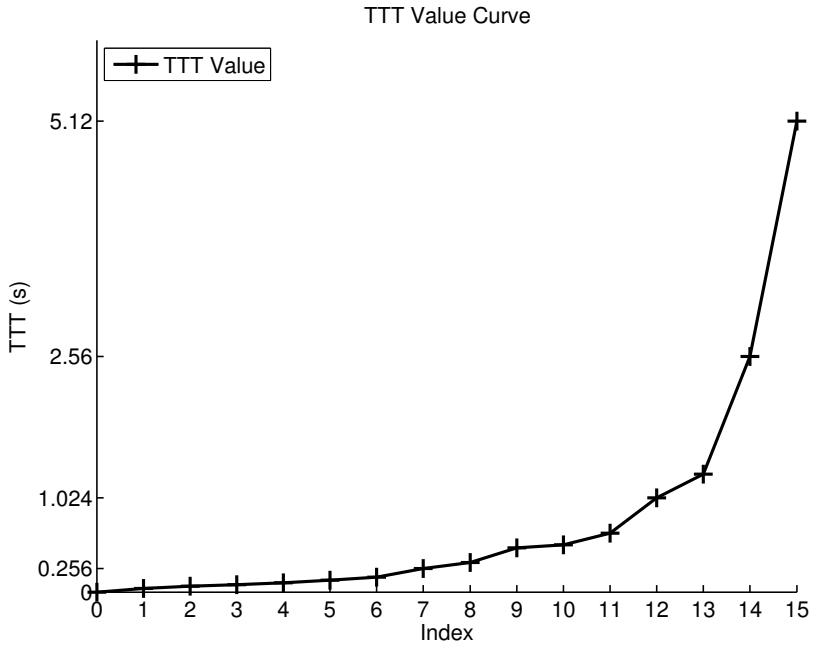


Figure 2.3: Graph of TTT values.

There are 336 different combinations of TTT and hys values. Having such a large range of combinations means that pairs of values can mean that a neighbouring eNodeB has to be better by a large value of hys but for a small value of TTT or vice-versa. This makes for an interesting dynamic for which pairs of values will work the best in any given environment.

2.3.2 Procedure

In LTE there are eight different triggers defined for initiating handovers. Table 2.3 shows different trigger events and how they are defined [8].

Event Type	Trigger Criteria
A1	Serving becomes better than a threshold.
A2	Serving becomes worse than a threshold.
A3	Neighbour becomes offset better than Primary Cell (PCell).
A4	Neighbour becomes better than threshold.
A5	PCell becomes worse than threshold1 and neighbour becomes better than threshold2.
A6	Neighbour becomes offset better than Secondary Cell (SCell).
B1	Inter RAT neighbour becomes better than threshold.
B2	PCell becomes worse than threshold1 and inter RAT neighbour becomes better than threshold2.

Table 2.3: Table of the different LTE Trigger types and their criteria.

Out of the eight triggers the A3 event is the most common and its definition is that a neighbouring eNodeB must give the UE better Reference Signal Received Power (RSRP) by an amount defined by the hys, for a length of time defined by the TTT. [9] The A3 event can be represented by the following equation:

$$RSRP_{Neighbouring} + hys > RSRP_{Serving} \tag{2.1}$$

When a handover event is triggered a measurement report is sent from the UE to the Serving eNodeB. The measurement report contains the information required for the Serving eNodeB to make a decision on whether to initiate a handover or not.

The full, high-level, procedure for a LTE handover is as follows:

1. If a Neighbouring eNodeB is found to be better than the Serving eNodeB a measurement report is sent by the UE to the Serving eNodeB.

2. The Serving eNodeB considers the information in the measurement report and decides whether or not a handover should take place.

3. If it is decided that a handover should take place then a message is sent to the Neighbouring eNodeB to prepare resources for the UE.

4. Once the resources are ready for the UE the new Serving eNodeB sends a message to the old eNodeB to release the resources it previously had for the UE.

5. Finally a message is sent to the MME to finalise the handover process.

Chapter 3

Machine Learning

Machine learning is a form of Artificial Intelligence (AI) that involves designing and studying systems and algorithms with the ability to learn from data. This field of AI has many applications within research (such as system optimisation), products (such as image recognition) and advertising (such as adverts that use a user's browsing history). There are many different paradigms that machine learning algorithms use. Algorithms can use training sets to train an algorithm to give appropriate outputs; other algorithms look for patterns in data; while others use the notion of rewards to find out if an action could be considered correct or not. [10] Three of the most popular types of machine learning algorithms are:

Supervised learning is where an algorithm is trained using a training set of data. This set of data includes inputs and the known outputs for those inputs. The training set is used to fine-tune the parameters in the algorithm. The purpose of this kind of algorithm is to learn a general mapping between inputs and outputs so that the algorithm can give an accurate result for an input with an unknown output. This type of algorithm is generally used in classification systems.

Unsupervised learning algorithms only know about the inputs they are given. The goal of such an algorithm is to try and find patterns or structure within the input data. Such an algorithm would be given inputs and any patterns that are contained would become more and more visible the more inputs the algorithm is given.

Reinforcement learning uses an intelligent agent to perform actions within an environment. Any such action will yield a reward to the agent and the agent's goal is to learn about how the environment reacts to any given action. The agent then uses this knowledge to try and maximise its reward gains.

3.1 Reinforcement Learning

In reinforcement learning an intelligent agent is learning what action to do at any given time to maximise the notion of a reward. In the beginning the agent has no knowledge of what action it should take from any state within the learning environment. It must instead learn through trial and error, exploring all possible actions and finding the ones that perform the best.

The trade-off between exploration and exploitation is one of the main features of reinforcement and can greatly affect the performance of a chosen algorithm. A reinforcement learning algorithm must contemplate this trade-off of whether to exploit an action that resulted in a large reward or to explore other actions with the possibility of receiving a greater reward.

Another main feature of reinforcement learning is that the problem in question is taken into context as a whole. This is different from other types of machine learning algorithms, as they will not considered how the results of any sub-problems may affect the problem as a whole.

The basic elements required for reinforcement learning is as follows:

- A Model (M) of the environment that consists of a set of States (S) and Actions (A).
- A reward function (R).
- A value function (V).
- A policy (P).

The model of the environment is used to mimic the behaviour of the environment, such as predicting the next state and reward from a state and taken action. Models are generally used for planning by deciding what action to take while considering future rewards.

The reward function defines how good or bad an action is from a state. It is also used to define the immediate reward the agent can expect to receive. Generally a mapping between a state-action pair and a numerical value is used to define the reward that the agent would gain. The reward values are used to define the policy where the best value of state-action pair is used to define the action to take from a state.

While the reward function defines the immediate reward that can be gained from a state, the value function defines how good a state will be long-term. This difference can create possible conflicts of interest for an agent; so while its goal is to collect as much reward as possible, it has to weigh up the options of picking a state that may provide a lot of up front reward but not much future reward

against a state with a lot of future reward but not much immediate reward. This trade-off is similar to that of exploration versus exploitation as it can define how successful a reinforcement algorithm is.

The policy is a mapping between a state and the best action to be taken from that state at any given time. Policies can be simple or complex; with a simple policy consisting of a lookup table, while more complex policies can involve search processes. In general most policies begin stochastic so that the agent can start to learn what actions are more optimal. [11]

3.2 Q-Learning

Q-Learning is a type of reinforcement learning algorithm where an agent tries to discover an optimal policy from its history of interactions within an environment. What makes Q-Learning so powerful is that it will always learn the optimal policy (which action a to take from a state s) for a problem regardless of the policy it follows, as long as there is no limit on the number of times the agent can try an action. Due to this ability to always learn the optimal policy, Q-Learning is known as an Off-Policy learner. The history of interactions of an agent can be shown as a sequence of State-Action-Rewards:

$$< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2... >$$

This can be described as the agent was in State 0, did Action 0, received Reward 0 and transitioned into State 1; then did Action 1, received Reward 1 and transitioned into State 2; and so on.

The history of interactions can be treated as a sequence of experiences, with each experience being a tuple.

$$< s, a, r, s' >$$

The meaning of the tuple is that the agent was in State s, did Action a, received Reward r and transitioned in State s'. The experiences are what the agent uses to determine what the optimal action to take is at a given time.

The basic process of a Q-Learning algorithm can be seen in Algorithm 3.1. The general process requires that the learning agent is given a set of states, a set of actions, a discount factor γ and step size α. The agent also keeps a table of Q-Values, denoted by $Q(s, a)$ where s is a state and a is an action from that state. A Q-Value is also an average of all the experiences the agent has with a specific state-action pair. This allows for good and bad experiences to be averaged out to giving a reasonable estimation of the actual value of state-action pair.

The process of averaging out experiences is done using Temporal Differences. It could be said that the best way to estimate the next value in a list is to take the average of all the previous values. Equation 3.1 shows this process.

$$A_k = \frac{(v_1 + \cdots + v_k)}{k} \tag{3.1}$$

Therefore:

$$kA_k = v_1 + \cdots + v_k \tag{3.2}$$

$$= (k-1)A_{k-1} + v_k \tag{3.3}$$

Then dividing by k gives:

$$A_k = (1 - \frac{1}{k})A_{k-1} + \frac{v_k}{k} \tag{3.4}$$

Then let $\alpha_k = 1/k$:

$$A_k = (1 - \alpha_k)Ak - 1 + \alpha_k + v_k \tag{3.5}$$

$$= A_{k-1} + \alpha_k(v_k - A_{k-1}) \tag{3.6}$$

The part of Equation 3.6 where the difference $v_k - A_{k-1}$ is seen is known as the Temporal Difference Error or TD Error. This shows how different the old value A_{k-1} is from the new value v_k. The new value of the estimate, A_k, is then the old estimate, A_{k-1}, plus the TD error times α_k.

The Q-Values, therefore, are defined using temporal differences and Equation 3.7 shows the formula to calculate the values, where α is a variable between 0 and 1 and defines the step size of the algorithm. If the step size were 0 then the algorithm would ignore any rewards received and if the step size were 1 the algorithm would consider the rewards gained just as much as the previous experiences of a state-action pair. The discount factor, γ, is also a variable between 0 and 1 and defines how much less future rewards will be worth compared to the current reward. If the discount factor were to be 0, then the future rewards would not be considered a lot. If the discount factor were to be 1, then the future rewards would be worth as much as the current rewards. The possible future rewards ($max_a Q(s, a)$) is the maximum of the Q-Values of all possible state-action pairs from the action selected.

$$Q[s, a] = Q[s, a] + \alpha(r + \gamma max_{a'} Q[s', a'] - Q[s, a]) \tag{3.7}$$

The table of Q-Values can either be initialised as empty or with some values pre-set to try and lead the agent to a specific goal state. Once the agent has initialised these parameters it observes the starting state. The starting state can either be chosen by random or be a pre-determined start state for the problem. The agent will then choose an action. Actions are chosen either stochastically or by a policy. Once an action has been chosen the agent will carry out the action and receive a reward. This reward is used to update the table of Q-Values using Equation 3.7. Finally the agent moves into the new state and repeats until termination; which can be either when the agent discovers a goal state or after a certain number of actions have be taken.

Require:
S is a set of states
A is a set of actions
γ the discount reward factor
α is the learning rate

1: **procedure** Q-LEARNING(S, A, γ, α)
2: real array $Q[S, A]$
3: previous state s
4: previous action a
5: initialise $Q[S, A]$ arbitrarily
6: observe current state s
7: **repeat**
8: select and carry out an action a
9: observe reward r and state s'
10: $Q[s, a] \leftarrow Q[s, a] + \alpha(r + \gamma max_{a'} Q[s', a'] - Q[s, a])$
11: $s \leftarrow s'$
12: **until** termination
13: **end procedure**

Algorithm 3.1: Q-Learing Procedure.

After a Q-Learning algorithm has finished exploring the model of the environment it creates a policy. The policy is generated by searching across all actions for a state and finding the next state with the greatest value. The policy is therefore a lookup table that maps a state with the best possible next state. The policy created can then be used to solve the problem that the Q-Learning agent was exploring. [12]

Chapter 4

Simulation Design and Implementation

The simulation is a very important part of the project. It is required to provide the basic functionality of a LTE network. For simplicity the simulation was broken down into two main components; the mobile (UE) and the base station (eNodeB). Due to the project revolving around the handover process in LTE, it made sense for the two main components of the simulation to be the mobile and the base station; it is the mobile that triggers the measurement report and the base station that makes the decision on whether a handover should take place or not. Each base station would also be given its own Q-Learning agent since each base station is unique. Since the A3 event trigger (Table 2.3) is the most common it was decided that it would be the only trigger implemented in the simulation to reduce the complexity within the simulation.

It was required for the mobile to be able to move freely around a group of base stations. It was decided that this movement should be random because if any machine learning algorithm can handle random movement then it should also be able to handle regimented movement. The movement that the mobile follows is defined by a Mobility Model and the choice of mobility model is explained in Section 4.3. It was decided that when a UE would get to the point where its next step would move it beyond the bounds of the simulation area that it would just move in the reflected direction away from the wall.

In wireless communications the received signal strength from a transmitter degrades the further away from the transmitter the receiver is. A propagation model can be used to define the way in which the signal strength degrades. The propagation model is very important to the simulation as it will define how far away from a base station the mobile can be without dropping the call. A comparison and explanation of the choice of propagation model can be seen in Section 4.4.

4.1 Simulation Parameters

Within the simulation there are many variable parameters that need to be assigned values. Such parameters are: the height of the base stations, the height of the UEs, the dimensions of the simulation area and the positioning of the base stations. Other parameters that also needed to be considered are the number of base stations and the number of UEs; as well as the transmission power of the base stations and the time limit for what would be perceived to have been a connection ping-pong. The type of environment also had to be chosen; whether the simulation would be within a rural, urban, small city, medium city or large city environment.

The type of environment was decided to be a medium sized city. It was decided that the height of any UE would be $1m$ as it would normally be in the possession of a human being. The height of any base station would be $60m$ because base stations are either placed upon tall buildings or structures so that they produce better coverage. Since it was decided that the environment would be a medium sized city, this height made sense since there would be some large multi-storey building. It was also decided that the transmission power of the base stations would be $46dBm$ and the time limit for ping-pongs to occur would be $5s$ as they were the values that had been used in similar projects [7]. The simulation area and positioning of the base stations both depended on the propagation model used and the area of coverage that could be expected from it. The decision of the propagation model used can be found in Section 4.4. From Figure 4.2 it was found that for the chosen propagation model the Path Loss would reach the LTE noise floor of $-97.5dB$ at around $2km$. Therefore, from this it was decided that the area for the simulation would be $6km$ by $6km$. It was then decided that 9 base stations would provide good coverage in this area. Each of these base stations will also have there own Q-Learning agent since each base station has there own unique TTT and hys values. The base stations were placed in the following locations and the coverage can be seen in Figure 4.1.

1. X Co-Ordinate: 500.0, Y Co-Ordinate: 500.0
2. X Co-Ordinate: 3000.0, Y Co-Ordinate: 0.0
3. X Co-Ordinate: 5500.0, Y Co-Ordinate: 500.0
4. X Co-Ordinate: 0.0, Y Co-Ordinate: 3000.0
5. X Co-Ordinate: 3000.0, Y Co-Ordinate: 3000.0
6. X Co-Ordinate: 6000.0, Y Co-Ordinate: 3000.0

7. X Co-Ordinate: 500.0, Y Co-Ordinate: 5500.0

8. X Co-Ordinate: 3000.0, Y Co-Ordinate: 6000.0

9. X Co-Ordinate: 5500.0, Y Co-Ordinate: 5500.0

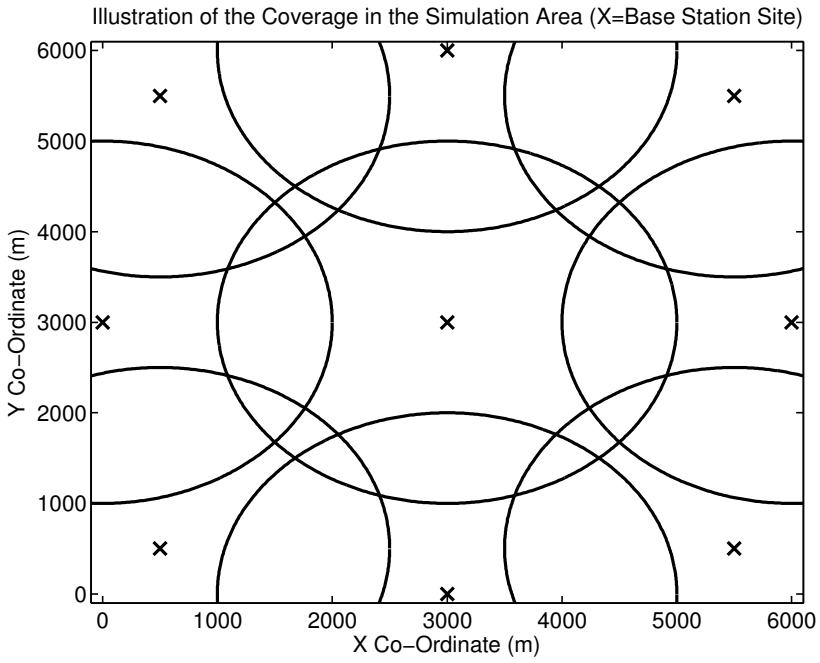


Figure 4.1: Illustration of Coverage within the Simulation Area.

Finally, it was decided that 10 UEs would be used in the simulation because this would allow for the learning done by the Q-Learning agents to happen faster than if there were just 1 UE. The UEs would also start on top of different base stations so that they will start in different places allowing for the Q-Learning agent for each base station to start learning straight away without having to wait for a UE to move into range, as at least 1 UE will start on top of it.

4.2 Discrete Event Simulation

The concept of time is very important in real world simulations. A popular method for creating this concept of time is Discrete Event Simulation (DES). It works on the basis of a scheduler where events, to be processed at a certain time, are passed to the scheduler and the scheduler then passes the event on to be processed. The way that DES creates the concept of time is by allowing events passed in the scheduler to have a time that it should be processed at, the scheduler will keep all the events that are currently still to be processed in an ordered list, the scheduler will then 'jump' to the time the first event in the list

is to be processed at and passes the event on to be processed. The process of 'jumping' to the next time that the first event in the list is to be processed at is what gives DES the concept of time.

Another advantage of DES is that the passing of events from the scheduler not only gives a concept of time but it also allows for a message to be sent with the event to tell a different part of the simulation what to do at a given time. This message can also pass parameters for the other part of the simulation to use.

4.3 Mobility Model

A mobility model defines the way in which an entity will move. For the purposes of the simulation the mobility model used needed to be random in nature. After some research it was decided that the mobility model to be used in the simulation would either be the Random Direction or Random Waypoint model because they are two of the most popular random mobility models. [13]

4.3.1 Random Direction Model

The Random Direction Model is defined as follows:

1. Select a direction randomly between 0 and 359 degrees.
2. Select a random speed to move at.
3. Select a random duration to move for.
4. Move in the selected direction at the selected speed for the selected duration.
5. Repeat until termination.

4.3.2 Random Waypoint Model

The Random Waypoint Model is defined as follows:

1. Randomly select the co-ordinates for a point within the environment.
2. Select a random speed to move at.
3. Select a random length of time to pause for when the destination is reached.
4. Move towards the selected co-ordinates at the selected speed
5. Pause for the randomly selected length of time.

6. Repeat until termination.

It was decided that the Random Direction Model would be used in the simulation because the Random Waypoint Model has the problem that it is possible to select the co-ordinates of a point very close to where you begin and then pause for a long period time. The possibility of that happening is undesirable within the simulation. Random Direction does not have this problem and it is also possible to set boundaries on the parameters to make sure that a minimum distance is travelled. [13]

4.4 Propagation Model

A propagation model defines how the received signal from a transmitter decays the further from the transmitter you are. There are many different models available, all with different functions and purposes. After some research three models were considered; the Okumura-Hata Model, the Egli Model and the Cost231-Hata Model.

4.4.1 Okumura-Hata Model

The Okumura-Hata model is very popular for simulating transmissions in built up areas. Equations 4.1, 4.2 and 4.3 show the formulas for the model.

$$L_u = 69.55 + 26.16log_{10}f - 13.82log_{10}h_B - C_h + [44.9 - 6.55log_{10}h_B]log_{10}d \quad (4.1)$$

For small or medium sized cities:

$$C_H = 0.8 + (1.1log_{10}f - 0.7)h_m - 1.56log_{10}f \quad (4.2)$$

For large cities.

$$C_H = \begin{cases} 8.29(log_{10}(1.54h_M))^2 - 1.1 & \text{if } 150 \leq f \leq 200 \\ 3.2(log_{10}(11.75h_M))^2 - 4.97 & \text{if } 200 < f \leq 1500 \end{cases} \quad (4.3)$$

Where:

- Lu is the path loss (dB).
- H_B is the height of the base station antenna 30 to 200m.
- H_R is the height of the mobile antenna 1 to 10m.
- f is the frequency of the transmission 150 to 1500MHz.

- C_H is the antenna correction factor.
- d is the distance between the base station and the mobile 1 to $20km$.

4.4.2 Egli Model

The Egli Model was another model that was considered for the simulation. Equations 4.4 and 4.5 shows the formulas for the model.

If h_m is less than or equal to $10m$ height:

$$P_L(dB) = 20log_{10}f_c + 40log_{10}d - 20log_{10}h_b + 76.3 - 10log_{10}h_m \tag{4.4}$$

If h_m is greater than or equal to $10m$ height:

$$P_L(dB) = 20log_{10}f_c + 40log_{10}d - 20log_{10}h_b + 83.9 - 10log_{10}h_m \tag{4.5}$$

Where:

- $P_L(dB)$ is the path loss (dB).
- h_B is the height of the base station antenna (m).
- h_M is the height of the mobile antenna (m).
- d is the distance between the base station and the mobile (m).
- f is the frequency of the transmission $3MHz$ to $3000MHz$.

4.4.3 Cost231-Hata Model

The Cost231-Hata model is an extension of the Okumura-Hata to work for frequencies between 1.5 GHz and 2 GHz. The formulas for this model can be seen in Equations 4.6, 4.7 and 4.8.

$$L = 46.3 + 33.9logf - 13.82logh_B - a(h_R) + [44.9 - 6.55logh_B]logd + C \tag{4.6}$$

$$a(h_R) = (1.1logf - 0.7)h_M - (1.58logf - 0.8) \tag{4.7}$$

$$C = \begin{cases} 0dB & \text{for medium cities and suburban areas} \\ 3dB & \text{for metropolitan areas} \end{cases} \tag{4.8}$$

Where:

- L is the path loss (dB).
- f is the frequency of the transmission 1500 to $2000MHz$.

- h_B is the height of the base station antenna 30 to 200m.
- h_M is the height of the mobile antenna 1 to 10m.
- d is the distance between the base station and the mobile 1 to 20km.
- $a(h_R)$ is the antenna correction factor.

A comparison of the three propagation models can be seen in Figure 4.2. For the graphs the height of the mobile was said to be 1m, while the height of the base station was said to be 60m. The frequencies used in the models were 2000MHz for the Egli and Cost231-Hata models and 1500MHz for the Okumura-Hata model. These values for the propagation models are then taken away from a base station transmit power of 46dBm so that the graphs are an estimate of the signal strength that would be received by a mobile at any distance up to 20km.

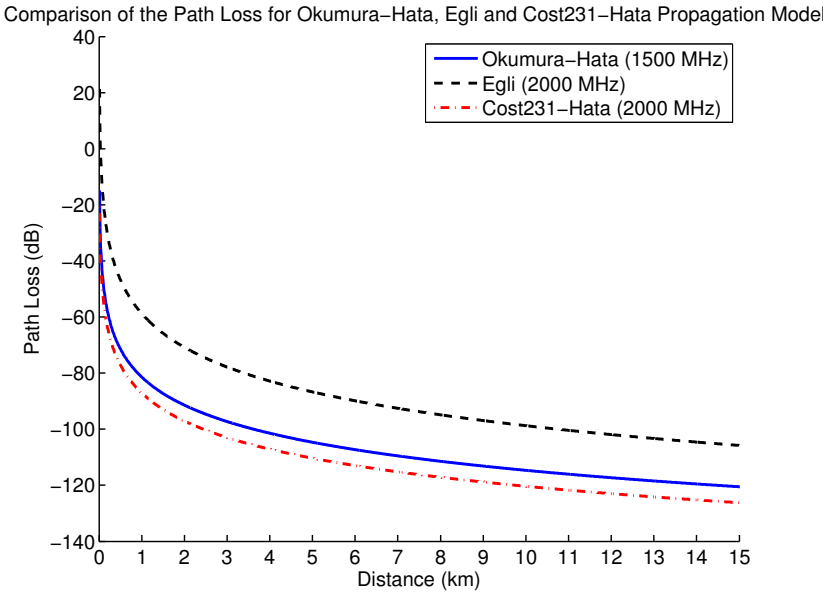


Figure 4.2: Graph of different Propagation Models.

After comparing the three models above it was decided that the Cost231-Hata Model would be the one used in the simulation due to it working with frequencies up to 2000MHz (which is the minimum operating frequency of LTE) unlike the Okumura-Hata model which only works up to 1500MHz. The Cost231-Hata Model was also picked over the Egli Model because the Egli Model used more parameters that would add more complexity to the simulation. [14, 15]

4.5 Simulation Implementation

The first step taken to implementing the simulation was to decide on the programming language to be used. After same deliberation it was decided that C++ would be used. It was chosen due to its Object Orientated nature being very powerful considering the multiple mobiles and base stations needing to be implemented.

The next step was to create the basic classes for the UEs and base stations, with basic functionality such as Accessors and Mutators for changing the basic parameters of the classes. Such parameters for the base stations would be if a mobile were currently connected to it and the X-Y co-ordinates representing its location. Such parameters for the UEs would be the ID number of the base station it is currently connected to and the X-Y co-ordinates representing its location.

It was important to get the Discrete Event Simulation framework implemented early into the project. Dr. Robert C. Atkinson kindly donated a DES library for C++ that he had. This meant that a DES framework did not have to be created from scratch, which allowed more development time to be spent on other aspects of the simulation. The DES library itself was very simple to use once some experience was had with it. There were two main parts to the library; they were the Scheduler and the Event Handler. The Scheduler is the class that provides the discrete time steps in the simulation as well as passing events to the event handlers. The Event Handler is an abstract class that is the super class to the UEs, base stations and Q-Learning agents; an example of the implementation of the inheritance can be seen in Listing 4.1. The characteristics that are inherited include the Handler method, which receives events from the Scheduler. The way in which the inheritance works as well as how the Scheduler and other classes interact with each other can be seen in the UML diagram in Figure 4.3.

Listing 4.1: Implementation of inheritance from Event Handler.

```
mobile::mobile(scheduler* gs) : event_handler(gs) {
```

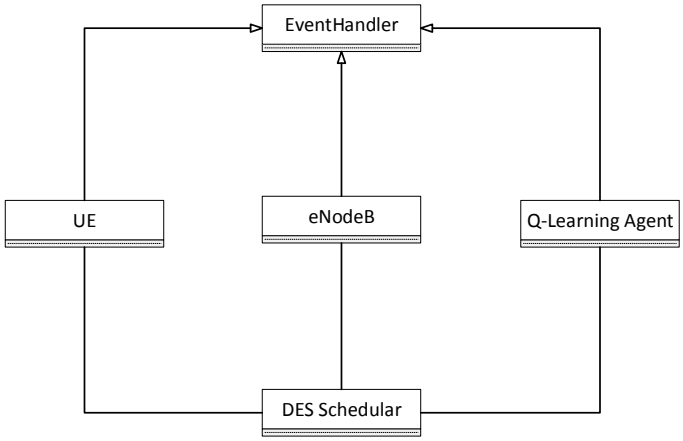


Figure 4.3: Illustration of UML Diagram.

After the DES library was integrated into UE and base station classes the next part to be implemented was the Random Mobility model (decision on the model chosen can be found in Section 4.3). The implementation of the mobility model made use of the DES library by using it to make the UEs take 'steps by sending itself an event after each step, with a delay of 0.01 seconds, telling itself to take another step. The full implementation of the model can be found in Appendix A.

Once the mobility model was implemented and tested the next part that needed to be implemented was the Propagation model (decision on the model chosen can be found in Section 4.4). The implementation of the chosen model can be seen in Listing 4.2 where d is the distance the UE is from the base station and hm is the height of the UE, which are passed into the function. The frequency is defined by the variable f, the height of the base station by the variable hb and the transmission power from the base station of tx; all these variables are private to the base station.

Listing 4.2: Implementation of Cost231-Hata Propagation model.

```
double basestation::getProp(double d, double hm) {
 //for small or medium sized city
 double ahr = 0.8 + ((1.1 * log10(f) - 0.7) * hm) - (1.56 * log10(f));
 double prop = 46.3 + (33.9 * log10(f)) - (13.82 * log10(hb)) - ahr +
     ((44.9 - (6.55 * log10(hb))) * log10(d/1000)); //divide by 1000
     for km

 return (tx-prop);
}
```

When both the mobility and propagation models had been implemented the next part to be done was the handovers along with detection for call dropping and connection ping-pong's. These are a very important parts of the simulation and there implementation can be seen in Appendix A. While all three of these components worked correctly the handover triggering was not implemented as efficiently as they could have been. It ended up using decrementing variables instead of the DES library due to bugs that ended up being unsolved.

After all the previous components were implemented, tested and found to be functioning correctly the Q-Learning agents were implemented. The implementation for the agent can be found in Appendix A. The Q-Learning agent was overall implemented quite well, with the one exception being the 2D array of actions. Due to the nature of the problem having 336 states all with a maximum of 8 actions from the states, this meant that the array ended up being very large. Also due to some states having less than 8 actions from them the array needed to have -1s added to it so that the program knew it would need to choose a different actions; this interactions adds both redundant data and wasted processing time into the simulation.

The final part that needed to be implemented was the way in which the simulation recorded the results. It was decided that the best way to record the results would be to store them in arrays and then output them to appropriate text files that could then be analysed. This analysis ended up being done using MATLAB and then MATLAB code used to do the analysis can be found in Appendix A.

4.6 Simulation Testing

It was very important for the simulation and Q-Learning algorithm to be tested so that there was confidence that they would produce the correct results when they were run. There are many different types of testing that can take place to ensure that a piece of software is working correctly. Testing methods that were used in this project include: Unit testing, Black Box testing and White/Clear Box testing.

Unit testing involves testing individual parts or processes of a program so that the stakeholders can be confident in them to be fit for use. The different parts of the simulation that were tested this way were the Mobility Model, handovers, drops, ping-pong's and the Q-learning algorithm changing the values of TTT and hys correctly.

Black Box testing involves making sure that a function works as required without any knowledge of the under laying code. This type of testing is performed

by giving a function an input, and comparing the output from the function with a previously determined expected output.

White Box testing is used to make sure that the underlying code used in a function works as required. This type of testing is done by inserting print statements in the code to see how the respective variables are changing while the code is running. These are compared against previously determined expected values to confirm whether the function is working as intended.

The test cases used to test the simulation to make sure it was working correctly were:

Movement:

M001 Positive X and Y movement.

M002 Positive X movement and Negative Y movement.

M003 Negative X movement and Positive Y Movement.

M004 Negative X and Y movement.

M005 Movement against left wall of simulation area.

M006 Movement against bottom wall of simulation area.

M007 Movement against right wall of simulation area.

M008 Movement against top wall of simulation area.

Path Loss:

PL001 Received Path Loss correct.

Network Functions:

NF001 Handovers working correctly.

NF002 Drops working correctly.

NF003 Ping-Pong's detected correctly.

Q-Learning:

QL001 Q-Values updated correctly.

QL002 Changing state correctly.

QL003 Changing TTT and hys values correctly.

After the simulation was tested using the test cases above it was decided that the simulation was functioning correctly and was fit for use. The results of all the test cases can be seen in Appendix B.

Chapter 5

Handover Parameter Optimisation

The approach taken for optimising the handover parameters in LTE uses a Q-Learning algorithm based on the process given in Section 3.2. In the approach the model of the environment has a state for every combination of TTT and hys; giving a total number of 336 states. An action within the model can move to any other state that is different by one of the following changes to the handover parameters:

1. A single value increase of TTT.
2. A single value increase of hys.
3. A single value increase of both TTT and hys.
4. A single value decrease of TTT.
5. A single value decrease of hys.
6. A single value decrease of both TTT and hys.
7. A single value increase of TTT and a single value decrease of hys.
8. A single value increase of hys and a single value decrease of TTT.

For example if the learning agent is in the state where the TTT equals $0.256s$ and the hys equals $5.0dB$ and did action 3 from the list seen above; then the new TTT would equal $0.32s$ and the hys would equal $5.5dB$. The full list of hys values can be seen in Table 2.1 and the full list of TTT values can be seen in Table 2.2.

Having the actions only change the parameters by one increase or decrease of the TTT and hys values each time not only allows for more refined optimisation

of the parameters but it also makes sure that no large changes can suddenly happen.

Due to the nature to the kind of problem that is being solved, the reward gained by an action is dynamic and is likely to be different each time it is taken. Rewards are based on the number of drop and ping-pong's accumulated in the simulation for current state in the environment model. The rewards are defined by the following equation:

$$Reward = Handover_{successful}(10 * Drops + 2 * PingPong's) \tag{5.1}$$

The coefficients in Equation 5.1 are given the values of 10 for drops and 2 for ping-pong's. Drops are extremely bad for the QoS of a communication system so it's given a large value and the reason ping-pong's are multiplied by 2 to remove the successful handover that was caused by the ping-pong and give the agent a penalty. The reward is given to the agent and the Q-Value for that state is updated just before agent selects the next action to take. The agent then selects new actions in discrete time steps, this allows for the simulation to run for fixed periods of time with TTT-hys pairs specified by a state in the environment model.

After the agent has been given enough time to try every action at least once the Q-Learning agent generates a policy. This policy can then be used to attempt to optimise the handover parameters by changing the TTT and hys values after a call is dropped or the connection ping-pongs between base stations. The Q-Learning agent still receives rewards every time a call is dropped or the connection ping-pong's while following the generated policy. Doing this allows for the system to always be learning; even after the initial learning process that generated the policy.

The optimisation system was tested in two scenarios. One scenario was to have 10 UE's moving randomly around 9 base stations, with each UE being 1 m in height and each base station being 60 m in height, with the layout as seen in Figure 4.1, using the Random Direction mobility model seen in Section 4.3, where the speed of the UE is 1 to $4m/s$, which is walking speed and the duration of the direction is between 100 and 200 seconds. The other scenario is to have the UE moving at 10 to $15m/s$, which is roughly $30mph$. In these scenarios each mobile begins on top of one of the base stations before it starts moving so that handovers are not required as soon as the simulation starts. These scenarios would also be run with no fading in the RSS calculations so in to make the environment easier to learn for the agents.

Each base station has its own Q-Learning agent to optimise the TTT and hys values for that specific base station. The agents are given 1000000 seconds

to attempt to learn the environment that are working within, with each state being given 180 seconds to gain their reward. This length of time was chosen because there are 336 state each with a maximum of 8 actions, therefore the time needed to do all actions would take approximately 483840 seconds if each action was given 180 seconds. This length of time is less than half of the total time given so even due to the randomness of selecting next actions when learning the environment there should be enough time to try all state and most of the actions available. After the agents have learned the environment they generate a policy for their base station to follow. The simulation is then run for 200000 seconds to test how well the policies perform. The results for the scenarios can be seen in Sections 5.2 and 5.3.

5.1 Assessment Process

The results are assessed by how well the machine learning algorithm performed compared to not making any changes to the TTT and hys values. The comparison is made by observing how to ratio of dropped calls and ping-pong's to successful handovers changes over time [16]. The ratio of dropped calls is given by the following equation:

$$DropRatio = \frac{\#DroppedCalls}{\#SuccessfulHandovers} \tag{5.2}$$

The ratio for the connection ping-ponging between base stations is given by the following equation:

$$PingPongRatio = \frac{\#PingPong's}{\#SuccessfulHandovers} \tag{5.3}$$

It is also important to see how the TTT and hys values are being changes when the system is attempting to optimise them. This will allow it to be seen if the agents for the base stations come to some kind of consensus on the most optimal values of TTT and hys or if them come up with there own unique solutions. It is likely that Base Station 4, being the central base station and whose coverage overlaps with the coverage from every other base station, as seen in Figure 4.1, will come up with a different solution to the other base stations because it will be involved in the most handover attempts.

The simulation is run for four different starting states when compiling results for 10 UE's moving at walking or vehicle speeds. The first starting state was to have the TTT and hys both start at their maximum values, 5.12 second and 10 dB respectively, to see how the system attempted to optimise the values as it

is expected that a lot of dropped calls would occur for this set of values. The second start state was to give the TTT and hys their middle values, which are 0.256 seconds for TTT and 5 dB for hys. This start state would be expected to perform relatively well without any optimisation taking place due to the values neither being very large or very small. The third start state has the TTT and hys being at their lowest possible values of 0 seconds for TTT and 0 dB for hys. This starting state is expected to cause ping-pong's to occur because a handover will be triggered as soon as a neighbouring base station becomes better than the serving base station, instead of waiting to see if the neighbouring base station continues to be better for a period of time. The final starting start is to have the TTT start at a low value of 0.08 seconds and the hys start at a high value of 7.5 dB. This type of state was chosen to see how the well the system could optimise the values when it is likely that one will need to be increased and the other decreased. Each of there scenarios were run 10 times so that the results could be averaged and 95% confidence limits could be established. Doing this would allow it to be seen if the machine learning algorithm was likely to produce better results over multiple runs but also give indications on the best and worst possible performance according to the confidence limits.

5.2 Walking Speed

For the walking speed testing it is expected that the first scenario of having both the TTT and hys at their highest values will produce a large number of dropped calls, with the likelihood of ping-pong's occurring being vey small, and prompt the optimisation system to reduce both values. For the second scenario with both the TTT and hys being middle values it is expected that the system will perform better than if no optimisation takes place. It is also expected that the system in likely to still reduced both values in an attempt to improve performance. For the third scenario with both the TTT and hys being their lowest possible values it is expected that no or very few dropped calls will occur and than ping-pong's are far more likely to happen. The system should compensate for this by increasing both values. In the final scenario where the TTT is a small value and the hys is a large one it is still expected that no ping-pong's will occur and that the ratio of called drops may be fairly high. It is expected that the system would attempt to improve the performance by increasing the TTT and decreasing the hys.

5.2.1 Large Starting Values

The results of how the optimisation system compared to the static values can be seen in Figure 5.1 when both the TTT and hys started with their largest possible values of 5.12 seconds and 10 dB respectively. The results show that the process of optimising the values initially generated a very large increase in the number of dropped calls. However, the system then managed to improve rapidly and ended up having a better dropped call ratio by the end to the simulation run than that of the non-optimised system.

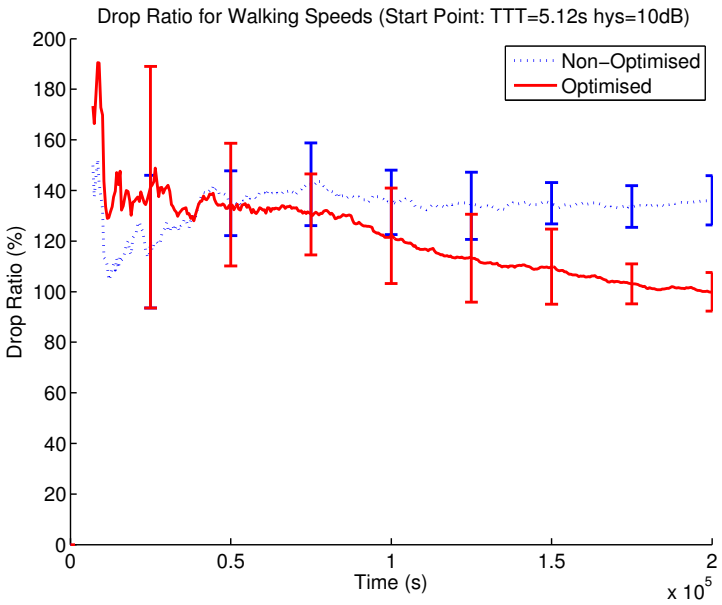
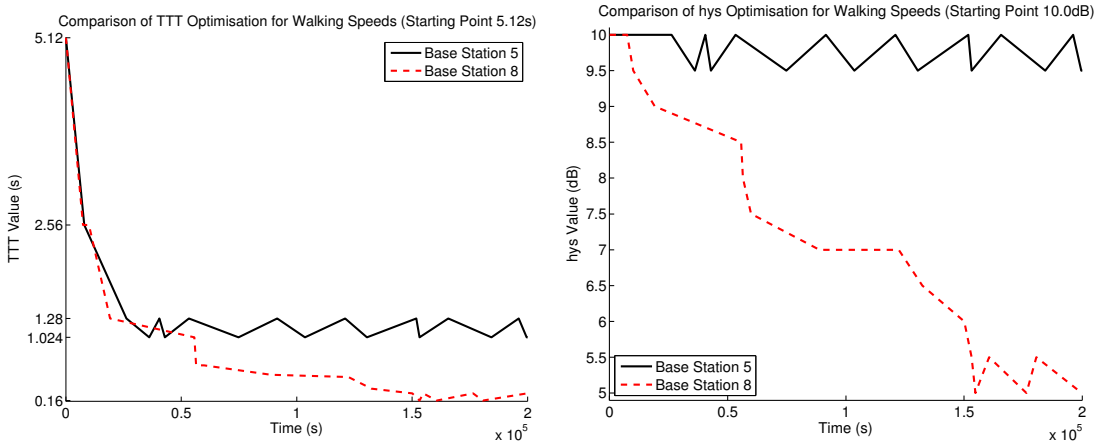


Figure 5.1: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=5.12s hys=10dB when UE traveling at walking speeds.

Figure 5.2 shows how the TTT and hys values for base stations 5 and 8 optimised over a simulation run. It can be seen in Figure 5.2a that both base stations were in consensus to reduce the value of their TTT from the starting value of 5.12s. However, both base stations were not in consensus for how much the value should be reduce by. Base station 8 reduced its TTT value to as low as 0.16s before settling between it and 0.256s. Base station 5 on the other hand reduced its TTT value a lot less, only going as low as to oscillate between 1.024s and 1.28s. It can also be seen that Base station 5 oscillated a lot between those two values and this could be an indication that the algorithm had got stuck between two non-optimal states and was not able to optimise the value anymore. This means that even though the optimisation improved the performance there was a large window of potential for further improvement.

While both base station 5 and 8 were in consensus about reducing their TTT values, they were not in consensus about their hys values. It can be seen in

Figure 5.2b that base station 5 did not try to reduce its value of hys at all and only oscillated between, the starting value of, $10dB$ and $9.5dB$. This is a stark contrast to base station 8 which greatly reduced its value of hys as low as $5dB$ before oscillating between it and $5.5dB$. Again, much like with the TTT values, base station 5 appears to oscillate between two non-optimal values as the oscillations are quite frequent and the learning agent for that base station has not tried any other values to see if they give an improvement. Also since base station 8 reduce its value greatly before oscillating between two values could mean that it reached what it thinks to be an optimal state.



(a) Changing TTT Values (b) Changing hys Values

Figure 5.2: Illustration of how the TTT and hys values changed over time for large values when UE traveling at walking speeds.

From these results it can be said that the system performed as expected by with the base stations shown having reduced both their values of TTT and hys. There were also a very high number for dropped calls and no ping-pong's, which was also expected in the simulation. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.1.1.

5.2.2 Middle Starting Values

Again as seen in Figure 5.3 the optimisation process performed a lot better than the static values when they were originally set to the their middle values of 0.256 seconds for TTT and 5 dB for hys. The results also show that unlike with the results seen in Figure 5.1 the optimisation process did not greatly increase the number of dropped call at first. Instead it was the non-optimised values that began with the large number of dropped calls. This means that when these dropped calls began to appear the machine learning algorithm changed the value of TTT and hys in such a way that the dropped calls seen for the non-optimised

values did not happen.

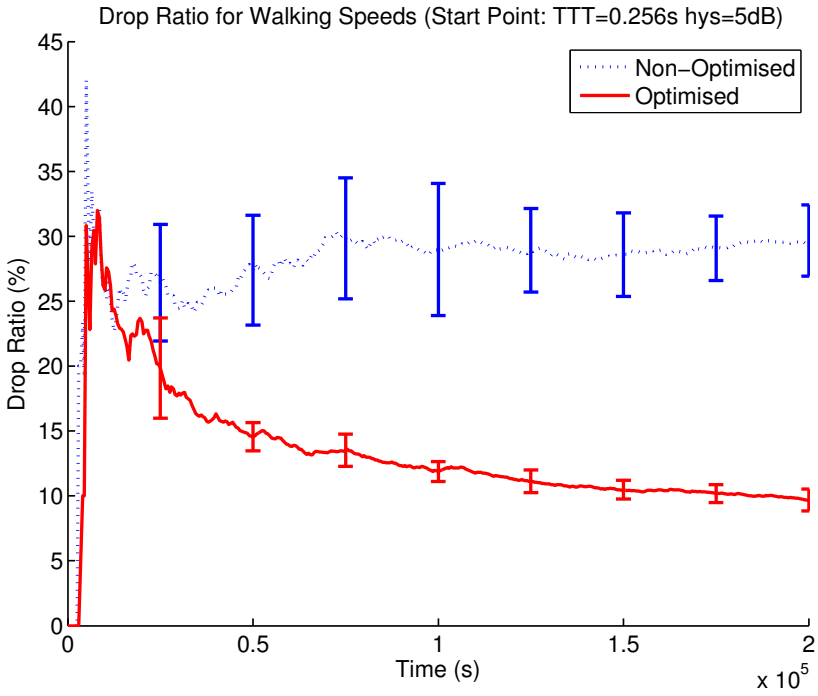
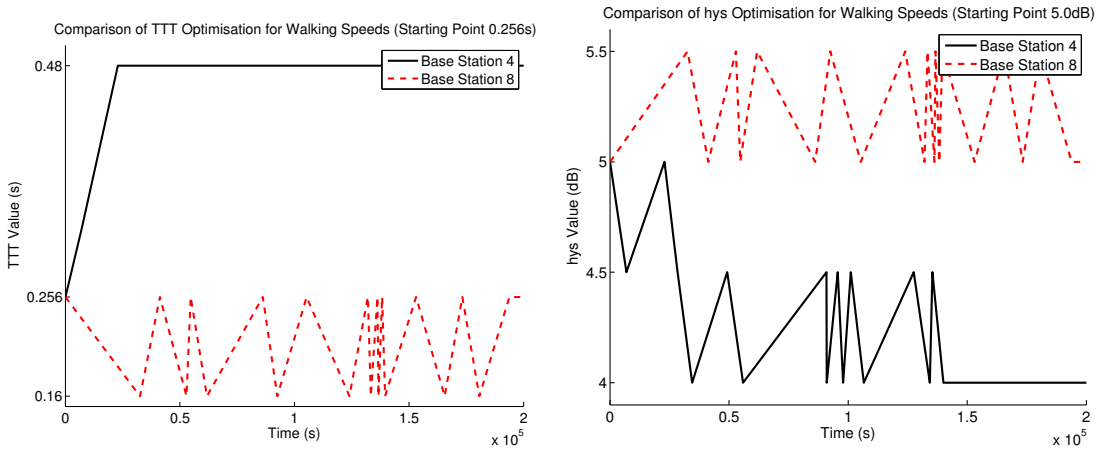


Figure 5.3: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0.256s hys=5dB when UE traveling at walking speeds.

A comparison of how base stations 4 and 8 optimised their TTT and hys values can be seen in Figure 5.4. It can be seen in that the two base station took different approaches when trying to optimise their TTT and hys values. Base station 4 can be seen to have increased its TTT value and decreased its hys value, while base station 8 decreased its TTT value and increased its hys value. When looking at Figure 5.4a it can be seen that base station 4 increases its value of TTT to $0.48s$ and then keeps that value steady. However, it can also be seen in Figure 5.4b that its value of hys actually ends up oscillating between $4dB$ and $4.5dB$ and oscillating quite frequently at first before settling at $4dB$ for the last quarter of the simulation. Therefore, while the state of TTT being $0.48s$ and hys being $4dB$ may not have looked like an optimal state at first, the fact that it made no changes to any values for the last quarter of the simulation means that it could be a very optimal state for the base station.

It is interesting to see that base station 8 actually ended up oscillating between the same two states in both this scenario and the scenario seen in Section 5.2.1. With both of these scenarios finished between the same states of TTT equalling $0.16s$, hys equalling $5dB$ and TTT equalling $0.256s$, hys equalling $5.5dB$ it means that the learning agent for that base station managed to find what it thinks be an optimal set of states and end up there when starting at two different states,

this is a very promising result.



(a) Changing TTT Values

(b) Changing hys Values

Figure 5.4: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at walking speeds.

From the base stations compared above, while the performance was increased a lot, these two base stations did not optimised the values as expected. Instead of reducing both values they both opted for a split approach increased one value while decreasing the other. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.1.2.

5.2.3 Small Starting Values

It turned out that ping-pong's were a very rare occurrence in the simulation. This was most likely due to there being no fading in the simulation and that Random Direction mobility model would having the UE moving in one direction for a long time meaning that the only likely occurrence of a ping-pong would be if a handover took place and the UE then turned around and moved the other direction.

Figure 5.5 shows how the optimisation system performed against the static values when the simulation started with the TTT being 0 seconds and the hys being 0 dB. It can be seen that the optimisation process and the static values performed very similar. It can be seen though that the error bars for the optimised system become a lot smaller, than those for the static values, the longer the simulation is run. This means that the optimisation system would be expected to perform better the majority of the time.

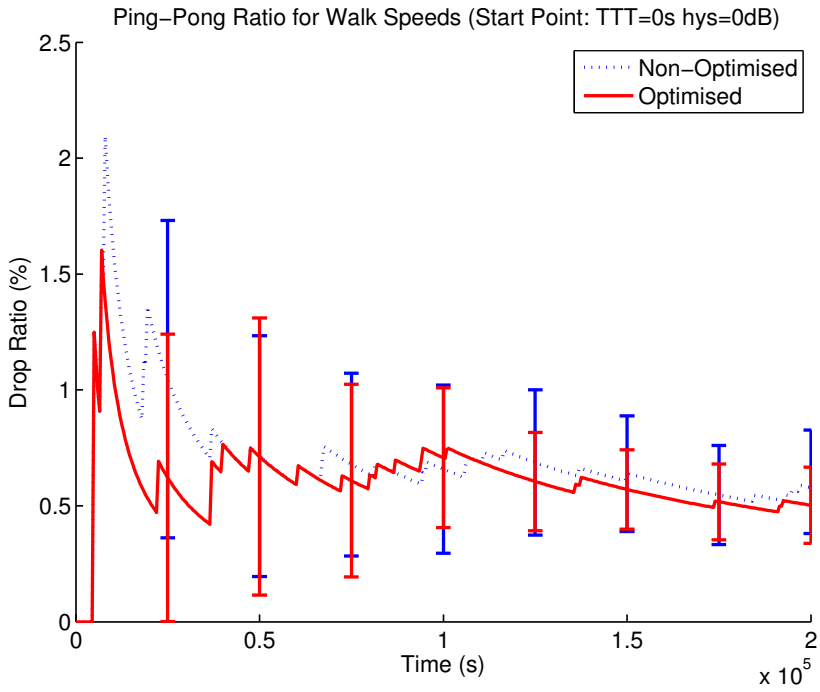
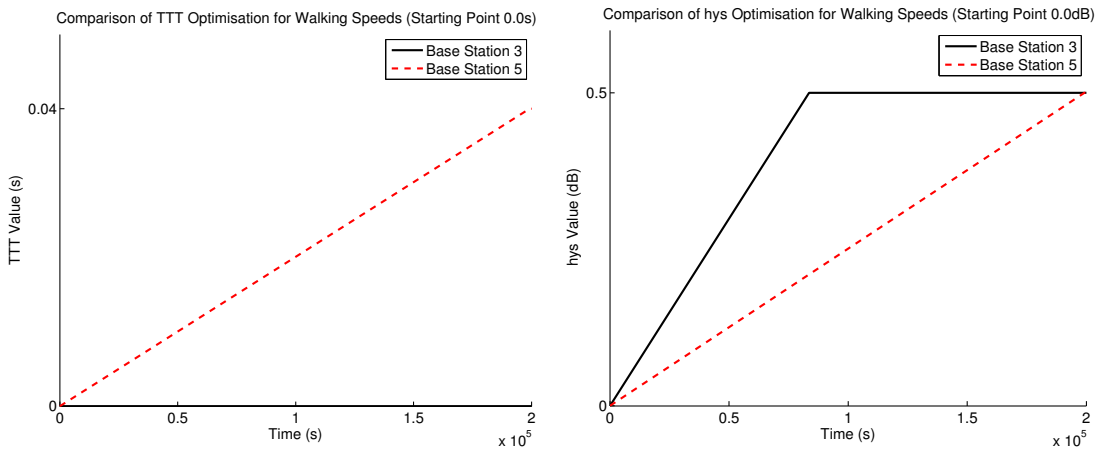


Figure 5.5: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0s hys=0dB when UE traveling at walking speeds.

A comparison of how base stations 3 and 5 changed their values of TTT and hys when a ping-pong occurred can be seen in Figure 5.6. It can be seen that while both base stations only encounter one ping-pong each they both reacted differently when they occurred. It can be seen that base station 5 increased both its values of TTT and hys from $0s$ and $0dB$ to $0.04s$ and $0.5dB$ respectively. Base station 3, however, decided to keep its value of TTT as $0s$ and only increase its hys to $0.5dB$. It is difficult to make any assumptions on which base station made the correct changes to their values as not a lot of changes actually happened.



(a) Changing TTT Values (b) Changing hys Values

Figure 5.6: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at walking speeds.

This scenario played out as expected with no dropped calls occurring; instead having ping-pongs determine the performance of the system. The system also compensated for this as expected by increase the values of TTT and hys as seen with base stations 3 and 5 and by the end of the simulation run the performance of the optimised values had become better than that of the non-optimised values. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.1.3.

5.2.4 Large hys and Small TTT Starting Values

Much like the first three scenarios, it can be seen in Figure 5.7 that, the optimisation system performed better than the static values when the TTT and hys started at 0.08 seconds and 7.5 dB respectively.

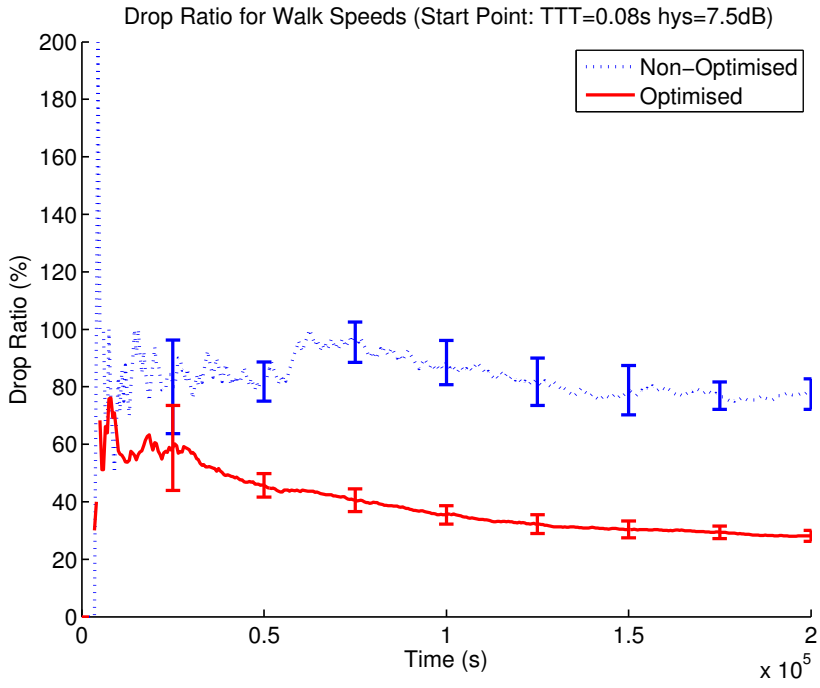
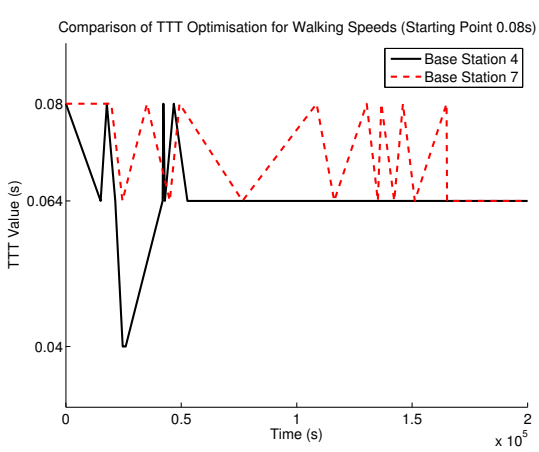


Figure 5.7: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0.08s hys=7.5dB when UE traveling at walking speeds.

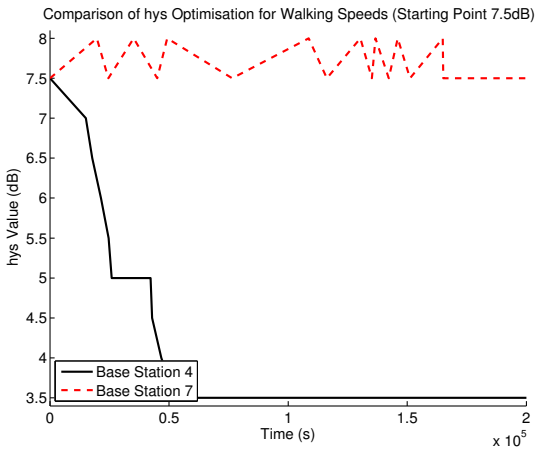
A comparison of how base stations 4 and 7 attempted to optimised the system can be seen in Figure 5.8. It can be seen in Figure 5.8a that both base stations were in consensus to reduce the value slightly with both base stations oscillating between the starting value of $0.08s$ and $0.064s$ before they both became steady at $0.064s$.

Figure 5.8b shows that while both base stations were in consensus about their TTT values they were not in consensus about their hys values. It can be seen that base station 7 actually increased its value of hys and ended up oscillating between the starting value of $7.5dB$ and $8dB$. This is vastly different to the changes made by base station 4 which reduced the value all the way down to $3.5dB$ and then stayed steady at that value.

It should also be noted that base station 4 appeared to reach an optimal state of TTT being $0.064s$ and hys being $3.5dB$ very quickly and stayed in this state for almost three quarters of the simulation run.



(a) Changing TTT Values



(b) Changing hys Values

Figure 5.8: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at vehicle speeds.

In the comparison seen above the system did not perform entirely as expected. While base station 4 did decrease its hys value, base station 7 actually increased it slightly, which was unexpected. Both base stations also performed expectantly decreasing their TTT values when they were expected to be increased. Even with this the system still performed better than the simulation when run with static values. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.1.4.

5.3 Vehicle Speed

For the vehicle speed scenarios it is expected that number of dropped calls would increase over those seen in the walking speed scenarios due to the UE moving a lot faster and therefore giving a lot less time for the decision to handover or not. It is still expected that for the first scenario the system will reduce both values for TTT and hys. In the second scenario of the middle values it can also be expected the both values for TTT and hys will be reduced as dropped calls will still be expected fairly often at the speed the UE will be moving at. For the third scenario where both values for TTT and hys are at their lowest it is expected that the system will increase these values has ping-pong's should still be likely to happen, even at the high speeds. In the fourth scenario it is expected that the system is more likely to reduce both values, unlike when the UE was moving at walking speeds, because of the speed the UE is moving at decisions need to be made quickly.

5.3.1 Large Starting Values

It can be seen in Figure 5.9 that the optimised system performed better than the system using the static values of 5.12 seconds for TTT and 10 dB for hys.

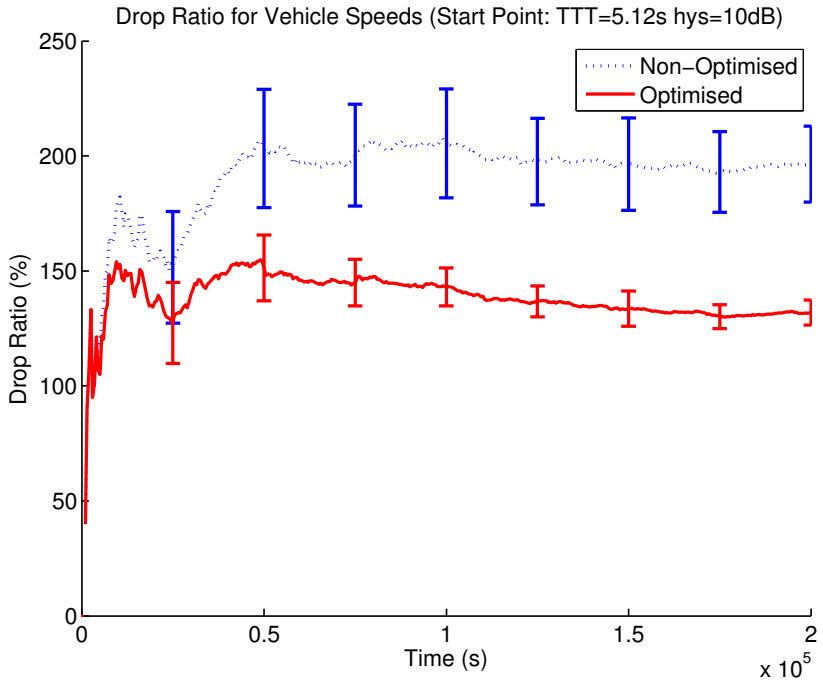
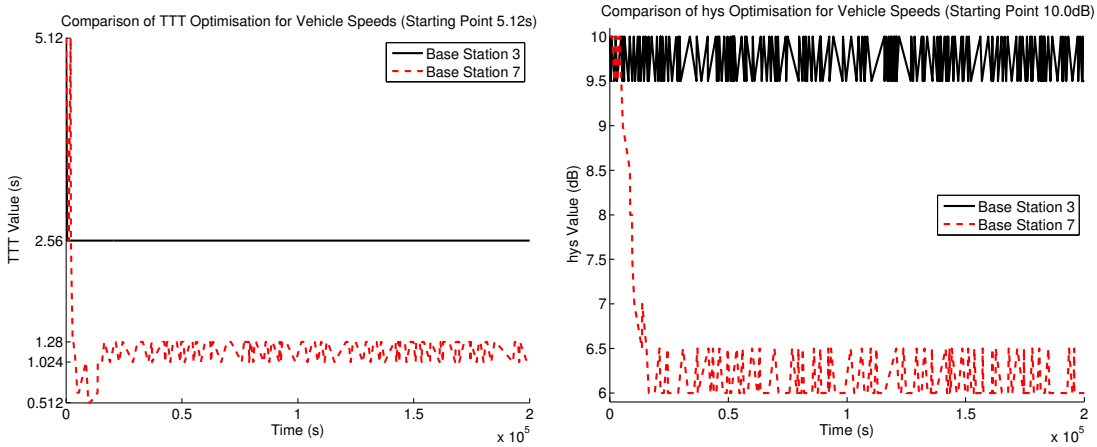


Figure 5.9: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=5.12s hys=10dB when UE traveling at walking speeds.

A comparison between how base stations 3 and 7 optimised their TTT and hys values can be seen in Figure 5.10. It can be seen in Figure 5.10a that both base stations were in consensus about what to do to their TT values which was to reduce them. However, both base stations were not in consensus about how much to reduce it by. Base station 3 reduced its value by only one single decrease to $2.56s$ and then stayed steady at that value. Base station 7, however, reduced its value as low as $0.512s$ before ending up oscillating between $1.024s$ and $1.28s$. the oscillation between those two values can be seen to be very frequent meaning the value should have been reduced more. This would make sense since the UEs in this scenario are moving quickly meaning that there is less time for handovers to be triggered.

It can be seen in Figure 5.10b that while base station 3 maybe have kept its value of TTT steady, its value of hys was oscillating extremely frequently between the starting value of $10dB$ and $9.5dB$. This means that those states are very likely non-optimal and the learning agent for the base station should have reduced that values a lot more. As for base station 7, it reduced its value of hys a lot more that base station 3 did and ended up oscillating between $6dB$ and $6.5dB$. Again

the oscillation is very frequent and it is likely that the learning agent should have reduced the value more.



(a) Changing TTT Values (b) Changing hys Values

Figure 5.10: Illustration of how the TTT and hys values changed over time for large values when UE traveling at vehicle speeds.

For this scenario while the system performed as expected and still performed better than that of the non-optimised system, it can be seen that it still did not perform all that well and appear be being getting stuck in non-optimal states. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.2.1.

5.3.2 Middle Starting Values

In this scenario the optimised system perform much better than of the static values, where it TTT and hys values were initialised to 0.256 seconds and 5 dB respectively. The graphs of the dropped calls ratios can be seen in Figure 5.11.

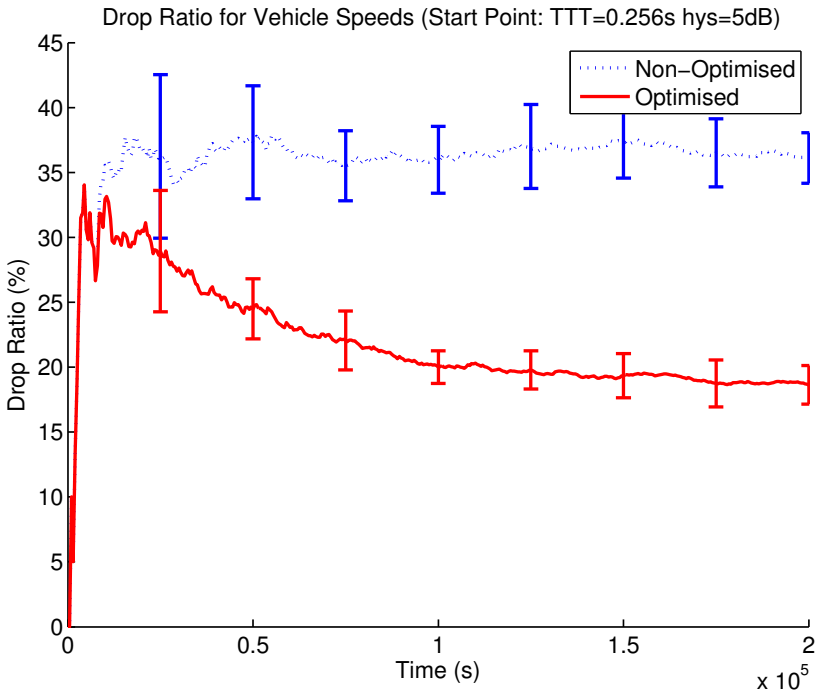
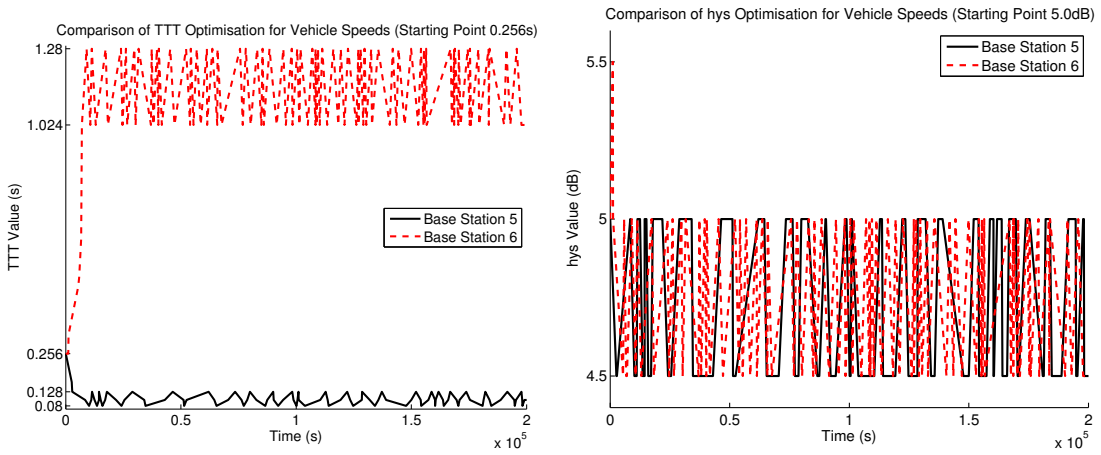


Figure 5.11: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0.256s hys=5dB when UE traveling at vehicle speeds.

Figure 5.12 shows a comparison of how base stations 5 and 6 attempted to optimised their values of TTT and hys. It can be seen in Figure 5.12a that the two base stations took completely different approaches to optimising their TTT values. Base station 5 reduced its value of TTT and actually ended up looping between $0.08s$, $0.1s$ and $0.128s$, with the frequency of the changes being fairly frequent. Base station 6, however, increased its value of TTT as high as $1.28s$ and ended up oscillating between that value and $1.024s$, with its movements being slightly more frequent that those of base station 5. It is difficult to say which base station may be correct since both still made changes frequently while in oscillation, but since the UEs are moving very fast in this scenario is would be assumed that base station 5s made changes would be more correct.

Unlike with the changes base stations 5 and 6 made to their TTT values, they were in consensus about what changes should be made to their hys values. Both base stations end up oscillating between the starting value of $5dB$ and $4.5dB$.



(a) Changing TTT Values

(b) Changing hys Values

Figure 5.12: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at vehicle speeds.

In this scenario it has been seen that base station 5 functioned as expected reducing both its values of TTT and hys. On the other hand base station 6 made very unexpected changes to its TTT value by actually greatly increasing its value. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.2.2.

5.3.3 Small Starting Values

It can be seen that for this scenario the optimised system and static values performed identically at first, but by the end the ratio of ping-pong's was better for the optimised system than that of the non-optimised values.

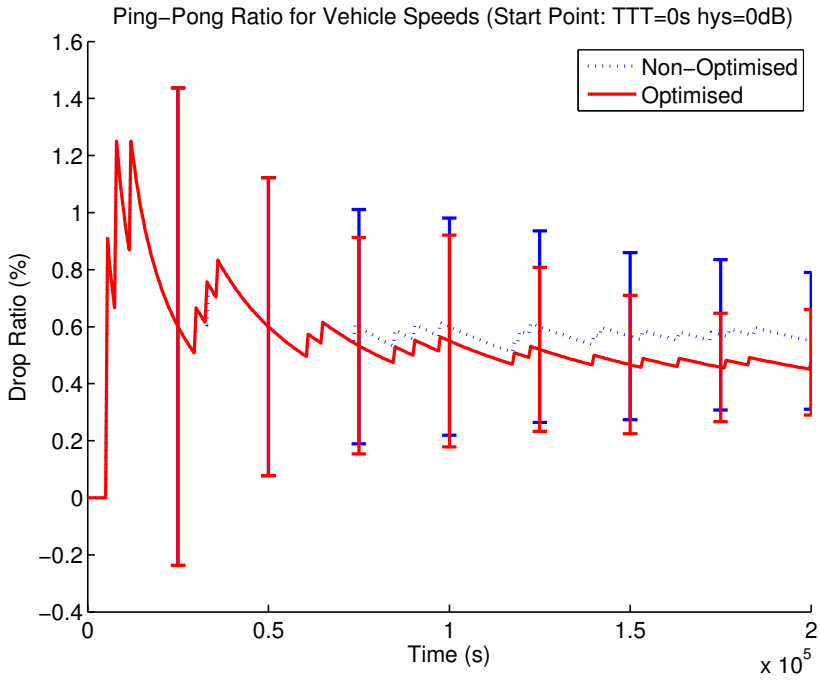
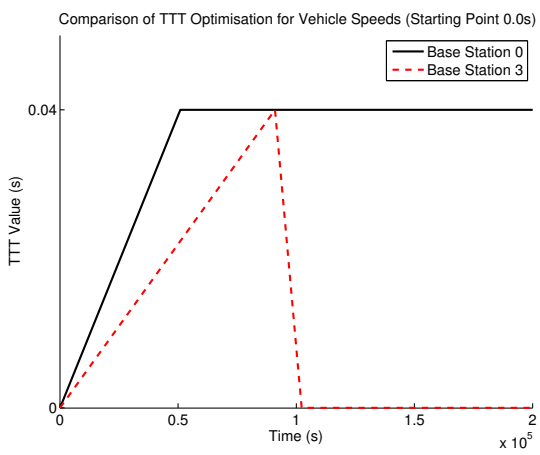


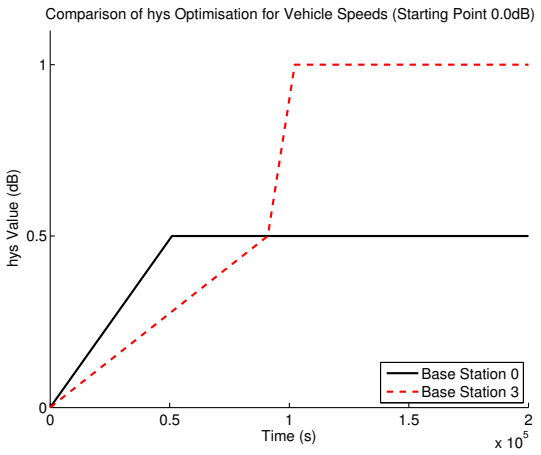
Figure 5.13: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0s hys=0dB when UE traveling at vehicle speeds.

Figure 5.14 shows a comparison of how base stations 0 and 3 reacted when they experienced connection ping-pongs. It can be seen that base station 3 actually encountered two ping-pongs during this simulation run. It actually increased is TTT value after the first ping-pong to $0.04s$ but after the second it actually reduced it back down to $0s$. However, the learning agent for the base station increased its value of hys both times it encountered a ping-pong reaching $1dB$ by the end of the simulation.

Unlike base station 3, base station 0 only encountered one ping-pong and it increased both its values of TTT and hys to $0.04s$ and $0.5dB$ respectively. Much like the results seen for this scenario when done with walking speeds it is hard to determined which base station made the correct changes as very few changes were actually made.



(a) Changing TTT Values



(b) Changing hys Values

Figure 5.14: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at vehicle speeds.

Again the optimisation system functioned as expected with increasing both the TTT and hys values due to ping-pong's occurring. This was expected because even though the UE is travel quickly the system still has to make sure that it does not trigger a handover too quickly. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.2.3.

5.3.4 Large hys and Small TTT Starting Values

As seen in the first three scenarios for the vehicle speed testing the optimised system managed to perform better than that of the non-optimised system when starting with the TTT being 0.08 seconds and the hys 7.5 dB. However, the improvement was not quite as great as seen in the other scenarios.

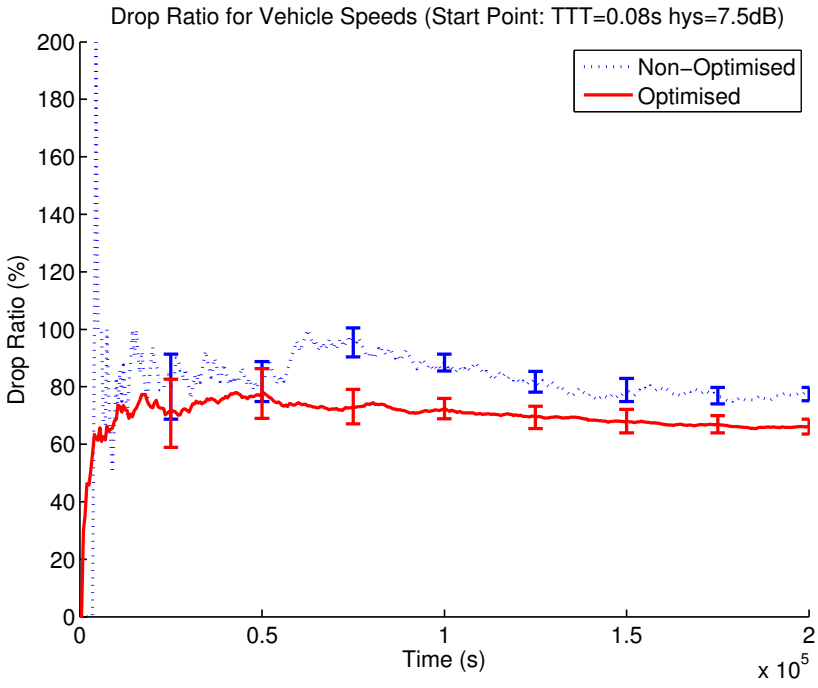
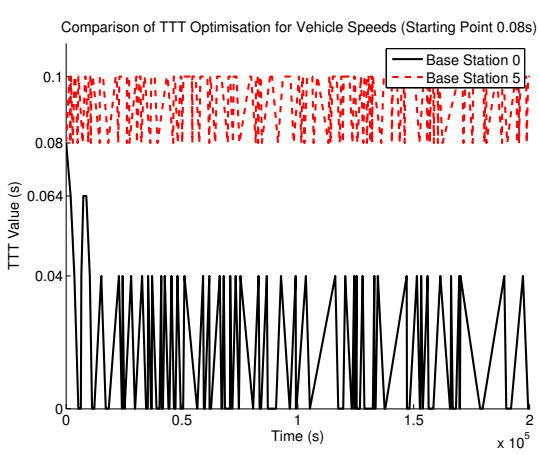


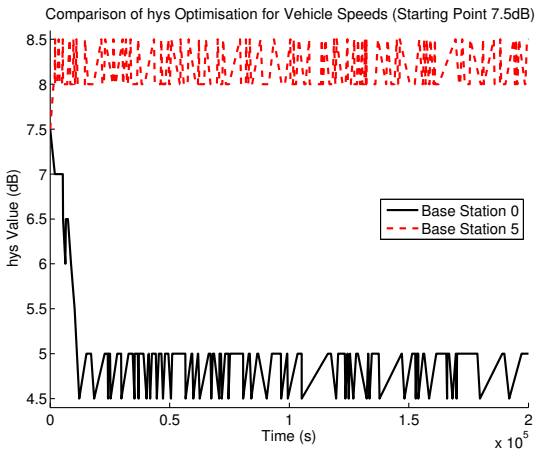
Figure 5.15: Graph of Optimised vs. Non-Optimised Results for Starting Point TTT=0.08s hys=7.5dB when UE traveling at vehicle speeds.

A comparison of how base stations 0 and 5 attempted to optimise their TTT and hys values can be seen in Figure 5.16. It can be seen that base stations 0 and 5 both took different approaches to trying to optimise values of TTT and hys. Base station 0 reduced both its values of TTT and hys end up oscillating between $0s$ and $0.04s$ for TTT and $4.5dB$ and $5dB$ for hys. The oscillations were still fairly frequent meaning that there could have been more of an improvement.

Unlike base station 0, base station 5 actually increased both its values of TTT and hys, which was highly unexpected. The base station ended up oscillating between $0.08s$ and $0.1s$ for TTT and $8dB$ and $8.5dB$ for hys. Due to the speeds that the UEs are moving at it could be said that these changes are wrong because due to the speed of the UEs the handovers need to be triggered quickly and increasing the values from their starting points in this scenario would not do that.



(a) Changing TTT Values



(b) Changing hys Values

Figure 5.16: Illustration of how the TTT and hys values changed over time for medium values when UE traveling at vehicle speeds.

Much like with the other scenarios it is seen that while the system may have been an improvement over the static values the system still could have performed far better, as base stations appeared to keep switching between values that could be considered non-optimal. All the graphs for the optimisation of the TTT and hys values for this scenario can be found in Appendix C.2.4.

5.4 Evaluation

Over the four scenarios used to test the optimisation system for walking speeds it can be seen that overall the system performed better than if no optimisation was done at all. It can also be seen that the system makes improvements to the performance very quickly. As seen in scenarios for the middle values and the large hys with small TTT the system managed to not have the large spike in dropped calls at the beginning of their simulation runs which were seen in there respective non-optimised runs.

In the scenarios for vehicle speeds it was seen that the system did performed better than if no optimisation was used. However, it was seen in the ways that the TTT and hys values were changed that they did not stop dropped calls constantly happening, which means that these values were likely non-optimal. Therefore it is possible that the system could have performed a lot better if the system did not get stuck between non-optimal states.

It can, however, be seen that when the system does not get stuck between non-optimal states that it will optimise the TTT and hys values as quickly as it is needed, i.e., whenever a dropped call or ping-pong occurs.

The optimisation system, however, also appears to have some drawbacks. It

was seen in the first scenario that the optimisation system caused a very large increase to the dropped call ratio before improving it. This is a usual downside in optimisation processes where 'things have to get worse before they can get better' and this process is a part of Q-Learning where the possible future rewards are taken into account when selecting a new state to move to. It was also seen in the scenarios using the vehicle speeds that the system appeared to keep getting stuck between two states that appear to be non-optimal which degrading the performance which had the potential to do a lot better.

Chapter 6

Future Work

So while the optimisation system created was able to make vast improvements to the performance of the simulated network, there was still the potential for the system to perform even better. Such ways that could possibly be explored to improve the system could be to add more complexity to the simulation and simulated network, as well as improving the machine learning algorithm used as a whole.

6.1 Slow Fading

An important expansion to this project would be the addition of Slow Fading to the Path Loss equations. This addition would simulate obstacles in the simulation environment such as wall and vehicles. This would add more randomness into the simulation and add the possibility of sudden drops in the signal strength resulting in dropped calls or sudden improvements triggering a handover only for the signal strength to drop again resulting in a handover ping-pong. This would be a good addition to the project because it would make the simulation more realistic.

6.2 Limited Resources Within the Network

A possible expansion on this project would be to have limited resources within the simulated network. This would mean that instead of the base stations having the resources to accommodate all the UE's within the simulation they would only have the resources for a limited number of UE's. This would create a new dynamic within the simulation where handovers could be refused and this in turn could increase the number of dropped calls. This could, therefore, mean that the Q-Learning agents for the base stations reduce the values of TTT and hys so that it takes less time to trigger a handover. This means that UE's connected to a

base station will handover to another base station faster freeing up the resources for another UE.

6.3 Improved Algorithm

Another important expansion of the project would be to improve the machine learning algorithm. It was seen that while the Q-Learning algorithm used did produce improved performance compared to if no machine learning was used; there was still a lot of improvement that could have taken place as the algorithm kept oscillating between what could be assumed to be non-optimal values as the oscillations were very frequent. Such improvements that could be made to the algorithm would be involve finding a way to 'break' out of the states that produced the oscillations. Other improvements would be providing mechanisms to ensure that all possible actions from every state were explored during the process of learning the environment. The algorithm currently learns the environment randomly which means that, while a lot of time was given to learning the environment, there is no insurance that all, or a large majority, of the states and actions were actually explored.

Chapter 7

Conclusions

Bibliography

[1] A. Ericsson, "Ericsson mobility report," tech. rep., tech. rep., Ericsson AB, February 2014.

[2] C. Cox, *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications*. John Wiley & Sons, 2012.

[3] 3GPP TS 23.002 V11.6.0, *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Network architecture (Release 11)*, June 2013.

[4] S. Feng and E. Seidel, "Self-organizing networks (son) in 3gpp long term evolution," *Nomor Research GmbH, Munich, Germany*, 2008.

[5] 3GPP TR 36.902 V9.3.1 Release 9, *LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions (Release 9)*, May 2011.

[6] H. Holma and A. Toskala, *LTE for UMTS-OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009.

[7] T. Jansen, I. Balan, J. Turk, I. Moerman, and T. Kurner, "Handover parameter optimization in lte self-organizing networks," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pp. 1–5, IEEE, 2010.

[8] 3GPP TS 36.331 V10.7.0, *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 10)*, November 2012.

[9] N. Sinclair, *Handover Optimisation using Neural Networks within LTE*. PhD thesis, University of Strathclyde, 2013.

[10] E. Alpaydin, *Introduction to Machine Learning*. MIT press, 2 ed., 2010.

[11] A. G. Barto and R. S. Sutton, *Reinforcement learning: An introduction*. MIT press, 1998.

[12] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 2010.

[13] R. Roy, *Handbook of Mobile Ad Hoc Networks for Mobility Models*. Springer, 2010.

[14] J. Chebil, A. K. Lwas, M. R. Islam, and A. Zyoud, "Comparison of empirical propagation path loss models for mobile communications in the suburban area of kuala lumpur," in *Mechatronics (ICOM), 2011 4th International Conference On*, pp. 1–5, IEEE, 2011.

[15] N. Shabbir, M. T. Sadiq, H. Kashif, and R. Ullah, "Comparison of radio propagation models for long term evolution (lte) network.," *International Journal of Next-Generation Networks*, vol. 3, no. 3, 2011.

[16] N. Sinclair, D. Harle, I. Glover, J. Irvine, and R. Atkinson, "An advanced som algorithm applied to handover management within lte," 2013.

Appendices

Appendix A

Code Listings

Appendix B

Simulation Testing

To be added.

Appendix C

Handover Optimisation Graphs

C.1 Walking Speeds

C.1.1 Large Starting Values

C.1.2 Middle Starting Values

C.1.3 Small Starting Values

C.1.4 Large hys and Small TTT Starting Values

C.2 Walking Speeds

C.2.1 Large Starting Values

C.2.2 Middle Starting Values

C.2.3 Small Starting Values

C.2.4 Large hys and Small TTT Starting Values