Handover Optimisation in 4G Systems

Iain Cuthbertson

201015895

Computer & Electronic Systems

University of Strathclyde

April 2014

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

3G	Third Generation
4G	Fourth Generation
AI	Artificial Intelligence
DL	Downlink
eNodeB	Evolved Node B
hys	Hysteresis
LTE	Long Term Evolution
MME	Mobility Management Entity
PCell	Primary Cell
QoS	Quality-of-Service
SCell	Secondary Cell
SON	Self-Organising Network
TTT	Time-to-Trigger
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Related Work

Chapter 3

LTE

Mobile communications is on to its Fourth Generation (4G) of network infrastructure with Long Term Evolution (LTE). This network infrastructure is an improvement upon Universal Mobile Telecommunications System (UMTS), which is a Third Generation (3G) network. LTE has Downlink (DL) speeds of up to 300 Mbit/s and Uplink (UL) speeds of 75 Mbit/s. This development was driven by the users want of faster download speeds for mobile services such as Video Streaming.

3.1 Self-Organising Network

Self-Organising Network (SON) [2]

3.2 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 User Equipment (UE) from one base station to another. This process is used to provide more 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. To keep with the QoS it is important that handovers are done fast, have little-to-no disruption to the users 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 Evolved Node B (eNodeB) in LTE, or the received signal strength to the UE drops below a certain threshold needed to maintain the call. Handovers are stated to take roughly 0.25 seconds to complete after the decision has been made for a handover to take place [3].

eNodeB Mobility Management Entity (MME)

3.2.1 Parameters

Time-to-Trigger (TTT) Hysteresis (hys)

Parameter	Value(dB)
hys	0.0
	0.5
	1.0
	1.5
	2.0
	2.5
	3.0
	3.5
	4.0
	4.5
	5.0
	5.5
	6.0
	6.5
	7.0
	7.5
	8.0
	8.5
	9.0
	9.5
	10.0

Table 3.1: Table of the different LTE hys values.

Parameter	Value(s)
TTT	0.0
	0.04
	0.064
	0.08
	0.1
	0.128
	0.16
	0.256
	0.32
	0.48
	0.512
	0.64
	1.024
	1.280
	2.56
	5.12

Table 3.2: Table of the different LTE TTT values.

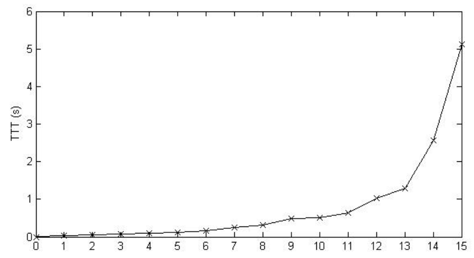


Figure 3.1: Graph of TTT values.

3.2.2 Procedure

Event Type	Trigger Criteria
A1	Serving becomes better than a threshold.
A2	Serving becomes worse than a threshold.
A3	Neighbour becomes offset better than 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 SCell.
B1	Inter RAT neighbour becomes better than threshold.
B2	PCell becomes worse than threshold1 and inter RAT neighbour becomes better than threshold2.

Table 3.3: Table of the different LTE Trigger types and their criteria.

Primary Cell (PCell) Secondary Cell (SCell)

[3] [4]

Chapter 4

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 users 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. [5] 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 algorithm would be given inputs and any patterns that are contained would become more and more common 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.

4.1 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 a lot of future reward against a state with a lot of future reward but not a lot of immediate reward. This is trade-off is similar to that of exploration versus exploit 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. [6]

4.2 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 from within an environment. What makes Q-Learning so powerful is that it will always learn the optimal policy for a problem regardless of the policy (which action a to take from a state s) it follows while learning 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.

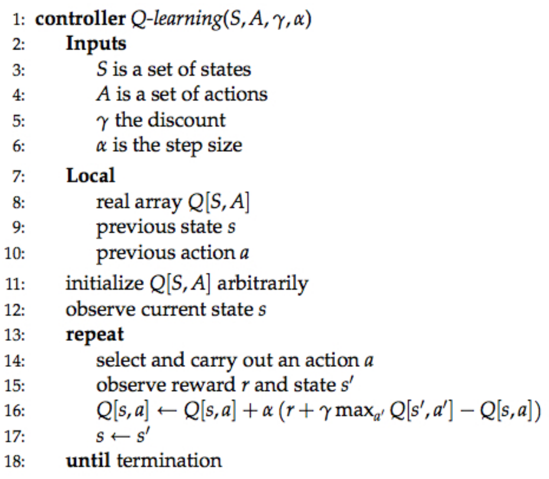
$$< s, a, r, s' >$$

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 Figure 4.1. 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 give a reasonable estimation of the actual value of state-action pair. The Q-Values are defined by Equation 4.1 where α is the step size which specifies how much the new Q-Value is averaged with the old one, γ is the discount factor which specifies how much the agent considers the possible future rewards it will gain and the possible future rewards ($max_{a'}Q(s', a')$) is the maximum of the Q-Values of all possible state-actions pairs from the action selected.

$$Q[s, a] = Q[s, a] + \alpha(r + \gamma max_{a'}Q[s', a'] - Q[s, a]) \tag{4.1}$$

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 4.1. 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.



```
controller Q-learning($S, A, \gamma, \alpha$)
    Inputs
        $S$ is a set of states
        $A$ is a set of actions
        $\gamma$ the discount
        $\alpha$ is the step size
    Local
        real array $Q[S, A]$
        previous state $s$
        previous action $a$
    initialize $Q[S, A]$ arbitrarily
    observe current state $s$
    repeat
        select and carry out an action $a$
        observe reward $r$ and state $s'$
        $Q[s, a] \leftarrow Q[s, a] + \alpha(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
        $s \leftarrow s'$
    until termination
```

Figure 4.1: Image of Q-Learning process [1].

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. [1]

4.2.1 Example

Chapter 5

Simulation Design

5.1 Simulation Parameters

5.1.1 Propagation Model

5.1.1.1 Okumura-Hata Model

$$L_u = 69.55 + 26.16 log_{10} f - 13.82 log_{10} h_B - C_h + [44.9 - 6.55 log_{10} h_B] log_{10} d \quad (5.1)$$

$$C_H = 0.8 + (1.1 log_{10} f - 0.7) h_m - 1.56 log_{10} f \quad (5.2)$$

$$C_H = \begin{cases} 8.29(log_{10}(1.54 h_M))^2 - 1.1 & \text{if } 150 \leq f \leq 200 \\ 3.2(log_{10}(11.75 h_M))^2 - 4.97 & \text{if } 200 < f \leq 1500 \end{cases} \quad (5.3)$$

5.1.1.2 Egli Model

$$P_{R50} = 0.668 G_B G_M [\frac{h_B h_M}{d^2}]^2 [\frac{40}{f}]^2 P_T \quad (5.4)$$

5.1.1.3 Cost231-Hata Model

$$L = 46.3 + 33.9 log f - 13.82 log h_B - a(h_R) + [44.9 - 6.55 log h_B] log d + C \quad (5.5)$$

$$a(h_R) = (1.1 log f - 0.7)h_R - (1.58 log f - 0.8) \tag{5.6}$$

$$C = \begin{cases} 0dB & \text{for medium cities and suburban areas} \\ 3dB & \text{for metropolitan areas} \end{cases} \tag{5.7}$$

5.1.2 Mobility Model

5.1.2.1 Random Direction

5.1.2.2 Random Waypoint

5.2 Simulation Testing

Chapter 6

Handover Parameter Optimisation

6.1 Approach

The approach taken for optimising the handover parameters in LTE uses a Q-Learning algorithm based on the process given in Section 4.2. In the approach the model of the environment has a state for every combination of TTT and hys; giving a total number of states of 336. 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 increase of TTT.
2. A single increase of hys.
3. A single increase of both TTT and hys.
4. A single decrease of TTT.
5. A single decrease of hys.
6. A single decrease of both TTT and hys.

Having the actions only change the parameters by one value each time not only allows for 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-pongs accumulated in the simulation for current state in the environment model. 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 selects a new action 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 is terminated and a policy is generated. 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.

6.2 Results

Chapter 7

Future Work

Chapter 8

Conclusions

Bibliography

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

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

[3] 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.

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

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

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

Appendices

Appendix A

System Design and Evaluation

The contents...