Physical Cell ID Allocation in Cellular Networks

Sofia Nyberg

Supervisor: Kaj Holmberg
Examiner: Niklas Carlsson
Upphovsrätt


Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/her own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Sofia Nyberg
Abstract

In LTE networks, there are several properties that need to be carefully planned for the network to be well performing. As the networks’ traffic increases and the networks are getting denser, this planning gets even more important. The Physical Cell Id (PCI) is the identifier of a network cell in the physical layer. This property is limited to 504 values, and therefore needs to be reused in the network. If the PCI assignment is poorly planned, the risk for network conflicts is high.

In this work, the aim is to develop a distributed approach where the planning is performed by the cells involved in the specific conflict. Initially, the PCI allocation problem is formulated mathematically and is proven to be NP-complete by a reduction to the vertex colouring problem. Two optimisation models are developed which are minimising the number of PCI changes and the number of PCIs used within the network respectively.

An approach is developed which enlarges the traditional decision basis for a distributed approach by letting the confused cell request neighbour information from its confusion-causing neighbours. The approach is complemented with several decision rules for the confused cell to be able to make an as good decision as possible and by that mimic the behaviour of the optimisation models. Three different algorithms are implemented using the approach as a basis. For evaluation purpose, two additional algorithms are implemented, one which is applicable to today’s standard and one inspired by the work by Amirijoo et al.

The algorithms were tested against three different test scenarios where the PCI range was narrowed, the number of cells was increased and the network was extended. The algorithms were also tested on a small world network. The testing showed promising results for the approach, especially for larger and denser networks.
Acknowledgments

This report is a master thesis at the program Master of Science in Information Technology at the Institute of Technology at Linköping University. The study was conducted at Ericsson in Mjärdevi, Linköping.

I would like to thank Ericsson and all people involved for giving me the opportunity and support to investigate this interesting subject. A special thanks to my supervisor Tobias Ahlström and Line Manager Ove Linnell. Also, I would like to thank my supervisor Kaj Holmberg at the Department of Mathematics (MAI) and my examiner Niklas Carlsson at the Department of Computer and Information Science (IDA) for their contribution in time and support.
# Contents

Abstract iii  
Acknowledgments iv  
Contents v  
List of Figures viii  
List of Tables x  

1 Introduction 1  
1.1 Motivation ............................................. 3  
1.2 Aim .................................................. 3  
1.3 Research Questions .................................... 4  
1.4 Limitations ............................................. 4  
1.5 Terminology .......................................... 5  

2 Background 6  
2.1 Long Term Evolution ................................. 6  
2.1.1 E-UTRAN ........................................... 6  
2.2 Self-organising Networks ............................. 8  
2.2.1 Automatic Neighbour Relation .................... 9  
2.3 Physical Cell Identity ................................ 11  
2.3.1 The PCI Construction ............................... 12  
2.3.2 The PCI Management ............................... 12  

3 Graph-based Optimisation Formulation 15  
3.1 Graph Theory .......................................... 15  
3.2 Network Graph Representation ....................... 16  
3.2.1 Network Matrix Representation ................... 16  
3.3 Problem Definitions ................................... 17  
3.4 Network Optimisation ................................ 17  
3.4.1 NP-completeness ................................ 18  
3.4.2 The Vertex Colouring Problem .................... 18  
3.4.3 Problem Reduction ................................ 19  
3.5 Optimisation Models .................................. 20  
3.5.1 Minimum Number of PCI Changes ................ 20  
3.5.2 Minimum Number of Used PCIs ................... 21  
3.5.3 Optimisation Model with Multiple Goals ........... 21  
3.5.4 Execution Limitations ............................. 21  

4 Allocation Algorithms 23  
4.1 Related Approaches .................................. 23
# List of Figures

1.1 Illustration of a collision. .................................................. 2
1.2 Illustration of a confusion. .................................................. 2
1.3 Terminology clarification. ................................................... 5

2.1 E-UTRAN structure. ............................................................ 7
2.2 Illustration of the overlapping area where the handover from one cell to another is performed. ................................. 10
2.3 ECGI construction. .............................................................. 12
2.4 Range separation schemes. .................................................... 13

3.1 Simple graph. ........................................................................ 15
3.2 (a) Graph representation of a small network. (b) Modified graph representation. ........................................ 16
3.3 Problem classes. ................................................................... 18
3.4 Graph colouring example. ...................................................... 19

4.1 Decision basis for (a) what the standard supports, (b) the approach suggested in Amirijoo et al. and (c) the approach suggested in this work. ........................................ 26
4.2 Illustration of the second decision rule. .............................. 27
4.3 Illustration of the fourth decision rule. .............................. 27

5.1 An example picture of a very small, easily overlooked network. .................................................. 33
5.2 An example picture of a larger, more realistic network. .................................................. 34
5.3 Illustrative example of a small world construction. ................. 36

6.1 Test Phase I - Results obtained from the scenario PCI Restriction. .................................................. 39
6.2 Test Phase I - The neighbour distribution for each test case in the scenario Neighbour Magnitude. .................................................. 40
6.3 Test Phase I - Results obtained from the scenario Neighbour Magnitude. .................................................. 41
6.4 Test Phase I - Results obtained from the scenario Network Extension, Test Case IIIa, with 2.5 % network extension. .................................................. 42
6.5 Test Phase I - Results obtained from the scenario Network Extension, Test Case IIIb, with 5 % network extension. .................................................. 43
6.6 Test Phase I - Results obtained from the scenario Network Extension, Test Case IIIc, with 10 % network extension. .................................................. 44
6.7 Test Phase I - Results obtained from the scenario Network Extension, Test Case IIId, with 20 % network extension. .................................................. 45
6.8 Test Phase II - The neighbour distribution for each test case in the scenario Neighbour Magnitude. .................................................. 46
6.9 Test Phase II - Results obtained from the scenario Neighbour Magnitude. .................................................. 46
6.10 Test Phase II - Results obtained from the scenario Network Extension. .................................................. 47
6.11 Test Phase III - Small world networks using $\beta = 0$, $\beta = 0.15$, $\beta = 0.5$ and $\beta = 1$. .................................................. 48
6.12 Test Phase III - Result of a varied $\beta$. ........................................ 49
7.1 Distribution of (a) neighbours and (b) neighbours’ neighbours for $N = 500$, $K = 25$ and $\beta = 0.15$. ................................................................. 53
7.2 Network constructed of different sized disks. ..................................................... 59
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Neighbour relation table example.</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Mapping between the PCI allocation problem and k-colouring.</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Test phase description.</td>
<td>35</td>
</tr>
<tr>
<td>6.1</td>
<td>Test Phase I - Test details for the test scenario PCI Restriction.</td>
<td>38</td>
</tr>
<tr>
<td>6.2</td>
<td>Test Phase I - Test details for the test scenario Neighbour Magnitude.</td>
<td>39</td>
</tr>
<tr>
<td>6.3</td>
<td>Test Phase I - Test details for the test scenario Network Extension.</td>
<td>41</td>
</tr>
<tr>
<td>6.4</td>
<td>Test Phase I - Relations and neighbours from the test scenario Network Extension.</td>
<td>42</td>
</tr>
<tr>
<td>6.5</td>
<td>Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.a.</td>
<td>43</td>
</tr>
<tr>
<td>6.6</td>
<td>Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.b.</td>
<td>43</td>
</tr>
<tr>
<td>6.7</td>
<td>Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.c.</td>
<td>44</td>
</tr>
<tr>
<td>6.8</td>
<td>Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.d.</td>
<td>45</td>
</tr>
<tr>
<td>6.9</td>
<td>Test Phase II - The number of confused nodes from the test scenario Network Extension.</td>
<td>47</td>
</tr>
<tr>
<td>6.10</td>
<td>Test Phase III - Test results when using ten different small world networks with $\beta = 0.15$.</td>
<td>49</td>
</tr>
</tbody>
</table>
1 Introduction

The global mobile usage is growing with around 5% every year [1]. In the first quarter of 2015, the number of Long Term Evolution (LTE) subscriptions reached around 600 million. By the end of 2020, the same year that 5G aim to be launched, the number of LTE subscriptions is forecasted to 3.7 billion. In June 2015, 300 different suppliers had together launched around 3,000 LTE user device models and approximately half of these were launched in the last twelve months [1].

The steady increase in mobile usage requires higher network capacity which implies denser networks. For these denser networks to work properly, the network structure needs to be well-planned for each unit to have access to a fair share of the available network capacity. One of the planning problems is the assignment of cell properties. These properties are in some cases limited and therefore need to be reused.

A Physical Cell Id (PCI) is the identifier of a cell in the physical layer of the LTE network, which is used for separation of different transmitters. Due to the construction of PCIs, the number of PCIs are limited to 504. Because of the large amount of cells in the LTE network, the PCIs needs to be reused and several cells will share the same PCI. There are two undesired events that may occur if this property is poorly allocated: collision and confusion.

For an allocation to be collision-free, there should not be any two neighbouring cells at the same frequency sharing the same id [2]. If a User Equipment (UE) is to be handed over from one cell to another, and the source and target cell is sharing the same id, there is no unambiguous way to notify the UE to which cell it should be handed over to. The UE could interpret the command as if it should stay connected to the service cell. This would eventually lead to a service interruption for the UE, as it would lose the connection with the source cell while entering the target cell. A collision is illustrated in Figure 1.1, where two neighbouring cells are sharing the same PCI.
In a confusion-free allocation there is no cell that has two neighbouring cells sharing the same id within the same frequency. When confusion occurs, the serving cell of a UE cannot command the UE to which of its neighbours to be handed over in an effective way [2]. This scenario could eventually also lead to a service interruption for the UE, if the UE is incorrectly handed over to the wrong cell. Figure 1.2 illustrates a confusion where the cell associated with PCI B has two neighbors both using PCI A.

Because of the large amount of cells and the limited amount of PCIs, the undesired events of collision and confusion is hard to eliminate. To establish a conflict free allocation, no cell within the entire network may have the same PCI as one of its neighbours (to avoid collision) or neighbour’s neighbours (to avoid confusion). Due to this property, the number of possible PCIs for each cell is narrowed by the amount of neighbours and neighbours’ neighbours, which consequently increases as the networks expand. The dynamism of the LTE networks also induce new conflicts as the signal strength of the eNB (evolved node B) varies, and new cells are regularly deployed.

By assigning the PCIs in a proper way, the event of a conflict may be prevented or at least handled in the best possible way. The assignment includes selecting PCIs to multiple cells in an entirely new network, but also to select a PCI for a few, newly deployed cells or reassign the PCI to a conflicting cell in an already established network. Today, the PCI allocation is traditionally handled in a centralised fashion [3], where a centralised authority is responsible for solving the conflicts. An alternative and desirable method is a distributed solution where the involved cells are responsible of handling the PCI resolution when a conflict occurs.
1.1 Motivation

It is in theory possible to obtain an optimal solution to the allocation problem using a centralised solution to allocate the PCIs. However, to obtain this optimal solution, the centralised authority (a centralised node or Operations Support System, OSS) needs to obtain a large amount of information from all of the nodes within the network. In excess of this, the PCI allocation problem is NP-complete and to obtain this optimal solution could be really time consuming if the network is large. When an optimal solution is obtained, this solution could imply that several other cells than the cell currently in conflict needs to change their PCI as well. In this situation the decision to be considered is if these cells should reboot and change their PCIs (and by that causing a service interruption within these cells) or if a non-optimal solution should be chosen to avoid this.

Distributed approaches for the PCI assignment problem are both important and relatively unexplored, making the design of such algorithms an interesting topic to look closer into. Using a decentralised solution, the PCI allocation might not end up as an optimal solution to the problem since an optimal solution for a small part of the network is not necessarily optimal for the entire network. Moreover, it is hard to know if the solution obtained is optimal for the entire network without looking at the allocation from a higher perspective. However, a distributed solution is beneficial in terms of speed and information gain. In a distributed solution, a smaller part of the network is considered and therefore even an NP-complete problem can be evaluated relatively fast. It is also possible to use greedy algorithms since the goal at this point is to obtain a solution as good as possible, not necessarily optimal. In terms of information gain, the nodes themselves do already contain a great part of the information needed to calculate a good solution, and therefore the signalling delay is smaller for a distributed solution compared to a centralised approach.

1.2 Aim

The aim of this work is to investigate the possibility of a distributed solution for the PCI assignment problem, were the cells or the eNBs are making the best possible decision regarding which PCI it should use. By looking at how the LTE network is constructed, what is realisable relative to the prescribed LTE standard and how other researchers have approached the problem, a graph theory based mathematical formulation of the problem is constructed. This model is reduced to an existing optimisation problem and two new optimisation models are formulated. The optimisation models aim to address two of the major problems with the PCI allocation: the number of PCI changes that needs to be performed for conflict resolution and the lack of possible PCIs to choose from. As the problem is NP-complete, a distributed approach is developed where the goal is to obtain a result close to optimum by mimicking the behaviour of the developed optimisation models. Three resulting algorithms of the approach are implemented in MATLAB together with two comparison algorithms to be able to evaluate the result. The five algorithms are run within a simulation environment, also implemented using MATLAB, were several realistic scenarios as well as small world networks are evaluated.
1.3 Research Questions

The following research questions aims to be answered within this study:

1. How has the PCI allocation problem been addressed in related studies?

2. What possibilities and limitations does the network standard induce in terms of a distributed approach?

3. What performance improvements can be achieved if relaxing communication limitations and constraints regarding which cells are allowed to resolve conflicts, imposed by current network standards?

1.4 Limitations

The PCI planning process consists of both the planning of an entirely new network where all cells needs to be assigned PCIs as well as planning for the PCI assignment of one or a few new cells in an already existing network. The planning may also include rebooting an already established cell because of the occurrence of collisions and/or confusions due to a badly planned network. Since the techniques are quite different when establishing a whole network compared to one or a few cells, this work will only focus on the latter. Conflicts already present within a network will also be considered.

This work will focus on how to assign the PCIs properly in binary terms, i.e. the work will omit the extremely dense scenarios were different traffic parameters needs to be considered to make proper decisions. Consequently, these denser networks require different techniques to be solved, and access to traffic parameters which are often confidential material that the operators are not willing to share. Because of this, the networks evaluated will have a feasible solution, as defined in Section 3.3.

When solving the networks, only confusions will be considered. Since two adjacent neighbours in a dense network often share at least one neighbour, a collision between these to cells will also cause confusion for their common neighbour. Because of this, solving the confusions will solve the majority of the conflicts in the network.

The testing of the algorithms will be performed in constructed, randomised networks. Since the networks used in reality is connected to different vendors, and to be able to construct these networks one need to take part of sensitive and/or confidential information, these networks will not be part of this work.

Because of the lack of time and computer capacity, no optimal solutions will be calculated using the developed optimisation models.
1.5 Terminology

This thesis distinguishes between the cells *experiencing* a confusion and the cells *causing* a confusion. In Figure 1.3, Cell 2 is experiencing the confusion, and will further be referred to as the *confused cell*. Cell 1 and 3 causes the confusion, and are further referred to as the *confusion-causing cells*. Moreover, Cell 2 is a neighbour to Cell 1 and Cell 3 is a neighbour’s neighbour to Cell 1.

![Figure 1.3: Terminology clarification.](image)

 PCI: A
 PCI: B
2 Background

To be able to fully understand how the Physical Cell Ids are constructed and how the allocation process is handled today, some background information regarding the LTE network is needed. The first part of this chapter will focus on clarifying some terminology and concepts important for this matter. The chapter continues with a description of the PCI standard used today and what possibilities that the standard supports.

2.1 Long Term Evolution

LTE is based on OFDM (Orthogonal Frequency Division Multiplexing) for downlink and Single Carrier FDMA (Frequency Division Multiple Access) for uplink [4]. The frequency band spans from 2-8 GHz and has a potentially estimated data throughput range of 100-300 Mbps. The network is designed by national and regional communication standards known as 3GPP or Third Generation Partnership Project and LTE has been part of their standard since release 8 in 2008.

2.1.1 E-UTRAN

E-UTRAN is the air interface of the LTE-network, which include communication between the UEs and the base stations (in LTE called eNBs) and between the base stations and Evolved Packet Core (EPC). The E-UTRAN structure is illustrated in Figure 2.1. The eNBs are connected to one or multiple Mobility Management Entities (MMEs) and Serving Gateways (S-GWs) [5]. The MMEs function include managing and storing UEs context, for example user security parameters. It also authenticates the user. The S-GWs functions as an anchor between LTE and other standards, i.e. between LTE and 3G or GSM. The routing and forwarding of user data packages is also handled by the S-GW. The communication between the eNB and the MMEs and/or S-GWs (i.e. the communication between E-UTRAN and EPC) is done through the S1 interface, which consists of a user plane and a control plane.
The communication between the eNBs are executed through the X2 interface [5] which is established through a X2 Setup [6]. The X2 Setup is performed through the MME given that the eNB is aware of the Cell Global Identity (CGI or ECGI, as described in Section 2.3) of a neighbouring cell. If the eNBs do not share the same MME, the Tracking Area Code (TAC) is also required. The eNB sends a request to MME asking for the IP address used for X2 communication by the eNB connected to the target cell (the cell associated with the CGI). MME sends the request onwards to the other eNB through S1 and returns the response to the first eNB, and the connection is established [6]. Once established, the eNBs can take part of the neighbour information stored by the respective eNB. The eNB can establish several X2 connections, one between every neighbouring eNB. It is also possible to have no X2 interface connections established at all, however, the establishment of a X2 connection streamline the communication between the eNBs.

**Evolved Node B**

An LTE base station is called evolved Node B (E-UTRAN Node B or eNB) [7]. The eNBs may be manufactured by different vendors and could be shared by several operators. There are several types of eNBs: Macro, Micro and Pico eNBs as well as HeNBs (Home eNBs) [8]. Different eNBs have different range, i.e. a limited size in which a receiver can successfully hear the transmitter. They also have different capacity in terms of the total amount of data rate that all UEs within a cell can transmit [6].

The Macro eNBs serves wide areas and are deployed by mobile network operators. The cells of a Macro eNB can be used by all subscribers of the operators associated with it. A Macro eNB typically has three to six cells which has a range of a few kilometers. The Micro eNB is smaller, often has one cell and has a range of a few hundred meters. These cells are typically used in denser urban areas due to the greater collective capacity. A Pico eNB has a range of tens of meters and are often used in large indoor environments, such as shopping centers.
centres or offices. The HeNB which is a LTE femto cell [9] uses low output power and are used for home installation. The range for a HeNB is a couple of meters. The access policy of the different cell types is either closed or open. A closed cell is mainly used by residential users and the cell is in this case defined as a Closed Subscriber Group (CSG) cell where the access control is located in the gateway. If open, all users are allowed access to the cell.

Since today’s networks are dense and rely on a heterogeneous network structure, the different cells are often overlapping, both at different frequency level and within the same frequency. Two adjacent cells sharing the same frequency is said to be intra-frequency neighbours and two adjacent cells at different frequency level are inter-frequency neighbours. If for example a macro cell gets overloaded, the overlapping Micro or Pico cells can be used for offload or to increase the capacity [8].

2.2 Self-organising Networks

The LTE network is controlled by a network management system as in common with other telecommunication technologies. However, the LTE network also contain techniques to reduce the manual intervention, known as Self-organising Networks (SON) [6]. SON is a network that is able to manage itself and by that reduce the manual workload. By reducing the manual workload, the operational cost will be lowered and the human errors will be minimised [4]. SON are often divided into three categories: self-configuration, self-optimisation and self-healing, and the architecture can either be distributed, centralised or a hybrid of them both [10]. Using a distributed SON approach, only high level parameters, for example policies, are handled by the management system. The autonomous calculations and decisions are in this approach performed by the nodes. This minimises the communication between the nodes and the management system. In a centralised approach, the management system executes all the calculations and makes the decisions, and the nodes only perform the necessary parameter changes. For this approach, the communication between the management system and the nodes are more intense. This will imply that delays because of the external communication will occur, as the decisions cannot be made instantly. In the hybrid version, there are naturally calculations and decisions performed at both the management system and the nodes.

3GPP defines the self-configuration process as [3]:
"...the process where newly deployed nodes are configured by automatic installation procedures to get the necessary basic configuration for system operation."

The self-configuration process of the eNB is performed in pre-operational state, which means that the process is carried out when an eNB is newly installed and not yet in use. The process includes for example configuration of Physical Cell Ids for the eNB cells and transmission frequency [10]. When the RF transmitter gets switched on, the self-configuration process should have terminated.

The self-optimisation process is defined by 3GPP as [3]:
"...the process where UE and eNB measurements and performance measurements are used to auto-tune the network."

Self-optimisation is a constant ongoing process with the goal of increase network performance by optimising capacity, coverage, handovers and interference. Three examples of self-optimisation techniques are Mobile Load Balancing (MLB), Mobility Robustness Optimisation (MRO) and energy saving [6].
Mobility load balancing is the function of equating the burden of the cells when some cells are congested whilst others have spare resources. For the technique to work, the eNBs needs to share information between one another regarding their load level and available capacity [10]. The goal of mobility robustness optimisation is to ensure proper mobility. The technique gathers information about problems related to measurement reporting thresholds and is using this to correct errors in the configuration. The errors can for example remark as repeated too early or too late handovers. To save energy, the cells that are not in use can be switched off [6]. This technique is suitable in for example office environments where the installed pico cells are unused during the nights.

Self-healing is a process triggered by different self-healing functions [11]. Each function monitors a specific trigger condition called a Trigger Condition of Self-Healing (TCoSH). This trigger condition is either an alarm or the detection of a fault, and determines if an appropriate recovery action needs to be started or not. When recovery actions are triggered, the self-healing function also monitors the execution of the recovery process, and determines the next step after each iteration. Which recovery action to be taken depends on which type of fault that has occurred. For software faults, the action may for example be a reload of backed up software or to perform a reconfiguration. For hardware faults, the action depends on if there are redundant resources or not, and what type of redundant resources there are [11].

2.2.1 Automatic Neighbour Relation

Automatic Neighbour Relation (ANR) is a SON-function that automates the management of adjacent neighbours in the eNBs [9]. The ANR function increases the handover success rate and improve network performance by maintaining the efficiency of neighbour cell lists and neighbour relation tables [12]. This will unburden the operators from manually having to manage the neighbour relations of eNBs within the network. 3GPP defines a Neighbour cell Relation (NR) as follows [3]:

"An existing Neighbour Relation from a source cell to a target cell means that eNB controlling the source cell:

1. Knows the ECGI/CGI and PCI of the target cell
2. Has an entry in the Neighbour Relation Table for the source cell identifying the target cell
3. Has the attributes in this Neighbour Relation Table entry defined, either by O&M or set to default values"

Whilst an X2 relation is a symmetric relation between two eNBs, a neighbour cell relation is a cell-to-cell relation which is unidirectional. However, given that there is a neighbour cell relation from cell A to cell B, the possibility that there is a corresponding relation from cell B to cell A is quite high.

In Figure 2.2, a UE is heading from cell A to cell B. As the UE enters the overlapping area between the cells (in the figure marked in blue) the signal strength from cell B will at some point be higher than the signal strength from cell A, and a handover will be performed from cell A to cell B. In a reverse scenario, where the UE is heading from cell B to cell A, there will be a similar break point where the UE will handover from cell B to cell A. This indicates a relation from cell A to cell B and a relation from cell B to cell A. The special case where the UE’s only travel in one direction, and therefore implies a one way relation, may for example occur if a cell is allocated covering a unidirectional street.
2.2. Self-organising Networks

Figure 2.2: Illustration of the overlapping area where the handover from one cell to another is performed.

Neighbour Relation Table

As the neighbour information is constantly used, for example in handover situations, the information needs to be stored efficiently. Therefore, the neighbour relations managed by ANR are stored within a Neighbour Relation Table (NRT) [9]. Each cell in the network has its own NRT, and the information in the table is updated when changes in the surrounding cells are reported. The NRT consists of the following information for E-UTRAN:

<table>
<thead>
<tr>
<th>NR</th>
<th>TCI</th>
<th>No HO</th>
<th>No Remove</th>
<th>No X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TCI1</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>TCI2</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>TCI3</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>TCI4</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Neighbour relation table example.

- TCI - The Target Cell Identity which includes the ECGI/CGI and the PCI for the target cell.
- No HO - An indicator if the target cell can be used for handover purposes or not.
- No Remove - Indicates if the target cell can be removed from the NRT or not.
- No X2 - A field that only exists for LTE neighbours. This field is an indicator of the absence of the X2 interface between the eNBs of the source cell and the target cell.

If a neighbouring cell is in UTRAN, e.g. using 3G, the TCI field contain information of the Neighbour Cell Identity (NCI) which is a CGI including the PLMN-id, the Cell Identity (CI) and Radio Network Controller Identifier (RNC-Id). In case of a neighbouring cell is in GSM/GPRS/EDGE, the NCI is a CGI including PLMN-Id, CI, Location Area Code (LAC) and Base Station Identity Code (BSIC) [9].

Connected to the NRT, the ANR management contains a detection function and a removal function. These functions are implementation specific which means that different operators may handle detection and removal in a different manner. However, the purpose of the detection function is to find new neighbours and add these to the NRT and the purpose of
the removal function is to delete outdated neighbours from the NRT. These two functions together with attribute changes may also be performed by Operations and Maintenance (O&M). O&M is also informed about all changes that are made to the NRT.

NRT Management

For the ANR function to work, the cells in the network needs to broadcast their global identity, e.g. their CGI/ECGI. For an eNB to receive this information, the eNB needs to rely on UE measurements. Depending on if two cells are located within the same frequency or not, the performance of the ANR management is slightly different.

If the serving cell and the targeted cell are located at the same frequency level, an intra-frequency/intra-LTE ANR procedure is performed. Firstly, the UE sends a measurement report to the eNB containing information about the PCI of the targeted cell. If the PCI is not within the eNBs NRT, the eNB schedules enough idle periods to allow the UE to perform extended measurements with the goal of finding the ECGI of the target cell. When the ECGI is found, the UE reports the information to the serving cell which updates the NRT.

In case the serving cell and the target cell are not located at the same frequency, an inter-frequency/intra-LTE ANR procedure is performed. In this case, the eNB might first needs to schedule idle periods for the UE (depending on the type of UE) to perform neighbour cell measurements within the targeted frequency and the UE reports the PCI of the detected cells to the eNB. To be able to receive the ECGI the eNB schedules additionally idle periods for the UE to perform further measurements. When the ECGI is found, the UE reports this along with other parameters (for example TAC and PLMN) to the eNB, which updates the NRT.

For inter-RAT, when the communication is performed between LTE and 3G/GSM, the procedure is performed similarly to the inter-frequency/intra-LTE ANR. However, in the case of 3G for example, the scrambling code is found within the first measurement and the goal of the second idle scheduling is the CGI. As mentioned above, the X2 field within the NRT is excluded.

2.3 Physical Cell Identity

The Physical Cell Identity (PCI) is, as described in the introduction, the identifier of a cell within the physical layer of the LTE network. The PCI itself is not a unique identifier, because of the limited amount of ids available. To be able to identify a cell in an absolute unique manner, the E-UTRAN Cell Global Identifier (ECGI) needs to be measured [5]. The ECGI can be used to unambiguously identify any cell in any E-UTRAN in the world. The construction of the ECGI is shown in Figure 2.3. It is a combination of the Public Land Mobile Network Identity (PLMN-Id) and the Evolved Node B IDentity (ENBID). The PLMN-Id is in turn the combination of the Mobile Country Code (MMC) and the Mobile Network Code (MNC) [5]. The ENBID consists of the E-UTRAN Cell Identifier (ECI) and its length depends on the cell type. For HeNB cells, the ENBID length is 28 bits, and for other cells, the length is 20 bits.
2.3. Physical Cell Identity

Figure 2.3: ECGI construction.

The detection and decoding of the ECGI is however a much more complex and time consuming procedure than detecting the PCI. For an eNB to receive the ECGI of an unknown adjacent neighbour, the eNB needs to rely on UE measurements which implies idle scheduled time for the UE to perform the measurements as described in the previous section. Because of this, the network planning cannot be based on the ECGI and therefore needs to rely on the more easily accessible PCIs.

2.3.1 The PCI Construction

The PCI is calculated by adding two different down link synchronisation signals, the primary synchronisation signal (PSS) and the secondary synchronisation signal (SSS) [13]. The SSS (or PCI-group) consists of 168 sequence numbers: \( N^{(1)}_{ID} = [0, 167] \), and the PSS (or PCI-ID) consists of three different sequence numbers: \( N^{(2)}_{ID} = [0, 2] \). A PCI is defined as [12]:

\[
    N^{\text{cell}}_{ID} = 3 \times N^{(1)}_{ID} + N^{(2)}_{ID},
\]

which gives the maximum PCI value 503 when \( N^{(1)}_{ID} = 167 \) and \( N^{(2)}_{ID} = 2 \) and the minimum value 0 for \( N^{(1)}_{ID} = N^{(2)}_{ID} = 0 \).

Beyond the construction of the PCIs, the two signals are used when a UE is initially switched on within the network. Within every radio frame, the signals are transmitted twice (within sub frame 0 and 5) for the UE to establish down link synchronisation. The three sequence numbers of the PSS are mapped to three different roots of the Zadoff-Chu sequence, which is a frequency domain sequence of length 63. The Zadoff-Chu sequence is chosen for the purpose because of its good correlation and orthogonality properties, which makes robust PSS detection possible [2]. The SSS is based on maximum length sequence, called the m-sequence, which is a binary pseudo random sequence. For the generation of the synchronisation signal, three different m-sequences, ˜s, ˜c and ˜z, are used which each has a length of 31. These sequences can be generated by cycling through every possible state of a shift register [2].

2.3.2 The PCI Management

The 3GPP standard 36.300 specifies a framework for PCI selection [3]. The selection process can either be performed in a centralised or a distributed manner. For the centralised approach, a PCI is specified by O&M and the eNB shall use this value for the PCI change. In a distributed approach, O&M signals a list of possible PCI values. From this list, the eNB gets to choose one PCI at random or alternatively pick one based on some implementation specific rule, for example lowest CGI. In this approach, the eNB itself restrict this list by removing PCIs reported by UE measurements as neighbours, PCIs reported via the X2 interface as neighbours or PCIs that is for example reserved in the specific implementation used [3].
Apart from this standardised restrictions, the PCI management is implementation specific. This means that different operators are using their own method and algorithm for the purpose. Because of this, it is hard to get hold of an exact description regarding how the unravelling of networks in conflict is made. However, a general description can be formulated using the literature as a basis.

A PCI conflict can be detected in several different ways. Firstly, a PCI conflict can be discovered as a handover failure [14]. The conflict is in this case suspected after logging multiple handover failures correlated to a certain PCI. A conflict may also be detected as part of the ANR process, as mentioned in Section 2.2.1. If one node has multiple cells sharing the same PCI within its neighbour relation table, a confusion is detected. Since the probability that two cells in collision shares at least one neighbouring cell is quite high, there is a great chance that a collision within the network will be detected as a confusion in the ANR process of their common neighbour. The denser the network is and the more frequencies defined in the network, the higher the probability is that a collision is also a confusion. For sparser networks, an alternative method to detect collisions is to let the serving cell switch off its reference signal while the UEs within the cell scans for the same reference signal within the surrounding [14]. If the same reference signal is discovered during this time, the serving cell is part of a conflict. This method will however cause a transmission gap when the cells reference signal is switched off.

When solving a PCI conflict, there is a possibility to either solve the conflict automatically or manually. For the automatic option, a predefined implementation specific algorithm is run which solves the PCI conflicts. The conflict resolution is in this case performed when the specific network is experiencing low traffic, for example in the night hours. In a manual option, the operator is involved in the resolution process, and may for example influence the choice of the new PCI.

The PCI range may or may not be divided into smaller sub ranges using different allocation schemes [15]. In a layer independent scheme, the entire PCI range is used irrespective of which cell type the PCI allocation is planned for. In a range separation scheme, the PCI range is split in ordered, disjoint ranges where one range for example is associated with the macro cells and another with the pico cells. The PCI range may also be split into disordered ranges where the different cell types get allocated PCIs from the entire range, but from disjoint groups of PCIs. This is called a continuous separation with cross-layered coordination. The three different range separation schemes can be seen in Figure 2.4.

![Range separation schemes](image)

Another way to divide the available PCI range is to dedicate different sub ranges for different purposes, for example using the Nokia Siemens (NS) approach [16]. This approach specifies a reserved range of PCIs for newly established nodes to choose from. When the node has performed the ANR process and gained information about its surroundings, the node is rebooted and a new PCI is selected.
In a dense network where there are lots of neighbour relations, the scenario of no available PCIs may occur. In this case, the PCI selection may be performed randomly or using different network and traffic parameters. Parameters that can be used for this purpose may be for example distance, signal strength or the utilisation rate of the different PCIs within the network. By combining these parameters, the network can be weighted so that the best possible choice of PCI can be made. There will still be a conflict within the network, but it will cause as little problem as possible.
In this chapter, the graph-based optimisation formulation is presented. The chapter begins with some basic graph theory and a description of how the cell network can be represented on this form. The chapter continues by defining some necessary concepts useful when reducing the PCI allocation problem to the vertex colouring problem. The chapter terminates with optimisation model definitions.

3.1 Graph Theory

A graph is a set of vertices and edges. Mathematically, a graph $G$ is defined as $G = (V, E)$ where the elements of $V$ are the vertices (or nodes) and the elements of $E$ are the edges [17]. An element $e_{ij} \in E$ is equal to one if there exist an edge from the vertex $v_i$ to vertex $v_j$, where $v_i, v_j \in V$, and zero otherwise. Similarly, $e_{ji} = 1$ if there exist an edge from $v_j$ to $v_i$, and zero otherwise. In an undirected graph, the relation between $v_i$ and $v_j$ is symmetric, so that $e_{ij} = e_{ji}$. When an edge between $v_i$ and $v_j$ exists, i.e. when $e_{ij} = e_{ji} = 1$, the vertices $v_i$ and $v_j$ are adjacent to each other, and can be called neighbours [17]. The degree of a vertex, denoted $\text{deg}(v)$, is the number of edges incident on $v$. In Figure 3.1 a simple graph is illustrated. Here, the degree of vertex $v_i$ is equal to 4.

![Figure 3.1: Simple graph.](image)
3.2 Network Graph Representation

To represent a network as a graph is a classical way to mathematically formulate the network. For this work, the representation can be used to model the cells and neighbour relations.

Let \( G \) be a network containing cells which in turn each containing their own list of neighbours. Let the set of vertices \( V = \{v_1, v_2, ..., v_n\} \) represent the cells which are present in the network, and let the set of edges \( E \) correspond to the neighbour relations between the cells. If a cell \( i \) contain the cell \( j \) in its neighbour list, then \( e_{ij} = 1 \), and zero otherwise. For any cell \( i \) in the network, the number of neighbours is now equal to the degree of the vertex \( v_i \) in the graph. The resulting graph will be a simple graph, since no neighbour list of any cell will contain the cell itself. An example is shown in Figure 3.2 (a).

For the PCI allocation problem, the neighbours’ neighbours are also of interest. If cell \( i \) contains cell \( j \) in its neighbour list, and if cell \( j \) contains cell \( k \) in its neighbour list, then cell \( k \) is a neighbour’s neighbour to cell \( i \). In the graph representation, this would correspond to if there is an edge from \( v_i \) to \( v_j \), and an edge from \( v_j \) to \( v_k \), i.e. if \( e_{ij} = 1 \) and \( e_{jk} = 1 \), then \( v_k \) is a neighbour’s neighbour to \( v_i \). By adding an edge between each vertex \( v_i \) and its neighbours’ neighbours, the graph in Figure 3.2 (a) will be modified to the graph in Figure 3.2 (b). The degree of a vertex in the graph now correlates to the (unique) number of neighbours and neighbours’ neighbours of each cell.

![Figure 3.2: (a) Graph representation of a small network. (b) Modified graph representation.](image)

Now, two sets \( N_i \) and \( N_i^2 \) can be defined, where \( N_i \) is the set of neighbours to cell \( i \) and \( N_i^2 \) is the set of neighbours’ neighbours to cell \( i \). A vertex \( v_j \in N_i \) if \( v_j \) is a neighbour to vertex \( v_i \). In Figure 3.2 (a), there is an edge between each vertex \( v_i \) and the vertices within the set \( N_i \). If \( v_k \) is a neighbour’s neighbour to the vertex \( v_i \), then \( v_k \in N_i^2 \). In Figure 3.2 (b), there is an edge between each vertex \( v_i \) and the vertices within the set \( M_i = N_i \cup N_i^2 \). For future use, another set \( K \) of PCIs is defined, where \( K_{max} = 504 \).

3.2.1 Network Matrix Representation

For a matrix representation of the network, the elements of \( G \) can be translated to an \( n \times n \)-matrix, where each row \( i \) in the matrix represents the neighbour list for cell \( i \). The index \( i \) can except from identifying the cell in the matrix also represent the unique ECGI, described in Section 2.3. Using the matrix representation it is easy to obtain the neighbours’ neighbours as well. As \( G \) contains information of which cells that can be reached from any cell at a distance one, \( G^2 \) is the representation of which cells that can be reached from any cell at a distance two. For this matrix representation to suit the network properties, the diagonal of the matrix, i.e. the position \( e_{ij} \) for each \( i \), should be set to zero since there is no interest in knowing that a cell is its own neighbours’ neighbour.
To represent the PCIs, a vector $P$ of size $1 \times n$ is introduced. By positioning the PCI associated with cell $j$ at the corresponding position within the vector $P$, and element-wise multiply each row in the matrix $G$ with the PCI vector $P$, the resulting matrix will contain the neighbouring PCIs of each cell $i$. Similarly, the matrix $G^2$ can be multiplied element-wise with the vector $P$ for the matrix to contain the PCIs of every cells neighbours’ neighbour.

### 3.3 Problem Definitions

To be able to fully describe the problem mathematically, some definitions needs to be made regarding the conflicts that can occur within the network. First, a collision free and a confusion free network needs to be defined:

- **Definition 3.3.1** A PCI allocation is collision free if no cell has a neighbouring cell sharing the same PCI as the cell itself.

- **Definition 3.3.2** A PCI allocation is confusion free if no cell has two or more neighbours sharing the same PCI.

With these two concepts defined, the conflict free network can be defined:

- **Definition 3.3.3** A network is conflict free if there exist no collisions or confusions.

Definition 3.3.3 describes the goal of the PCI allocation. The aim when allocating the PCIs is to obtain a conflict free distribution where no ambiguity exist between the cells. A conflict free allocation can therefore be equalised as a feasible solution to the allocation problem.

The definitions stated this far describe the allocation problem in a centralised manner or for the network as a whole. For a distributed approach, only a partial set of the network may be considered when trying to solve conflicts locally. This partial set of the network will further be referred to as a sub network. For these sub networks, the following definition is applicable:

- **Definition 3.3.4** A sub network is conflict free if there exist no collisions or confusions within the sub network.

However, it is important to note that a conflict free sub network does not imply that all cells within the sub network are conflict free. One or several cells may be involved in a collision or confusion as part of another sub network. A feasible solution in a sub network using a distributed algorithm does also not imply that the sub network is conflict free. A feasible solution to a sub network is found when the conflict which triggered the algorithm is solved.

The above stated definitions defines a conflict free allocation, and how a feasible solution is found using both centralised and distributed approaches. However, a feasible solution does not imply an optimal solution. To be able to find a good or even optimal solution, the goal of the allocation needs to be determined. The goal of the allocation could, for example, be to minimise the number of PCIs used, the number of changes required to obtain a feasible solution or the run time of the algorithm.

### 3.4 Network Optimisation

Since the cell network is easily represented in graphical form, it is natural to look for optimisation methods which apply to graphs. This section begins with a short review of NP-completeness and continues with a description of the vertex colouring problem and how this can be mapped on the PCI allocation problem.
3.4. Network Optimisation

3.4.1 NP-completeness

Problems can be divided into different classes depending on their complexity: \( P, NP \) and \( NPC \) [18]. The problems belonging to \( P \) are the problems which can be solved in polynomial time. This means that for an input of size \( n \), a problem belonging to \( P \) can be solved in \( O(n^k) \), for some constant \( k \). If a problem belongs to \( NP \), the problem can be verified in polynomial time. Given a solution to a problem in \( NP \), this solution can be proved to be correct or incorrect in \( O(n^k) \) time. If a problem belongs to \( P \), it also belongs to \( NP \), since if a problem can be solved within polynomial time, it can also be verified within the same amount of time. Mathematically, this means that \( P \subseteq NP \).

The NPC class contains problems that is said to be \textit{NP-complete} [18]. A problem that belongs to NPC is a problem belonging to \( NP \) which is \textit{NP-hard}. If a problem is NP-hard, the problem is at least as hard as the hardest problem in \( NP \). For these problems, there are no known polynomial time solving algorithms. Reduction is used to prove that a problem \( B \) is NP-complete. If a problem \( A \) known to be NP-complete can be reduced to problem \( B \) in polynomial time, then \( B \) is at least as hard as \( A \), and therefore \( B \) is also NP-complete. Some of the NP-hard problems which are not in NPC are the \textit{undecidable problems}, for example Turing’s halting problem.

3.4.2 The Vertex Colouring Problem

Vertex colouring (or graph colouring) is the problem of colouring the vertices of a graph so that no connected vertices shares the same colour [18]. Using two colours, a proper vertex colouring can only be applied on graphs forming a simple path or a cycle containing an even number of vertices. To colour a graph with three or more colours is a problem which belongs to NPC.

There are two types of graph colouring problems: one optimisation problem and one decision problem. The optimisation problem related to vertex colouring aims to colour a graph \( G = (V, E) \) with as few colours as possible. The minimum number of colours that can be used to colour a graph is called the Chromatic number, denoted \( \chi \) [19], and the calculation of the chromatic number is NP-complete. The chromatic number is larger than or equal to the number of nodes in the largest clique of a graph, denoted \( n_k \). If the graph is perfect, then \( \chi = n_k \) [20]. The corresponding decision problem is called \( k \)-colouring, and aims to answer the question if the graph \( G \) can be coloured with \( k \) colours. If \( k \) is larger than or equal to the chromatic number, the answer would be yes. According to Brook’s theorem, the chromatic number of a graph \( G \) is at most the maximum vertex degree \( \Delta \), with two exceptions. If \( G \) is a complete graph or if \( G \) is an odd cycle, the chromatic number is at most \( \Delta + 1 \) [19].
3.4.3 Problem Reduction

As shown in Section 3.2, the cell network can easily be translated to graphical form. This representation can be used to map the PCI allocation problem to the vertex colouring problem by letting the set $K$ of PCIs map to colours. Recall from Section 3.3 that a feasible solution for the allocation is obtained when the allocation is conflict free, i.e. free from both collisions and confusions. Since there is a limited amount of PCIs, namely 504, a feasible solution can be found if there exist a conflict free allocation using these 504 PCI values. This correlates to the decision problem of determine if the graph may be coloured with $k$ colours where $k = 504$.

For the allocation to be collision free, no cell in the network may have any neighbour using the same PCI as the cell itself. This would map to that no two connected vertices in the graph may have the same colour. Applying the vertex colouring algorithm on the graph in Figure 3.2 (a), the resulting solution would be collision free, as seen in Figure 3.4 (a).

![Graph colouring example](image)

The coloured graph in Figure 3.4 (a) is however not confusion free. For an allocation to be confusion free, no cell may have two neighbours using the same PCI, i.e. no vertex may be connected to two other vertices sharing the same colour. Here, vertex $v_j$ has two neighbours $v_i$ and $v_k$ sharing the same colour. To be able to avoid confusions in the graph using the vertex colouring algorithm, the graph in Figure 3.2 (b) needs to be used instead. Since graph (b) has one edge for every vertex in $M_i$, and by that has additional edges between every vertex $v_i$ and its neighbours’ neighbours present in $N^2_i$, the vertex colouring algorithm is forced to give these neighbours different colours. As illustrated in Figure 3.4 (b), vertex $v_j$ and $v_k$ now have different colours.

To summarise, the PCI allocation problem can be translated into the k-colouring problem by doing the mappings in Table 3.1:

<table>
<thead>
<tr>
<th>PCI allocation problem</th>
<th>k-colouring</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell network</td>
<td>Graph $G = (V, E)$</td>
<td>$n = \text{number of cells in the network}$</td>
</tr>
<tr>
<td>Network cells</td>
<td>Vertexes, $v \in V$</td>
<td>Cell $i \rightarrow v_i \in V$ for each $i = 1, \ldots, n$</td>
</tr>
<tr>
<td>Network conflicts</td>
<td>Edges, $e \in E$</td>
<td>$e_{ij} = 1$ if $v_j \in M_i$ for each $i, j = 1, \ldots, n$</td>
</tr>
<tr>
<td>PCI’s (504)</td>
<td>Colours, $k$</td>
<td>$k = 504$</td>
</tr>
</tbody>
</table>

Table 3.1: Mapping between the PCI allocation problem and k-colouring.

This shows that the PCI allocation problem is at least as hard as the k-colouring problem, which is NP-complete.
3.5 Optimisation Models

As described in Section 3.3, the goal of the allocation is to obtain a conflict free PCI allocation. We consider a binary network model based on if cells interfere or not, i.e. if they are in conflict or not. We do not consider the degree of interference, conflict, or any workload characteristics (i.e. traffic volumes etc.). Since traffic and network parameters are traditionally used to be able to obtain a solution in networks not containing a feasible solution, the only networks considered in this work are the ones which contain a feasible solution. Because of this, the goal of the objective function will not be to obtain a feasible solution. This will instead be a constraint.

3.5.1 Minimum Number of PCI Changes

There are several possible goals of the objective function. The first one to be discussed is to **minimise the number of PCI changes**. To be able to formulate this mathematically, the following input parameters are defined:

- Let $K$ be the set of available PCIs with $K_{\text{max}} = 504$.
- Let $\Gamma$ specify the maximum value of the PCI range, where $\Gamma \in K$.
- Let $p_i = k$ if cell $i$ is initiated with PCI $k$.
- Let $N_i$ be the set of neighbours to cell $i$, and $N_i^2$ be the set of neighbours’ neighbours to cell $i$.
- Let $M_i$ be the set of cells in cell $i$’s surrounding, i.e. the set of neighbours and neighbours’ neighbours, so that $M_i = N_i \cup N_i^2$.

Some of these input parameters were first introduced in Section 3.2, but summarised here to ease the reading of the problem formulation. The output of the allocation algorithms are determined by the following variable:

- Let $x^k_i = 1$ if cell $i$ is allocated PCI $k$, and 0 otherwise.

For $k = p_i$, a change will occur if $x^k_i$ is set to zero, while no change will occur if $x^k_i$ is set to one. For $k \neq p_i$, a change will occur if $x^k_i$ is set to one, while no change will occur if $x^k_i$ is set to zero. The objective function can now be formulated as follows:

$$
\begin{align*}
\min \quad & z = \sum_i (1-x^{p_i}_i) + \sum_i \sum_{k \neq p_i} x^k_i \\
\text{s.t.} \quad & x^k_i + \sum_{j \in M_i} x^k_j \leq 1 \quad \forall i, k, \\
& \sum_{k=1}^{\Gamma} x^k_i = 1 \quad \forall i, \\
& x^k_i \in \{0, 1\} \quad \forall i, k.
\end{align*}
$$

(3.1)

The objective function will be minimised when performing as few PCI changes as possible. The first condition guarantees a feasible solution for the network, i.e. make sure that a PCI value $k$ only can be used by the cell $x_j$ if the PCI value is not in $M_i$. The second condition prevents a cell from being allocated with more than one PCI, and the last condition sets $x^k_i$ as a binary variable.
3.5.2 Minimum Number of Used PCIs

Another possibility of the optimisation is to try to reduce the number of used PCIs within the network. Since the networks are getting denser, it can be beneficial to reuse the PCIs already present within the network. By doing this, one is able to save unused PCIs for the future where no other possibilities might exist. To be able to formulate this mathematically, a new variable is defined:

- Let $\gamma_k = 1$ if PCI $k$ is used, and zero otherwise.

Now, the objective function can be formulated as follows:

$$
\min \quad z = \sum_{k=1}^{\Gamma} \gamma_k \\
\text{s.t.} \quad x_i^k + \sum_{j \in \mathcal{M}_i} x_j^k \leq 1 \quad \forall i, k, \\
\sum_{k=1}^{\Gamma} x_i^k = 1 \quad \forall i, \\
x_i^k \leq \gamma_k \quad \forall i, k, \\
x_i^k \in \{0, 1\} \quad \forall i, k, \\
\gamma_k \in \{0, 1\} \quad \forall k.
$$

(3.2)

Here, the sum of the used PCIs is minimised. Condition one and two are the same as in Model 3.1 and the third condition only allow $x_i^k$ to use PCI $k$ if it is used within the network. The fourth and fifth condition sets $x_i^k$ and $\gamma_k$ as binary variables.

3.5.3 Optimisation Model with Multiple Goals

Models 3.1 and 3.2 can be used to obtain two different optimal solutions in a cell network. The goal of the allocation could however be to combine these two objectives, i.e. to do as few PCI changes as possible and try to reduce the number of PCIs present within the network. To be able to reach this goal, the two optimisation models could be combined. If one of the objectives is more prioritised than the other, the combined objective function may be weighted. To do so, the following definition is made:

- Let $\zeta$ be a value between 0 and 1.

Now, the combined objective function can be written as:

$$
\min z = (1 - \zeta)(\sum_i (1 - x_i^P)) + \sum_k \sum_{i \neq p} x_i^k + \zeta(\sum_{k=1}^{\Gamma} \gamma_k).
$$

(3.3)

By choosing different values for $\zeta$, the prioritisation when allocating the PCIs will be different. If $\zeta$ is given a value close to 1, minimising the number of PCI changes will be the most prioritised goal. Similarly, if $\zeta$ is given a value close to zero, minimising the number of PCIs will be prioritised. The constraints would for this combined model be the union of the constraints in Models 3.1 and 3.2.

3.5.4 Execution Limitations

The optimisation models presented in Models 3.1, 3.2 and the combination of them are suitable to gain an optimal solution when approaching the cell network in a centralised manner.
The problem does however get very large even for smaller input data. A small example with 1000 cells \((n = 1000)\) using a range of all possible PCIs \((\Gamma = 504)\) would result in 504,000 variables and 505,000 conditions when trying to minimise the number of PCI changes. A duplication of the number of cells would result in twice the number of variables and conditions, respectively. If a realistic cell network is to be calculated where the number of cells is larger, the number of variables and conditions would be enormous. As the equipment used in this work is limited, no optimal solution will be calculated. It is possible to obtain an optimal solution, but it is not possible to obtain it within reasonable time. Instead, a distributed method is implemented, where the approach needs to make decentralised decisions which will mimic the behaviour of a centralised method when looked at from a higher perspective.
In this chapter, the allocation algorithms used within this work is presented. The chapter starts with a presentation of the studied related work, including both centralised and distributed approaches. The chapter continues with a description of the suggested approach for the allocation problem, and terminates by specifying the algorithms designed and implemented in this work.

4.1 Related Approaches

To be able to gain inspiration and to get a clearer picture of what has been done and what has been proven to be effective, articles regarding related approaches were studied. To enhance as much inspiration and knowledge as possible, both centralised and distributed approaches of different quality were considered. The following two sections gives a summary of what approaches has already been evaluated.

4.1.1 Centralised Approaches

There are several articles using graph theory as a basis for a centralised solution of the PCI allocation [21, 22, 23]. Using graph theory is a classical method to represent networks, not only for the case of PCI planning.

Wei et al. are representing the graph $G = (V, E)$ using a $m \times m$ matrices $C$, where $m$ is the number of vertices within the graph [21]. The values of $C$, namely $c_{ij}$, is calculated by $c_{ij} = w_{ij} \times p_{ij}$ where $w_{ij}$ is defined as calculated linear propagation loss value and $p_{ij}$ is a binary variable which is equal to one if the $i$th and $j$th cell are assigned with the same PCI. The algorithm is creating a minimum spanning tree based on the weights $w_{ij}$ and the PCIs are randomly picked from the set of unused IDs. When all available IDs are used, a suitable PCI is selected by calculating the reuse effect, i.e. the PCI is chosen by creating an ordered set of PCI choices considering the number of uses within a certain distance. If collision and confusion cannot be avoided, the first ID within the ordered set is selected.
4.1. Related Approaches

Different matrix-based approaches have been used to model the PCI allocation as a graph theoretical problem. For example, Abdullah et al. [22] used an incidence matrix as a basis. The algorithm takes benefit from the structure of the incidence matrix to convert the graph colouring problem to a constraint problem solving. By modifying the incidence matrix to a minimised version where unlimited vertices were stored in the same column, the matrix becomes more compact.

Bandh et al. modified the graph \( G = (V, E) \) to avoid conflicts when assigning PCIs by adding extra edges in the graph from every vertex to its neighbours’ neighbours [23]. The problem is after that solved as a classical graph colouring problem. They propose to reuse PCIs of a distance three from the concerned cell to avoid collisions and confusions but at the same time be economic in the use of new PCIs. Their approach were tested against two data sets. The first set contained geo positions of Vodafone Germany’s 3G sites. The number of cells could unfortunately not be found because of a broken link in the references. The second set were an artificial network containing 750 cells.

In an article written by Krichen et al. [24], the distance three (D3) method used by Bandh et al. [23] is evaluated against Random Relabeling (RR) and Smallest available Value algorithm (SV) in terms of the best performing PCI selection method. Krichen et al. showed that among these three PCI selection methods, D3 seems to be the least effective. RR, which is the algorithm defined by 3GPP standards [10], and SV performed a lot better.

Another centralised approach is presented by Schmelz et al. [25]. This approach is using Magnetic Field Model (MFM) techniques for self-configuration of both physical cell ids and physical random access channel (PRANCH). Inspired by the basic behaviour of magnets (i.e. their magnitude and direction) each cell in the network is represented at the OSS by an electromagnet. The conflicts in the network are thereby represented as repulsion effects of the electromagnets.

Sanneck et al. is analysing the performance using range separation for efficient multi-layer, multi-vendor networks [15]. The article discusses three different types of PCI allocation: layer independent, continuous with cross-layer coordination and range separation. Since range separation is claimed to be the most effective and least complex, it was tested against the layer independent allocation. The simulation network contained both a macro layer and a pico layer and different safety margins were used, i.e. different distances for PCI reuse. The result showed that range separation and layer independent allocation perform similar with a low safety margin, but range separation was slightly better when using a higher margin.

4.1.2 Decentralised Approaches

Liu et al. presents a distributed approach using a consultation mechanism that is to be implemented in the eNB [26]. The eNB first receive a PCI list from O&M and then neighbour information from other eNBs through the X2 interface. Using this information and the consulting mechanism, the eNB is responsible for creating a suitable PCI for itself. The mechanism is tested and compared against an ordinary assignment. The test does however only demonstrate the occasion of a new eNB being installed. Moreover, the test is only performed once and with as little as 50 cells.

Another distributed solution is presented by Ahmed et al., using a graph colouring approach [27]. In their study, they consider both Primary Component Carrier Selection and PCI allocation. The authors used four different distributed local search algorithms for the graph colouring and the result was compared against two complete constraint satisfaction algorithms. The simulations were performed on a model of a number of multiple-floor build-
ings in a Manhattan grid with several pico cells placed within the buildings. Ahmed et al. concluded that binary confusion prices were as effective as real pricing. They also conclude that a range of temporary PCIs is counterproductive if the number of cells are large enough.

Two of the authors from the previous study are in another article looking at another approach of graph colouring [28]. In this study, the authors consider a distributed multi-operator HetNet (Heterogeneous Network) where the assumption is that the single eNB can only access extended information from the other eNBs that belongs to the same operator. The problem is modulated as an interference graph with binary valued asymmetric edges, where directional edges are used between cells belonging to different operators and multi-directional edges are used between cells belonging to the same operator. The algorithm used is Stochastic Local Search with Focused Plateau moves (SLS-FP) which is compared to Steepest Descent Local Search with Focused Plateau moves (SDLS-FP). The first algorithm randomly selects a new PCI to a cell in conflict whilst the second algorithm tries all possible PCIs and selects the best. The result and conclusion of this study is vaguely presented, but indicates that both algorithms can be used for collision and confusion freeness in HetNets.

Baba et al. are comparing three different approaches for self-configuration in self-organising networks [29]. The study describes the LTE standard approach, the Graph Colouring approach and the Nokia Siemens (NS) approach [16]. Their conclusion was that the NS approach seems to be the best method because of its fast network convergence and relatively few resulting conflicts. This study does however only consider Femto cells.

Lim et al. are presenting a number of self-organising schemes in [30]. The proposed schemes are considering initial configuration and maintenance of the Neighbour Cell List (NCL) as well as PCI allocation for newly added eNBs. For PCI allocation, the proposed method is to eliminate collisions by creating a list of possible PCIs and eliminating neighbouring PCI values from the list using UE scanning. Based on the Reference Signal Received Power (RSRP), the eNB selects an available PCI given a certain parameter \( \alpha \). This parameter \( \alpha \) determines the geographical distance between the node to be allocated and another cell in the network sharing the same PCI. The approach was evaluated against random selection, and was showed to outperform random selection when a proper \( \alpha \) was chosen.

Diab et al. are using a self-organised solution called Stable Graph Colouring (SGC) in [31]. The method tries to minimise conflicts caused by newly switched on eNBs and to improve the PCI utilisation ratio. To minimise conflicts, the method uses a sub-range of PCIs reserved for temporary initial assignment as used in the Nokia Siemens (NS) approach [16]. To improve PCI utilisation ratio, the operating eNB uses 3-hop neighbour information to find a suitable PCI not used by its own neighbours or 2-hop neighbours. The algorithm was tested on a simulation environment of 100 km\(^2\) using both micro- and macro cells. There were 18 static scenarios performed and the number of eNBs were between 50 and 900. The proposed algorithm was compared against the NS approach, an ordinary graph colouring (GC) approach and the LTE standard approach. The NS approach was slightly better than the SGC scheme in terms of average number of PCI conflicts, but SGC performed much better than both the GC and the LTE standard approach. In terms of PCI utilisation rate, SGC was beaten by both GC and LTE standard, but performed better than the NS approach.

Amirijoo et al. present an approach where the confused node is the node to make the decision regarding which one of the confusion-causing nodes that should be the one to change its PCI [14, 32]. The decision is based on a neighbour search performed by the confused node and they suggest three different options to base the decision on: (i) the node with the lowest GID should change its PCI, (ii) the node with the shortest neighbour list should change its PCI, or (iii) the node which changed its PCI most recently should change its PCI.
The approach is evaluated using a smaller test scenario and a more realistic urban scenario, and the performance of the three options were similar. Amirijoo et al. does however accentuate the use of GID as decision basis since this option requires no additional information passing.

4.2 The Suggested Approach

The approach suggested in this work is today not compatible with the 3GPP standard. As mentioned in Section 2.3.2, the standard specifies that for a distributed approach, the cell causing confusion is the one to decide when to change its PCI and when doing it, randomly pick one from its list of available PCIs. For this work, the proposal is that the cell experiencing the confusion should make the decision, in other words, the cell that has at least two neighbours sharing the same PCI. The proposal resembles the decision approach investigated by Amirijoo et al. in [14], but for the approach presented in this work, the decision basis is expanded. Instead of making the decision based on the confused cells neighbour information, the suggestion is that this cell shall request the neighbour information from the cells causing the confusion. By doing this, the decision basis is enlarged and the decision will probably favour a larger part of the network. Figure 4.1 illustrates the decision basis for what the standard supports, what the algorithm presented by Amirijoo et al. supports and for the suggested approach in this work. In the figure, the confused cell is using PCI X and the cells causing the confusion is using PCI Y. The dark red cell is in this case the one to make the decision and the light orange cells illustrates the decision basis. As seen in the figure, the suggested approach would include a larger amount of cells than the other two.

![Figure 4.1: Decision basis for (a) what the standard supports, (b) the approach suggested in Amirijoo et al. and (c) the approach suggested in this work.](image)

The suggested approach was initially divided into two versions: one basic version and one extended. The basic version uses the proposed decision basis to determine which one of the cells causing confusion that has the most PCIs available. This cell will be ordered to change its PCI, and the new PCI will be randomly selected from its list of available PCIs. If both cells have equally long lists of available PCIs, the cell to change is chosen at random.
The extended version is complemented with a decision function which picks the best cell and a suitable PCI based on four decision rules which can be combined into re-allocation rules. The rules are stated as follows:

1. Prioritise changing the PCI of the cell that has the longest list of available PCIs.
2. Prioritise changing the PCI of the cell among these that causes more than one confusion.
3. Select a new PCI from the PCIs already present within the sub network, and
4. Prioritise cells with a new PCI value at furthest distance.

The priority of the rules will be specified for each algorithm in Section 4.3. The first rule is in compliance with the basic version of the approach. Rule two, three and four are added to mimic the behaviour of the optimisation models by trying to reduce the number of PCI changes and reduce the number of new PCIs inducted in the sub networks. The second rule is illustrated in Figure 4.2. Here, the dark red cell associated with PCI Y is the one making the decision. In this case, the smarter choice would be to let cell 2 be the one to change its PCI, since that would solve two confusions at the same time. If cell 1 would be the one to change, the cell using PCI Z would still be confused.

![Figure 4.2: Illustration of the second decision rule.](image)

The third rule promote the reuse of already utilised PCIs within the network. By reusing an already utilised PCI if possible, the unused PCIs can be stored for a situation where there are no other possibilities to chose from. This will prevent the network from experiencing unfeasible sub networks for as long as possible. The forth rule is illustrated in Figure 4.3. If the choice is between cell 1 changing to PCI V and cell 2 changing to PCI W, the latter would be the preferred choice. Since the cell distance is larger between cell 2 and the cell with PCI W than between cell 1 and the cell with PCI V, even the geographical distance is probably larger. If the geographical distance is larger, the risk of a future conflict between the cells are less than if the distance is smaller and therefore the larger cell distance should be prioritised.

![Figure 4.3: Illustration of the fourth decision rule.](image)
The combination of the four rules tells the cell to change and the new PCI that should be pursued. This means that the most desirable choice is to select the cell that has the most available PCIs on its list, which in its list has a reusable PCI on distance four and besides this is part of at least two confusions. If none of the cells meet all of the decision rules, the rules are prioritised to make an as good as possible decision. Details about the prioritisation of the decision rules is described in Section 4.3.4.

Later in this work, the need for a more goal focused version of the extended implementation was discovered, and therefore a third version of the approach was suggested. In this version, further called the focused version, the first and the fourth decision rule were considered, and the second and third decision rule were scoped out. By doing this, the algorithm is able to fully focus on minimising the number of PCI changes, which match the goal of Model 3.1 defined in Section 3.5.2.

4.3 The Algorithms

To be able to determine how well the proposed solution would perform in comparison to standardised methods, five different distributed algorithms were designed and implemented. The first algorithm performs according to what today’s 3GPP standard supports in a distributed manner. The second algorithm is inspired by the algorithm investigated by Amirijoo et al. in [14]. The third algorithm is an implementation of the basic version described in Section 4.2 and the fourth algorithm is the extended version mentioned in the same section. These four algorithms are all part of Test Phase I, described in Section 5.2. For Test Phase II, the fourth algorithm is compared to the focused algorithm, also described in Section 5.2. This algorithm will further be entitled Algorithm V. The following sections will describe each implementation in more detail.

4.3.1 Algorithm I - "Standard"

Algorithm I is, as mentioned above, implemented according to what today’s standard supports. The algorithm initiates with detecting all of the confusions present within the network. For each node being confused, the algorithm extracts the nodes causing this confusion. The nodes causing confusion gets instructed about this and starts searching for alternative PCI values by checking what PCIs are already taken by their neighbours and their neighbours’ neighbours. If the node has multiple alternative PCI values to choose from, the node randomly selects one from this set and the PCI vector is updated. If there are no alternative PCI values available, there is no action taken.

In a real situation, each node aware of it causing a confusion may or may not decide to switch its PCI at any time within a certain time interval. If two or more nodes decide to change their PCI at the same time, they will make their decision based on the same PCI vector and may because of this make decisions that are not valid. To simulate this, each node causing confusion for one node are given a random "time" between 0 and 1. These values are sorted and their distance are calculated. If the interval between two times is less than a defined T, the nodes are to perform their PCI change simultaneous. In this case, the changes in the PCI vector is performed after all of the nodes within the same “solving block” have performed their changes.
4.3. The Algorithms

4.3.2 Algorithm II - "Research"

The second algorithm inspired by the work by Amirijoo et al. in [14] initiates in a similar way as the previously mentioned algorithm. First, all nodes experiencing confusion are found. For each of these nodes, the nodes causing the confusion are extracted. In this implementation, the decision regarding which node that should be changing its PCI is taken by the node experiencing the confusion. As described in Section 4.1.2, Amirijoo et al. propose three different decision bases for the confused cell: choose the cell with the lowest ECGI, choose the cell with the shortest neighbour list, or choose the cell which changed its PCI most recent [14]. For this implementation, the first decision basis was chosen since it is the one preferred by the authors, even though their simulations showed similar results for all of the approaches. This is because the option of using ECGI does not result in extra signalling.

To be able to make the decision based on the ECGI, an ECGI vector was constructed containing a unique value for each node in the implementation. The algorithm looks up the ECGI corresponding to the nodes causing its confusion and orders the node with the lowest ECGI to perform a PCI change. This node checks its surrounding (i.e. checks the PCIs of its neighbours and its neighbours’ neighbours) and randomly picks a PCI from the set of available PCIs. If no PCIs are available, no action is taken.

4.3.3 Algorithm III - "Basic"

The third algorithm is the first one using the PCI selection method approached in this work. As for the other two previously described algorithms, this algorithm initiates by finding all of the nodes experiencing confusion. When these nodes are found, all of the nodes causing confusion for each node are found. The confused node command the two nodes causing the confusion to return a list of their possible PCIs, which is based on a neighbour search performed by both nodes respectively. The list of their possible PCIs will contain the all the permitted PCIs minus the PCIs used by their neighbours and neighbours’ neighbours. Using these lists as a basis, the node containing the longest list of available PCIs is chosen to change its PCI. The new PCI will be chosen randomly from this list. If both of the nodes have equally long PCI lists, the node changing its PCI is chosen randomly. If none of the nodes have any available PCIs, no action will be taken.

4.3.4 Algorithm IV - "Extended"

Algorithm IV is working similarly to the third algorithm but is complemented with decision rules described in Section 4.2. As for the previous algorithm, the node experiencing the confusion decides which of the nodes causing the confusion that should perform the PCI change. For this algorithm however, the confused node demands two different lists from each of the confusion-causing nodes: one containing the PCIs used by its neighbours and one containing the PCIs used by its neighbours’ neighbours. These lists can be used to calculate basic data for the decisions, for example the available PCIs for each node. Using this information, the confused node tries to make the smartest PCI change possible, based on the decision rules.

As mentioned in Section 4.2, if not all of the decision rules can be met, some prioritisation of the importance of the rules needs to be made. For this implementation, the following (simplified) priority has been made:
1. Choose a node that is causing more than one confusion, preferably the one with the longest list and an available distance four PCI.
   a) ..else the node with the shorter list (if not empty) causing a double confusion and has an available distance four PCI.
   b) ..else the node with the longest list with an available already present PCI.
   c) ..else the node with the shorter list with an available already present PCI.
   d) ..else the node with the longest list.

2. Choose a node that has an available distance four PCI, preferably from the node with the longest list of available PCIs.
   a) ..else the node with the shorter list with an available PCI of distance four.

3. Choose the node with an available PCI already present within the network, preferably the one with the longest list
   a) ..else choose the node with the shorter list with an available PCI already in the list.

4. Choose a PCI from the one with the longest list

If the two nodes within the conflict resolution has equally long lists, the prioritisation looks like the following. If the statement is true for both nodes, the one to change its PCI is picked at random.

1. Choose the node causing more than one confusion having a distance four PCI available.
   a) ..else the node causing more than one confusion with a PCI already present in the network available.
   b) ..else the node causing more than one confusion with an available PCI.

2. Choose the node that has a distance four PCI available.

3. Choose the node with an available PCI already present within the network.

4. Randomly choose one of the nodes to change its PCI.

If both nodes have empty lists, no action is taken. For both the case of equally and unequally long lists, if a node is chosen and has several PCIs that matches the statement, one of these will be picked at random.

4.3.5 Algorithm V - "Focused"

The fifth algorithm is implemented using the first and the fourth decision rule as a basis. The node experiencing the confusion demands the same lists from its confusion-causing neighbours as for Algorithm IV, namely one list of neighbours’ PCIs and one of neighbours’ neighbours’ PCIs. Using the information in this list, the following prioritisation is made:

1. Choose the node causing more than one confusion, preferably the one with the longest list of available PCIs.
   a) ..else the node with the shorter list (if not empty) of available PCIs causing more than one confusion.

2. Choose the node with the longest list of available PCIs.

If both nodes have equally long lists, the following action is taken:
4.3. The Algorithms

1. Choose the node causing more than one confusion.

2. Randomly choose one of the nodes to change its PCI.

As for Algorithm IV, if both nodes have empty lists of available PCIs, no action is taken. When the node to change its PCI has been chosen, the new PCI value is randomly picked from the list of available PCIs.
5 Evaluation Methodology

Since it is easier to detect a PCI confusion than a PCI collision within a realistic network, the created algorithms are solving the PCI conflicts based on confusion information. The algorithms and the simulation environment are both implemented using MATLAB version R2015b [33]. This chapter begins with a description regarding how the simulation environment is constructed, and the extended functionality added to mimic a realistic scenario as well as possible. The chapter continues by describing the evaluation scenarios and terminates with a discussion regarding reliability and validity.

5.1 The Simulation Environment

To be able to test the algorithms implemented, a test environment was needed. This environment was implemented using matrices, vectors, predefined variables and the plot-function. The algorithms take a binary adjacency matrix, a vector of PCI values used by the different cells and a vector containing permitted PCI values as a standard input. Using these inputs, the algorithms can extract the neighbour list of each of the cells by multiplying the rows of the adjacency matrix with the PCI vector. The vector of permitted PCI values is used when calculating possible PCI changes. The cells will further in this section be referred to as nodes, as they are plotted as nodes in the implementation environment.

There are several predefined variables set in the beginning of the script. The variable \( n \) sets the number of nodes for the simulation, the variable \( pp \) sets the maximum id value for the permitted PCI range and the variable \( area \) defines the size of the area in which the nodes are allocated. Beyond these variables, there are three distances and three probabilities set. These variables are used to determine whether or not two nodes may have neighbour relation. Lastly, there is a probability variable set representing the probability that node \( j \) has a neighbour relation to node \( i \) given that node \( i \) has a neighbour relation to node \( j \). To match the reality, this probability should be quite high.
Creating the network environment, the first step is to set the previously mentioned variables. With this as an input, the script generates x- and y-coordinates at random, \(n\) of each, within the set area. These coordinates represent the positions of the nodes in the network. The script generates the PCI vector with length \(n\) and the maximum value \(pp\) and generates the vector of permitted PCIs containing values from 1 to \(pp\). The adjacency matrix initiates containing all zeros.

To create the network, the script loop through all of the possible relations by calculating the Euclidean distance between any two nodes \(i\) and \(j\). The script compares these distances to the distance parameters and, given a certain probability, creates a relation by adding a 1 in the adjacency matrix in position \(A(i, j)\) and drawing a line between these two nodes on the plot. If a relation is established, the symmetric relation \(A(j, i)\) is also established given the probability mentioned above. For illustrative purposes and as a basis for the evaluation, the script calculates the number of relations within the network, and also specifies how many of these that are asymmetric.

Figure 5.1: An example picture of a very small, easily overlooked network.

Figure 5.1 is an example of a very small network where the nodes and relations can be easily overlooked. The black numbers on the plot is the number of the node (i.e. the number of the row in the adjacency matrix, which is also used as ECGI), and the red number is the randomised PCI values. The number of nodes are 30 and the number of relations are 135, which leads to an average of 4,5 neighbours. In a real scenario, the number of neighbour relations would be a lot higher. Figure 5.2 shows another example where the average number of relation are approximately 29, given the 1000 nodes and the 29 124 neighbour relations within the network. This amount of neighbours could figure in a sparse realistic environment. For a denser network, the plot would be even harder to read, since the number of nodes and relations would be a lot higher. Because of this, the networks constructed in the test phase will not be plotted.
5.1. The Simulation Environment

Figure 5.2: An example picture of a larger, more realistic network.

5.1.1 Adding Additional Nodes

The simulation environment described above generates a static version of a modern LTE network. To make the model more dynamic, and by that more realistic, the possibility of removing relations and adding additional nodes was implemented. This script takes the old network parameters as input parameters and creates a new, updated version according to specifications. For this script, the parameters that needs to be predefined are extended_n which sets the number of additional nodes that should be added to the original network and stay which specifies the probability that an already existing neighbour relation should be preserved to the new network.

When the parameters are set, the script constructs the network in a similar manner as the original network was created. The script loops through the binary adjacency matrix which has been extended with the new nodes by zeros. If $A(i,j) = 1$, the script generates a random number $r$ between 0 and 1, and if $r$ is less than the set parameter stay, the relation is preserved. Else, $A(i,j)$ is set to zero and the relation is by that deleted. If $A(i,j) = 0$, a relationship is set using the same distance and probability measure as in the original script. This means that two original nodes failing to establish a relationship at the first run might end up with a relationship when the relation is reconsidered in the enlargement of the network.

5.1.2 Real Time Simulation

To make the simulation even more realistic, the possibility that two or more nodes are experiencing and reporting a confusion at the same time is added. This is implemented by randomly generating a time $t$ for when the node will start to perform the confusion resolution process. The performing times are ordered and the distance between the adjacent times are calculated. If the distance between two times $t$ is less than a previously set $T$ the two nodes will perform their process at the same time. For the implementation, this means that the PCI list will be updated after both of the nodes have performed their healing process. For the simulation to be realistic, this $T$ will be a really small value. However, since there is a
possibility that two or more nodes perform their healing process at the same time and by that has a risk of choosing an invalid PCI, this should be included within the implementation to make it more realistic.

5.2 Evaluation Scenarios

To evaluate the performance of the algorithms, each algorithm will be run against several test scenarios. In Table 5.1 the plan for the testing is illustrated. As seen in the table, the testing is divided into three different phases. The first test phase includes Algorithm I-IV where these algorithms will be run against the scenarios "PCI Restriction", "Neighbour Magnitude" and "Network Extension" (TS-I, TS-II and TS-III respectively). In the second test phase, the extended and the focused version of the approach, i.e. Algorithm IV and V, will be compared to each other. This phase includes the scenarios Neighbour Magnitude and Network Extension. The third test phase will include all of the five algorithms, which are tested against a Watts-Strogatz small world network (WS-SWN).

<table>
<thead>
<tr>
<th>Test phase</th>
<th>Inclusion</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>III</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 5.1: Test phase description.

For all of the test scenarios, the environment where the network will be plotted is set to 1000 × 1000 unit of area with equal neighbour relation establishment parameters. Two nodes separated by an euclidean distance of at most 50 unit of length, the probability of a neighbour relation is 80 %. If the nodes have a distance between 50 and 100, the probability of a neighbour relation is 60 % and if the distance is between 100 and 150, the probability is 40 %. Given that node \( a \) establishes a neighbour relation to node \( b \), the probability of a symmetric relation from \( b \) to \( a \) is 90 %. These properties does however not apply to Test Phase III, since this test environment is based on an extended script, as described in Section 5.2.4.

5.2.1 PCI Restriction

The first test scenario aims to evaluate how the algorithms perform when the PCI range is reduced. To do so, a test environment is created containing 2000 nodes within the defined area. The PCI range is initially set to all possible PCIs, i.e. to 504. When the PCI range 504 has been evaluated, the range will be reduced to 400 permitted PCIs and reduced again to 300 permitted PCIs. The least range of 300 PCIs was chosen to challenge the algorithms but still induce feasible solutions for the given network size. For this scenario, the environment is static, which means that for each test case, the PCI vector will be rewritten. The network and the relations are however the same in each test case.

5.2.2 Neighbour Magnitude

The second scenario aims to evaluate how the algorithms perform in denser environments. This time, instead of reducing the number of possible PCIs, the number of nodes will be increased. The number of nodes that will be evaluated are 2000, 3000 and 3500 and the neighbour distribution for each test case will be extracted. The neighbour distribution will in this case ensure that the denseness of the networks are realistic. The largest network size of 3500 nodes was chosen to challenge the algorithms, but still induce feasible solutions. As this is a static test scenario, the network will be rewritten between the different test cases.
5.2.3 Network Extension

In the last test scenario, an attempt to mimic a realistic, dynamic environment is made. For this scenario, the function described in Section 5.1.1 is used. The test cases will include a network expanding from 2500 to 2563 nodes (2.5% increase), 2500 to 2625 nodes (5% increase), from 2500 to 2750 nodes (10% increase) and from 2500 to 3000 nodes (20% increase). Because the amount of temporary data that needs to be stored is quite high just by doing one dynamic network expansion, the network will be rewritten between the different test cases. The behaviour of the algorithms is also forecasted to be similar if further additions would be made to the same network, so therefore only one expansion will be done in each test case. The given percental values were chosen randomly to induce a notable change within the networks, but not to induce an entirely new network.

5.2.4 Watts-Strogatz Small World Network

For the third test phase, each algorithm will be tested in Watts-Strogatz small world networks [34]. This test phase aims to evaluate the overall performance of the algorithms, as the network structure is not typical for cell networks. The phrase “Six degrees of separation” is grounded from the small world studies, which carry the idea of a short chain between any two people in the world. Apart from social science, the small world networks have been applied in various contexts, such as computer science, mathematics and biological science [35]. The Watts-Strogatz small world network is a random network with small world properties in terms of a clustering coefficient and a small shortest path length. The script used to generate the small world can be found in [36]. The algorithm in the script performs the following two steps given the input parameters $N$, $K$ and $\beta$:

1. A ring lattice is created containing $N$ nodes. The nodes gets connected to its $K$ nearest neighbours on each side.

2. With the probability $\beta$, the target node is rewired for each edge within the graph. The restriction is that no rewired edge can be duplicated or a self-loop. The mean degree of the nodes will be $2K$.

An illustrative example is shown in Figure 5.3. The $\beta$ value, that should be set to a number between 0-1, determines the appearance of the output graph. If $\beta$ is set to zero, the graph remains a ring lattice. If beta is set to one, the result is a random graph. For this testing, the same parameters will be used as the example graphs are initiated with in [36], which is a constant $N = 500$, $K = 25$ and a varying $\beta$ with value 0, 0.15, 0.5 and 1. To complement this, one parameter setting will be used to construct ten different networks, to see how much variation will occur when the same $\beta$ is repeated. The parameters chosen for this are again $N = 500$, $K = 25$, and $\beta$ is randomly chosen as $\beta = 0.15$.

![Illustrative example of a small world construction.](image)

Since the output data returned by the script had a different structure than the input data required by the implemented algorithms, a minor MATLAB script was created to convert the
data to the desired form. The output data consisted of a set of nodes and a set of relations. Using the relations from the output data the input matrix $A$ (see Section 3.2.1) was created. As the script in [36] is designed for undirected graphs, the resulting $A$ matrix was symmetric.

### 5.3 Reliability and Validity

It is tempting to let as many procedures as possible be randomised to mimic a realistic behaviour within the simulation. The procedures may be which PCI each of the nodes within the network will be initiated with, in which order the nodes will be managed and what new PCI that will be chosen within the rebooting. However, since there are five different algorithms to be tested, too much randomness will lower both the validity and the reliability of the test result. If too many parameters are chosen at random, the result will most likely differ a lot between the different executions of the algorithms. Because of this, the reliability of the result decreases since it is hard to determine if a good result is due to a successful algorithm or because of a favourable randomisation of the parameters. Regarding the validity of the result, it is hard to draw conclusions about which algorithm performed the best when the different algorithms were fed with different input parameters.

There are several interventions performed to try to reduce this problem. Firstly, the initial network and PCI allocation will be the same for each of the algorithms, i.e. the PCI allocation will be reset between the different algorithms when executing the test cases. When the network is extended with additional nodes, these nodes will have equal geographical allocation, the same relations will be established between the nodes and the nodes will be given the same initial PCI. Beyond this, the order of which the nodes experiencing confusion is examined will be the same for each algorithm. These interventions will increase the validity of the result since the correlation between the different algorithms will be better.

For every test case, each algorithm will be executed ten times, and for each of these ten executions, the algorithms gets to iterate five times as a maximum. For the third test scenario, this means five executions on the original network and five executions for each extended network. For each test case, this means 30 executions per algorithm. Between every execution, the initial PCI allocation will be reset which will give each algorithm ten chances to perform. By letting the algorithm run several times for each test case, it will be easier to determine if one result was caused by favourable or unfavourable randomisation of the parameters. This will eventually lead to a more reliable result. The result is moreover complemented with confidence intervals.
6 Evaluation Results

In this section, the evaluation results of each test phase will be presented. Here, the mean value of the ten executions for each algorithm is presented within the figures, together with a confidence interval with a confidence level at 95%.

6.1 Test Phase I

As described in Section 5.2, the first test phase consists of three different scenarios: PCI Restriction, Neighbour Magnitude and Network Extension. The algorithms taking part of this phase is Algorithm I-IV, namely Standard, Research, Basic and Extended.

6.1.1 PCI Restriction

The test details for the PCI restriction is presented in Table 6.1. In this scenario, the purpose was to evaluate how the four different algorithms would behave using different permitted PCI ranges. The algorithms were successful in solving all of the confusions in every execution of every test case.

<table>
<thead>
<tr>
<th>Test Case</th>
<th># of nodes</th>
<th>Average # of neighbours</th>
<th>PCI range</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.a</td>
<td>2000</td>
<td>90</td>
<td>504</td>
</tr>
<tr>
<td>I.b</td>
<td>2000</td>
<td>90</td>
<td>400</td>
</tr>
<tr>
<td>I.c</td>
<td>2000</td>
<td>90</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 6.1: Test Phase I - Test details for the test scenario PCI Restriction.

The result of the first test scenario is shown in Figure 6.1. Here, the number of PCI changes are shown for each of the four algorithms for the three test cases. As seen in the figure, the fourth algorithm is the one needing to make the fewest PCI changes to reach a feasible solution within the network, independent of the PCI range used. The other three algorithms performs similarly in every test case with small variations among each other. Algorithm II is somewhat the best performer among these three with the fewest number of PCI changes performed in two out of three test cases. The range of the confidence intervals for Algorithms
I-III varies a lot between the different test cases, and no obvious pattern can be seen in the interval width for any of these algorithms.

6.1.2 Neighbour Magnitude

The aim of the second test scenario was to evaluate how the different algorithms would handle larger, more complex networks resulting in an increasing amount of neighbour relations for each node. Details regarding the amount of nodes present within the test cases and the resulting average number of neighbour relations is presented in Table 6.2. The exact neighbour distribution for each of the test cases is shown in Figure 6.2. All of the algorithms managed to solve all of the confusions within the networks with two exceptions. Both Algorithms II and III failed two times each when trying to solve Test Case II.c. In each of these cases, the algorithms failed by one confused node.

<table>
<thead>
<tr>
<th>Test Case</th>
<th># of nodes</th>
<th>Average # of neighbours</th>
<th>PCI range</th>
</tr>
</thead>
<tbody>
<tr>
<td>II.a</td>
<td>2000</td>
<td>90</td>
<td>504</td>
</tr>
<tr>
<td>II.b</td>
<td>3000</td>
<td>135</td>
<td>504</td>
</tr>
<tr>
<td>II.c</td>
<td>3500</td>
<td>158</td>
<td>504</td>
</tr>
</tbody>
</table>

Table 6.2: Test Phase I - Test details for the test scenario Neighbour Magnitude.
6.1. Test Phase I

The result of this scenario is shown in Figure 6.3. As illustrated in the figure, the larger number of nodes that are present within the network, the more effective the fourth algorithm is in comparison to the other three algorithms. For Test Case II.a, the fourth algorithm performs approximately 16% less PCI changes than the rest of the algorithms. For Test Case II.b, the number of PCI changes is decreased by approximately 35%, and for the largest test case, II.c, the number of PCI changes are almost halved. The other three algorithms are again performing similarly in the entire test scenario. In this scenario, Algorithm II is no longer the best performer among the three, as the relative performance of the three algorithms varies for every test case. Notable is that as the number of nodes in the network increases, the larger the confidence interval gets for Algorithms I-III whilst the confidence interval for Algorithm IV is stable. This is assumed to depend on the larger amount of randomised parameters implemented in Algorithms I-III.
6.1.3 Network Extension

For each of the test cases in the third scenario, the original network contains 2500 nodes where between 2490 and 2495 are experiencing confusion. This test scenario aims to investigate how the different algorithms handle network manipulations of different sizes. In each test case, the original network will first be solved by each algorithm. Using the resulting solution, the network will be extended with a predefined percentage of the original network, as seen in Table 6.3. As mentioned in Section 5.1.1, all of the already established relations within the network will also be reconsidered, which means that a few relations may be erased. In this scenario, all of the algorithms managed to solve all confusions, both within the original networks and the extended ones.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Extended network</th>
<th>Perceptual extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.a</td>
<td>2563 nodes</td>
<td>2.5 %</td>
</tr>
<tr>
<td>III.b</td>
<td>2625 nodes</td>
<td>5 %</td>
</tr>
<tr>
<td>III.c</td>
<td>2750 nodes</td>
<td>10 %</td>
</tr>
<tr>
<td>III.d</td>
<td>3000 nodes</td>
<td>20 %</td>
</tr>
</tbody>
</table>

Table 6.3: Test Phase I - Test details for the test scenario Network Extension.

In Table 6.4, the number of relations present within both the original networks and the extended networks for each test case are shown. Even though the four original networks used are unique for each test case, the average number of relations per node is 112 for all of the networks. The number of neighbours in the extended networks increases along with the higher percentage of extension presented in Table 6.3. Notable is that it is the average number that is presented, which means that there are nodes with fewer neighbours as well as nodes with a lot more neighbours within the network.
### Test Phase I

#### Test Case III.a

In Figure 6.4 the result from Test Case III.a is shown. As seen in the figure, the fourth algorithm is the one making the fewest PCI changes to solve the original network, which is in line with what was shown in the previous test scenarios. However, Algorithm IV is also the one making the most PCI changes to reestablish a feasible solution after the network extension. Looking at the total number of PCI changes that needs to be performed to both solve the original network and handle the extension, the fourth algorithm is still better than the other three. Comparing the first three algorithms, Algorithm I and II performed almost identical solving both the original and the extended network. The third algorithm performed slightly better than these two solving the original network but was similar to the other two solving the extended network.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Network</th>
<th># of relations</th>
<th>Average # of neighbours</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.a</td>
<td>Original</td>
<td>279 992</td>
<td>112 per node</td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td>333 808</td>
<td>130 per node</td>
</tr>
<tr>
<td>III.b</td>
<td>Original</td>
<td>280 868</td>
<td>112 per node</td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td>348 578</td>
<td>133 per node</td>
</tr>
<tr>
<td>III.c</td>
<td>Original</td>
<td>281 172</td>
<td>112 per node</td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td>378 437</td>
<td>138 per node</td>
</tr>
<tr>
<td>III.d</td>
<td>Original</td>
<td>280 868</td>
<td>112 per node</td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td>439 947</td>
<td>148 per node</td>
</tr>
</tbody>
</table>

Table 6.4: Test Phase I - Relations and neighbours from the test scenario Network Extension.

#### Test Case III.b

Table 6.5 shows the average number of nodes that were confused after the network had been extended with 2.5 % additional nodes. As seen in the table, Algorithm I and II has clearly the fewest confused nodes after this network extension. In the second place comes Algorithm III with approximately 50 more confused nodes and Algorithm IV has the most confused nodes, approximately 80 more than Algorithm I and II. This is probably due to the reuse of PCIs already present within the network.

Figure 6.4: Test Phase I - Results obtained from the scenario Network Extension, Test Case III.a, with 2.5 % network extension.
6.1. Test Phase I

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average # of confused nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1389</td>
</tr>
<tr>
<td>II</td>
<td>1383</td>
</tr>
<tr>
<td>III</td>
<td>1437</td>
</tr>
<tr>
<td>IV</td>
<td>1469</td>
</tr>
</tbody>
</table>

Table 6.5: Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.a.

Test Case III.b

The result of Test Case III.b is shown in Figure 6.5. As for the previous test case, the forth algorithm is the most effective solving the original network. This time, the forth algorithm is also the most effective solving the extended network with a few PCI changes less than the other algorithms. Algorithm I and II performs again very similar solving both the original and the extended network, but the third algorithm is this time slightly worse. Compared to the previous test case, Algorithm III performs the opposite of what it did in Test Case III.a: it is this time worse than the first two algorithms solving the original network, but slightly better than these two solving the extended network.

![Figure 6.5: Test Phase I - Results obtained from the scenario Network Extension, Test Case III.b, with 5% network extension.](image)

In Table 6.6, the average number of confused nodes is shown. This time, Algorithm II has clearly the fewest confused nodes within its network. Algorithm I and III has relatively equal nodes confused and Algorithm IV is again the one with the most nodes confused, with 100 more confusions than Algorithm II.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average # of confused nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1966</td>
</tr>
<tr>
<td>II</td>
<td>1925</td>
</tr>
<tr>
<td>III</td>
<td>1954</td>
</tr>
<tr>
<td>IV</td>
<td>2026</td>
</tr>
</tbody>
</table>

Table 6.6: Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.b.
Test Case III.c

The third test case once again shows similar results for Algorithm I, II and III, which can be seen in Figure 6.6. This time, the second algorithm is slightly better than Algorithm I and III solving the original network, and the third algorithm is a little bit better than the other two solving the network extension. The fourth algorithm is the one performing the best solving both the original network and the extended network. With the network extension of 10%, the difference between Algorithm IV and the other three is starting to signify.

![Graph showing Test Phase I - Results obtained from the scenario Network Extension, Test Case III.c, with 10% network extension.](image)

As shown in Table 6.7 the difference in number of confusions present within the extended networks is starting to level out. As the number of new nodes and new neighbour relations increases, all of the algorithms are affected with a lot of new confusions to handle, regardless of the reuse of PCIs.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average # of confused nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2510</td>
</tr>
<tr>
<td>II</td>
<td>2490</td>
</tr>
<tr>
<td>III</td>
<td>2477</td>
</tr>
<tr>
<td>IV</td>
<td>2492</td>
</tr>
</tbody>
</table>

Table 6.7: Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.c.

Test Case III.d

In Test Case III.d Algorithm IV is again the one performing the best solving both the original network and the extended network. This time, the difference between Algorithm IV and the other three is even more significant. Algorithm II is the one performing second best solving the original network, but is the least performing solving the extended network. Summarising the results from this test case, Algorithm II does however end up at second place after Algorithm IV.
6.2 Test Phase II

After the result from Test Phase I was extracted, and the defect of reusing the PCIs was observed, the need for a more focused version of the suggested approach was discovered. This version was to be called Algorithm V. In this test phase, this new algorithm will be tested against Algorithm IV to be able to compare the two, and to see if a more narrow focus can make the algorithm perform even better. The test phase will consist of two test scenarios: Neighbour Magnitude and Network Extension, as mentioned in Section 5.2. The first scenario aims to evaluate if Algorithm V still performs equally to Algorithm IV in large networks even though two decision rules have been scoped out. The second scenario will tell if the removal of these two decision rules had the desired effect.

6.2.1 Neighbour Magnitude

As the parameters and network sizes for this scenario were set to the same as in Test Phase I, the resulting neighbour distribution for each test case were almost the same, as can be seen in Figure 6.8.

With a network extension of 20 %, the difference in confused nodes within the extended network is almost eliminated, as shown in Table 6.8. Nearly all of the nodes within the network are experiencing at least one confusion after the additional nodes have been added.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average # of confused nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2948</td>
</tr>
<tr>
<td>II</td>
<td>2953</td>
</tr>
<tr>
<td>III</td>
<td>2946</td>
</tr>
<tr>
<td>IV</td>
<td>2945</td>
</tr>
</tbody>
</table>

Table 6.8: Test Phase I - The number of confused nodes from the test scenario Network Extension, Test Case III.d.
6.2. Test Phase II

Figure 6.8: Test Phase II - The neighbour distribution for each test case in the scenario Neighbour Magnitude.

Figure 6.9 illustrates the result obtained from this scenario. The two algorithms perform almost exactly the same within each of the test cases, which implies that the reduction of decision rules did not affect the ability to solve confusions.

Figure 6.9: Test Phase II - Results obtained from the scenario Neighbour Magnitude.
6.2.2 Network Extension

The result of the network extension can be found in Figure 6.10. As the figure illustrates, the two algorithms do again perform almost exactly the same when solving the original network with 2500 nodes. When the network is extended, some difference is starting to show. With a node increase of 2.5 %, Algorithm V does almost 20 % less PCI changes than Algorithm IV to restore a conflict free distribution. As a larger number of nodes are added to the original network, the percentage difference between the algorithms is however decreased to 7 % when increasing with 5 % and 10 % new nodes, and about 1 % when 20 % new nodes are added.

![Figure 6.10: Test Phase II - Results obtained from the scenario Network Extension.](image)

In Table 6.9 the number of confusions after each node increase is shown. The table reveals that the less PCI changes performed by Algorithm V is not related to fewer confusions to solve since Algorithm V has the most confusions to handle when increasing the network with both 2.5 % and 5 %. Notable is that when increasing the network with 10 %, Algorithm V has fewer confusions to handle, and succeeds in keeping the better performance of 7 %. This was also the case when increasing with 5 %.

<table>
<thead>
<tr>
<th>Increase</th>
<th>Extended</th>
<th>Focused</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 %</td>
<td>1357</td>
<td>1369</td>
</tr>
<tr>
<td>5 %</td>
<td>1885</td>
<td>1913</td>
</tr>
<tr>
<td>10 %</td>
<td>2432</td>
<td>2421</td>
</tr>
<tr>
<td>20 %</td>
<td>2949</td>
<td>2943</td>
</tr>
</tbody>
</table>

Table 6.9: Test Phase II - The number of confused nodes from the test scenario Network Extension.

6.3 Test phase III

The third test phase consists of an evaluation of all of the implemented algorithms against small world graphs. The test phase is divided into two different scenarios: variation of $\beta$
and repetition of $\beta$. The first scenario aims to evaluate how the different algorithms behave for varying graph structures. Therefore, the network size $N$ and the neighbour parameter $K$ will be constant, and the rewrite probability factor $\beta$ is varied. The second scenario aims to evaluate how much the result will vary when running the algorithms on several networks with the same properties. In this scenario, ten different networks will be generated using constant $N$, $K$ and $\beta$.

### 6.3.1 Variation of $\beta$

Figure 6.11 illustrates the different graph structures that are generated using a varied $\beta$. As described in Section 5.2.4, using $\beta = 0$ the graph remains a ring lattice where each node is connected to its $K$ nearest neighbours on each side. Using a $\beta$ between 0 and 1, the relations is randomly relocated with a probability $\beta$. For $\beta = 1$, all relations are randomly relocated, which generates a completely random graph.

![Figure 6.11: Test Phase III - Small world networks using $\beta = 0$, $\beta = 0.15$, $\beta = 0.5$ and $\beta = 1$.](image)

The result from the testing of each network is presented in Figure 6.12. Since the network generated is quite small, the variation between the different algorithms is not that significant. Notable however, is that Algorithm IV and V were really precise in each execution. Both algorithms got the exact same result in almost every re-execution of each network, which is displayed by the narrow confidence interval. In some cases, both algorithms got an interval range of zero. Algorithm I-III generated a more spread result, which is displayed by the wider confidence interval.
6.3. Test phase III

Figure 6.12: Test Phase III - Result of a varied $\beta$.

6.3.2 Repetition of $\beta$

Table 6.10 shows the result for each of the ten networks evaluated. The $\beta$ chosen for the evaluation is $\beta = 0.15$, where the first network is the same as displayed in Figure 6.11. For each algorithm, the minimum number of PCI changes, the maximum number of PCI changes and the average of the ten executions on each network can be read off. As seen in the table, the results are varying, but within a constant range. Algorithm I-III shows the most variation, which is natural due to the implemented randomisation. Notable is however that none of these algorithms does not even once touch the average for Algorithm IV-V with their minimum values. Another notable thing is that Algorithm IV and V does for all ten test cases have a minimum and maximum value which is equal, or at most differ with one PCI change.

<table>
<thead>
<tr>
<th>Test</th>
<th>Standard</th>
<th></th>
<th></th>
<th>Research</th>
<th></th>
<th></th>
<th>Basic</th>
<th></th>
<th></th>
<th>Extended</th>
<th></th>
<th></th>
<th>Focused</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
</tr>
<tr>
<td>1</td>
<td>146</td>
<td>164</td>
<td>154.6</td>
<td>153</td>
<td>161</td>
<td>155.2</td>
<td>150</td>
<td>162</td>
<td>155.2</td>
<td>137</td>
<td>138</td>
<td>137.2</td>
<td>138</td>
</tr>
<tr>
<td>2</td>
<td>157</td>
<td>174</td>
<td>163.5</td>
<td>157</td>
<td>168</td>
<td>162.6</td>
<td>157</td>
<td>169</td>
<td>161.9</td>
<td>138</td>
<td>138</td>
<td>138</td>
<td>138</td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>170</td>
<td>166</td>
<td>168</td>
<td>182</td>
<td>172.4</td>
<td>162</td>
<td>170</td>
<td>166.4</td>
<td>144</td>
<td>145</td>
<td>144.1</td>
<td>144</td>
</tr>
<tr>
<td>4</td>
<td>175</td>
<td>189</td>
<td>181.3</td>
<td>174</td>
<td>190</td>
<td>180.7</td>
<td>171</td>
<td>184</td>
<td>176.6</td>
<td>160</td>
<td>160</td>
<td>160</td>
<td>159.9</td>
</tr>
<tr>
<td>5</td>
<td>147</td>
<td>163</td>
<td>157.3</td>
<td>154</td>
<td>178</td>
<td>165.1</td>
<td>144</td>
<td>158</td>
<td>150.3</td>
<td>130</td>
<td>131</td>
<td>130.9</td>
<td>130</td>
</tr>
<tr>
<td>6</td>
<td>157</td>
<td>174</td>
<td>163.5</td>
<td>157</td>
<td>168</td>
<td>162.6</td>
<td>157</td>
<td>169</td>
<td>161.9</td>
<td>138</td>
<td>138</td>
<td>138</td>
<td>138</td>
</tr>
<tr>
<td>7</td>
<td>163</td>
<td>173</td>
<td>166.7</td>
<td>164</td>
<td>182</td>
<td>171.4</td>
<td>163</td>
<td>175</td>
<td>168.4</td>
<td>148</td>
<td>149</td>
<td>148.3</td>
<td>148</td>
</tr>
<tr>
<td>8</td>
<td>165</td>
<td>184</td>
<td>171.7</td>
<td>170</td>
<td>185</td>
<td>177.3</td>
<td>168</td>
<td>180</td>
<td>171.7</td>
<td>149</td>
<td>150</td>
<td>149.3</td>
<td>149</td>
</tr>
<tr>
<td>9</td>
<td>155</td>
<td>173</td>
<td>162</td>
<td>161</td>
<td>171</td>
<td>164.9</td>
<td>156</td>
<td>166</td>
<td>162.2</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>10</td>
<td>157</td>
<td>174</td>
<td>163.5</td>
<td>157</td>
<td>168</td>
<td>162.7</td>
<td>157</td>
<td>169</td>
<td>161.9</td>
<td>138</td>
<td>138</td>
<td>138</td>
<td>138</td>
</tr>
</tbody>
</table>

Table 6.10: Test Phase III - Test results when using ten different small world networks with $\beta = 0.15$. 
7 Discussion

In this chapter, the work performed will be discussed. This chapter initiates with a discussion regarding the results from the three test phases. This chapter continues with a discussion of the implemented algorithms where Algorithm IV is the one to be most highlighted. This discussion includes its performance, the complexity of the algorithm and the network trade-off. The next section discuss the defined optimisation models followed by a discussion about the test environment and alternative evaluation possibilities. The chapter terminates with a discussion about the method used and the work in a wider context.

7.1 Results

In this section, the result from the three test phases are discussed. The section is divided into the subsections, one for each test phase, where the scenarios considered in each phase is commented.

7.1.1 Test Phase I

Before the formal testing was made, the code was tested several times to make sure that the code had no errors and to estimate how time consuming the "real" testing would be. At this point, smaller test cases were considered which resulted in quite similar results between the different algorithms. For a network with 1000 nodes using the same distances and probabilities as in the real testing, the improvement using Algorithm IV compared to the others was about 11%. Using smaller test cases, the diverseness of the result is naturally quite low, which resulted in modest expectations for the real testing.

PCI Restriction

The result of the first test scenario was a little better than expected based on the results from the pre-testing. As seen in Figure 6.1, Algorithm IV is 16-20% more effective than the other algorithms through the entire test scenario, which was better than the results obtained from the smaller network testing. Obtaining these results for a network size of 2000 nodes, the expectations for the second test scenario, Neighbour Magnitude, were increased.
Notable from this test scenario was that Algorithm III did not perform any better than Algorithm I and II. The expectations were that this algorithm would be not as effective as Algorithm IV, but at least better than Algorithm I and II. This was unfortunately not the case, especially not in the third test case with a restriction of 300 PCIs, where Algorithm III turned out to be the least effective of them all. Looking at all the test cases together, there is no indication that Algorithm III should be the worst algorithm, rather that the bad performance in the third test case may be caused by unfavourable randomisation.

The PCI range could have been narrowed even more to see how the algorithms would perform if the range was even more restricted. However, as one of the limitations of the work is to look at feasible networks, and a too narrow PCI range makes it impossible to solve the problem, further restrictions were scoped out. Even if the algorithms would manage to gain solutions (infeasible solutions, but still solutions), it would be difficult to draw conclusions from the result.

**Neighbour Magnitude**

The second test scenario ended up with much better results than expected. Algorithm IV turned out to be increasingly more effective than the other algorithms when the number of nodes and neighbours per node increased. The performance of Algorithm III was again a disappointment. The result for Algorithm II and III in the third test case is however quite misleading, since both of them failed in two out of ten executions trying to solve the network with 3500 nodes.

The promising performance from Algorithm IV is believed to be a result from the decision to prioritise a node which is causing more than one confusion. Since several confusions are solved at once, the number of confusions within the network will decrease faster, which leads to the need for fewer PCI changes. The reusing of PCIs already present within the network is believed to be of less importance here.

The network size was delimited to the maximum of 3500 nodes because of the behaviour of Algorithm II and III. When all of the confusions are solved by each algorithm, the result can easily be compared between the different algorithms, but when one or several algorithms fail to get feasible solutions, the comparison gets tougher. The neighbour distribution shown in Figures 6.2 and 6.8 shows that the networks used in this scenario were quite dense, with several cells containing over 200 neighbours in the largest test case. Both Algorithms I and IV were tested against a network size of 4000 nodes, but at this size both of the algorithms failed in most of the executions. The failure is due to the large amount of neighbours and neighbours’ neighbours which makes it hard to find possible PCIs. For this network size, the average number of neighbours was 181. The result from this testing was judged to be too futile to be part of the report. The process of iterating this input size was moreover really time consuming.

**Network Extension**

The result of the third test scenario was not entirely as good as hoped, but was not unexpected. Since Algorithm IV is implemented to reuse PCIs within its sub network when possible, this naturally risk to lead to more conflicts when the network is expanded with randomised nodes. Looking at Test Case III.a, Algorithm IV is the one making the most PCI changes after the network has been expanded, but it is also the one having the most confusions to solve. Algorithm IV does 6-9 PCI changes more than the other algorithms, but also has 32-86 more confusions to solve. This implies that Algorithm IV does not need that many more PCI changes to solve more confusions than the rest of the algorithms. Looking at Test
Case III.b, Algorithm IV is as effective as the other algorithms, even a bit better, but still has more confusions to solve. In Test Case III.c and III.d, the effectiveness of Algorithm IV is increasing compared to the other algorithms at the same time as the difference in amount of confusions is levelling out. At a network extension at 20% and higher, the number of new nodes and new relations reach the point where the reuse effect can no longer be reflected in the number of nodes that are confused.

### 7.1.2 Test Phase II

Due to the results seen in the third test scenario in Test Phase I, the idea of a fifth algorithm was born. The idea was that this algorithm should be less affected by the randomness of position and PCI of the added nodes in the network expansion by scoping out the PCI reuse criteria in the decision rules.

**Network Magnitude**

As mentioned in Section 6.2, this scenario was used to verify that Algorithm V would be as effective as Algorithm IV when it comes to minimising the number of PCI changes performed. As the result showed, the performance of the two algorithms were almost identical. Since the decision rule which premiered a PCI change of a node which caused at least two confusions, this result came as no surprise.

**Network Extension**

When extending the network, Algorithm V performed better than Algorithm IV in every test case, but the performance was not outstanding. As seen in Figure 6.10 there are no difference in PCI changes when solving the original network, and when solving the extended one, the difference is minimal. As mentioned in Section 6.2.2, Algorithm V does have more confusions to solve after the extension in half of the test cases. This implies that the PCI reuse performed by Algorithm IV does not affect the number of confused nodes after the extension that dramatically after all.

### 7.1.3 Test Phase III

Test Phase III included several small world networks, which is a network structure used within several research areas. The structure of these small world networks is however not very typical in a realistic LTE cell networks, but the structure is rather interesting. The resulting connection topology eventuate somewhere between completely regular and completely random, which is more applicable to technical as well as biological and social networks [34]. But as a consequence of the non-typical structure, the goal of this phase was not to evaluate a realistic scenario, but rather to investigate the overall performance of the five algorithms.

For the entire test phase, the parameters $N$ and $K$ were initiated to $N = 500$ and $K = 25$, which resulted in a network of 500 nodes with an average of $2 \times K = 50$ neighbours. The network may seem quite small compared to the size of the networks tested in the previous two test phases, but due to the structure of a small world graph, the network gets unfeasible for quite small parameter values. Recall from Section 5.2.4 that a Watts-Strogatz small world graph has high clustering and a small shortest path length. The neighbour distribution for a graph with $N = 500$, $K = 25$ and $\beta = 0.15$ is shown in Figure 7.1 (a). In Figure 7.1 (b), the distribution of neighbours’ neighbours for the same graph is shown. Since a feasible solution for the PCI allocation assume that no cell has the same PCI as any of its neighbours or neighbours’ neighbours, it is easy to see why larger small world networks cannot be used if a feasible solution should be possible. As the network size increases, the number of neighbours’ neighbours will make it impossible to obtain a feasible solution.
Since the correlation between the small world graphs and a realistic cell network is not that significant, it is a bit tricky to draw unbiased conclusions from the result. For smaller values of $\beta$, Algorithm IV and V were clearly the most effective algorithms, but when $\beta$ increases, the difference was evened out. However, for all the tested values of $\beta$, Algorithm IV and V were very stable, whilst the other algorithms delivered a wider range of PCI changes.

7.2 The Algorithms

In this section, each algorithm is discussed further regarding expectations and performance. The discussion regarding Algorithm IV is extended with discussion about the decision rules, the complexity and the network trade-off.

7.2.1 Algorithm I - Standard

Theoretically, Algorithm I is the most unpredictable of all of the implemented algorithms. As described in Section 4.3.1, the algorithm does nothing to influence the decision of which PCI should be chosen. Instead, the algorithm just picks a valid PCI at random. Looking at the results however, the confidence intervals seen in Figure 6.1 and 6.3 indicates that Algorithm I is no more unpredictable than neither Algorithm II nor Algorithm III. In some of the cases, Algorithm I even has the narrowest interval among these three. It was expected that Algorithm I should be one of the least performing algorithms, but the expectation was that the interval would be broader than the rest of the algorithms. If the test cases were run more than ten times each, perhaps the result would have turned out like the expectations.

7.2.2 Algorithm II - Research

As the result indicates, Algorithm II is one of the least performing among the five implemented algorithms. This is however not that surprising. The large number of PCI changes needed to solve all confusions, and the two cases of failing to solve the network can be explained by two key factors. Firstly, the article by Amirijoo et al. [14], where the inspiration has come from was published in 2008, which means that the method was created no later than this year. At 2008, the traffic situation was a lot different than it is today, and the networks were far less dense. At this point, networks so dense that all the available PCIs were to be occupied by the neighbours and neighbours’ neighbours of one cell was hardly imaginable. For that reason, the method might not be developed for networks as dense as the
one tested in this work. Secondly, the article does not specify how these denser scenarios should be handled. When confusion occurs, the confused cell command the neighbouring cell with the lowest CGI to change its PCI. If the cell has no available PCIs, nothing happens, and the confusion remains. At the next iteration, the exact same scenario may occur, and the outcome will be identical. This means that the neighbouring cell with the higher CGI might have available PCI candidates, but is never ordered to change its PCI. Because the article did not specify how this should be handled, the functionality is not implemented in Algorithm II. For this reason, the confusions left unsolved when tested on a graph with 3500 cells could possibly be solved if the cell with the higher CGI was ordered to change its PCI.

7.2.3 Algorithm III - Basic

Whilst the mediocre performance of Algorithm I and II were somewhat expected, the mediocre performance of algorithm III was more unexpected. Even though the algorithm is implemented with more randomness than Algorithm IV and V, it was expected to be at least closer to the results that were seen for these algorithms. The result obtained does however confirm the effectiveness of the decision rules used by Algorithm IV and V. The implementation of Algorithm III-V has the exact same foundation and the only thing separating them from each other are the decision rules. Because of this, one can clearly see that if some extra computational effort is performed by the confused cell or eNB, there are great network improvements to be expected.

7.2.4 Algorithm IV - Extended

Since Algorithm IV is the main character in this work, and fully represent the suggested approach, this algorithm will be discussed in depth. Interesting topics to discuss are the implemented decision rules, the complexity of the algorithm and the network trade-off which the additive information passing induce.

The Decision Rules

The result of the testing is most definitely affected by the decision rules which were used for the choice of PCI and which confusion-causing cell to be the one to change its PCI. However, the decision rules stated in Section 4.2 might be incomplete and the order of which they effect the decision can be discussed. The rules were created to prevent two of the main problems when allocating PCIs in today’s dense networks: there are too many PCI changes and it is hard to find a PCI suitable for a cell in confusion due to the denseness. The most prioritised rule in the implementation was rule number 2 (to solve more than one PCI confusion at the same time if possible). The idea of this rule was to match the desired lowering of the number of PCI changes performed within the network. Rule three and four were added to try to prevent the event of no available PCIs for as long as possible. The problem with these two types of decision rules implemented within the same algorithm is that the algorithm strives to minimise multiple types of parameters at the same time, which could have a negative effect for both of the parameters. By dividing the algorithm into two algorithms with more well defined purpose may be a way to obtain even better results. Since the result in this work is presented by number of PCI changes, the natural choice here was to continue to develop rule two and let go of rule three and four. This led to the implementation of Algorithm V.

Another thing justifying the choice of moving forward with the PCI changes instead of PCI utilisation is that is is not obvious that utilisation is the best way to go. The advantage of reusing the PCIs is that one are able to “save” unused values for the future when other possibilities might be missing. Also, if the PCIs used within a sub network is reduced, the number of possible choices are hopefully increased. However, since the cell networks are
7.2. The Algorithms

not static, the probability for new conflicts are increased since adjacent cells not already
neighbours might discover each other. With high reuse rate, the possibility of a new conflict
is higher than if the entire PCI range is used. The choice is consequently between having to
resolve conflicts more frequently or to risk ending up with a less structured network where
no valid PCIs can be found. To be able to determine if the utilisation of PCIs is a proper rule
to include, the algorithm needs to be evaluated in a more realistic environment where the
dynamism of the network is clearly simulated. Without proper simulation of dynamism and
the networks over-time behaviour, no confident statement can be made.

An alternative to choosing the cell causing at least two confusions is to choose the cell
causing the highest number of confusions, using the PCI which it is currently associated with.
This method would however increase the amount of data to be sent from the confusion-
causing cells to the confused cell, or increase the computational costs for this cell. In the
current implementation, the data that needs to be sent from the confusion-causing cell to
the confused cell is two binary vectors of length 504, one representing the PCIs used by the
confusion-causing cells neighbours and one representing the PCIs used by its neighbours’
neighbours. If the algorithm should be able to keep track of the number of confusions that are
caused by the cell, the neighbours’ neighbour vector could be linked so that the occurrence
of each PCI was attached. Alternatively, an additional variable representing the number of
times the current PCI of the cell is present within its sub network can be attached. Another
method would be to send a numerical vector instead of a binary, where the position in the
vector represent the PCI and the number on the position represents the occurrence. All of
these alternative does however imply additive computational complexity to the confusion-
cauSing cell. Since a lot of the cell’s neighbours will share neighbours, the cell needs to sort
out the duplicates somehow, which a binary representation avoids.

The Complexity

The time complexity of the extended algorithm is $O(n^2)$, which will be motivated in this
section. When a confusion occurs, the confused cell requests a vector of neighbours and
a vector of neighbours’ neighbours from each of the confusion-causing cells. If each of
the confusion-causing cells has an average of $n$ neighbours, each cell will need to make $n$
neighbour list requests, one for each of its neighbours. To create the vectors, the cell needs
to iterate through each neighbour list acquired one time (which has an average length $n$).
By initially creating two all zero vectors of length 504, the modification of the vector can
be done in linear time in each of the iterations. The total complexity of these operations
will be $O(n^2)$. Since the confusion is handled pairwise, these operations will happen twice,
i.e. $2 \times n^2 \in O(n^2)$ and will therefore not cause any extra complexity. When the confused
cell receives the requested vectors from the confusion-causing cells, some computational
preparatory work needs to be performed before the decision rules can be applied. This work
consist of merging two vectors (to obtain information about available PCIs) and comparing
two vectors (for example to know which of the cells has the most available PCIs). Since the
vectors are sorted, and when manipulated have a maximum length of 504, the complexity
of these types of operations is a constant, which would be $2 \times 504$ in a worst case scenario.
Using the prepared data, the decision of which cell should change its PCI, and to which
PCI this should be can be handled by if/else statements. Operations like these have linear
complexity and will not affect the total complexity. Summarised, the total complexity of
the entire algorithm would as stated be $O(n^2)$ since all of the operations performed by the
confused cell will not affect the total complexity.

The implementation of the algorithm should however not be associated with this complexity.
The implementation does in a centralised way use the suggested, distributed algorithm to
solve all the confusions present within the entire network. The algorithm is not intended to
be implemented like this in a real scenario. The real implementation would run the algorithm one time to solve one specific confusion in a sub network. The centralised implementation used in this work is therefore produced for illustrative purposes only, as this implementation gives an overview of how well the algorithm would perform in a larger network. It also makes it easier to get an estimation of how the different distributed algorithms performs compared to each other.

**The Network Trade-off**

Algorithm IV performed better than Algorithm I in almost every test case, but Algorithm I has several beneficial properties which will be discussed in this section. Firstly, Algorithm I is naturally faster since it does only consider the neighbours and neighbours' neighbours of one cell, an then choose an available new PCI value at random. Complexity wise, Algorithm I is however equal to Algorithm II. Since a cell needs to extract its neighbours' neighbours for each of its own neighbours, the complexity is $O(n^2)$ (if the average number of neighbours per cell is $n$, as assumed earlier). Time wise, however, this algorithm could be implemented more efficiently, since it performs a lot less operations.

Another advantage of Algorithm I is that it could be implemented and used today. This algorithm is applicable with the standard, and is therefore possible to get in use quite fast. Algorithm IV on the other hand is not directly applicable with today's standard. Firstly, to let the confusion solving decisions be handled by the confused cell is not part of the standard. The standard suggest that this should be handled by the cell which is causing the confusion. Secondly, Algorithm IV includes information passing between the cells, i.e. the neighbour vectors. Information passing between the cells does occur today, but not in the event of solving conflicts.

The extended information passing between the cells in Algorithm IV will of course affect the network traffic, since all sort of additional traffic will burden the network. If all of the cells are to be constantly updated, this additional traffic could be really costly. It is also memory costly if each cell should store all of this information, especially for large, dense networks. The suggestion is that the cells should not be constantly updated, and the cells should not be burdened with all of this extra information. The suggestion is that this information should be requested when needed and erased from the confused cell directly after the algorithm has finished. This would eventually lead to larger information packages to be sent when needed, and if the network is quite stable, retransmission of the same information each time a confusion occurs. However, the overall network weighting will probably be smaller.

**7.2.5 Algorithm V - Focused**

The goal of Algorithm V was as mentioned in Section 7.2.4 to create a more single-tracked algorithm where the number of PCI changes were in focus. It was natural to keep the rule about solving multiple confusions, rule two, but the rule regarding choosing the cell with the longest list of available PCIs, rule one, was kept as well. This rule is probably more related to rule three and four, but was still kept with rule two in the more focused algorithm. Looking at large, dense networks where the number of confusions are high, the possibility that both of the cells in a confusion also is part of another confusion is quite high. In this case, rule number one could be the conclusive factor to which of the cells that should make the PCI change. An alternative method to this was discussed in Section 7.2.4.

Algorithm V did perform somewhat better than Algorithm IV in terms of PCI changes in Test Phase II, but the implementation of the node adding function is a little unfavourable to Algorithm IV. As the algorithm tries to plan for the future by saving PCI values, the
addition of cells with randomised PCIs seems badly thought through. If the algorithm was to be implemented in reality, the smart thing to do would be to take advantage of the utilisation planning and allocate a valid PCI to a cell already in the configuration phase. With this in mind, the fact that Algorithm V only lowers the number of PCI changes by a few percent may interpret that the difference between the algorithms is negligible. If a more proper estimation should be made, the testing needs to be extended with more fair conditions.

7.3 The Optimisation Models

Studying the related articles, no optimisation models like the ones defined in this work have been found. In other studies, the goal has mostly been to obtain a valid solution, not an optimal solution. Also, the goal has not been to minimise the number of PCI changes performed. The goal of utilising the PCIs within the network has however been studied in other articles, for example by Bandh et al. in [23] where PCIs of a distance three from a conflicting cell is reused if possible.

7.3.1 Model Modifications

In this work, there were two different optimisation models defined which assumed that the input data came from a solvable network. As both models contains the condition that no cell may have the same PCI as one of its neighbours or neighbours’ neighbours, a network with no feasible solution will be unsolvable by the optimisation models. To be able to return a solution for unfeasible networks, this condition needs to be modified so that it is permitted for one cell to have the same PCI as one of its neighbours or neighbours’ neighbours, but it should be costly to have it. This could for example be handled by adding a new variable $v_{ki}$ where $v_{ki} \leq x_{ki}$ for all $i, k$ and $v_{ki} \in [0, 1]$. The condition would be rewritten as:

$$x_{ki} + \sum_{j \in M_i} x_{kj} - \sum_{j \in M_i} v_{kj} \leq 1 \quad \forall i, k.$$ (7.1)

Here, if no feasible solution is to be found, the condition can still hold. To avoid using the variable $v_{ki}$ and by that avoiding an unfeasible solution, the sum of all variables $v_{ki}$ will be added to the objective function multiplied with a large constant. Now, a conflicting PCI will only be chosen if there are no other possibilities.

There are of course other interesting models that could have been created as well. As mentioned in Section 3.5, one possibility could have been to create a combination of the two models to obtain one model which consider both the number of PCI changes and the number of PCIs used in the network. Another possibility would be a model which optimised networks with no feasible solution, i.e. solved the conflicts in a prioritised order based on their impact of the network. This model would however look a lot different than the two defined in this work, and would be hard to implement in a binary manner. If the conflicts should be prioritised, the decision should probably be based on relevant parameters such as traffic data.

7.3.2 Execution Possibilities

With access to a more powerful computer, the optimisation models could have been used to generate the best possible solutions in terms of PCI utilisation and PCI changes. These solutions could have been compared to the results obtained from executing the distributed algorithms, which would give a real estimation regarding how well the algorithms actually performed, not only in relation to each other. As described in Section 3.5, the number of variables and conditions gets rather large even for smaller input data. This makes the execution hard on a regular computer. Also, as the results indicates in Section 6.1.2, the difference
between the algorithms is reduced when the input size decreases. This implies that even if Algorithm IV and V would perform closely to optimum for small input data, so would Algorithm I-III, and the result would be meaningless. For an algorithm to be well performing for smaller test cases is just an indication that it perhaps will perform well for larger input data, it is not a guarantee. This can clearly be seen in Figure 6.3 where the difference between the results of Algorithm IV compared to the rest gets better and better as the input size increases. For the solution obtained using the optimisation models to be really interesting, the input size would have to match at least the input sizes of the test cases, else the result would not contribute to any conclusions.

One possibility could however be to execute the optimisation models to a larger sized input data but with a reduced number of variables. To do so, part of the variables could be fixed parameters, and the optimisation models would then return the optimal solution for a smaller sub network. Using this method, there is however no guarantee that the solutions obtained for the smaller sub networks can be combined to an optimal solution for the whole network. This matches the behaviour of a distributed algorithm. By choosing to retain variables matching the ones that the implemented algorithms are using as a decision basis, the optimisation models could mimic the behaviour of the different algorithms.

7.4 The Test Environment

From the beginning, the plan was to test the algorithms in both a randomised simulation environment and an environment constructed from network data. To construct an environment from network data was however harder than expected. The data received was on impractical form, implying that the data needed to be manipulated to be useful. Furthermore, the amount of data needed to create a small network of 249 nodes was huge. To be able to create networks of equal sizes as the ones evaluated in the test phase, the data amount needed would be enormous.

7.4.1 Pros and Cons

The randomised simulation environment implemented in MATLAB was quite easy to work with, but of course has its flaws. Firstly, the network that was plotted by MATLAB got messy for relatively small networks, as seen in Figure 5.2. The run time was also heavily increased by the plotting functionality. Secondly, the structure of the environment made it hard to simulate the dynamism of the cell network. In the third test scenario, Network Extension, the extension was only performed one time on each network, since further extensions would make it difficult to guarantee an equal network initiation for each algorithm in every execution. Even with one extension, the manual work with temporary parameters and small adjustments in the implementation between each execution needed to be made. If some kind of network simulation tool was used, it might have been easier to extend the network.

One advantage was however, opposed to the data constructed networks, that it was easy to vary the network properties and to create large networks. As described in Section 5.1, the environment was implemented with several pre-definable parameters to be set before the network was built. Also, it was easy to add additional functionality, for example counting the total number of relations in the network or the number of confused cells. If a network simulation tool was used, the possibility to extract the specific information wanted might have been harder. This is also implied by the related works studies, since the simulation environments used are self constructed.

One should also have in mind that in a real scenario, the entire cell network does not contain randomised PCIs, as in the test environment, but rather the major part of the net-
work is well-planned. Because of this, the number of confusions that needs to be solved are much less, and the number of possible PCIs for each cell are perhaps larger. This indicates that if the algorithms were used in a real cell network, they could probably handle denser networks and still obtain feasible solutions.

7.4.2 Alternative Methods

Another possible way to create a cell network model is to randomly locate disks with different diameter within a predefined area. The sizes of the disks could match the different cell types available, and the neighbour relations could be based on the overlap between the disks. Alternatively, a relation could be established if the centre point of one disk was in the area of another, which could be a method to also establish asymmetric relations. An illustrative example of a disk based network is shown in Figure 7.2.

![Network constructed of different sized disks.](image)

Figure 7.2: Network constructed of different sized disks.

The test environment is implemented with a 90% chance of a symmetric relation between node \( j \) and \( i \) if a relation between \( i \) and \( j \) has been established. This property was added to make the environment more realistic as the cell relations can be unidirectional, as mentioned in Section 2.2.1. Looking at the related work, the majority of the test environments used were implemented containing only symmetrical relations. As the user movement are in most of the cases multi-directional, the environment could perhaps have been implemented containing only symmetric relations. This would have simplified the implementation and reduced the number of special cases caused by the asymmetry. The asymmetry is however believed to have a minor effect on the result, due to the high probability of symmetric relations. If the probability of an asymmetric relation was higher, i.e. if the probability of a symmetric relation was lower, the result would probably be more affected.

7.5 Method

As discussed in Section 5.3, the validity and reliability is strengthened by executing each algorithm multiple times and give the algorithms as equal initiation as possible in each network setting. But since there are randomness included in the implementations, the exact outcome when replicating the study would be hard to achieve. However, the result would with a 95% probability be within the confidence intervals which is shown in Chapter 6. For the third test scenario, Network Extension, where the network was extended with additional nodes, the confidence intervals were omitted.
Both the test environment and the algorithms could probably be implemented more efficiently than they were in this work. If the method was to be replicated, the implementations could probably be improved to run faster. As the main focus of the implementation was to test the algorithms rather than to create perfectly written MATLAB code, no time was spent on optimising the code. The running time of the implementation is also not connected to the running time of the distributed algorithm, as discussed in Section 7.2.4.

When studying related work on the subject of PCI allocation, articles of varying quality were part of the research. As there was not that much published within the subject and most of the articles read referred to each other, the choice was to include even the articles with few citations. By doing so, the inspiration gained were greater than if only the more well cited articles were considered. But since the quality of these articles may be arguable, no important conclusions or statements have been made using these as a basis. As for the theory chapter, the basis for the information has been books issued by well known publication companies, standard documentation by 3GPP and internal product information from the company part of this project. As the development of the LTE network is a constantly ongoing process, the consideration when choosing literature was if to use newly published books and articles which were rather unrevised or to use more revised literature which might not be up to date. The former were chosen to make sure that the information used in this thesis was up-to-date.

7.6 The Work in a Wider Context

As the algorithms are using network information, which cannot be associated with any of its users, the introduction of any of the developed algorithms would not harm the integrity of the user. The technique introduced in Algorithms IV and V could however be beneficial for the users of the network, as the minimised number of PCI changes results in fewer cell reboots and therefore increases the availability of the network. Also, a well-planned network implies fewer call drops and less handover failures, which are both key factors for user satisfaction.

The algorithms developed are suitable as SON functions, as the only input needed is data already present within the network. When a confusion is detected, one of the algorithms could be used as part of the self-healing process to solve the conflict. This will help to reduce the manual labour, as the network is able to heal itself without any additional workload.

The problem of allocating proper PCIs to different cells in the LTE network can be compared to several other allocation problems. The problem has similarities with allocation of different resources, for example allocating employees on a company. As the PCI allocation problem has the special properties where no neighbours or neighbours’ neighbours may share the same PCI, the algorithms may not be directly applicable to other allocation problems. But with minor modifications, the algorithms could be suitable for other allocation problems as well. Looking at the example stated above, the allocation of employees can be formulated as a graph by letting the nodes represent the employees. The PCIs (or colours, if reducing from the vertex colouring problem) could be substituted by different teams or projects, and an edge could be established between two nodes if the two corresponding employees should not be in the same team (or project). The goal here may be to minimise the number of teams or adjust non-functional teams by letting as few employees change their team as possible. Given this structure, the algorithms are able to solve different allocation problems, if the properties are adjusted to suit the specific problem.
The work performed generated five different distributed algorithms. It also generated two optimisation models and a promising approach for minimising the number of PCI changes.

The study of related work generated a broad spectrum of ideas for the approach developed in this work. There were both centralised and distributed approaches investigated, and the most commonly used method was to formulate the PCI allocation problem on graphical form. As most of the articles tried to find a valid solution rather than find the optimal one, two optimisation models were developed where the number of PCI changes and the number of used PCIs were minimised, respectively.

The 3GPP standard supports a distributed algorithm where a confusion-causing cell gets to choose from a spectrum of PCIs derived from the O&M. This property matches the implementation of Algorithm I. The idea of Algorithm II was derived from work performed by Amirjoo et al. [14], where the command of a PCI change is derived from the confused cell. Algorithms III-V reuse the idea of the confused cell as a decision maker, but adds a larger decision basis. In addition, Algorithms IV and V, try to mimic the behaviour of the optimisation models. Algorithms II-V are however not directly applicable to today’s standard as all of these algorithms transfer the conflict resolution from the confusion-causing cells to the confused cell. Also, Algorithms III-V suggest extra signalling between the cells involved in the conflict.

The results showed that improved results can be achieved if relaxing the limitations of the network standard, as per Algorithms II-V. However, as Algorithm III performed similarly to Algorithm I and II, the enlarged decision basis is not enough to obtain better results. To lower the number of PCI changes, the decision rules were proven really effective, especially for larger and denser networks. The minimised randomness seen in Algorithms IV and V also generated a more stable result.

8.1 Future Work

For future work, the decision rules needs to be further evaluated. The evaluation should include if or if not all of the decision rules should be part of the algorithm, if additional rules
8.1. Future Work

should be considered and in which order the rules should be prioritised to generate the best solution. Further, the evaluation of the conclusive algorithm needs to be performed in larger scale, where the dynamism of the cell network is properly simulated, and multiple frequency layers are added. Also, if the algorithm should be used in reality, some changes in the LTE network standard needs to be made.

Apart from this, it would be interesting to take a closer look at the optimisation models defined in Section 3.5. Using a computer with higher capacity, the models could possibly be run against realistic test scenarios with a large number of cells. By doing this, there would be a chance to see how well the algorithms performed compared to an optimal solution, not only compared to each other. This would reveal if there are still possible improvements to be made or if the distributed algorithms perform close enough to an optimal centralised solution.
Bibliography


