

AMERICAN UNIVERSITY OF BEIRUT

TABU SEARCH BASED APPROACH TO SOLVE THE TAS
RECONFIGURATION PROBLEM IN LTE NETWORKS

by
NADINE NABIH AHMAD

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

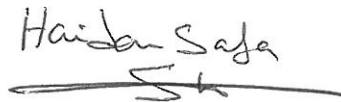
Beirut, Lebanon
January 2015

AMERICAN UNIVERSITY OF BEIRUT

TABU SEARCH BASED APPROACH TO SOLVE THE TAS
RECONFIGURATION PROBLEM IN LTE NETWORKS

by
NADINE AHMAD

Approved by:



Dr. Haidar Safa, Associate Professor
Computer Science

Advisor

Dr. Wassim El Hajj, Chairperson
Computer Science



Member of Committee

Dr. Hassan Artail, Professor
Electrical and Computer Engineering



Member of Committee

Date of thesis/dissertation defense: January 23, 2015

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Ahmad Nadine Nabih
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

Nadine 6-Feb-2015
Signature Date

ACKNOWLEDGMENTS

First and foremost, my special thanks to my advisor Dr. Haidar Safa for his time, supervision, and continuous support throughout the thesis. Working with Dr.Safa has always been smooth and encouraging which lead greatly in accelerating the completion of the thesis.

I would also like to thank my thesis committee members, Dr. Wassim El Hajj and Dr. Hassan Artail, whose assistance, guidance and constructive criticism had made this dissertation possible. Thank you all for your time and assistance.

Last but not least, I am grateful for being surrounded by a caring family, a loving fiancé and great friends. I especially thank them for their invaluable support and understanding throughout this time.

AN ABSTRACT OF THE THESIS OF

Nadine Ahmad for Master of Science
Major: Computer Science

Title: Tabu Search Based Approach to Solve the TAs Reconfiguration Problem in LTE Networks

Designing tracking areas (TAs) in LTE networks affects the signaling cost of user equipment (UE) during mobility. Indeed, when the latter is called, the mobility management entity (MME), which records the TA in which the UE is registered, broadcasts a paging message in the UE's TA to determine the exact cell through which the call should be delivered. When a UE moves to a new TA, a TA update (TAU) might be performed either to update the MME or the home subscriber server (HSS), which is a database maintaining the subscriber's profile, depending on whether the move is intra- or inter-MME. TA update and paging result in a significant amount of signaling overhead. The TA design is usually optimized during the network planning phase. However, the mobility of UEs might turn the initial TA configuration inefficient requiring TA reconfiguration. In this thesis, we use the tabu search (TS) heuristic to solve TAs reconfiguration problem in LTE networks taking into consideration both signaling overhead and reconfiguration cost. We compare our solution with a variation of the genetic algorithm (GA) based solution found in the literature and with the optimal solution obtained by formulating the problem using an Integer Programming Model (IPM) and implementing it using Gurobi optimizer.

ABBREVIATIONS

2G: Second Generation Wireless

3G: Third Generation Wireless

3GPP: 3rd Generation Partnership Project

4G: Fourth Generation Wireless

APN: Access Point Name

AS: Access Stratum

CA: Carrier Aggregation

CC: Component Carrier

CS: Circuit Switched

E-UTRAN: Evolved UTRAN

ECGI: E-UTRAN cell global identifier

ECI: E-UTRAN cell identity

ECM: EPS Connection Management

EDGE: Enhanced Data Rates for GSM Evolution

EMM: EPS Mobility Management

eNB: Evolved Node B

EPC: Evolved Packet Core

EPS: Evolved Packet System

GA: Genetic Algorithm

GERAN: GSM EDGE Radio Access Network

GPRS: General Packet Radio Service

GSM: Global System for Mobile Communications

GTP-C: GPRS Tunneling Protocol Control Part

GUMMEI: Globally Unique MME Identifier

GUTI: Globally Unique Temporary Identity

HSS: Home Subscriber Server

IMEI: International Mobile Equipment Identity

IMSI: International Mobile Subscriber Identity

IP: Internet Protocol

LTE: Long Term Evolution

M-TMSI: M Temporary Mobile Subscriber Identity

MCC: Mobile Country Code

MME: Mobility Management Entity

MMEC: MME Code

MMEI: MME Identifier

MNC: Mobile Network Code

NAS: Non-Access Stratum

OFDM: Orthogonal Frequency Division Multiplexing

P-GW: Packet Data Network Gateway

PDN: Packet Data Network

PLMN-ID: Public Land Mobile Network Identity

PS: Packet Switched

PSTN: Public Switched Telephone Network

RNC: Radio Network Controller

RRC: Radio Resource Control

S-eNB: Source eNB

S-GW: Serving Gateway

S-TMSI: S Temporary Mobile Subscriber Identity

SAE: System Architecture Evolution

SC-FDMA: Single Carrier Frequency Division Multiple Access

S1-AP: S1 application protocol

T-eNB: Target eNB

TA: Tracking Area

TAC: Tracking Area Code

TAI: Tracking Area Identity

TAL: Tracking Area List

TAU: Tracking Area Update

TS: Tabu Search

UE: User Equipment

UICC: Universal Integrated Circuit Card

UMTS: Universal Mobile Telecommunications System

UMTS: Universal Mobile Telecommunication System

USIM: Universal Subscriber Identity Module

UTRAN: UMTS Terrestrial Access Network

VoIP: Voice Over IP

CONTENTS

ACKNOWLEDGMENTS.....	IX
ABSTRACT	X
LIST OF ABBREVIATIONS	XI
LIST OF ILLUSTRATIONS	XIX
LIST OF TABLES	XXII

Chapter

1. INTRODUCTION	1
1.1 UMTS and GSM	1
1.2 LTE Main Motivations	2
1.3 LTE Overview	3
1.4 Thesis motivation	4
1.5 Problem definition	6
1.6 Objectives	7
1.7 Thesis plan	8
2. LTE	9
2.1 LTE Architecture	9

2.1.1	The User Equipment (UE)	10
2.1.2	The evolved UMTS terrestrial radio access network (E-UTRAN)	11
2.1.3	The evolved packet core (EPC)	12
2.1.3.1	The home subscriber server (HSS).....	12
2.1.3.2	The packet data network gateway (P-GW) ...	12
2.1.3.3	The serving gateway (S-GW).....	13
2.1.3.4	The mobility management entity (MME).....	13
2.2	Interfaces	14
2.3	Cells, TAs and TALs.....	15
2.4	Component Identification	17
2.5	LTE to LTE-Advanced	20
2.6	Mobility Management in LTE	22
2.6.1	Communication protocols	24
2.6.2	LTE Mobility and connection states	26
2.6.3	Tunnelling	27
2.6.4	Paging	28
2.6.5	Tracking Area Update	29
2.6.6	Handover	33
2.6.6.1	Intra-LTE handover	34
2.6.6.2	Inter-LTE handover	35
2.6.6.3	Inter-RAT handover	35
2.7	Problems related to mobility	35
3.	RELATED WORK.....	37
3.1	A dynamic paging scheme for long term evolution mobility management [10]	37
3.2	An investigation on LTE mobility management [6]	38
3.3	Reducing signaling overhead for femtocell/macrocell networks [12]	39
3.4	Exploiting Tracking Area List Concept in LTE Networks [13]	40

3.5 Performance and cost trade-off in Tracking Area Reconfiguration: A pareto-optimization approach [14].....	41
3.5.1 Genetic Algorithm.....	43
3.5.2 Integer programming model (IPM).....	47
3.5.3 Results	49
3.6 Optimization of TAC configuration in mobile communication systems: A tabu search approach [16].....	50
3.7 Automatic Replanning of Tracking Areas in Cellular Networks [20]	53

4. USING TABU SEARCH TO SOLVE THE TA RECONFIGURATION PROBLEM 54

4.1 Notions	54
4.2 Basic Principle	56
4.3 Problem formulation	58
4.4 Features of Tabu search	59
4.4.1 Encoding	59
4.4.2 Fitness	59
4.4.3 Algorithm Intuition	61
4.4.4 Tenure Matrix (TM).....	64
4.4.5 Aspiration Criterion	65
4.4.6 Call back mechanism.....	66
4.4.7 Frequency Matrix (FM).....	66
4.4.8 Long-term memory mechanism	68
4.4.9 Termination conditions	68

5. FORMULATING THE RECONFIGURATION PROBLEM USING GA AND IPM 70

5.1 Genetic Algorithm.....	70
5.1.1 Definition	70
5.1.2 Notations	71

5.1.3 Features of GA	71
5.1.3.1 Encoding.....	71
5.1.3.2 Fitness.....	72
5.1.3.3 Adaptation of the GA	73
5.1.4 Number of runs	77
5.2 Integer Programming Model	78
COMPUTATIONAL EXPERIENCE AND RESULTS.....	81
6.1 Test generation.....	82
6.2 Evaluating the performance of the proposed solution.....	82
6.2.1 Number of runs	84
6.2.2 Call back mechanism	84
6.3 Long-term mechanism (Diversification).....	88
6.4 Comparison with GA and IPM	91
6.4.1 Network 1	93
6.4.2 Network 2.....	95
6.4.3 Network 3.....	97
6.4.4 Reconfiguration cost	99
6.5 Complexity of the algorithms.....	100
CONCLUSION	101
7.1 Limitations and future work.....	102
REFERENCES	103

ILLUSTRATIONS

Figure	Page
Figure 1.1 UMTS and GSM Architecture	2
Figure 1.2 Evolved packet Core Architecture for LTE	4
Figure 1.3 TAs and MME association	6
Figure 2.1 Evolved Packet System	9
Figure 2.2 Evolved packet Core Architecture for LTE	10
Figure 2.3 Relationship between tracking areas, MME pool areas and S-GW service areas	16
Figure 2.4 MME identities	18
Figure 2.5 Temporary identities used by the mobile	19
Figure 2.6 OFDMA vs. SC-FDMA	20
Figure 2.7 Carrier aggregation scenarios	22
Figure 2.8 High level protocol architecture of LTE	24
Figure 2.9 Relationship between the access stratum and non-access stratum on the air interface	26
Figure 2.10 High level protocol architecture of LTE	28

Figure 2.11 Paging procedure	29
Figure 2.12 TAU Procedure with MME change.....	32
Figure 2.13 TAU Procedure without MME change	33
Figure 3.1 Genetic Algorithm flow chart.....	44
Figure 3.2 Crossover procedure.....	46
Figure 4.1 General algorithm of a TS method	58
Figure 4.2 Encoding example of using an array to represent a solution.....	59
Figure 4.3 Memory mechanisms in TS.....	62
Figure 4.4 Proposed TS based Algorithm.....	64
Figure 4.5 (a) Initial Matrix, (b) Tenure Matrix after swap (3,4), (c) Tenure Matrix after swap (2,1).....	65
Figure 4.6 (a) Initial Frequency Matrix, (b) Frequency Matrix after swap(3,4), (c) Frequency Matrix after swap (2,1)	67
Figure 4.7 Tabu search algorithm	69
Figure 5.1 General Genetic Algorithm	71
Figure 5.2 Encoding example of using an array to represent a solution.....	72
Figure 5.3 Implemented GA	75

Figure 5.4 An Iteration illustrating the GA.....	76
Figure 6.1 Effect of activation of the call-back mechanism on (a) 164 and 225 cells and (b) 1000 cells	86
Figure 6.2 Effect of the variation of the maximal intensity of the callback mechanism on the solutions	87
Figure 6.3 Effect of tenure value on the fitness function.....	88
Figure 6.4 Effect of activation of the long memory mechanism on (a) 164 and 225 cells and (b) 1000 cells.....	89
Figure 6.5 Effect of changing number of TAs	90
Figure 6.6 Effect of change of weight on average signaling	91
Figure 6.7 Pareto-optimal solutions in Network 1.....	94
Figure 6.8 Pareto-optimal solutions in Network 2.....	96
Figure 6.9 Pareto-optimal solutions in Network 3.....	98
Figure 6.10 Average reconfiguration cost obtained in both TS and GA	99

TABLES

Table	Page
Table 2.1 Interfaces linking components in LTE	14
Table 3.1 Advantages and disadvantages of GA	50
Table 3.2 Advantages and disadvantages of IPM.....	50
Table 5.1 Number of runs in GA	78
Table 6.1 Numerical assumptions and values for performance evaluation	83
Table 6.2 Number of runs in TS	84
Table 6.3 Numerical assumptions and values for performance evaluation in the GA	92
Table 6.4 Numerical assumptions and values for performance evaluation in the TS	93
Table 6.5 Signalling overhead generated in TS,GA and IPM associated with the fixed reconfiguration costs in Network 1	94
Table 6.6 Signalling overhead generated in TS,GA and IPM associated with the fixed reconfiguration costs in Network 2 (* 1000).....	96
Table 6.7 Signalling overhead generated in TS and GA associated with the fixed reconfiguration costs in Network 3 (* 1000).....	98
Table 6.8 Running time in seconds of the TS and GA.	100

CHAPTER 1

INTRODUCTION

Because of the rapid growth of the Internet, the Global System for Mobile Communications (GSM), most deployed 2G system, had to evolve to support, in addition to the circuit switching domain, the packet switched domain and to increase the data rates over the internet. As a result Universal Mobile Telecommunication System (UMTS), most deployed 3G systems was introduced. However because of certain UMTS limitations, UMTS itself has evolved into Long Term Evolution (LTE) networks. In this chapter, we briefly introduce GSM, UMTS, the main motivations behind LTE and its basic concepts, thesis motivations, problem and objectives.

1.1 UMTS and GSM

UMTS architecture is similar to that of 2.5 GSM and is shown in Figure 1.1 [1]. It consists of three major components, namely the user equipment (UE), the radio access network and the core network. The core network is made up of two components, namely the circuit switched (CS) domain and the packet switched (PS) domain. The CS domain is responsible for the transfer of voice. The CS exchanges information with the public switched telephone network (PSTN) allowing users to make calls to other CS domains of other operators or to land lines. The PS domain is responsible for transferring of data streams, such as emails and web pages between the UE and a packet data network (PDN). The radio access networks is further divided into two components namely, the GSM

Enhances Data Rates for GSM Evolution (EDGE) radio access network (GERAN) and UMTS terrestrial access network (UTRAN) each of which use different radio communication techniques but share, a core network in common.

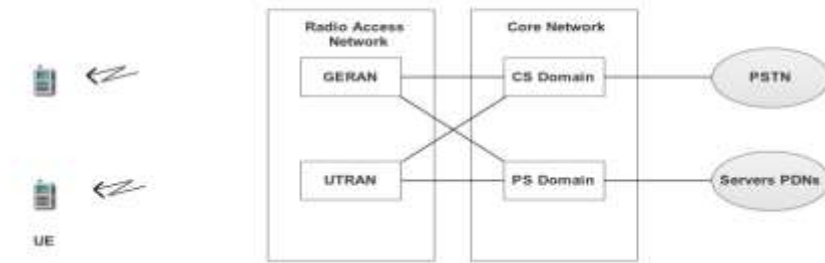


Figure 1.1 UMTS and GSM Architecture

1.2 LTE Main Motivations

To address certain limitations in the UMTS network, LTE was introduced. The key guiding principles to build up its architecture are summarised as following:

- Improve handling user data traffic: The wide spread of smartphone devices increased significantly the user data traffic and resulted in congesting 2G and 3G networks. Hence the need for introducing a new architecture that increases the network capacity and scales efficiently its infrastructure to support significant increase in user data traffic. This was achieved in LTE, which designs a flat architecture that reduces as much as possible the number of nodes processing the user data load to reach the objective.

- Separate the control and user plane functions: In LTE the control and user plane functionalities were separated because of several reasons such as: 1) to optimize the functionality of both independently of each other; 2) to accommodate the increasing demand on bandwidth by scaling only the nodes involved with the end-user traffic rather than signalling and traffic nodes, as in a UMTS network; and 3) to support distributed management for user data functionalities by certain infrastructure equipment, while maintaining centralized deployment of the equipment managing signaling in a way that reduces the delay between two end users and uses efficiently the transmission resources [2].

1.3 LTE Overview

Long term evolution (LTE) [1, 22] was developed by 3GPP to cope with the increasing demand for better QoS and the emergence of bandwidth consuming multimedia applications. It offers revolutionary performance when compared to its predecessors. To make this possible, a new architecture, the system architecture evolution (SAE) was designed, and several technologies such as orthogonal frequency division multiplex (OFDM) [8] and multiple inputs multiple outputs (MIMO) [10] were introduced to the cellular network turf. The SAE is an all-IP network, consisting of a user plane, the evolved UTRAN (eUTRAN), and a control plane, the evolved packet core (EPC), as shown in Figure 1.2. The eUTRAN contains only one simplified entity, the evolved NodeB (eNB) that is responsible for radio resource management, making the core of the network flatter and less complex, thus allowing for reduced latencies. Moreover, a new interface between the eNBs was incorporated, allowing for faster data routing. EPC is comprised of several

components such as: 1) the packet data network gateway (P-GW) which connects the EPC to the external world such as Internet; 2) the serving gateway (S-GW) whose main function is to transfer data between the eNB and P-GW just like a router; 3) the mobility management entity (MME) which handles the UEs mobility and other signaling such authentication, tracking the location of idle-mode users, security negotiations, controlling other relevant components in the network, etc; and 4) the home subscriber server (HSS), which is a database maintaining the subscriber's profile that includes information related to UEs current locations on MME level and the services granted to them.

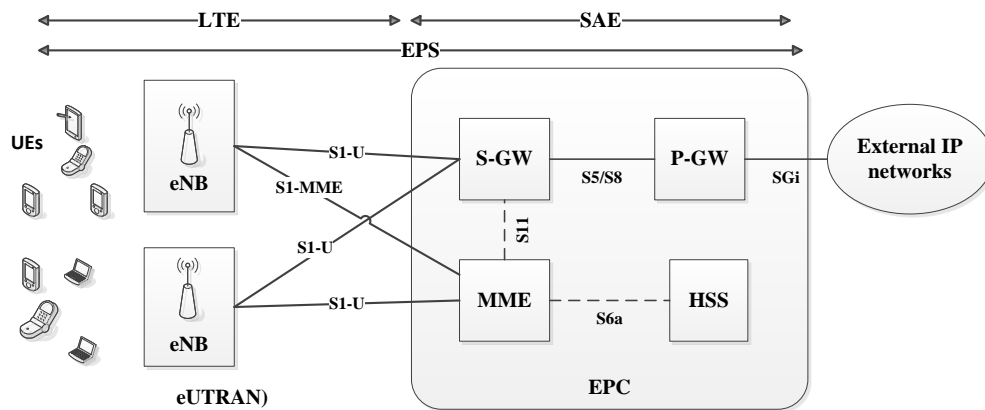


Figure 1.2 Evolved packet Core Architecture for LTE

1.4 Thesis motivation

Location management is one of the main mobility management functions in LTE networks [22]. It aims to keep track of the UE's position to allow calls and data delivery to them. In LTE mobility management architecture, cells (i.e. eNBs) are grouped to form Tracking Areas (TA) as shown in Figure 1.3. Each cell is assigned to one TA and each UE registers with one TA. A TA may be connected to more than one MME. The MME is

responsible for all its registered UEs. In the standard, MME records the current TAs of its UEs and the HSS records the current MME for each UE. A TA update (TAU) procedure occurs when a UE moves to a new TA. It consists of updating the MME only in case of intra-MME move (if both TAs belong to the same MME) and the HSS in case of inter-MME move (the two TAs situated in the vicinity of two different MMEs). Hence, a TA is defined as an area in which a user may move freely without having to update the MME or the HSS. The paging procedure occurs when the UE is being called. First the HSS should be queried to determine the current MME of the UE and then in order to place the call, the MME broadcasts a paging message in all cells of the UE's current TA. In order to reduce the mobility signalling overhead, TAs can be further grouped to form TA list (TAL) and then a list of one or more TAs can be allocated to the UE, which can move freely in these TAs without having to perform the TA update procedure (Figure 1.3 illustrates 3 TALs). Indeed, when a UE moves from one TA to another, it checks if this new TA belongs to its TAL; if it does (for example moving from TA1 to TA2 in Figure 1.3), then there is no need to perform the TA update procedure. LTE networks are also able to assign different TALs to different UEs. TA1 and TA2 belong to the same TAL assigned to UE1 and hence must belong to the same MME. when a UE moves from one TA to another, it checks if this new TA belongs to its TAL; if it does (for example moving from TA1 to TA2 in Figure 1.3), then there is no need to perform the TA update procedure for UE.

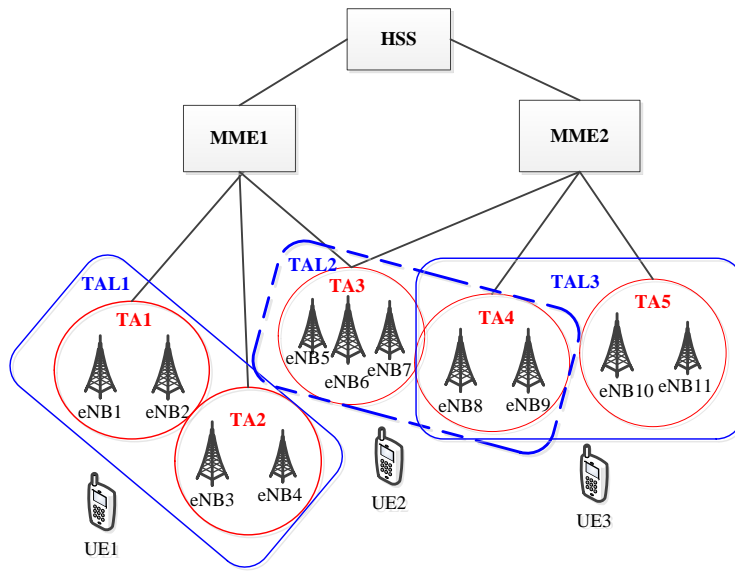


Figure 1.3 TAs and MME association

1.5 Problem definition

The size of the TA and the TAL affects both the TA update procedure used to keep the network aware of the latest UE region and the paging procedure used to determine the eNB through which the UE is reachable. The smaller the TAs are, the smaller the number of cells needed to be paged but more often TA updates. On the other hand with a larger TA and TAL, the paging cost increases and the TAU signalling decreases. Hence designing TAs is a problem as it affects both paging and TAU signalling cost and a compromise must be made. However, an initial optimal TA configuration cannot guarantee a low signalling overhead because the UEs mobility might alter their distribution. Therefore, the TAs needs to be reconfigured to account for the new distribution of UEs. Some questions have to be addressed such as how are cells grouped into TAs? May combining them in a certain way

produce less signalling cost during mobility of the UE? Should the behaviour history of the UE be considered? Will an optimized model remain efficient when UEs change positions? When a UE joins a network how MME selection decision is made by eNB? Why is a given MME preferred over the others?

1.6 Objectives

In this thesis, we use the tabu search (TS) heuristic to solve the tracking areas reconfiguration problem in LTE. The proposed approach not only avoids moves leading to non-feasible solutions but it also penalizes such moves by enlarging the period of time in which they are considered as tabu. Our contribution can be summarized as following:

1. Survey related work to LTE mobility management in general and TA reconfiguration in particular.
2. Formulate and implement a tracking area reconfiguration solution using tabu search heuristic.
3. Implement a variation of the genetic algorithm based approach found in the literature [14].
4. Formulate the problem as an integer programming model and implement it using gurobi optimizer.
5. Evaluate the performance of the proposed tabu search solution and compare its performance to the genetic algorithm based approach using different

network topographies and show how far or close to the optimality generated by the integer programming model.

1.7 Thesis plan

The remaining of this thesis is organized as follows: In chapter 2 we define the major architecture in LTE along with the different procedures and protocols involved in the mobility management and define our problem. In chapter 3 we survey related work. In chapter 4 we define the basic concept of Tabu search (TS) and use it to formulate a solution for the TA reconfiguration problem. In chapter 5 we introduce our implementation of the GA and the Integer programming model. Performance evaluation and analysis of the proposed TS based approach is presented in chapter 6 along with a comparison to Genetic algorithm (GA) and Integer Programming Model (IPM). We conclude in chapter 7.

CHAPTER 2

LTE

In this chapter we first present the LTE architecture and introduce its main elements, and then we describe the mobility management in LTE and related challenges. We end the chapter by defining the problem that we aim to solve in this thesis.

2.1 LTE Architecture

The new LTE architecture consists of two major components namely: the evolved packet core (EPC) and the evolved UMTS terrestrial radio access network (E-UTRAN). Figure 2.1 shows how the EPC is a substitution of the packet switched domain in UMTS and GSM. In the new architecture there is nothing equivalent to the circuit switched domain, instead voice is transported using voice over IP (VoIP). Hence the packet switch domain transports both data and voice. The E-UTRAN manages the radio communication between the EPC and the mobile users. It is a substitution of the UTRAN. In UMTS the mobile user is still called user equipment even though its internal processing has changed [1].

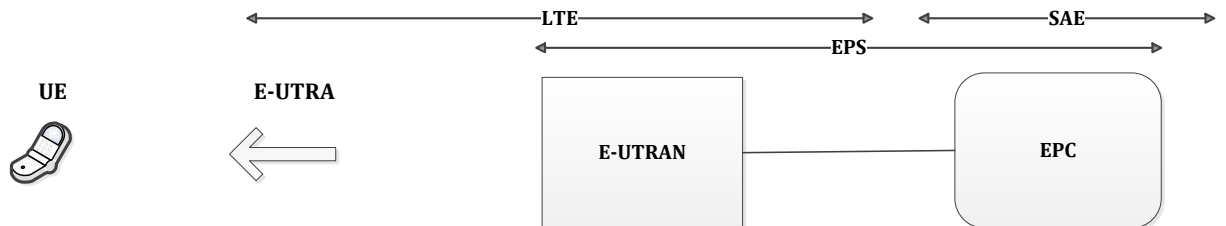


Figure 2.1 Evolved Packet System

Officially the whole network architecture is named evolved packet system (EPS), which is composed of the system architecture evolution (SAE), being the core network and long term evolution (LTE) that includes the air interface, radio access network and mobile as shown in Fig 2.2. In spite of this, LTE is the name used to refer to the whole network [1].

There are three main components, namely: the user equipment (UE), the evolved UMTS terrestrial radio access network (E-UTRAN) and the evolved packet core (EPC) as shown in Figure 2.2. The EPC is responsible for communication with the outside world. The interfaces between the different components are Uu, S1 and SGi.

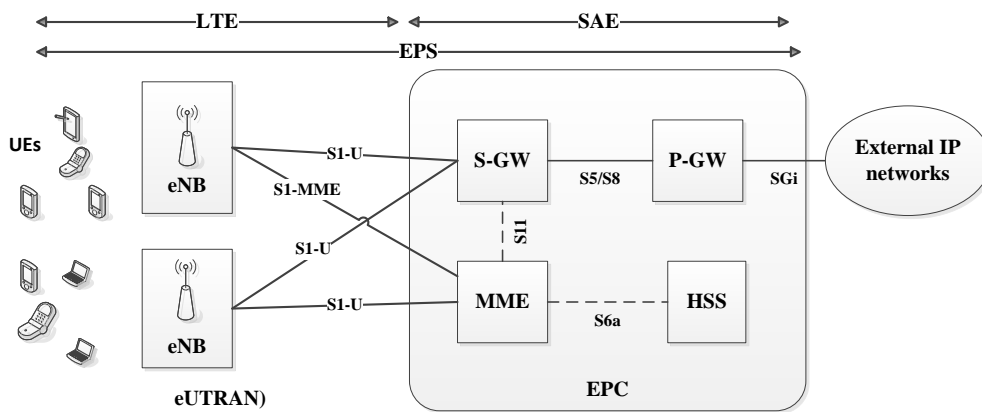


Figure 2.2 Evolved packet Core Architecture for LTE

2.1.1 The User Equipment (UE)

UE is either a handheld device such as our smart phone or a data card used in 2G or 3G that can also be contained in a laptop. UE contains the Universal Subscriber Identity Module (USIM) which is an application that is located in a removable smart card called

Universal Integrated Circuit Card (UICC). USIM's main function is to recognize and authenticate the user and to come up with security keys to secure the radio interface transmission. The UE is responsible for providing the user interface to the end user. In addition, the UE provides an environment for communication applications, which tell the network when to set up, maintain and remove communication links with the end user. These include mobility management functions such as handover and location update. During the execution of these functions, the UE listens to the network and performs the received instructions [4].

2.1.2 The evolved UMTS terrestrial radio access network (E-UTRAN)

E-UTRAN is composed of only a single type of node, the evolved Node B (eNB). Its main function is to manage the radio communication between the UE and the evolved packet core (EPC). Each eNB is a base station that serves one or more cells. A UE communicates with only one eNB at a time, and the eNB serving the UE is called its serving eNB.

The eNB is responsible for sending radio transmissions on the downlink channel to all its UEs and receiving transmissions from them on the uplink channel. In addition, the eNB manages the low-level operation of all its UEs, by sending handover messages. The eNB merges the functionalities of the earlier UMTS Node B and the radio network controller (RNC) in order to reduce the latency due to a fewer number of hops when a UE exchanges information with the network [4].

2.1.3 The evolved packet core (EPC)

EPC is comprised of several components such as: the home subscriber server (HSS), the serving gateway (S-GW), the packet data network gateway (P-GW) and the mobility management entity (MME), these elements are described next:

2.1.3.1 The home subscriber server (HSS)

HSS is a database that maintains the subscriber's profile. It contains information such as the location of the UE on MME level, the service granted to UE, roaming information, etc. [4].

2.1.3.2 The packet data network gateway (P-GW)

P-GW is the node that connects the EPC to the external world through the SGi interface. During the process data is exchanged with more than one external device such as IP multimedia subsystems, Internet or network operator servers [1]. The PDN is identified through its unique access point name (APN). The P-GW assigns IP address to the UE, which uses it to communicate with hosts, in the external network. It is also responsible for the traffic gating and filtering functions. Each P-GW may be connected to one or more S-GW(s) and external network(s). A UE associated with a given P-GW can only be involved with a single S-GW although it can be connected to several external networks if multiple PDNs are supported by the P-GW [4].

2.1.3.3 The serving gateway (S-GW)

S-GW's main function is to transfer data between the eNB and P-GW just like a router. Every UE is associated with a single S-GW, which can change during handover [1]. The S-GW is responsible for setting its own resources, which are assigned based on requests from the MME and P-GW. The MME and P-GW are responsible for setting up, clearing or modifying bearers for the UE. When the UE is in idle mode, the resources of the eNB allocated for a given UE are released and the data is held at the S-GW. Packets received from the P-GW on any tunnel will be buffered, and the MME is requested to initiate paging to the UE. Once the UE is found through paging, it switches to the connected mode, tunnels are reestablished and packets are forwarded. Here the S-GW manages data in the tunnels and obtains data needed for billing information [4].

2.1.3.4 The mobility management entity (MME)

MME is the only entity that involves signaling in the LTE architecture. The design was altered from previous implementation so that operators can manage the signaling and traffic independently. The MME's main function is to handle the mobility of the user. It is also responsible for authentication, tracking the location of an idle-mode user and security negotiations. User data packets do not go through the MME [4]. A network might involve several MMEs, each responsible for a given geographical area. A UE can only be associated with a single MME, known as the serving MME. This serving MME might be changed when the UE is far enough. The MME through its signaling messages controls

other components of the network [1]. The MME requests the S-GW to switch the tunnel from one eNB to another and to provide tunneling resources during the forwarding of data from the source eNB to the target eNB during handover. Other functionalities of the MME include clearing tunnels in the old S-GW and assembling them in a new S-GW as a result of change in S-GW.

2.2 Interfaces

A new important feature added to the LTE architecture is the S1 interface connecting the E-UTRAN to the core network as shown in Table 2.1. This feature is known as “S1 flex”. This was introduced so that several core networks (MME/S-GW) can serve a given geographical area, as will be described in the next section. An eNB is connected to its serving MMEs through the S1-MME interface and to its serving S-GW through the S1-SGW interface. The interface that exists between the P-GW and S-GW can either be the S5 or S8. If the two entities belong to the same network then the S5 interface is used and if they belong to different networks then the S8 interface is used [1].

Table 2.1 Interfaces linking components in LTE

	<i>MME</i>	<i>S-GW</i>	<i>P-GW</i>	<i>eNB</i>
MME	S10	S11		S1-MME
S-GW	S11	-	S5/S8	S1-U
P-GW	-	S5/S8	-	-
eNB	S1-MME	S1-U	-	X2
HSS	S6a	-	-	-

2.3 Cells, TAs and TALs

The area that a base station covers is called a cell. Every cell has a unique cell identity. Cells are grouped to form Tracking Areas (TA) [6]. TAs are further grouped to form Tracking Area Lists (TAL) to which UEs are assigned. TALs reduce the TAU cost because a UE moving in the TAs belonging to its TAL does not perform TAU and hence resulting in less the signaling overhead. However this might increase the paging cost because more TAs must be paged to locate the serving eNB of a called UE. Hence the TAL must be chosen very carefully. Every cell must be associated with only one TA and every UE must be assigned to a TAL. All the tracking areas in a Tracking Area List to which a UE is registered are served by the same serving MME [15].

The group of MME/S-GW that serves a given region is called MME/S-GW pool, and the region that contains this MME/S-GW pool is called the pool area. An MME pool area is an area in which a UE can move without having to change the serving MMEs. A pool area might be served by one or more MMEs. Therefore TAs in the same TAL must belong to the same MME/SGW pool area and TAs can be associated to more than one MME as shown in Figure 2.3:

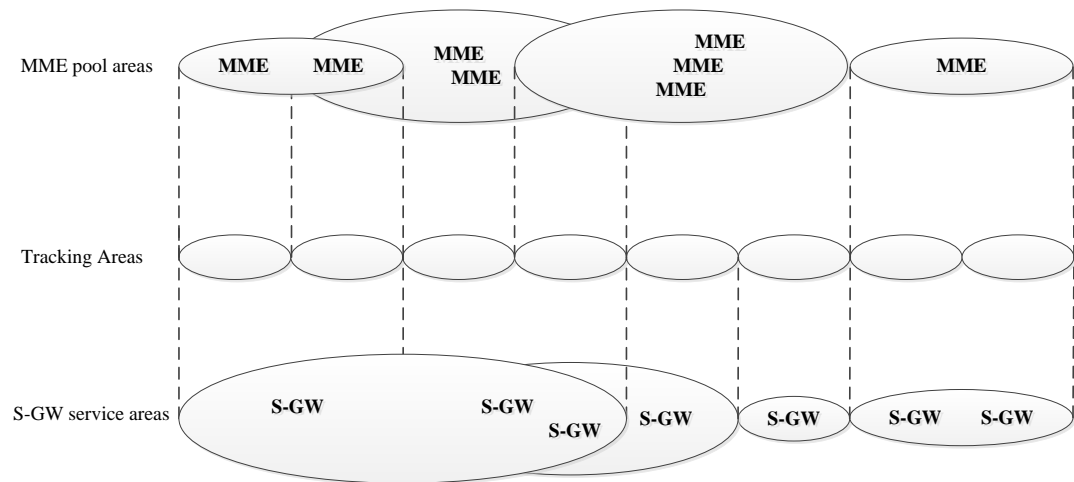


Figure 2.3 Relationship between tracking areas, MME pool areas and S-GW service areas

A S-GW pool area is an area in which a UE can move without the need to change the serving gateways. A pool area might be served by one or more S-GWs. Pool areas can overlap.

The MME/S-GW pool areas consist of smaller non-overlapping sections called TAs. The MME pool area and S-GW service area are not associated as shown in Figure 2.3. They are equivalent to the location and routing areas from UMTS and GSM. The importance of this added feature is that multiple core networks associated with a single eNB are responsible for a given UE providing load sharing and excluding the possibility of single point of failure for core networks. As long as the UE remains within a given pool area, its context will be located within the same MME [5]. MME and S-GW pool areas are not necessarily associated with each other.

2.4 Component Identification

Each network is assigned to a public land mobile network identity (PLMN-ID), which is made up of two parts namely the mobile country code (MCC) and the mobile network code (MNC). The MCC is a three-digit code and the MNC is a two or three digit code. Cells can be recognized by three different codes namely: E-UTRAN cell identity (ECI), E-UTRAN cell global identifier (ECGI) and the physical cell identity. A cell should be recognized in the network it belongs to through the ECI and across the world through the ECGI and from its neighbors through the physical cell identity. MMEs can be recognized by three different codes as shown in Figure 2.4, namely: the 8 bits MME code (MMEC) that distinguishes a MME in its pool area; by adding it to the 16 bits MME group identifier (MMEGI) we get the 24 bits MME identifier (MMEI) that distinguishes a MME in the network associated with. By also considering the network identity (PLMN-ID) we get the globally unique MME identifier (GUMMEI) that distinguishes the MME in the world.

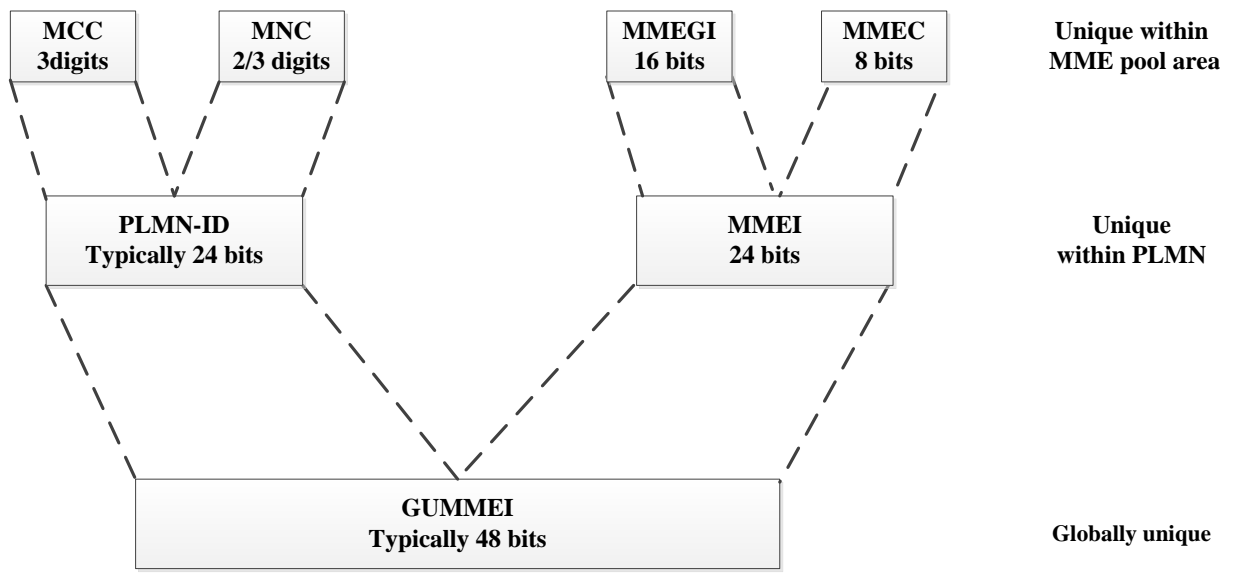


Figure 2.4 MME identities

Just like other components in the network a TA is recognized by two different codes namely: tracking area code (TAC) that distinguishes a TA within a given network and a tracking area identity (TAI) that distinguishes a given TA globally [1]. In a UE a universal integrated circuit card (UICC) also known as the SIM card is executed on an application known as the universal subscriber identity module (USIM) that contains user's information such as the user's phone number and the network it belongs to. A mobile can be recognized by two different identities namely: the international mobile equipment identity (IMEI), which is different for each UE and the international mobile subscriber identity (IMSI), which is different for the Universal Integrated Circuit Card (UICC) and the Universal Subscriber Identity Module (USIM). The IMSI should not be transmitted in an air interface since it is one of the information needed to clone a mobile device and hence must be secured. Mobile devices are recognized by their serving MMEs through three different

temporary identities. The 32 bit M temporary mobile subscriber identity (M-TMSI) which recognizes a mobile to the MME it is being served by, adding the MMEC we get the S temporary mobile subscriber identity (S-TMSI) which recognizes a mobile within its pool area [1]. Adding the MME group identifier and the PLMN-ID we get globally unique temporary identity (GUTI) as shown in Figure 2.5:

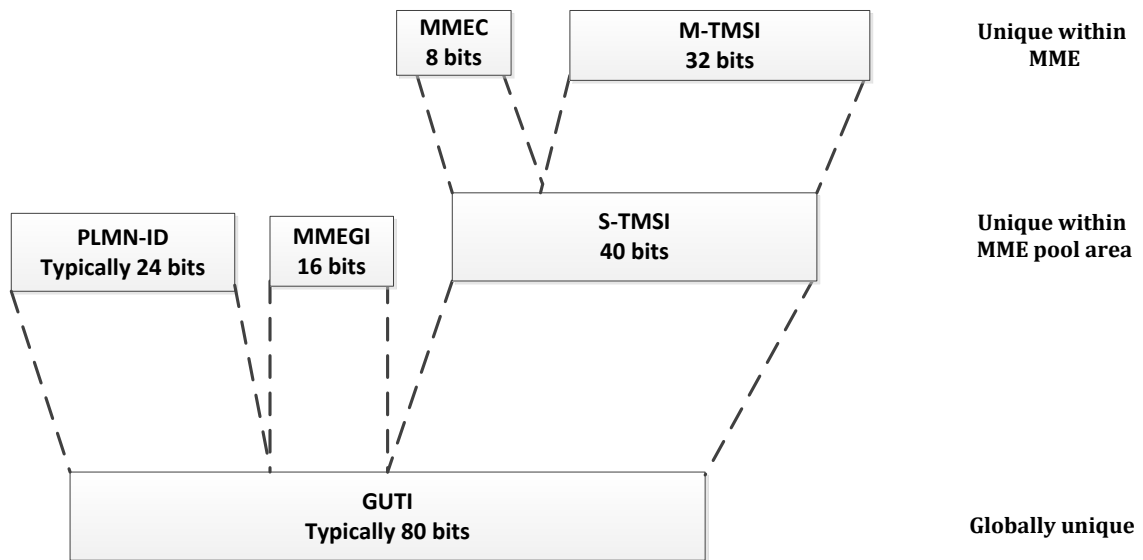


Figure 2.5 Temporary identities used by the mobile

2.5 LTE to LTE-Advanced

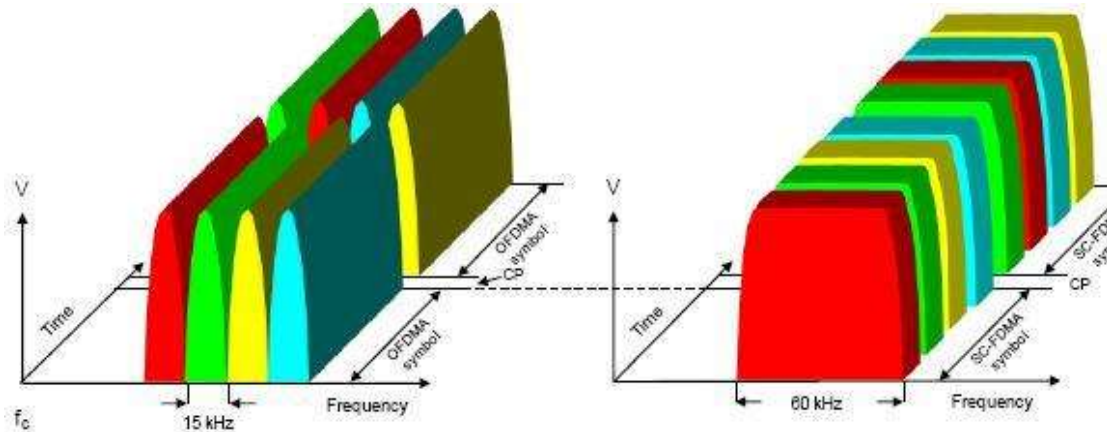


Figure 2.6 OFDMA vs. SC-FDMA

OFDM (Orthogonal Frequency Division Multiplexing) transmission scheme is used for the downlink channel i.e. from the eNB to the UE. Single Carrier Frequency Division Multiple Access (SC-FDMA) is used for the uplink channel i.e. from the UE to the eNB. In OFDMA the information is not sent along a single stream but along several sub-streams simultaneously, each on a different frequency known as a subcarrier i.e. the time and frequency domains can be managed when consuming the available capacity in parallel as shown in Figure 2.6 [2]. In SC-FDMA only a single carrier is available. This reduces the peak-to-average ratio i.e. the variation in the amplitude of the power does not alter as it does in the OFDMA scenario.

LTE-Advanced was introduced to support a maximum bandwidth of 100MHz. This is most probably unavailable in a contiguous pattern because of its large size. As a result of this arising problem LTE-Advanced provides the mobile user with up to five

component carriers (CCs) for transmission and reception, each having a maximum bandwidth of 20MHz. This mechanism is known as carrier aggregation (CA). Carrier Aggregation can occur in three different ways, namely: inter-band aggregation, non-contiguous and contiguous intra-band aggregation as shown in Figure 2.7.

In inter-band aggregation the carrier components do not belong to the same frequency band making this scenario the most complicated one. This is because the radio components supporting each band might be required to be different by the mobile user and the coverage area provided by each band might be very different. Here the carrier components are separated by a multiple of 100kHz, which is the same as the LTE carrier spacing [1].

In non-contiguous intra-band aggregation the carrier components fall within the same band simplifying the design of the network and mobile. Due to the non-contiguous pattern of frequencies, it shares some of the problems described in inter-band aggregation [1].

In contiguous intra-band aggregation, the carrier components belong to the same band and are adjacent to each other separated by a multiple of 300kHz [1]. LTE-Advanced is designed such that an LTE mobile can communicate with a base station of an LTE-Advanced network and vice versa [1]. All three carrier aggregation scenarios are represented in Figure 2.7.

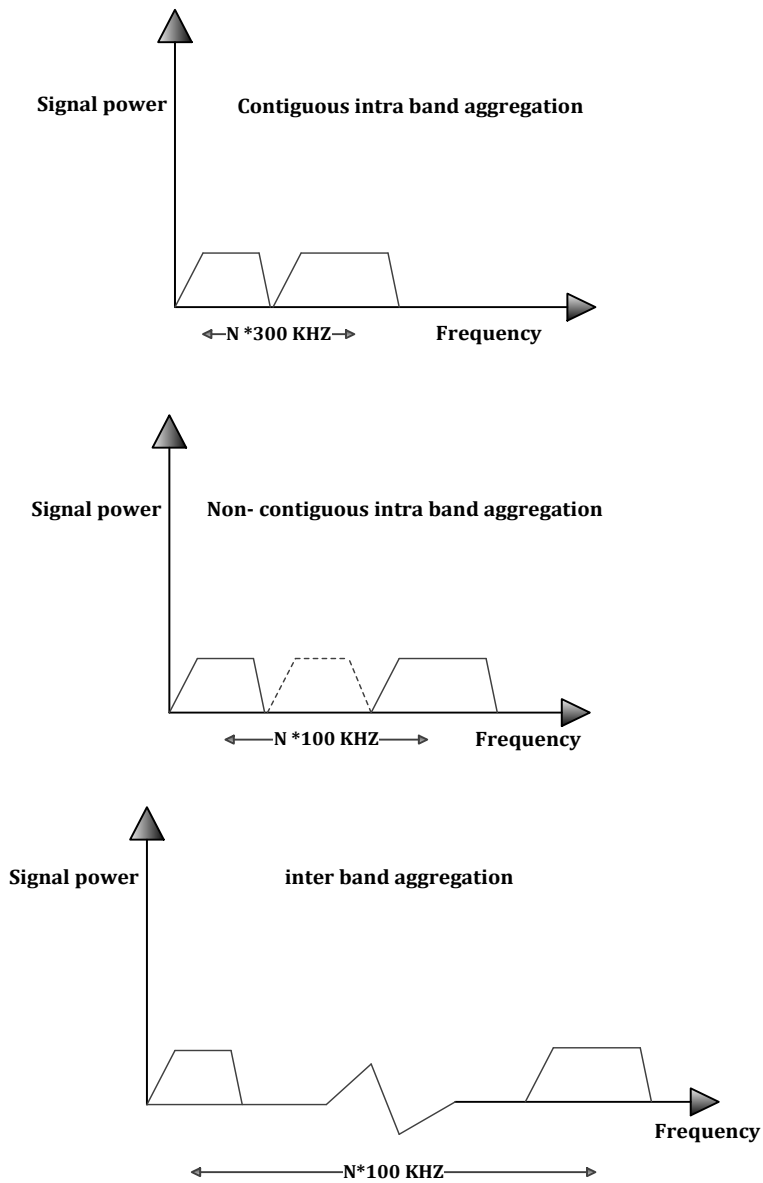


Figure 2.7 Carrier aggregation scenarios

2.6 Mobility Management in LTE

In LTE, the mobility management tasks are performed by the signaling only entity, the mobility management entity (MME) that is connected to a set of eNBs. The area that an

eNB covers is called a cell. As mentioned earlier, every cell has a unique cell identity. Cells are grouped to form the TAs. Just like cells, every TA has a unique TA identity (TAI). In LTE, TAs are further grouped to form TALs to which UEs are assigned. The main mobility management procedures are: paging, location update and handover.

The first procedure the UE executes is the attach procedure. The attach procedure provides several objectives such as registering the location of the UE with a serving MME. It is also responsible for managing the signaling radio bearer 2, which is responsible for transmitting non-access stratum signaling messages in the air interface. In addition it grants the UE an IP version 4 address and/or an IP version 6 address. It also provides the UE with a continuous connection to a default PDN.

Location update is triggered when a UE moves from one location to another reporting its new location to the HSS or the MME. When an incoming call arrives, the location of the UE is established through the paging procedure [6].

The mobility management of a UE depends on its mobility state, which can be in detached, active or idle mode. At first when the UE does not belong to any network, it is in the detached mode. When the UE registers to a network, it switches to the active state. After choosing a given network the UE sends a request message to the eNB, which chooses a MME to serve the UE and forwards the UE's request to this MME. The UE's request needs to be authorized by the MME before the latter creates the UE-context. In addition the MME is also responsible for sending a message request to the S-GW and P-GW commanding them to allocate resources for the UE. After allocating resources, the MME

sends all the information to the eNB. Later the bearer is established between the UE and the EPC. The UE enters the idle state whenever there is no data packet flowing towards it. This transition is performed in order to conserve power. In this state, the UE has no link with the eNB and the context of the UE is only located in the MME. This context doesn't contain the location of the cell the UE resides in but only the TAL, which can be comprised of several TAs.

2.6.1 Communication protocols

To exchange data and signaling messages between the different network entities, a protocol stack that consists of two planes, user plane and control plane was designed. Protocols in the user plane manage the data transferred to the user while those in the control plane manage the signaling messages between the network components as shown in Figure 2.8.

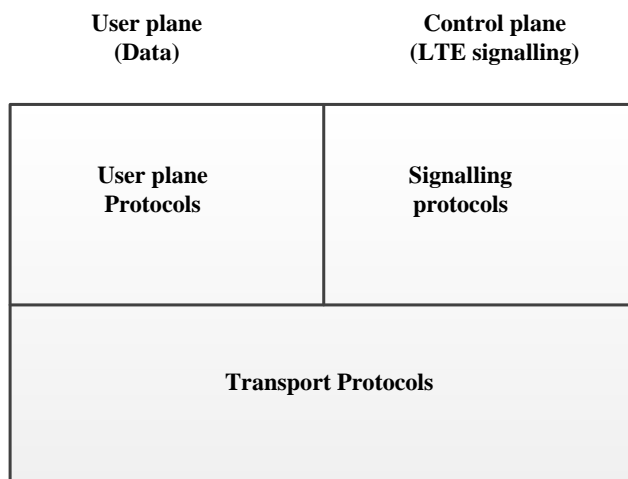


Figure 2.8 High level protocol architecture of LTE

The protocol stack has two main layers: the radio network layer which handles data in a precise way in LTE and transport network layer while the transport network layer which transports information from one point to another. Protocols are classified into three types:

1. Signalling protocols that define a communication language between network components,
2. User plane protocols that handle data in the user plane, and
3. Transport protocols that transport data and signalling messages between network components.

The MME is responsible for managing the high level behaviour of the mobile by sending signalling messages to the user. However there is no direct link between these two components. As a result the air interface consists of two levels, the access stratum (AS) and the non-access stratum (NAS) [1]. These signalling messages are found in the NAS and are delivered using S1 and Uu interfaces of the AS as shown in Figure 2.9.

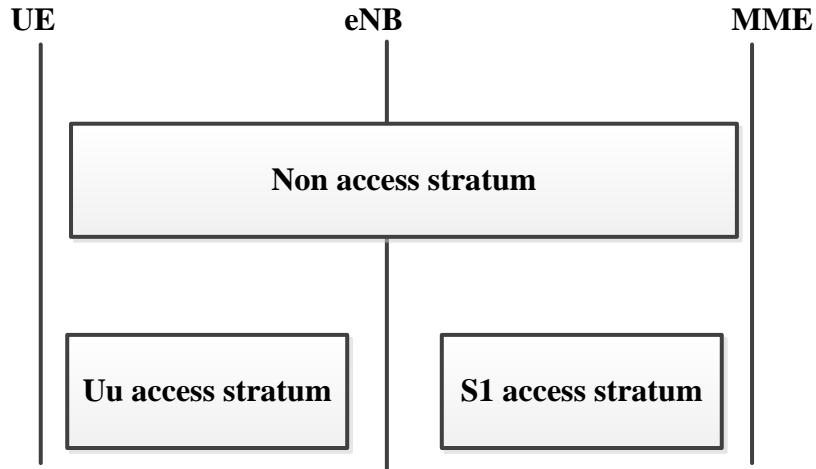


Figure 2.9 Relationship between the access stratum and non-access stratum on the air interface

2.6.2 LTE Mobility and connection states

Based on the information in the MME about the UE, the UE can be in one of the two states:

1. EPS Mobility Management (EMM) state: in which the UE can either be in the EMM-DEREGISTERED or the EMM-REGISTERED state. In the EMM REGISTERED state the MME contains information about the location of the UE on TA level. In this state the UE performs periodic TAU and replies to paging messages. In the EMM-DEREGISTERED state the MME holds no information about the UE's location. Attach and TAU procedures change the UE's state to the EMM-REGISTERED state.

2. EPS Connection Management (ECM) state: in which the UE can be in two ECM states: the ECM-IDLE state and the ECM-CONNECTED state. In the ECM-CONNECTED state there is a signalling connection between the UE and the MME, which is made possible by the RRC-CONNECTION between the UE and the E-UTRAN and the S1 link between the E-UTRAN and the MME. Here the location of the UE, is known on cell level and mobility is managed by handovers. In the ECM-IDLE state there is no NAS signalling connection between the UE and the MME and the E-UTRAN doesn't contain the UE's context. Mobility is managed by TAUs and the location of the UE is known on TA level [1].

2.6.3 Tunnelling

The GPRS Tunnelling Protocol (GTP) is an IP/UDP protocol used for tunnelling in LTE's core network as shown in Figure 2.10. It transfers both signalling and data traffic between the components in the core network. The data traffic is encapsulated when it is being tunnelled. There are three different types of GTP namely: GTP-C, GTP-U and GTP'. The GTP-C is the control unit of GTP. It is used for bearer activation, deletion and modification of tunnels between the core components. The GTP-U transfers data traffic in the user plane between the components in the core network. Finally the GTP' is used to carry charging/billing data [19].

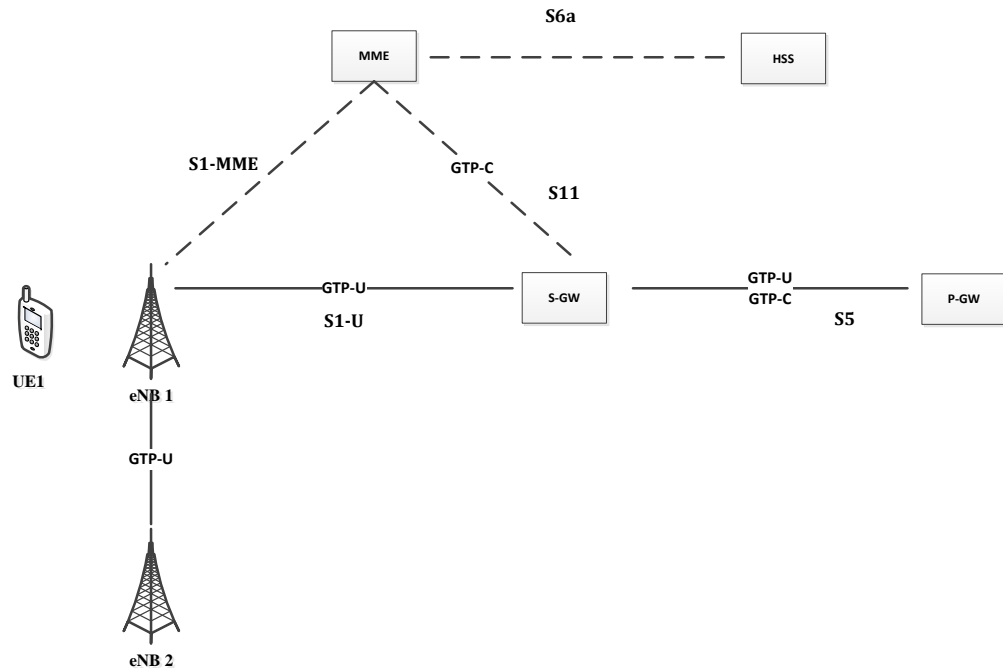


Figure 2.10 High level protocol architecture of LTE

2.6.4 Paging

When the mobile user becomes inactive the S1 release procedure is executed and as a result the user is put in the ECM-IDLE and RRC_IDLE state where the S1 and radio bearers are destroyed. When a packet arrives at the P-GW, it forwards it to the S-GW (step 1) as shown in Figure 2.11. However the S-GW does not know the current eNB of the destination UE. A GTP-C Downlink Data Notification is sent by the S-GW to the MME to notify it of the arriving packet (step 2). The MME then sends an acknowledgement message to notify the S-GW that it received the message to prevent it from sending more messages

on the arrival of more packets (step 2). The MME then creates a S1-AP paging message, which contains the mobile's S-TMSI and all the data required by the base stations to calculate the paging frame and occasion and sends it to the eNBs (step 3). Each eNB broadcasts the paging message to its UEs (step 4). The called UE receives a message from the serving eNB and responds using the service request procedure which switches the UE to the ECM-CONNECTED state, restores the signaling radio bearer 1(SRB 1), signaling radio bearer 1(SRB 2), the mobile's data radio bearers and S1 bearers (step 5)[1].

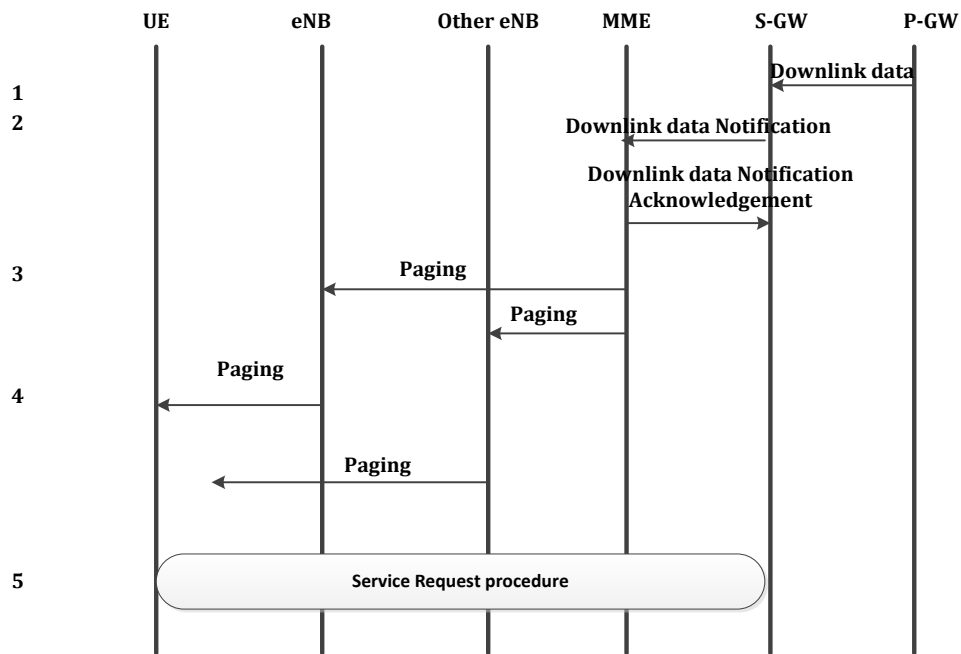


Figure 2.11 Paging procedure

2.6.5 Tracking Area Update

When a UE moves to a new cell, the TAC to which this cell belongs is examined and compared to the previously registered TA. If it is different, then the EPC is informed

through the tracking area update procedure (TAU), which is shown in Figure 2.12. It is assumed that the UE is in the ECM-IDLE and RRC_IDLE state and uses the contention based random access and the RRC connection establishment procedures to setup signaling connection with the serving eNB to switch to RRC_CONNECTED state. The TAU is represented in Figure 2.12 and can be summarized as follows:

1. If the new cell TA code is different than that of the old cell, the UE starts the TAU procedure.
2. The UE then forms an EMM Tracking area update request and embeds it in the RRC Connection Setup Complete and sends it to the eNodeB. The message contains the RRC parameters indicating the Selected Network and the old GUMMEI.
3. Upon receiving this message, the eNB retrieves the UE's current MME from the RRC parameters and forwards to it the TAU request. If the MME has nothing to do with the eNB then the latter selects a new MME using the MME selection function.
4. The new MME uses the old GUTI received to send to the old MME a context request message to obtain the user's information.
5. The old MME replies with a Context Response message.
6. The new MME authenticates the UE by contacting the HSS.
7. Because the MME has changed, the old S-GW might not be able to serve the UE. In this case the new S-GW is selected using the S-GW selection function. The new MME sends a context acknowledgement message to the old MME

indicating that the S-GW has changed. The old MME updates its context in case another TAU procedure is executed before the completion of this.

8. If a new S-GW is selected by the MME, the MME sends a Create Session Request message that contains the MME address, IMSI, serving network per PDN connection to the selected new S-GW.

9. The new S-GW sends a Modify Bearer Request message per PDN connection to the P-GW.

10. Upon receiving the Modify Bearer Request message the P-GW changes its bearer contexts and replies with a modify bearer response message.

11. The Serving GW returns a Create Session Response to the new MME, which contains the address of the new S-GW.

12. The new MME sends an Update Location Request message to the HSS.

13. The HSS sends the message Cancel Location (IMSI, Cancellation Type) to the old MME.

14. The old MME acknowledges with the message Cancel Location Acknowledgement (IMSI).

15. HSS acknowledges the Update Location Request message by sending an Update Location Acknowledge message to the new MME.

16. The new MME send a TAU accept to the UE that contains the TAL the UE is now assigned to and an optional new GUTI.

17. If the GUTI changes the UE send a TAU complete [15].

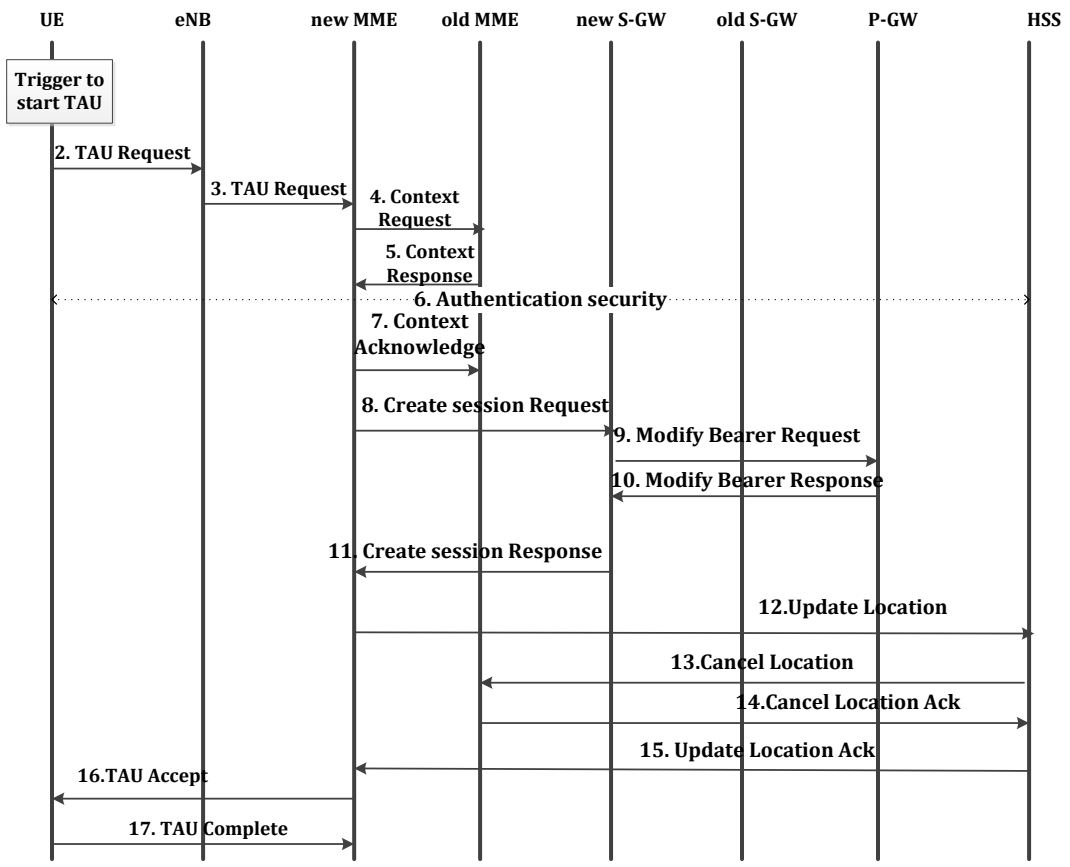


Figure 2.12 TAU Procedure with MME change

In case of Tracking Area Update without MME change the signalling in steps 4, 5, 7 and steps 8-15 are not executed [15] as shown in Figure 2.13.

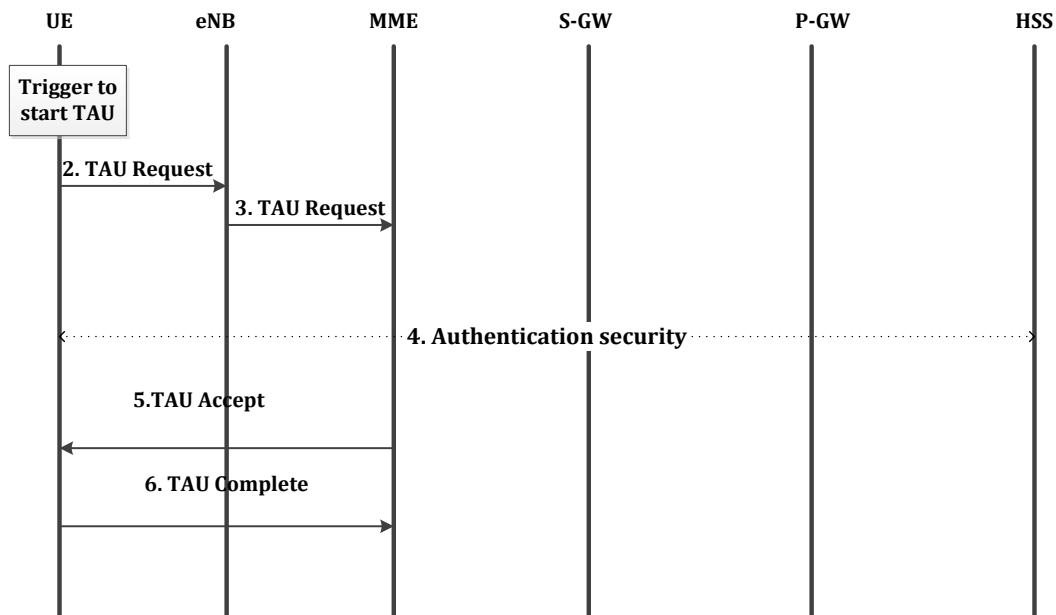


Figure 2.13 TAU Procedure without MME change

2.6.6 Handover

It is important to have the quality of service (QoS) maintained not just before and after handover but during handover as well, with minimal handover latency. Handover decision can be either network evaluated where the network determines the handover decision, or mobile evaluated where the UE determines the handover decision and then makes the network aware of it. The network makes the final decision based on the Radio Management Resources.

In LTE networks, handover decisions are determined as a result of a hybrid method. Both the UE and network contribute in making the HO decision. The UE measures the

neighboring cells and conveys this information to the network. The network then decides upon the target eNB and the handover timing based on the received information.

There are three types of handovers:

1. Intra-LTE handover (intra-MME and Intra-SGW)
2. Inter-LTE handover (inter-MME and Inter-SGW)
3. Inter-RAT handover

2.6.6.1 Intra-LTE handover

This type of handover occurs within the same LTE nodes (MME, S-GW). This can either happen using the X2 or S1 interface. It is only possible to use the X2 interface if the source and target eNBs (S-eNB) (T-eNB) are directly connected via the X2 interface. The serving MME and S-GW are unchanged. This is performed without the involvement of the EPC. Messages between the S-eNB and T-eNB are directly exchanged. After handover is complete, the T-eNB asks the MME to switch the downlink data path from the S-eNB to the T-eNB. The MME then commands the S-GW to switch the data path from the S-eNB to the T-eNB. If the S-GW has not yet switched paths towards the new eNB and the downlink packet arrives, the S-eNB transfers the packet via the X2 interface [9] [2]. When there is no X2 connectivity between the S-eNB and T-eNB, handover is triggered by the S-eNB that sends a handover message via the S1-MME interface. The core network has no say in the decision taken by the eNB. This handover is similar to the previous one except the

involvement of the MME in transferring the handover signaling between the S-eNB and T-eNB.

2.6.6.2 Inter-LTE handover

This type of handover occurs between two different LTE nodes (MME, S-GW). This is further divided into two types. The first type is inter-MME handover. Here the MMEs that control the S-eNB and T-eNB take part in the handover. As the name implies this type of handover occurs when the UE moves between two eNBs belonging to two different TAs associated with two different MMEs [9]. The second subtype is the Inter-MME/SGW handover using the S1 Interface. This is very similar to the previous scenario except that the S-eNB and T-eNB are served by different MME/S-GW [9].

2.6.6.3 Inter-RAT handover

In the presence of other technologies, handover can happen such that the T-eNB belongs to a network having a different radio technology (UMTS or GSM) [9].

2.7 Problems related to mobility

The size of the tracking area and the tracking area list affects two factors, namely: the Tracking Area Update (TAU) load and the paging load. The smaller they are, the smaller the number of cells needed to page the UE and hence the smaller the paging load

but more often TAU when the UE changes locations. On the other hand with a larger TA and TAL, the paging cost increases and the TAU signalling decreases. Hence designing TA and coming up with the set of TALs in a network is a problem as it affects both paging and TAU signalling cost, therefore a compromise must be made. Some questions have to be addressed such as how are cells grouped into TAs? May combining them in a certain way reduce signalling cost during mobility of the UE? Should the behaviour history of the UE be considered? Will an optimized model still be efficient when UEs change positions? When a UE joins a network the eNB chooses a MME to serve the UE. How is this decision made? That is, why is a given MME preferred over the others?

In the next chapter, we survey the literature that addresses some of these questions and the different proposed solutions.

CHAPTER 3

RELATED WORK

Several enhancements were proposed in the literature [6, 10, 12, 13, 14, 16] to improve mobility management in LTE. In this chapter we survey those that are relevant to location management.

3.1 A dynamic paging scheme for long term evolution mobility management [10]

As we have mentioned, when an incoming call arrives, the MME pages all the cells in the UE's TAL. This leads to enormous paging traffic using the limited radio resources. To reduce the paging traffic, a dynamic paging scheme was introduced in [10]. Three paging schemes were considered, namely: scheme Cell-TA-TAL (CTT), scheme TA-TAL (TT) and scheme 3G. In all three schemes an interacted cell is defined as a cell where the UE has interacted with the network either by initiating location update or upon receiving a call. In CTT when an incoming call arrives, the MME pages the last interacted cell to awaken the UE. If no response is observed after a given timeout period, the MME pages all the cells in the TA of the last interacted cell. Again, if no response is received from the UE, the MME pages all the cells in the TAL. In TT when an incoming call arrives the MME pages all the cells in the TA of the last interacted cell. If no response is received from the UE, the MME pages all the cells in the TAL. In 3G when an incoming call arrives the MME pages all the cells in the TAL simultaneously. A polling cycle is defined as the time

the MME has to wait before the paging response from the UE is received after sending the paging signals or the duration for timeout to occur. The expected number of paged cells is calculated on the arrival of the i^{th} incoming call for all three schemes. If two or more schemes happen to have the smallest number of paged cells the MME chooses the one with the smallest N_p (maximum number of polling cycles before the UE is found) to reduce paging delay. The ‘best’ paging scheme (i.e. the one with the smallest number of paged cells) is automatically selected based on UE movement and call behaviour. Using the UE’s recent ‘behaviour history’. Let V be the variance in location. Results showed that when V is small i.e. UE’s movement is regular; the UE is most probably to be found in the last interacted cell, in this case the CTT and the dynamic scheme have similar performance. When V is large, i.e. UE’s movement is dynamic; the dynamic scheme outperforms the CTT scheme since it is more likely to choose the TT or 3G schemes that provide less paging delay.

3.2 An investigation on LTE mobility management [6]

In this paper the performance of paging in a network is taken into consideration in order to provide guidelines to the best paging sequence used in cells. The three schemes defined in [10] are addressed. Under different conditions of the UE’s mobility, different schemes are selected. If the number of polling cycles is important and the mobility of the UE is high with regular movement patterns then scheme 3G must be used. However if the paging and TAU costs are to be considered then the CTT scheme must be selected.

3.3 Reducing signaling overhead for femtocell/macrocell networks [12]

Different methods used to assign Location Areas (LA) in femtocells / macrocell network were discussed in [12]. In the first method, femtocells overlapped with a macrocell share the same LA with that macrocell. When paging needs to be performed for a given UE in that LA all the femtocells get involved. This results in a very high paging cost. In the second method, femtocells overlapped with a macrocell are partitioned into several groups each assigned to a LA different from that of the macrocell. This significantly lowers the paging cost in the femtocells/macrocell network but increases signaling overhead. The increase of signaling overhead is as a result of the frequent registration updates that occur every time the UE moves between overlapped femtocells and macrocell. In the second method a Delay Registration (DR) algorithm is proposed to postpone the registration until a given time t has elapsed. This avoids registration during transient timing, decreasing the traffic offloading capability of the femtocell and the signaling overhead.

As a justification, let L_m denote the LAI associated with a macrocell and let L_f be the LAI associated with the overlapped femtocells. Initially a UE is located in a non-overlapped area in the macrocell; hence the LAI it is assigned to is L_m . The UE at time t moves into the overlapped femtocells and stays there for time t_f . At time $t + t_f$ the UE moves back into the macrocell and hence it can no longer receive L_f . If the standard 3GPP algorithms were used, two registrations would have been performed. If the UE passes by the femtocells for a short period of time, i.e. t_f is small, the UE's movement is considered to be transient and registration should not be performed since during the transient period, calls

are not likely to be received. Hence the t_f is compared to a threshold value below which no registration is preformed and above which registration is executed.

When the UEs' mobility is more dynamic, the DR algorithm seems to reduce signaling overhead ratio and provides higher offload traffic ratio. However for this to be very efficient, accurate estimation of the UE information must be obtained.

3.4 Exploiting Tracking Area List Concept in LTE Networks [13]

In [13] the main goal is to come up with a design that generates lower overhead than the conventional TA design already used in LTE. Two designs are introduced: The UE-trace based TAL and the Rule of thumb TAL designs. A UE-trace scenario is a set of UE movements and call patterns over a period of time that corresponds to a set of cell load and handover data. To build up UE-trace scenarios, UEs in the network are assigned to TAs each consisting of one cell. This is to keep track of the UE's current location. When a UE moves to another cell the TAU procedure is executed.

Results showed that the UE-based TAL design produced lower signalling overhead compared to the conventional TA design in LTE no matter how small or large the network is. With the use of more UE traces, the signalling overhead in the TAL design is further reduced. The drawback here is the cost in collecting these traces. It is very expensive since UEs are assigned to a single cell; hence a UE moving from one cell to another has to update its location to the MME. As a result of the high cost in collecting the UE traces, the UE-based TAL design is not recommended for large networks. The Rule of thumb used in

designing the TAL performed better than the UE-based TAL design scheme. In the Rule of thumb approach no data is collected since there is no need for UE traces in designing the TAL.

3.5 Performance and cost trade-off in Tracking Area Reconfiguration: A pareto-optimization approach [14]

Reducing the signaling overhead is important for location management in LTE.

Because the UE movement and distribution will change over time an optimized TA configuration will no longer be enough. TA reconfiguration becomes necessary. However, this is expensive and may lead to service interruption. Hence there is always a tradeoff between the signaling overhead and reconfiguration cost. The tradeoff is modeled as a bi-objective optimization problem. A solution classified by pareto-optimality was proposed in [14] which is unlike mono-objective optimization problems that have a single optimal value. The problem is formalized as an integer-programming model, which is solved for several budget values to come up with a pareto-optimization scheme. The Genetic algorithm embedded with local search is used to generate pareto-optimal solutions in one single execution. The different obtained solutions are compared to each other and each is assigned a dominance value called a Preference Value (PV). Two approaches were used to solve this problem: The Genetic algorithm (GA) and the Integer programming model

The following notations are used in modeling the problem:

- T: is the set of TAs {1...T}
- n: is the set of cells {1 ... n}

- t_i : is the new TA of cell i
- t^0 : is the old TA of cell i
- u_i : is the number of UEs in cell i for a given period in time.
- h_{ij} : is the number of UEs moving from cells i to j .
- c^p : is the overhead resulting from one paging
- c^u : is the cost of one location update
- α : is the probability that a UE has to be paged.
- P_{it} : is 1 when i belongs to TA t and 0 otherwise
- $S_{ij}(t)$: represents whether two cells are in the same tracking area or not

i.e

$$S_{ij} = \begin{cases} 1 & t_i = t_j \text{ (two cells in the same TA)} \\ 0 & \text{otherwise} \end{cases}$$

The total overhead resulting from paging and location update is calculated as follows:

$$C_{so} = \sum_{i \in n} \sum_{j \in n: j \neq i} (c^u h_{ij} (1 - S_{ij}(t)) + (\alpha c^p u_i S_{ij}(t))) \quad (3.1)$$

In equation (3.1) the first addend represents the update overhead resulting from the UE moving from cell i to j . The second addend represents the paging overhead as a result of an incoming call. To understand it better, suppose a topology of n cells and T tracking areas then let C^u denote the cost of one location update, h_{ij} the number of UEs moving from cell i to j , $S_{ij}(t)$ an element in a symmetric and binary matrix $S(t)$ representing whether two cells are in the same tracking area t or not, α the probability that a UE has to be paged, C^p the overhead resulting from one paging it. Suppose a network is made up of two cells a

and b , if each of the cells belong to a different TA then when a UE moves between these two cells the TAU procedure is executed which costs $c^u h_{ij} (1 - S_{ij}(t))$. $S_{ij}(t)$ is 0 since cells a and b belong to different TAs. However the paging cost $(\alpha c^p S_{ij}(t))$ resolves to zero.

The problem is re-optimized by moving a given cell from one TA to another. At this stage, addition of a new TA is not allowed but deleting an empty TA is allowed. The UE distribution parameter u_i is used to measure the effect of interruption of cell i .

Let $d(t, t^0)$ be a binary vector that contains cells that have been moved to a new TA i.e.

$d_i(t, t^0) = 1$ if and only if $t_i \neq t_i^0, i \in n$. The cost of reconfiguration due to service interruption is calculated as follows:

$$C_R(t) = u_i * d(t, t^0) \quad (3.2)$$

3.5.1 Genetic Algorithm

There are two reasons in approaching this problem with a genetic algorithm:

1. The solution vector can be easily represented as a chromosome of size n , where n is the number of cells.
2. The GA generates a group of solutions that can be used in determining the pareto-optimal solution in one run.

The GA algorithm is summarized in Figure 3.1:

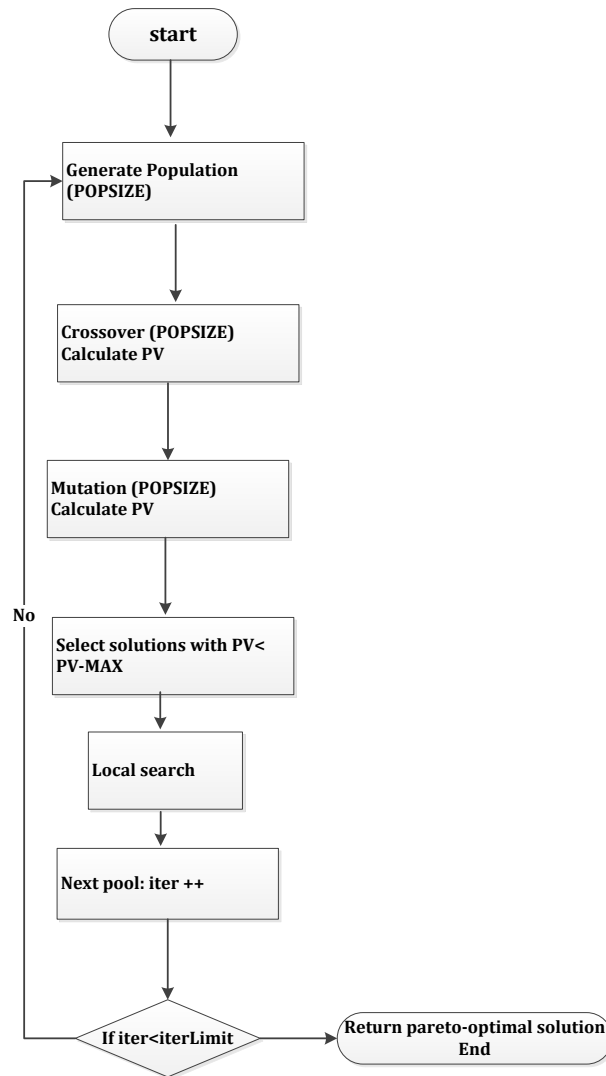


Figure 3.1 Genetic Algorithm flow chart

The GA algorithm can be summarized through several stages: *population initialization, crossover, mutation*, and then *PV local search algorithm*. In the phase of *population initialization*, the initial pool generation plays a key role in coming up with good solutions. Hence, the initial pool must be highly populated. A pareto-optimal solution is used as a starting point. A local search algorithm is applied to come up with a variety in the initial pool. In this algorithm cells that can be moved in respect of the remaining budget

are taken into consideration in order to come up with a movement with the largest improvement in signalling overhead. This is repeated until no further improvement can be obtained.

The first population will be comprised of the initial pool. The genetic algorithm randomly chooses a configuration from the initial pool and changes the TA configuration of 20% of the cells generating the rest of the population. This is repeated until a fixed population size is reached. Reconfiguration is carried out if the cell is on the boundary of its TA and the TA of this cell can only be altered to that of a neighbouring TA. This is to ensure good configurations.

In the *Crossover phase*, first the PV of the entire population is determined. Two parents who have low preference values are chosen. Two points in the configuration are randomly chosen and swapping is done between these two parents at these points as shown in Figure 3.2. This is done so that characteristics of the parents are inherited by the offspring. The crossover operation is repeated until the number of offspring becomes equal to POPSIZE. It is important to make sure that the parents are not identical so that the offspring generated are not the same and that the parents are different in at least one position of the randomly chosen points.

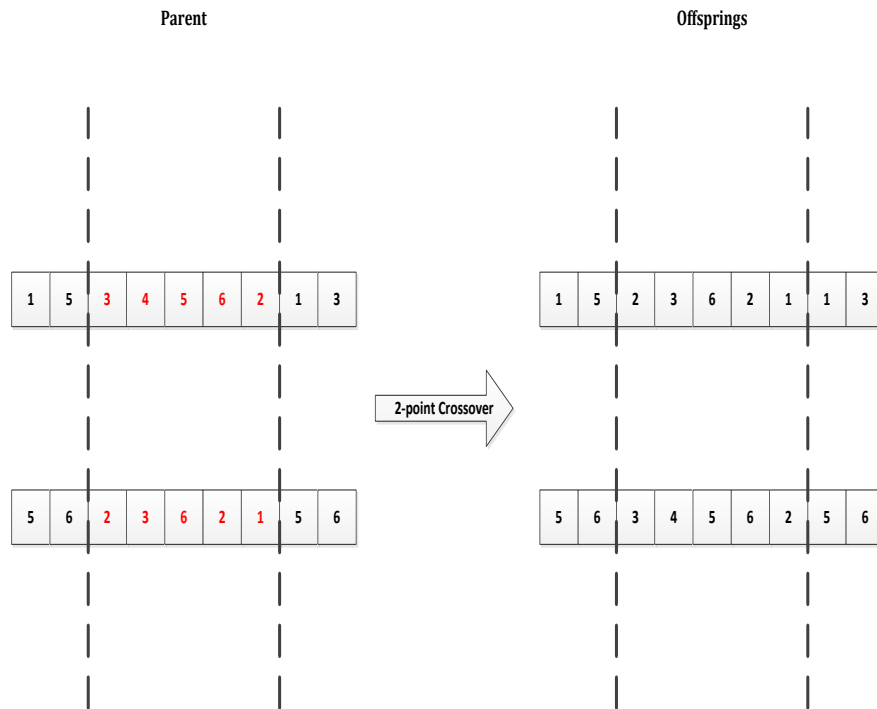


Figure 3.2 Crossover procedure

In the mutation phase, chromosomes that have not been formed during crossover are generated during mutation to enhance diversity. Mutation can only take place if the cell is on the boundary of its TA and the TA of this cell can only be altered to that of a neighbouring TA. Configurations with low PVs are preferred here. Only 5% of the elements are mutated. The mutation operation is repeated POPSIZE times.

Finally the *PV local search algorithm* is used to improve the results generated by the GA algorithm. Solutions having a PV less than a set threshold PVMAX are used as an input to provide improved solutions. Cells in a given solution are assigned to neighbouring TAs one by one and the move which results in low PV is chosen as long as the point defined by signalling overhead and reconfiguration has not been visited before. If unvisited

points with low PV no longer exist then points with equal PV are considered. The visited points at every iteration are stored to prevent redundancy. The LS is used at this stage to provide new solutions with lower PV and to search for new pareto-optimal solutions. Termination occurs when all remaining points are already visited or moves are of higher PV.

In order to improve the efficiency of the GA, the visited and the PV matrix are implemented. Every pixel in the visited matrix represents a given interval of the signalling overhead of a given number of solutions. When an element in the visited matrix is one this implies that a solution with signalling overhead that falls within a given interval has been already visited otherwise the value is zero. The PV matrix is of the same size as the visited matrix; with every pixel representing a given interval of signalling overhead, every time a new solution is found all the dominating elements to the right and up of the pixel of the new solution are incremented by one.

3.5.2 Integer programming model (IPM)

To describe the trade-off between $C_{so}(t)$ and $C_R(t)$ the problem is modelled with these bi-objective functions as follows:

Minimize: $(C_{SO}(t), C_R(t))$,

Subject to:

$$s_{ij} = \begin{cases} 1 & t_i = t_j \\ 0 & otherwise \end{cases} \quad (3.3)$$

$$d(t, t^0) = \begin{cases} 1 & t \neq t^0 \\ 0 & otherwise \end{cases}$$

$d(t, t^0)$ is a binary vector where t^0 represents the initial TA of a cell and t is the new TA of the cell. If the TA has changed $d(t, t^0) = 1$.

Coming up with a pareto-optimal solution is the main goal in solving this problem. A pareto-optimal solution is one in which it is impossible to improve an objective function without worsening the other. The TA re-optimization problem is solved repeatedly for different budget values B as shown in the equation below:

$$C_R(t) \leq B$$

This model contains two sets of binary variables:

- P_{it} is 1 when i belongs to TA t and 0 otherwise
- S_{ij} is 1 when i and j are in the same TA and 0 otherwise

The integer-programming model is represented as follows:

$$\text{Minimize: } C_{so} = \sum_{i \in n} \sum_{j \in n: j \neq i} (c^u h_{ij} (1 - S_{ij}(t)) + (\alpha c^p u_i S_{ij}(t))) \quad (3.4)$$

Subject to:

$$\sum_{t \in T} p_{it} = 1 \quad (\forall i \in n) \quad (3.5)$$

$$p_{it} + p_{jt} - 1 \leq s_{ij} \quad (i, j \in n, t \in T) \quad (3.6)$$

$$s_{ij} + p_{it} - 1 \leq p_{jt} \quad (i, j \in n, t \in T) \quad (3.7)$$

$$s_{ij} + s_{jk} - s_{ik} \leq 1 \quad (i, j, k \in n) \quad (3.8)$$

$$\sum_{i \in n} u_i (1 - p_{it}^0) \leq B \quad (3.9)$$

Constraint (3.5) assures that each cell belongs to a single TA. If $P_{it} = P_{jt} = 1$ this implies that j and i belong to the same TA t and $S_{ij} = 1$ as stated by constraint (3.6). If $P_{it} = 1$ and $P_{jt} = 0$ this implies that i belongs to TA t and j does not and $S_{ij} = 0$ as implied by constraint (3.7). Constraint (3.8) ensures that if two cells k and i belong to the TA as cell j then they all must belong to the same TA. Constraint (3.9) limits the number of UEs affected by reconfiguration by using the budget limit as a threshold. When p_{it} is 1 this means that i belongs to the currently assigned TA t and 0 otherwise.

The dominance-based approach is used to solve the problem. A performance metric called Preference Value (PV) is defined to measure the solution in terms of pareto-optimality. To determine the PV of a solution the entire solution space must be examined. This is not practical. In the proposed algorithm the points in signalling overhead and reconfiguration cost are accumulated to obtain an improved approximation of the PV.

3.5.3 **Results**

Experiments were conducted on different networks using the integer programming model and the dominance-based GA algorithm to solve the problem. The integer-programming model provided the optimal solutions. On the other hand the performance of

the GA algorithm differs between networks and provides close to optimal solutions for large-scale networks. The limitations of both approaches are shown in the tables 3.1 and 3.2.

<i>Advantages</i>	<i>Disadvantages</i>
Provides close to optimal solutions in large-scale networks	Not efficient for small networks
	High reconfiguration cost
	Requires lot of memory when keeping track of the visited solutions

Table 3.1 Advantages and disadvantages of GA

<i>Advantages</i>	<i>Disadvantages</i>
Provides optimal solutions for small networks	Cannot be used to solve large networks

Table 3.2 Advantages and disadvantages of IPM

3.6 Optimization of TAC configuration in mobile communication systems: A tabu search approach [16]

In [16] the main goal is to configure the Tracking Area Code (TAC) (group of cells) to maximize the paging success rate. The mobility (handover) patterns of mobile users as well as the TAC size and the capacity of paging traffic are used to construct a TAC. The paging procedure is explained as follows:

When a packet arrives for a given UE a paging message is broadcast to all the UEs in the TAC. If paging fails the passing message is broadcast to all the cells in that area. The aim is to maximize the paging success rate of the first paging procedure executed. For TAC optimization, the system model for TAC configuration is as follows:

- h_{ij} : handover ratio that a user moves from cell i to cell j , where

$$\sum_{i \in N} h_{ij} = 1$$
- λ_i : average paging traffic load for cell i , which is calculated as the number of RRC connections established in the cell;
- N : a group of total cells in the area, with the size of n ;
- M : a group of TACs in the area, with the size of m ;
- S_{TAC} : the maximally allowable number of cells for a TAC
- C_{TAC} : the maximum paging traffic load allowable for a TAC
- x_{ik} : a decision variable, $x_{ik} = 1$ if cell i is assigned to TAC k , $x_{ik} = 0$ otherwise.

The optimization model for TAC optimization is formulated as follows:

$$\text{Maximize: } \sum_{k \in M} \sum_{i \in N} \sum_{j \in N} \lambda_i * h_{ij} * x_{ik} * x_{jk}$$

Where $\lambda_i * h_{ij} = \text{Paging traffic load}$

Subject to:

$$\sum_{k \in M} x_{ik} = 1, \forall i \in N$$

$$\sum_{i \in N} x_{ik} \leq S_{TAC}, \forall k \in M$$

$$\sum_{i \in N} \lambda_i * x_{ik} * x_{jk} \leq C_{TAC}, \forall k \in M$$

$$x_{ik} = 1 \text{ or } 0, \forall k \in M, i \in N$$

The first constraint makes sure that each cell is assigned to one TAC. The second constraint limits the number of cells associated to a given TAC. The third constraint guarantees that the paging load in TAC does not exceed CTAC.

The paging success probability (PSP) represents the ratio of successfully paged traffic (paging success ratio (PSR)) divided by the total paging traffic in the network. The performance of the network is measured with respect to the PSP.

$$PSP = PSR / \sum_{i \in N} \lambda_i$$

The main difference between [14] and [16] is the formulation of the problem and the constraints they are subjected to. In the GA the signaling overhead C_{so} consists of the TAU overhead which is zero when the cells are in the same tracking area and the paging overhead which is zero when the cells are in different tracking areas. The reconfiguration cost is limited to a given value when choosing neighboring solutions to be visited in the embedded local search algorithm. In the tabu search algorithm cells are partitioned into TACs. The size of a TAC is limited and every cell must belong to a TAC. The handover ratio h_{ij} and the paging traffic λ_i determine the paging traffic load. The signaling overhead objective function only takes into account the paging overhead without considering the TAU overhead as mentioned in C_{so} . In the tabu search algorithm the performance of the

network is measured with respect to the PSP. In the GA results are measured in terms of the signalling overhead and reconfiguration cost.

3.7 Automatic Replanning of Tracking Areas in Cellular Networks [20]

In this paper, the TA is reconfigured using an automatic method that specifies when and how this is done in order to minimize the signaling overhead. Handover and paging statistics are used to generate graphs representing UEs mobility and traffic. Then, clustering algorithms are applied to these graphs to obtain similar user trends in order to come up with a TA replan at the end of each period. Generate a TA plan of correlated periods using the graph-partitioning algorithm based on previous similar characteristics. Results showed that there was a decrease in the number of TAU and paging requests by only altering the TA plan twice a week.

CHAPTER 4

USING TABU SEARCH TO SOLVE THE TA RECONFIGURATION PROBLEM

In this chapter, a Tabu Search (TS) heuristic method is used to solve the TA reconfiguration problem. The first section of this chapter describes all the terms that are used to formulate the problem. The second section defines the general TS. Next, the problem is formulated using a TS algorithm along with some added features to improve the performance.

4.1 Notions

To formulate the problem the following notations will be used:

- s^0 : is the current solution
- s^* : is the improved solution in terms of signalling overhead.
- t_i : TA of cell i
- u_i : is the number of UEs in cell i for a given period in time.
- h_{ij} : is the number of UEs moving from cell i to j .
- C^p : is the overhead resulting from one paging
- C^u : is the cost of one location update
- N : set of cells in the network

- n : number of cells in the network
- α : is the probability that a UE has to be paged
- t_0 : is the TA of cell a before applying the move $m(a,b)$ which is the swapping of the TAs cells a and b
- K_{max} : is the threshold used to measure the number of iterations since there has not been an improvement in the fitness of the solution, in the long-term memory.
- Nb_{iter} : is the number of iterations so far.
- $Best_{iter}$: is the iteration at which the best solution is found.
- ik_{max} : is the threshold used to measure the number of iterations since there has not been an improvement in the fitness of the solution, when $Nb_{iter} - Best_{iter} > ik_{max}$ the algorithm terminates.
- $IterSinceLastImpv$: is the number of iterations since there has not been an improvement in the fitness of the solution
- NB_{START} : is the maximum number of iterations before the long-term memory is executed.
- $CountToRestart$: keeps the count of iterations since $Nb_{iter} - Best_{iter} > K_{max}$ until it reaches the threshold value NB_{START} .
- $S_{ij}(t)$: is an element in the symmetric and binary matrix $S(t)$ that represents whether two cells are in the same tracking area or not.
- $d(t,t^0)$: is a binary vector that contains cells that have been moved to a new tracking area i.e. $d_i(t,t^0)=1$ if and only if $t_i \neq t_i^0, i \in N$.

- s_i : is solution at iteration i .
- $N(s)$: is the set of solutions that are generated by applying one move to solution s . i.e the neighbouring solutions.
- $maxVal$: is the maximum tenure threshold that can be assigned to non feasible solutions.
- k : is size of tabu List

4.2 Basic Principle

Tabu search is a meta-heuristic function because it can be applied to different occurrences of the same problem to generate good solutions. It is a refinement to other descending algorithms since it prevents the occurrence of local minima. The algorithm starts by marching towards non-feasible solutions and their corresponding swaps will be stored in the tabu list to prevent occurrence of the same solutions. Solutions that do not necessarily improve the objective function are tolerated hoping that they can lead to better solutions later on. However, accepting these solutions might lead the search to already visited solutions i.e. a cycle, hence the requirement to use a tabu list TL . Tabu list TL contains the latest m visited solutions. When searching for neighboring solutions $N(s)$, solutions that are already in the tabu list are removed. It is used to avoid cycles as mentioned above; however it requires a lot of memory. It is also beneficial to return to already visited solutions to take the search in other directions hence the size of the tabu list, which is denoted as k must be chosen carefully. The algorithm terminates when ik_{max}

iterations is reached or all of the neighbouring solutions are already in the tabu list i.e $N(s)$ -
 $TL = \emptyset$ [17-18].

To minimize the objective function we start with an initial solution and apply moves that allow the transition from solution s_i to s_{i+1} where s_{i+1} belongs $N(s_i)$. The tabu search can be differentiated from other descending algorithms by the short memory structure used. This mechanism tolerates the transition to less good solutions to prohibit local optima and prevents the risk of cycles [17-18]. Figure 4.1 represents a simple TS method. Every time $N(s_i)$ are generated, the move that is associated with the solution is guaranteed not to be in the TL unless it provides a solution with a lower cost than the current best solution. Every time a move is executed the Tabu List is updated. The algorithm terminates when all possible moves are already in the tabu list or a maximum number of iterations since last improvement has been reached.

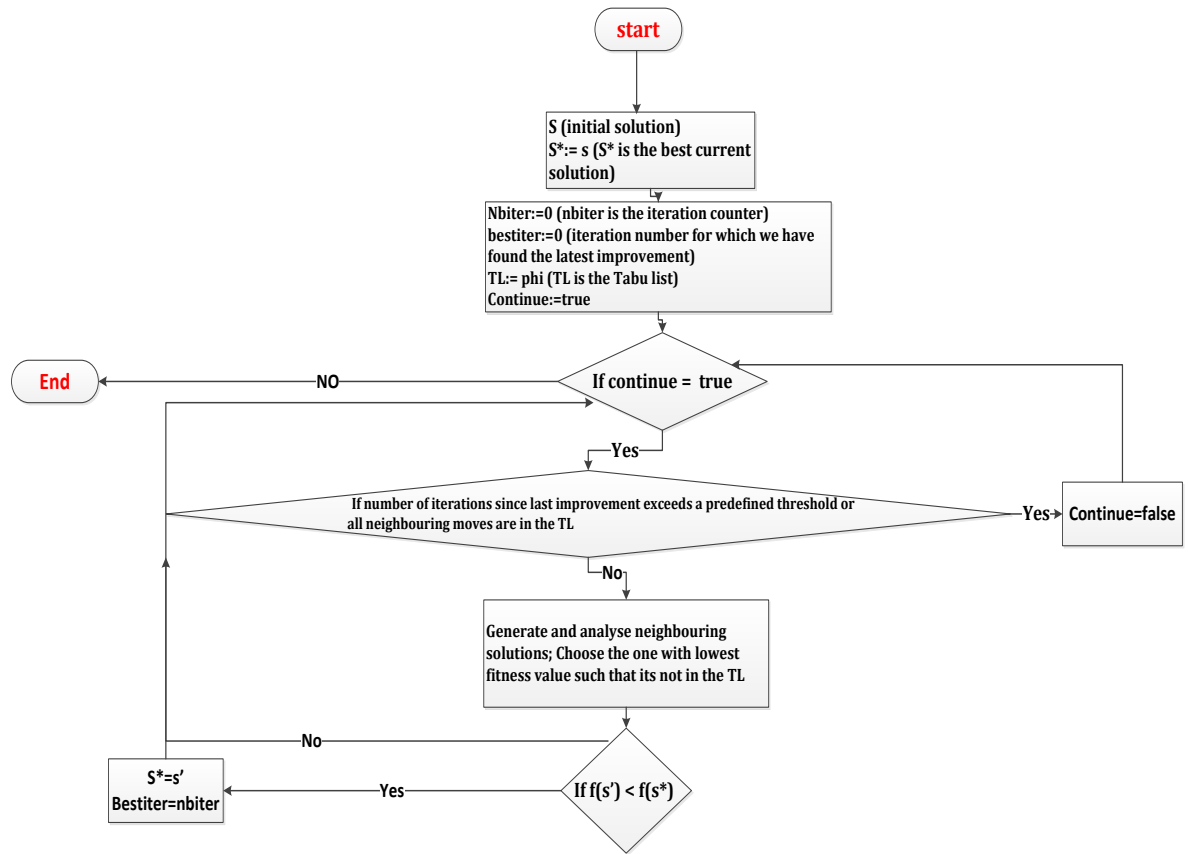


Figure 4.1 General algorithm of a TS method

4.3 Problem formulation

A cell is assigned to new TA to come up with new improved solutions. The subset of the new solutions will contain cells that are assigned to new TAs and these newly assigned TAs. This is known as reconfiguration. The main goal here is to obtain a solution with a reduced paging and TAU overhead. This implies that whether a cell can be assigned to a new TA and which TA it should be allocated to depends on the assignment of other cells. To improve the performance of the algorithm a Tenure Matrix (TM) is used instead

of a tabu list. In TM, tabu moves are managed by assigning to them tenures instead of maintaining them in a list. This makes it easier to check whether a given move is tabu by just checking its indexes instead of going over the list as will be described later on.

4.4 Features of Tabu search

4.4.1 Encoding

A solution is represented by an array of integers with a length, n , equal to the number of cells in the network as shown in Figure 4.2 i.e. $n=|N|$. The indices in the array represent the IDs of the cells. The value of each array element represents the ID of the TA to which the cell is assigned. Hence, $E(i)$ is the TA to which cell i is assigned. Figure 4.2 shows an example in which $E(0) = E(3) = E(8) = 6$, $E(1) = E(2) = E(4) = E(7) = 4$, $E(5) = E(9) = E(8) = 7$, and $E(6) = 3$; which means cells 0, 3, 8 are assigned to TA 6, cells 1, 2, 4, 7 to TA 4, and cells 5, 9 to TA 7 and cell 6 belongs to TA 3.

i	0	1	2	3	4	5	6	7	8	9
E	6	4	4	6	4	7	3	4	6	7

Figure 4.2 Encoding example of using an array to represent a solution

4.4.2 Fitness

The fitness of every solution is based on the signalling overhead resulted from the assignment of the TAs to the cells. To assess the fitness of a solution, the objective function of a solution S , $f(S)$, is formulated as the total signalling overhead, C_{so} , resulting from

paging and location update as per equation (4.1). The first addend in equation (4.1) represents the update overhead resulting from the UE moving from cell i to j while the second addend represents the paging overhead as a result of an incoming call. To understand it better, suppose a topology of n cells and T tracking areas then let C^u denote the cost of one location update, h_{ij} the number of UEs moving from cell i to j , $S_{ij}(t)$ an element in a symmetric and binary matrix $S(t)$ representing whether two cells are in the same tracking area t or not, α the probability that a UE has to be paged, C^p the overhead resulting from one paging it.

The objective function is formulated as follows (signalling overhead):

$$f(S) = C_{so} = \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} \left(c^u h_{ij} (1 - S_{ij}) + \alpha c^p S_{ij}(t) \right) \quad (4.1)$$

Subject to:

$$\sum_{i \in T} p_{it} = 1 \quad (\forall i \in n) \quad (4.2)$$

Where p_{it} is 1 when cell i belongs to TA t (i.e. $E(i)=t$) and 0 otherwise. This assures that each cell belongs to a single TA. The problem is re-optimized and new solutions are generated by swapping two given cells. Hence, *move* $m(i,j)$ is equivalent to swap ($E(i)$, $E(j)$) which is defined as assigning cell i to $E(j)$ (i.e. TA of cell j) and assigning cell j to $E(i)$ (i.e., previous TA of cell i). Assume that t_i is the TA of cell i , then altering the assignment of cells to TAs results in a reconfiguration cost, $C_R(t)$ given in equation (4.3) in which u_i is the number of UEs in cell i for a given period in time and $d(t, t^o)$ is a binary vector that contains cells that have been moved to a new tracking area i.e. $d_i(t, t^o)=1$ if and only if

$t_i \neq t_i^0, i \in N$ and t_i and t_i^0 are the new and old TAs of cell i respectively (i.e., cell i was assigned to a new TA).

$$C_R(t) = u_i * d(t, t^0) \quad (4.3)$$

4.4.3 Algorithm Intuition

The intuition of our proposed approach is shown in Figure 4.3. We normally start with a good feasible solution as initial solution, then we move towards a better less costly solution by performing a move to generate the set, $N(S)$, of neighborhood solutions that are reached from current solution S , and we choose the one with the lowest fitness value. In our algorithm, we use two memory techniques, the short memory structure and the long memory structure. After generating a good initial solution from real data, the short memory is used to refine it while preventing cycles. To do so, an aspiration criterion is defined to allow a tabu move to be executed only when yielding a better solution. To reduce the number of non-feasible solutions, a penalty is assigned to the moves leading to them through a call back mechanism. Based on the statistics of the solutions already visited and if no improvement is witnessed after a predefined number of iterations, the long term memory [18] is entered to diversify the search to make it possible to restart with a new initial solution that can be generated by swapping the least frequently permuted cells before switching back to the short memory structure.

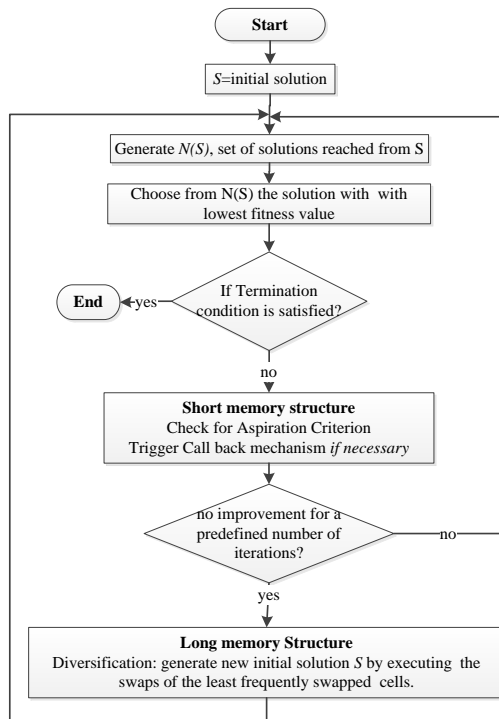


Figure 4.3 Memory mechanisms in TS

A low level algorithm of our proposed approach is shown in Figure 4.4 and is described shortly. As previously mentioned, we normally start with good initial solution as shown in the input section of Figure 4.4 (lines 1-8), then we move toward a better neighboring solution (lines 11-16). The constraint of having each cell assigned to a TA is taken into consideration but the capacity constraint of the TA is not considered in order to traverse as many solutions as possible to increase the possibility of finding better solutions. Each solution is associated with an intrinsic cost, which is simply the value of the objective function.

Input:

1. E : encoding vector in which cell i is assigned to TA $E(i)$
2. D : Delay threshold before triggering call back mechanism
3. Starting solution = S
4. Initialize Tenure Matrix TM , Frequency matrix FM
5. bestSolution $S^* \leftarrow S$
6. Set Nb_{iter} , $best_{iter}$, $IterSinceLastImpv$, $CountToRestart \leftarrow 0$
7. Set $maxVal \leftarrow 8$ (maximum threshold tenure).
8. Set $ik_{max}, k_{max} \leftarrow$ Average values

Output:

9. bestSolution S^*
10. **repeat for each $E(i)$ where i goes from 0 to $n-1$**
11. **for each $E(j)$ where j goes from i to $n-1$**
12. Swap ($E(i)$, $E(j)$) to generate a solution S' in $N(S)$
13. Calculate the objective function $f(S')$
14. Store $f(S')$ in list L sorted in ascending order
15. **endfor**
16. Choose the best S' from L
17. $Nb_{iter}++$
18. **if** ($Nb_{iter} - best_{iter} > ik_{max}$) or (no available moves in the TM)
19. Termination Condition
20. **endif**
21. **if** (tenure value in $TM > 0$) and ($f(S') < f(S^*)$)
22. ASPIRATION CRITERIA
23. $S^* \leftarrow S'$
24. **else**
25. **if** ($f(S') < f(S^*)$)
26. $IterSinceLastImpv \leftarrow 0$
27. $S^* \leftarrow S'$
28. $best_{iter} \leftarrow Nb_{iter}$
29. **else**
30. $IterSinceLastImpv++$
31. **endif**
32. **endif**
33. **if** ($IterSinceLastImpv > D$)
34. trigger CALL BACK MECHANISM
35. penalize the non-feasible solution by incrementing tenure until $maxVal$
36. **else**
37. execute permutation
38. decrement all tenures
39. update TM with tenure of current move
40. update FM
41. **if** ($Nb_{iter} - best_{iter} > k_{max}$)
42. $countToRestart++$
43. **if** ($countToRestart > NB_{START}$)
44. execute diversification
45. $countToRestart=0$
46. **endif**
47. **endif**
48. **endif**
49. **endrepeast**
50. **return** S^*

Figure 4.4 Proposed TS based Algorithm

The short memory improves the initial solution by performing permutations in order to improve the current solution. This is achieved by minimizing the cost function. To describe it further we define the move $m(a,b)$ as: $m(a,b)$: swap cells a and b . This move does not consider capacity constraint and constitutes of permuting two cells. As mentioned earlier the neighborhood $N(S)$ are the solutions that are reached from S by executing a single move $m(a,b)$ to S .

4.4.4 Tenure Matrix (TM)

In our approach a tenure matrix (TM) is used to track tabu moves that might lead the search toward non-feasible solutions. Indeed, suppose a tenure matrix (TM) of size $n*n$ (where n represents the number of cells in the network) is used to store the tabu moves. Every time a swap is executed the cells in the matrix corresponding to the swap is set to a tenure value representing the number of iterations during which a given move is considered as tabu (lines 37-39 in Figure 4.4). For example, consider a network of four cells $\{1, 2, 3, 4\}$ then the initial TM is shown in Figure 4.5 (a) in which the first row and column's values represent the cells and the other values are the tenure values assigned to each cell permutation. After executing the $m(3, 4)$ as shown in Figure 4.5 (b), the cells $(3, 4)$ and $(4, 3)$ will be set to a tenure value of 3 (i.e., the move will tabu during next three iterations); then this value will be decremented at every other iteration until it reaches a value of 0 (i.e., the move can be considered as a non-tabu). It is worth to mention that both cells $(3, 4)$ and

(4, 3) are set to same tenure value in order to prevent reverse iterations in the future. In the next iteration, the performed swap/move (i.e., $m(2,1)$ in Figure 4.5 (c)) is also accounted for in the TM and all other previous non-zero values are decremented.

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

(a)

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	3
4	0	0	3	0

(b)

	1	2	3	4
1	0	3	0	0
2	3	0	0	0
3	0	0	0	2
4	0	0	2	0

(c)

Figure 4.5 (a) Initial Matrix, (b) Tenure Matrix after swap (3,4), (c) Tenure Matrix after swap (2,1)

When the solutions visited are non-feasible, it is important to make sure that the search is redirected towards feasible solutions, this is achieved by implementing the call back mechanism.

4.4.5 Aspiration Criterion

After starting with a good initial solution S , and generating the set $N(S)$ of neighboring solutions reached from S , it is sometimes beneficial to consider a tabu move in order to come up with a better solution. That is why an aspiration criterion is defined in the short memory structure to allow a tabu move to be executed only if it provides a better solution than the current best one. The algorithm (lines 11-16 of Figure 4.4) shows that all possible moves that can be executed to generate neighboring solutions are analyzed; sorted in a list based on their objective function values, and then the move with the lowest objective function is selected. Lines 21-23 check if this move is tabu (i.e., has a tenure

value > 0) but yields a fitness value lower than the value of the best current solution. If it is, then the aspiration criterion consists of switching it to a non-tabu move.

4.4.6 Call back mechanism

It is obvious that when the visited solutions are non-feasible after a predefined number of iterations, the search is redirected towards feasible solutions by implementing the call back mechanism as shown in lines 33-35 of Figure 4.4. Indeed, the algorithm uses *IterSinceLastImpv* to track the number of iterations since there has not been an improvement (line 30 in Figure 4.4). If this number exceeds a certain threshold, D , the call back mechanism is activated by penalizing the next occurring non-feasible solutions through incrementing their tenure values in TM to prevent, in the next iterations, permutations that do not result in better solutions. The tenure value is incremented gradually at every non-feasible solution until it reaches a maximal value of *maxVal* (i.e., the maximum tenure threshold that can be assigned to non-feasible solutions).

4.4.7 Frequency Matrix (FM)

A frequency matrix (FM) of size $n*n$ (n represents the number of cells) is used to keep track of the number of times each cell is being swapped. After a certain number of iterations, the elements in the FM with the lowest values might be considered in order to come up with unexplored solutions. This is known as the diversification. Its main objective is to produce a better initial solution than the original one if no improvement is witnessed

after a certain number of iterations. In Figure 4.6 (a) the first row and column's values represent the cells and the other values are the number of times each cell permutation has occurred. After executing the swap (3,4) as shown in Figure 4.6(b), the cells (3,4) and (4,3) will be incremented. In the next iteration the permutation executed (swap (2,1)) is also accounted for in FM (Figure 4.6(c)). Diversification is only allowed when reaching a threshold of K_{max} iterations with no improvement as shown in line 41 of the algorithm in Figure 4.4. To describe further this phase, let $Best_{iter}$ denote the iteration number at which the best current solution has occurred, and Nb_{iter} be the number of iterations so far then when $nb_{iter} - best_{iter} > k_{max}$ (i.e., more than k_{max} iterations were performed without yielding an improvement), as shown in lines 41-46 of Figure 4.4, the algorithm prepares to switch to long memory by monitoring NB_{START} iterations with no improvement. If no improvement obtained in NB_{START} iterations after reaching the threshold k_{max} , the algorithm generates a new starting solution, by swapping the cells with lowest value in each column in FM. Having now, new initial solution, the short memory is executed once again. NB_{START} is a constant that represents the number of restart cycles.

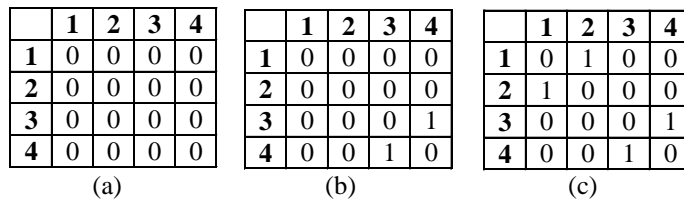


Figure 4.6 (a) Initial Frequency Matrix, (b) Frequency Matrix after swap(3,4), (c) Frequency Matrix after swap (2,1)

4.4.8 Long-term memory mechanism

This is only allowed when $nb_{iter}-best_{iter}>k_{max}$. The main objective of this mechanism is to produce better solutions than the ones generated in the short memory mechanism. This is achieved by coming up with new starting solutions. The frequency matrix, which keeps track of the number of swaps of each cell, is evaluated to come up with new starting solutions. The lowest number of swaps of each column is generated and that swap is executed to come up with these new solutions. This is also known as the diversification phase. Having the initial solutions, the short-term memory is executed once again. NB_{START} is a constant that represents the number of restart cycles. Different values of NB_{START} will be analyzing and the performance will be highlighted.

4.4.9 Termination conditions

The termination condition of our algorithm is when the number of iterations since last improvement has reached a value of ik_{max} or when there are no more possible moves i.e. all possible moves are already in the TM (lines 18-20 in Figure 4.4). Increasing the value of ik_{max} might increase the possibility of leading the search towards more feasible regions. However it is also possible for very large ik_{max} values to drift the search away from feasible solutions with a little chance of bringing it back to the right track. Figure 4.7 represents a flow chart of the tabu search algorithm in Figure 4.4.

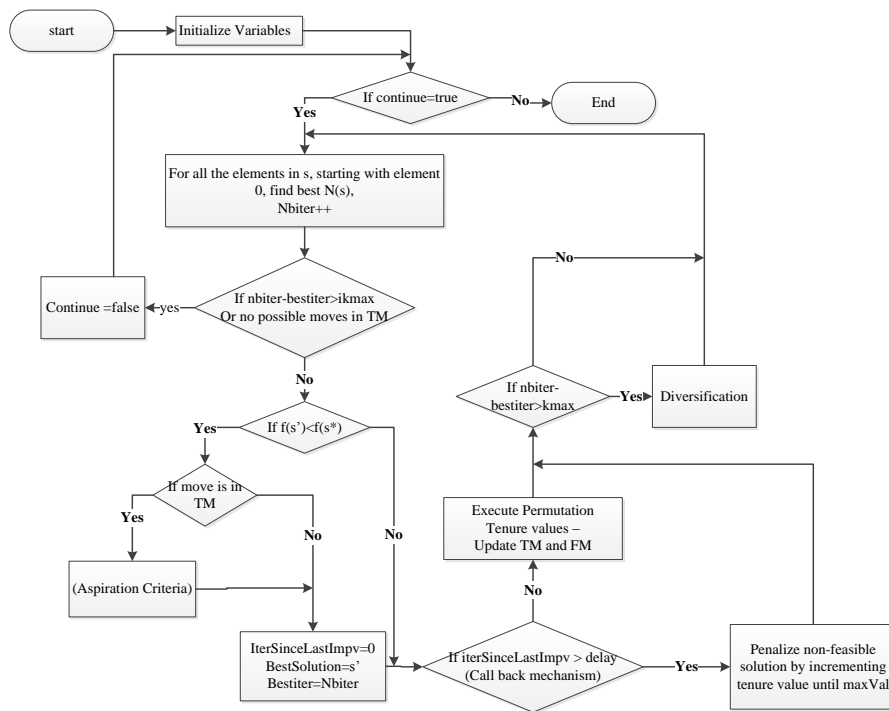


Figure 4.7 Tabu search algorithm

CHAPTER 5

FORMULATING THE RECONFIGURATION PROBLEM USING GA AND IPM

In this chapter, we describe the GA and IPM formulation to solve the TA reconfiguration problem. We have used [14] as our reference but altered the description to reflect our implementation of the GA. The first section of this chapter defines the general GA, followed by a definition of all the terms that are used to formulate the problem. A brief definition of the algorithm used is represented followed by the formulation of the problem using the GA. Next a system model is defined to represent the TA reconfiguration problem. The formulated Integer programming model is solved repeatedly for different budget values B as in [14]. When all the variables in Integer programming (IP) are required to be integers then the model is called pure integer programming model [23].

5.1 Genetic Algorithm

5.1.1 Definition

The Genetic Algorithm is an adaptive search mechanism that is an imitation of the natural biological processes. It is used to solve search and optimization problems. Starting with an initial pool, chromosomes are randomly selected to generate offspring through crossover and mutation operators [21]. A general definition of a GA is represented in Figure 5.1.

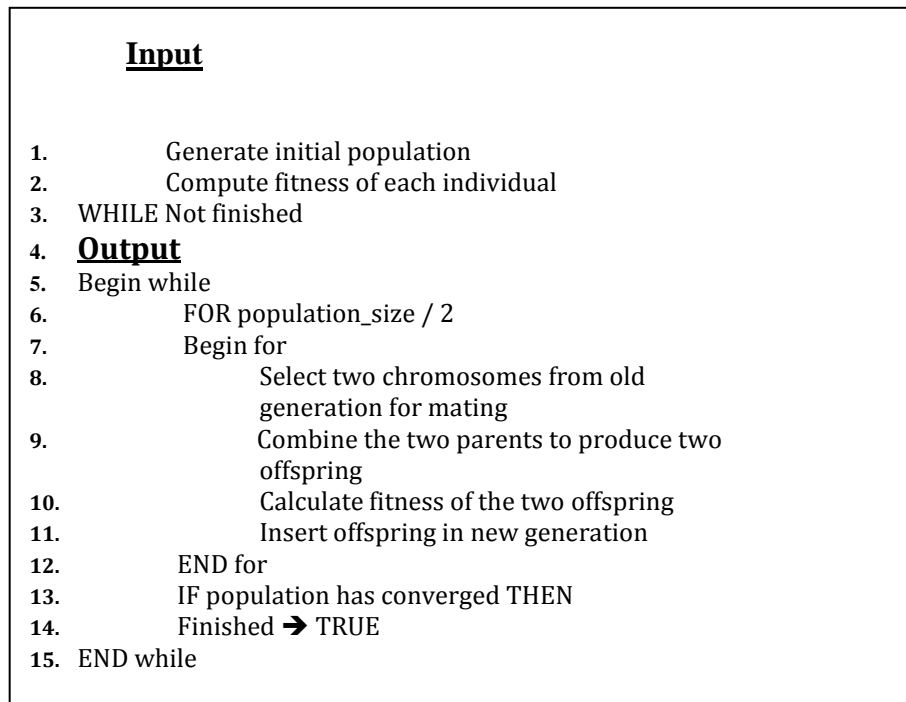


Figure 5.1 General Genetic Algorithm

5.1.2 Notations

Refer to the notations used in chapter 3, section 3.5.

- POPSIZE: Number of children to be generated in the network.

5.1.3 Features of GA

5.1.3.1 Encoding

A solution is represented by an array of integers with a length equal to the number of cells in the network as shown in Figure 5.2. The indices in the array represent the IDs of

the cells. The value of each array element represents the ID of the TA to which the cell is assigned. Hence, $E(i)$ is the TA to which cell i is assigned. Figure 5.2 shows an example in which $E(0) = E(3) = E(8) = 6$, $E(1) = E(2) = E(4) = E(7) = 4$, $E(5) = E(9) = E(8) = 7$, and $E(6) = 3$; which means cells 0, 3, 8 are assigned to TA 6, cells 1, 2, 4, 7 to TA 4, and cells 5, 9 to TA 7 and cell 6 belongs to TA 3.

i	0	1	2	3	4	5	6	7	8	9
E	6	4	4	6	4	7	3	4	6	7

Figure 5.2 Encoding example of using an array to represent a solution

5.1.3.2 Fitness

The fitness of every solution is based on the signalling overhead resulted from the assignment of the TAs to the cells. To assess the fitness of a solution, the objective function of a solution S , $f(S)$, is formulated as the total signalling overhead, C_{so} , resulting from paging and location update as per equation (5.1).

The objective function is formulated as follows (signalling overhead):

$$f(S) = C_{so} = \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} (c^u h_{ij} (1 - S_{ij}) + \alpha c^p S_{ij}(t)) \quad (5.1)$$

subject to:

$$\sum_{i \in T} p_{it} = 1 \quad (\forall i \in n) \quad (5.2)$$

where p_{it} is 1 when cell i belongs to TA t (i.e. $E(i)=t$) and 0 otherwise. This assures that each cell belongs to a single TA. The problem is re-optimized and new solutions are

generated by executing the GA. Assume that t_i is the TA of cell i , then altering the assignment of cells to TAs results in a reconfiguration cost, $C_R(t)$ given in equation (5.3) in which u_i is the number of UEs in cell i for a given period in time and $d(t, t^0)$ is a binary vector that contains cells that have been moved to a new tracking area i.e. $d_i(t, t^0)=1$ if and only if $t_i \neq t_i^0$, $i \in n$ and t_i and t_i^0 are the new and old TAs of cell i respectively (i.e., cell i was assigned to a new TA).

$$C_R(t) = u_i * d(t, t^0) \quad (5.3)$$

5.1.3.3 Adaptation of the GA

The GA algorithm can be summarized through several stages as shown in Figure 5.3: population initialization, crossover, and then mutation. In the phase of population initialization, the initial pool generation plays a key role in coming up with good solutions. Hence, the initial pool must be highly populated. Good feasible solutions are used as a starting point.

The first population is compromised of the initial pool. The genetic algorithm randomly chooses a configuration from the initial pool and changes the TA configuration of 20% of the cells generating the rest of the population. This is repeated until a fixed population size is reached. Reconfiguration is carried out if the cell is on the boundary of its TA and the TA of this cell can only be altered to that of a neighboring TA. This is to ensure good configurations.

In the Crossover phase, first the fitness value of the entire population is determined. Two parents who have low fitness values are chosen. Two points in the configuration are randomly chosen and swapping is done between these two parents at these points. This is done so that characteristics of the parents are inherited by the offspring. The crossover operation is repeated until the number of offspring becomes equal to POPSIZE. It is important to make sure that the parents are not identical so that the offspring generated are not the same and that the parents are different in at least one position of the randomly chosen points.

In the mutation phase, chromosomes that have not been formed during crossover are generated during mutation to enhance diversity. Mutation can only take place if the cell is on the boundary of its TA and the TA of this cell can only be altered to that of a neighbouring TA. Configurations with low fitness values are preferred here. Only 5% of the elements are mutated. The mutation operation is repeated POPSIZE times.

Termination occurs when the iteration limit is exceeded. The main difference in our GA implementation and the one in [14] is that we have not implemented the local search algorithm and the visited matrix that were implemented in the GA in [14]. The visited matrix only keeps track of the visited solutions without keeping track of the moves that lead to these solutions so keeping track of these solutions makes no improvement. However we have used the local search algorithm in proposed TS approach.

Figure 5.4 illustrates an example of a single iteration of the implemented GA. For simplicity we assume that the size of the chromosome is 5 and the set of TAs T is $\{1,2,3\}$.

As long as the iteration limit is not exceeded step 1 and step 2 are executed.

In step one (crossover phase) the red highlighted elements in the array are swapped in chromosomes b and c to form new chromosomes e and f. This phase is repeated

$\text{POPSIZE}/2$ times. Next the mutation phase is executed where a chromosome of a low fitness value is chosen and only 5% of its elements are swapped. In this example

chromosome e is mutated to form chromosome g. This is added to the initial pool. Mutation is repeated POPSIZE times.

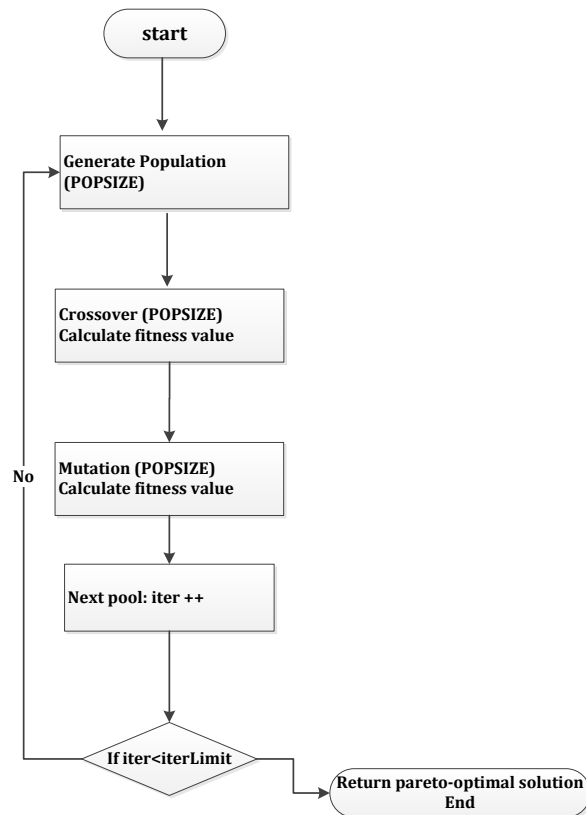


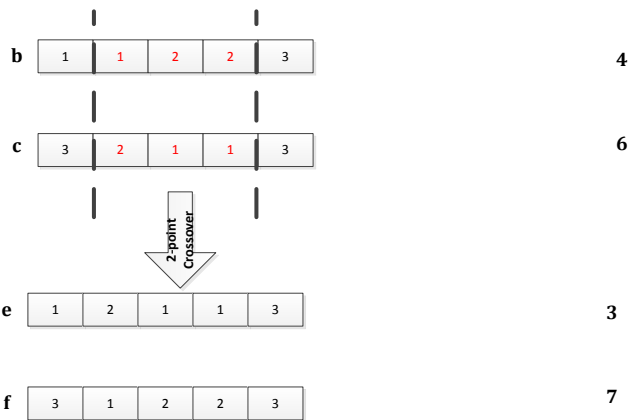
Figure 5.3 Implemented GA

	Initial population	Fitness Value
a	2 1 3 2 1	10
b	1 1 2 2 3	4
c	3 2 1 1 3	6
d	3 3 1 2 2	9

Step 1- Cross Over

Repeat POPSIZE/2 time (i.e until we get POPSIZE offsprings)

Choose chromosomes with low fitness values



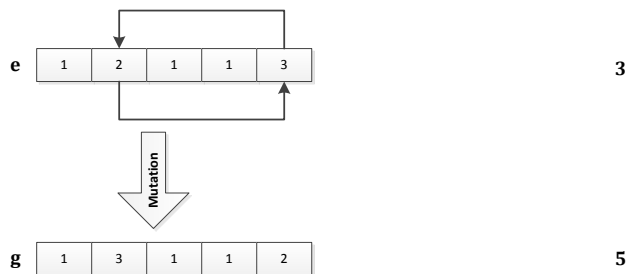
e and f are added to the initial population

Step 2- Mutation

Repeat POPSIZE time (i.e until we get POPSIZE offsprings)

Choose chromosomes with low fitness values

5% of the elements are mutated



g is added to the initial population

Figure 5.4 An Iteration illustrating the GA

5.1.4 Number of runs

To ensure statistical validity and credible simulation results, we calculated the number of runs required for achieving at least a 90% confidence level. First, we conducted 6 different runs. The results for the runs of simulations in GA are given in Table 5.1.

The standard deviation is calculated using the following formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n - 1}}$$

where σ is the standard deviation, X_i is the reading at run i , \bar{X} is the mean and n is the number of runs.

To achieve 90% confidence level, the required number of runs is calculated using the central limit theorem [43]. The +/- precision value used is 5%. The central limit theorem formula used in [25] to get the required number of runs is:

$$n = \left(\frac{100 \times z \times s}{r} \times \bar{X} \right)^2$$

where n is the number of runs, \bar{X} is the sample mean, r is the precision level, z is the normal variate, which is 1.645 constant for 90 percent confidence interval, and s is the standard deviation.

The required numbers of runs to achieve 90% confidence level were found to be 5, 1, 1 and 3 for the GA.

Table 5.1 Number of runs in GA

<i>Runs</i> \ <i>Recon. cost</i>	<i>10</i>	<i>20</i>	<i>30</i>	<i>40</i>
<i>1</i>	212073.4	209623.4	197848.4	195765.7
<i>2</i>	211980	210000	198600	195756
<i>3</i>	190000	220000	196848.9	210000
<i>4</i>	198000	200000	210000	180000
<i>5</i>	220000	209623.4	190000	195765.7
<i>6</i>	230800	209623.4	193789	195765.7
<i>Mean</i>	210475.6	209811.7	197847.7	195508.9
<i>Standard Deviation</i>	14720.75	6327.918	6742.424	9495.018
<i>z</i>	1.654	1.654	1.654	1.654
<i>r</i>	5	5	5	5
<i>Number of runs</i>	5.35288	0.99539	1.27087	2.58101

5.2 Integer Programming Model

The system model is an extension of the notations used in chapter 5.1.2. The signalling overhead $C_{so}(t)$ follows equation 5.1 and the reconfiguration cost, $C_R(t)$ follows equation 5.3. The goal in solving the problem is to produce a trade-off between $C_R(t)$ and $C_{so}(t)$. The bi-objective formulation of the problem is defined below:

$$\text{Min } C_{so}(t), C_R(t)$$

Subject to

$$S_{ij}(t) = \begin{cases} 1 & t_i = t_j \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

$$d_i(t, t^0) = \begin{cases} 1 & t_i^0 \neq t_i \\ 0 & \text{otherwise} \end{cases}$$

To solve the system model of equation (5.4) the $C_R(t)$ is minimized for several Budget values B . The model has two sets of binary variables:

- $s_{ij}(t)$ is 1 when i and j are in the same TA and 0 otherwise.
- p_{it} is 1 when cell i belongs to TA t and 0 otherwise.

The integer-programming model is represented as follows:

$$\text{Minimize: } C_{so} = \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} \left(c^u h_{ij} (1 - S_{ij}) + \alpha c^p S_{ij}(t) \right) \quad (5.5)$$

$$\sum_{t \in T} p_{it} = 1 \quad (\forall i \in n) \quad (5.6)$$

$$s_{ij} \leq p_{it} \quad (i, j \in n, t \in T) \quad (5.7)$$

$$s_{ij} \leq p_{jt} \quad (i, j \in n, t \in T) \quad (5.8)$$

$$s_{ij} \geq p_{it} + p_{jt} - 1 \quad (i, j \in n, t \in T) \quad (5.9)$$

$$\sum_{i \in n} u_i (1 - p_{it_i^0}) \leq B \quad (5.10)$$

Constraint (5.6) assures that each cell belongs to a single TA. If $P_{it} = P_{jt} = 1$ this implies that j and i belong to the same TA t and $S_{ij} = 1$ as stated by constraint (5.7, 5.8 and 5.9). If $P_{it} = 1$ and $P_{jt} = 0$ this implies that i belongs to TA t and j does not and $S_{ij} = 0$ as implied by constraint (5.7, 5.8 and 5.9). Constraint (5.10) limits the number of UEs affected by reconfiguration by using the budget limit as a threshold. When p_{it} is 1 this means that i belongs to the currently assigned TA t and 0 otherwise.

The model has been implemented using the Gurobi optimizer [23]. In Gurobi the Linear programming models are solved using the branch and bound algorithm (divides and conquers). First, all the constraints are ignored, the model is solved to produce the linear

programming (LP) relaxation of the original model. If the solution happens to satisfy the constraints then we are lucky to have found the optimal solution this fast. If not, then we search for a variable that is restricted to be an integer but whose value in the LP relaxation is a fraction. For example, let variable x be assigned to a value of 5.6 in the LP relaxation, two new constraints $x \geq 5$ and $x \leq 6$ are added to the model. If the original model is denoted as P^0 then the two new sub-models denoted as P^1 and P^2 are solved separately each considering one of the newly added constraints. The variable x is now called the branching variable, and it is said that we have now branched at x generating the two sub-optimal models P^1 and P^2 . Now that we have divided the problem into two sub-problems we take the one that generates the more optimal results. We recursively do the same to P^1 and P^2 to produce the linear programming relaxation and then the branching variables. This will eventually produce a search tree with P^0 as the root node, and every other node representing an Integer programming model. When we reach the leaf nodes then we have solved the original IPM. Each node the tree is a new IPM. The nodes shaded with straight lines in Figure 5.5 represent the leaf nodes and the node shaded with dots represents the optimal solution found.

.

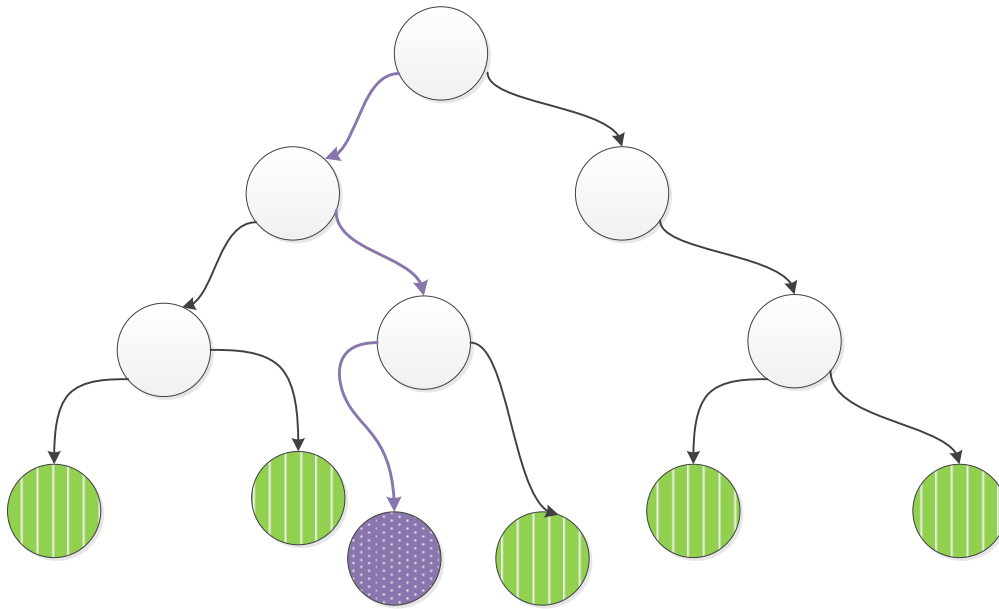


Figure 5.5 : Branch and bound search tree

CHAPTER 6

COMPUTATIONAL EXPERIENCE AND RESULTS

In this chapter, we evaluate the performance of the proposed solution and then compare it with that of the genetic algorithm and the integer programming model. In this regard, parameters of each memory mechanism in the TS approach are varied to highlight their effects on the algorithm.

6.1 Test generation

To choose a global reliable behaviour we were inspired by the test scenarios in [14] and [17]. We have used java in our implementation which assumes that the cells are arranged on a hexagonal platform of equal length and width with antennas at the centre of the cells. The IPM is implemented in Gurobi optimizer [23] and the computations for TS and GA were executed on an Intel Core i5 processor with 2.4 GHz speed and the IPM is executed on Intel(R) Xeon(R) CPU E5-2650 0 with 2 GHz speed.

The relationship between c^u and c^p is related to the radio resource consumption. The values of both the signaling and reconfiguration cost have no physical units. Hence, the values of the signaling cost and reconfiguration cost have no meaning unless compared to values computed by the same formula.

6.2 Evaluating the performance of the proposed solution

We first evaluate the performance of the proposed solution measuring the signaling and the reconfiguration cost while varying short term memory parameters such as: the number of iterations since last improvement in the call back mechanism and the tenure value, the *MaxTenure* value which is the maximum tenure assigned to moves that generate non-feasible solutions is altered to study its effect on the algorithm. In the diversification phase the number of NB_{START} is the most important parameter. Simulations of different NB_{START} values are executed while keeping the parameters of the short memory unchanged. At the end we compare the performance of both the GA and TS approaches in terms of signalling and reconfiguration cost relative to the optimal solutions the implemented

Integer Programming Model (IPM) generates. For several parameters we have used values similar to the one used in [14] [17] [18]. Table 6.1 lists some of the used parameters and their values.

Table 6.1 Numerical assumptions and values for performance evaluation

<i>Parameter</i>	<i>Description</i>	<i>values</i>
Delay	Number of non-improving iterations before the call back mechanism is triggered	2,4 and 8
NB _{START}	Number of restart cycles in the long memory	3,7 and 16
MaxTenure	Penalty assigned to moves that generate non-feasible solutions.	8,15 and 18
iK_{max}	Iteration limit in TS	12 and 17
N ₁	Number of cells in Network 1	164
N ₂	Number of cells in Network 2	225
N ₃	Number of cells in Network 3	1000
T ₁	Number of TAs in Network 1	7
T ₂	Number of TAs in Network 2	22
T ₃	Number of TAs in Network 2	7
α	Call intensity factor	0.05
C ^p	Cost of one paging	0.1
C ^u	Cost of one tracking area update	1

In TS three different values are assigned to each parameter as shown in Table 6.1. Different networks are used to test the different parametric values. The number of cells varies between 100 and 1000 and the number of TAs assigned to these cells varies between 7 and 22.

6.2.1 Number of runs

To ensure statistical validity and credible simulation results, we calculated the number of runs required for achieving at least a 90% confidence level. First, we conducted 6 different runs. The results for the runs of simulations in TS are given in Table 6.2. The standard deviation is calculated using the same formula as in section 5.1.4

The required numbers of runs to achieve 90% confidence level were found to be 1 for the TS.

Table 6.2 Number of runs in TS

<i>Runs</i> / <i>Recon. cost</i>	<i>10</i>	<i>20</i>	<i>30</i>	<i>40</i>
<i>1</i>	189490.3	189733.5	188551.4	188504.5
<i>2</i>	190000	187000	187551	190000
<i>3</i>	189000	191600	189551	187850
<i>4</i>	189490.3	189733.5	188551.4	187706
<i>Mean</i>	189495.2	189516.8	188551.2	188515.1
<i>Standard Deviation</i>	408.2866	1894.548	816.4966	1049.133
<i>z</i>	1.654	1.654	1.654	1.654
<i>r</i>	5	5	5	5
<i>Number of runs</i>	0.00508	0.109357	0.02052	0.033892

6.2.2 Call back mechanism

We first analyse the impact of the call back mechanism delay (D in Figure 4.4) on the signalling cost. The call back mechanism is a short memory component that can be

used to bring the search towards feasible regions after a number of consecutive non-feasible solutions. Hence, the delay is a threshold representing the number of iterations with no improvement that the algorithm must reach before activating the call back mechanism. It is a very important factor, as very large values will take the search away from promising regions and very small values will lead the search back to the same feasible solutions hence reducing the possibility in finding the best solution. Figure 6.1 (a) (b) shows the impact of different delay values on the signalling cost for three different network topologies of 164 cells grouped in 7 tracking areas, 225 cells grouped in 22 tracking areas and 1000 cells grouped in 7 tracking areas, while setting the tenure matrix value to 3 (which means a swap will be considered as tabu for 3 iterations). Three different delay values of 2, 4 and 8 were considered. The maximum delay value was set to 8 since larger values will take the search towards non-feasible solutions. The figure shows that the average signalling overhead generated when the call back mechanism is activated after two consecutive non-feasible solutions is the least. This is because having the delay being assigned to 2, the search is diverted towards feasible solutions as soon as non-feasible ones are visited. Delay values of 4 and 8 produce larger average signalling cost because when the search is in non-feasible regions it is difficult to divert it towards feasible regions. In small-scale networks (164 and 225) iK_{max} is assigned to a value of 12 and in the network containing 1000 cells iK_{max} is assigned to a value of 17 since a greater number of cells implies a larger search space and hence more iterations are executed in order to reach feasible regions. The base solutions represent non-feasible solutions i.e. the proposed TS approach has not been applied on these solutions.

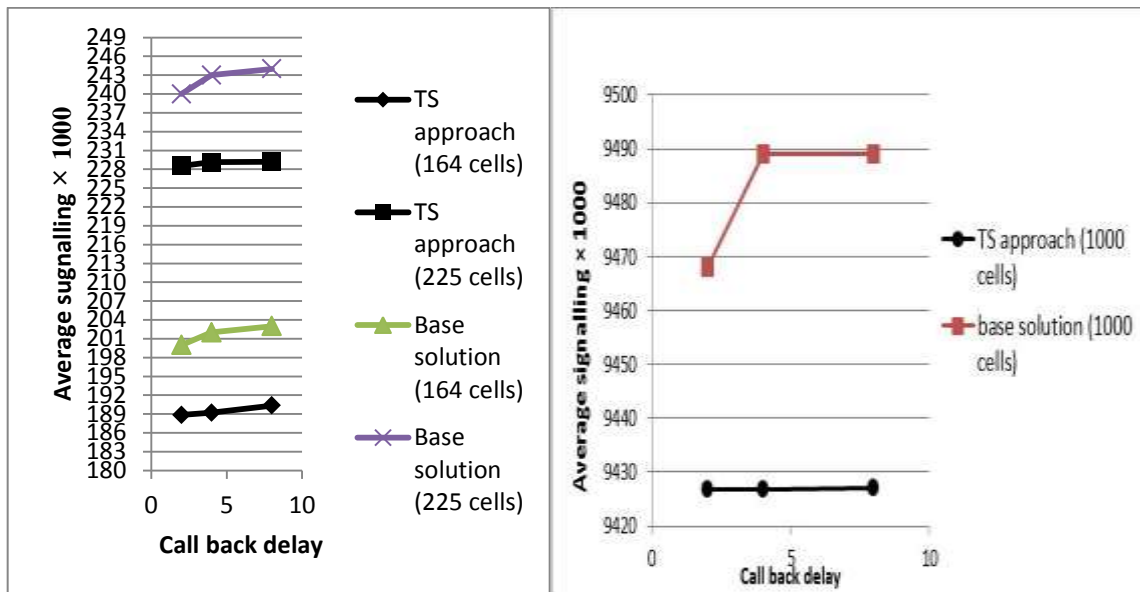


Figure 6.1 Effect of activation of the call-back mechanism on (a) 164 and 225 cells and (b) 1000 cells

The call back mechanism's effect is measured by the maximum tenure value, \maxVal (i.e., penalty) assigned to swaps resulting in non-feasible solutions. The tenure value of these swaps is incremented until it reaches the \maxVal value and hence the moves leading to non-feasible solutions are penalised. Figure 6.2 shows the results of the average signalling overhead while varying the \maxVal values (8, 15 and 18). The different values of \maxVal produced the same results for all three networks. This shows that it is not possible to have more than 8 consecutive non-feasible solutions in the generated solutions.

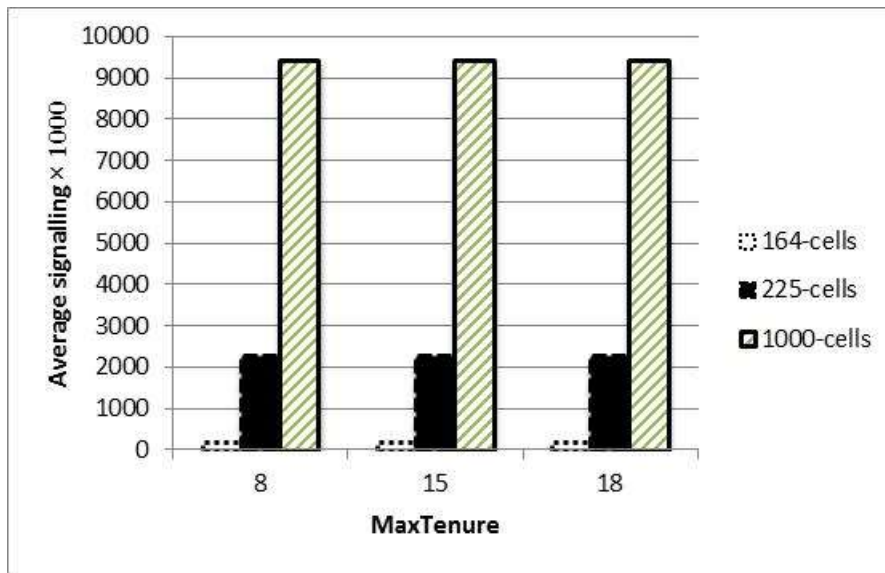


Figure 6.2 Effect of the variation of the maximal intensity of the callback mechanism on the solutions

According to Glover [24] a tabu list is used to store the last k moves executed and its size should be around size 7. In order to improve the performance of the algorithm in terms of retrieving the tabu moves since the short memory is accessed many times a tenure matrix (TM) is used instead of a tabu list. Different values of the tenure are examined as shown in Figure 6.3. Results showed that the tenure value of 3 produced the least average produced the least average signalling cost. This is because very large tenure values will take away the search from feasible regions and very small values will lead to cycles hence an average value must be used.

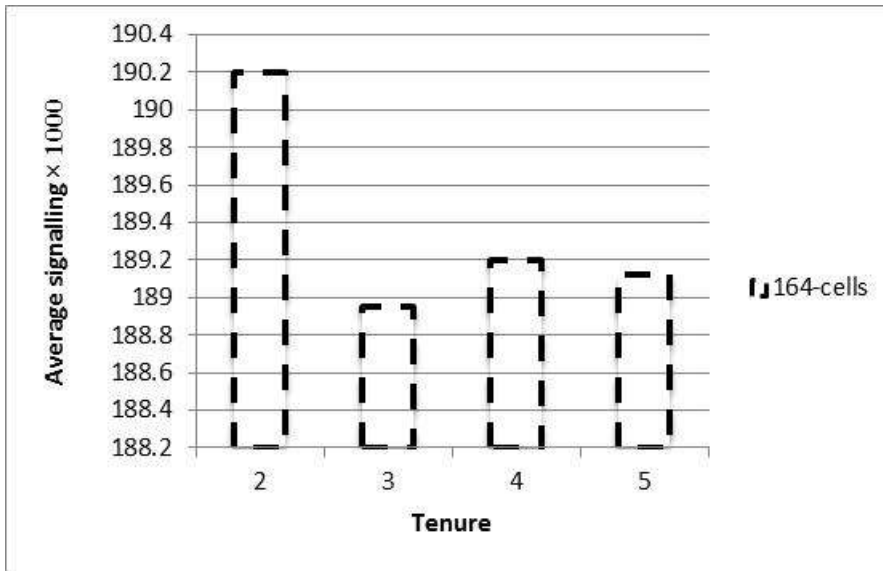


Figure 6.3 Effect of tenure value on the fitness function

6.3 Long-term mechanism (Diversification)

In the diversification phase the most important parameter is the number of restarts, NBSTARTS (i.e., the long memory delay) which is a threshold representing the number of iterations with no feasible solutions before switching to long memory, to generate new initial solutions to restart with, by swapping the cells with lowest value in each column in the frequency matrix. We have studied the impact of this delay on signaling using the network topologies of the previous scenarios and setting the tenure matrix value to 3 and the call back delay to 2. In Figure 6.4, we have measured the signalling cost while using different values of long memory delay (NBSTART in the algorithm). It is observed that for all network sizes smaller values of NBSTART produced lower average signalling overhead. Higher NBSTART values means more execution time of the algorithm and hence the

search is taken away from feasible solutions resulting in higher signalling overhead. Three re-starts produces the best results since the search is diverted towards unexplored new feasible regions which will be used as an initial pool.

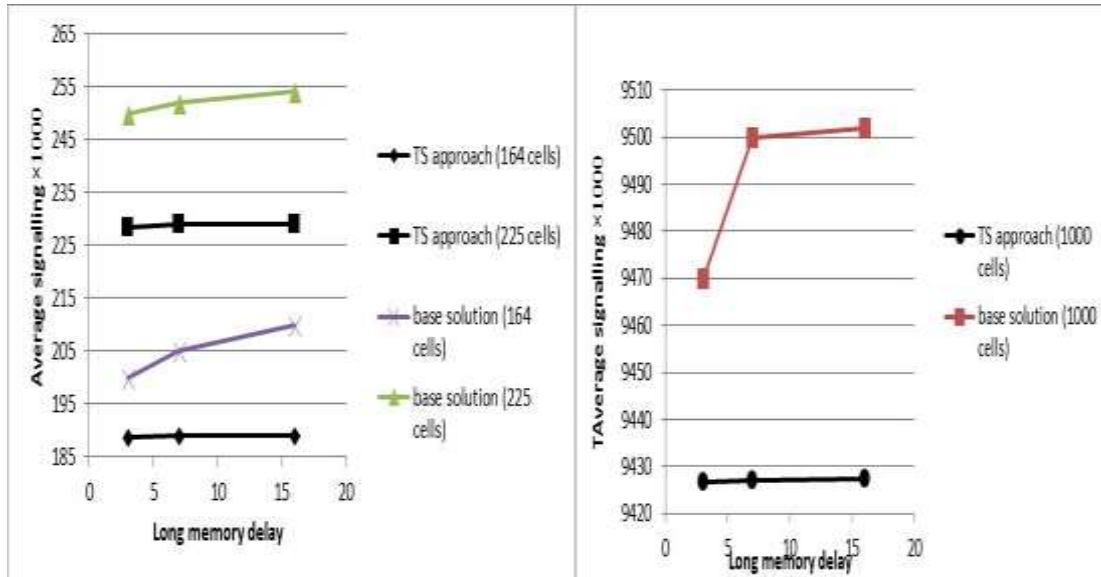


Figure 6.4 Effect of activation of the long memory mechanism on (a) 164 and 225 cells and (b) 1000 cells

Even though we used [14] to build our networks, we altered the number of TAs assigned to the cells to observe the effect this has on the signalling cost. Results showed that as the number of TAs increases in the network more signaling is generated this is because when the number of TAs increases more TAU will be accounted for and hence there will be more TAU overhead accounted for in the signaling cost formulation. Figure 6.5 shows the effect the number of TAs in a network has on the average signaling cost with 1000 and 164 cells. Both networks seem to perform better with lower number of TAs. This

implies that the number of TAs in a network must be chosen carefully in order to reduce the TAU cost.

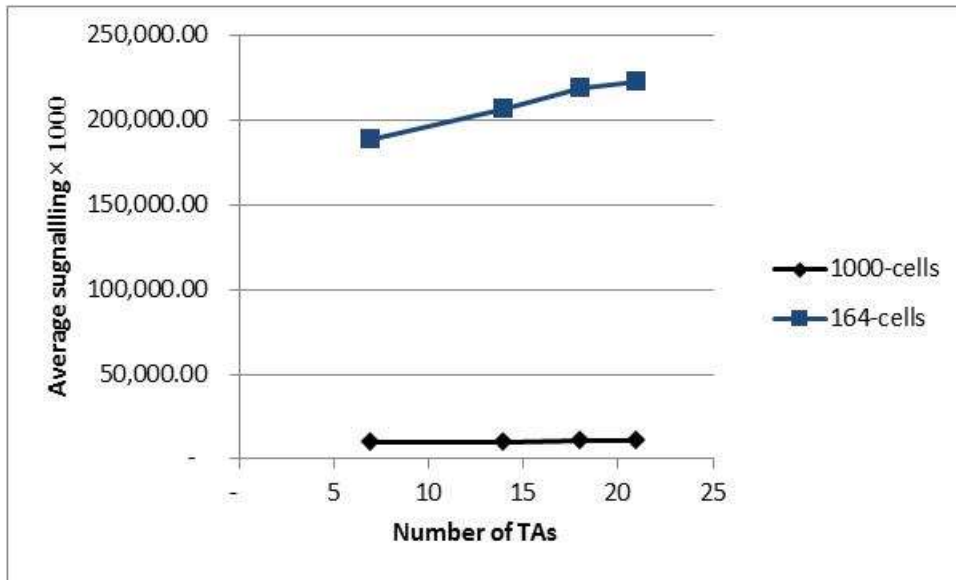


Figure 6.5 Effect of changing number of TAs

By assigning various weight values (p) to the TAU and paging procedures, it was concluded that the higher the weight assigned to the TAU procedure the higher the average signalling cost and vice versa. This indicates that the TAU overhead is greater than the paging overhead in the signalling cost equation. This is because the motion of the user is not known and hence all the possibilities in terms of TAU are taken into consideration resulting in this high overhead. The TAU procedure is multiplied by $(1-p)$ and the paging procedure is multiplied by p . The results in 1000 and 225 cells are represented in Figure 6.6.

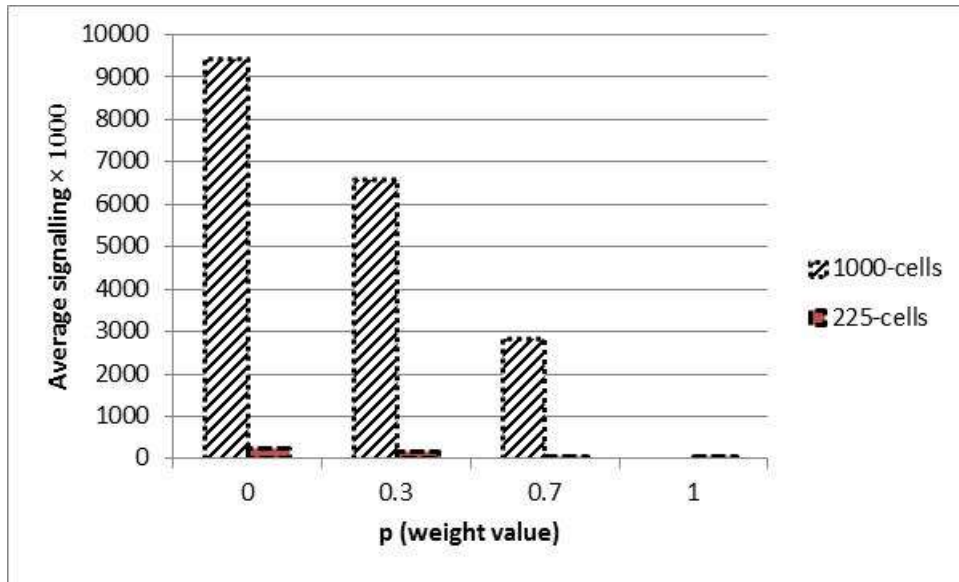


Figure 6.6 Effect of change of weight on average signaling

6.4 Comparison with GA and IPM

In this section we compare the TS based solution to the GA-based one and show how far from optimality they are using the IPM. Tables 6.3 and 6.4 show parametric values used in both the GA and TS. We use three network topologies (164 cells grouped in 7 TAs, 225 cells grouped in 22 TAs and 1000 cells grouped in 7 TAs) while setting the tenure matrix value to 3, the call back delay to 2, and long memory delay to 3. All three algorithms (TS, GA and IPM) start with the same initial pool and the performance of the GA, TS and IPM is evaluated in terms of the average signalling overhead and reconfiguration cost.

Table 6.3 Numerical assumptions and values for performance evaluation in the GA

<i>Parameter</i>	<i>Description</i>	<i>values</i>
α	Call intensity factor	0.05
C^p	Cost of single paging	0.1
C^u	Cost of single TAU	1
Iteration Limit	Limit of number of iterations in Networks	40
N_1	Number of cells in Network 1	164
N_2	Number of cells in Network 2	225
N_3	Number of cells in Network 3	1000
T1	Number of TAs in Network 1	7
T2	Number of TAs in Network 2	22
T3	Number of TAs in Network 3	7
POPSIZE	GA population size of Networks	50

Table 6.4 Numerical assumptions and values for performance evaluation in the TS

<i>Parameter</i>	<i>Description</i>	<i>values</i>
Delay	Number of non-improving iterations before the call back mechanism is triggered	2
NB _{START}	Number of restart cycles in the long-term memory	3
MaxTenure	Penalty assigned to moves that generate non-feasible solutions.	8
iK_{max}	iteration limit (termination condition)	12, 17
K _{max}	threshold used to measure the number of iterations	2

6.4.1 Network 1

This network consists of 164 cells and 7 TAs. Each cell must belong to one TA. Starting with feasible solutions as the initial pool the GA is executed until *Iteration Limit* is reached. Higher values than *Iteration Limit* will bring the search to already visited solutions. The TS terminates when the number of iterations since there has been an improvement in the fitness of the best solution exceeds iK_{max} or when all of the moves are in the TM. The behaviors of the GA and the TS approach relative to the optimal solutions as shown in Figure 6.7 by accounting for the average signaling cost for different reconfiguration limits. There are four pareto-optimal solutions in Figure 6.7. Both the GA and TS were run 5 times to obtain average values. It took almost 30 minutes to obtain an

optimal point corresponding to a given reconfiguration cost. Table 6.5 represents the values in the x and y axis in Figure 6.7

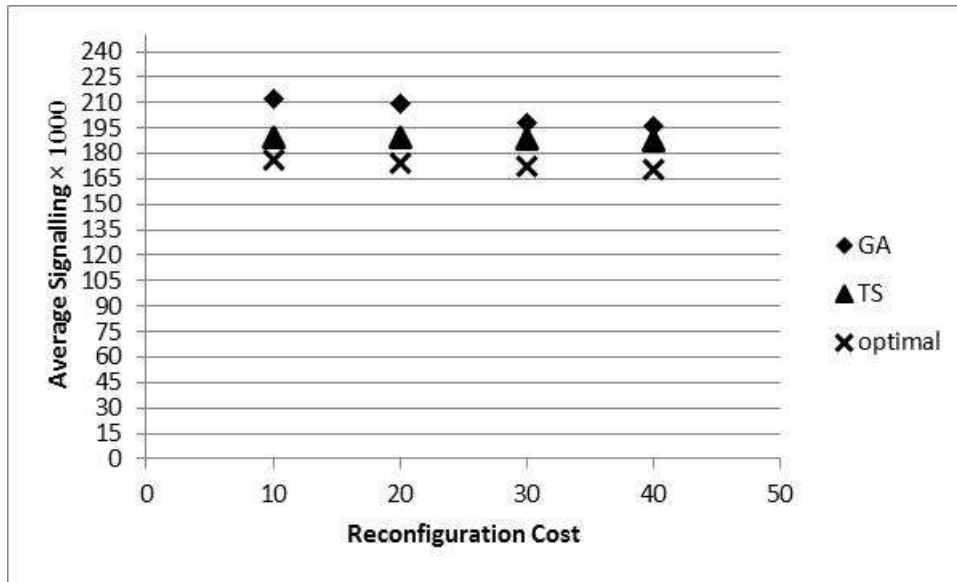


Figure 6.7 Pareto-optimal solutions in Network 1

Table 6.5 Signalling overhead generated in TS,GA and IPM associated with the fixed reconfiguration costs in Network 1

<i>Reconfiguration cost</i>	<i>10</i>	<i>20</i>	<i>30</i>	<i>40</i>
TS	189490.3	189733.5	188551.4	188504.5
GA	212073.4	209623.4	197848.4	195765.7
IPM	176532	174083	172137	170184

As shown in Figure 6.7 the solutions generated by the TS approach are closer to optimality than the ones generated by the GA. When there is an improvement in the signalling overhead the reconfiguration cost serves as a trade-off. Even though both start with the same initial values, when the TS method starts moving towards non-feasible regions non feasible moves are stored in a TM in order to prevent executing these moves again and hence avoiding local minima for a given number of iterations. The diversification phase also plays an important role in intensifying the search in new unexplored regions. In the genetic algorithm nothing guarantees from avoiding non-feasible solutions after being trapped in a local minima region. Both the GA and TS were run 5 times to obtain average values. The solutions generated by TS are far from optimality by 6% while the solutions generated by the GA are far by almost 21%.

6.4.2 Network 2

This network consists of 225 cells and 22 TAs. The behavior of the GA and the TS approach is highlighted in Figure 6.8 by accounting for the average signaling overhead for different reconfiguration costs. There are seven pareto-optimal solutions in Figure 6.8. The IPM has been run more than 7 times to come up with these values, every time taking up to 1 hour on average. It took almost 8 hours to come up with these values. Both the GA and TS were run 5 times to obtain average values. The higher the reconfiguration cost, the higher the time required to come up with the optimal solutions. The signalling cost decreases when the reconfiguration cost increases. For large-scale networks the TS has also proven to provide better signalling cost than the GA as shown in Figure 6.8. The solutions

provided by the TS are only 6% far from optimality while the solutions provided by the GA are far by 31% from optimality. This shows that even though the size of the networks has increased the TS still outperforms the GA and provides close to optimal solutions. Table 6.6 represents the values in the x and y axis in Figure 6.8.

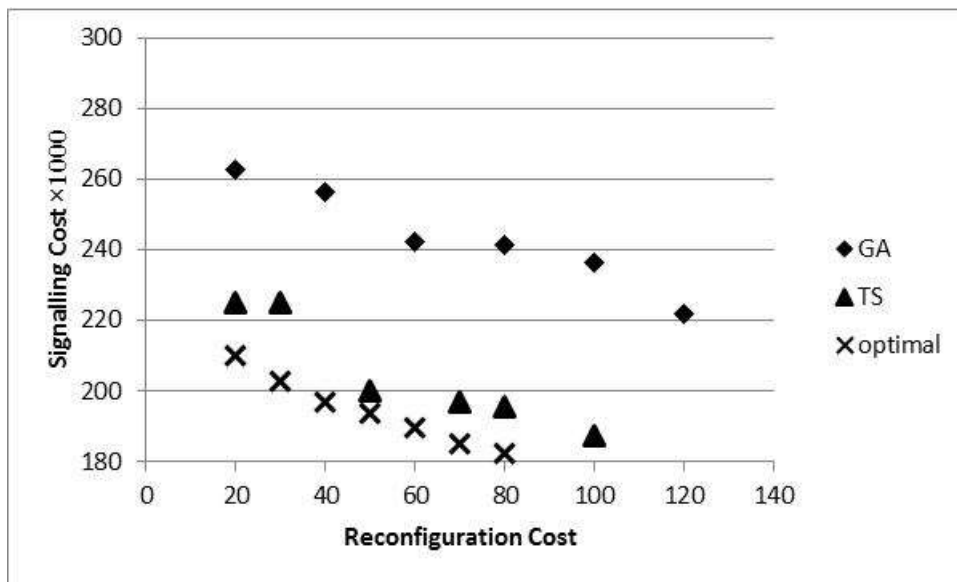


Figure 6.8 Pareto-optimal solutions in Network 2

Table 6.6 Signalling overhead generated in TS,GA and IPM associated with the fixed reconfiguration costs in Network 2 (* 1000)

<i>Reconfiguration cost</i>	20	30	40	50	60	70	80
-----------------------------	----	----	----	----	----	----	----

TS	225.29	225.2163		200.3054		197.1468	195.4
GA	262.596		256.363		242.523		241.391
IPM	210.083	203.004	196.782	193.656	189.494	185.2	182.623

6.4.3 Network 3

The network consists of 1000 cells and 7 TAs. Due to the large size of the network and the limitation of the computer memory, it was impossible to run the integer programming model. The behavior of the GA and the TS approach is highlighted in Figure 6.9 by accounting for the average signaling overhead for different reconfiguration costs. The higher the reconfiguration cost the more improvement is in the signaling cost. For large-scale networks the TS has also proven to provide better signaling cost than the GA over the same reconfiguration limits. Table 6.7 represents the values in the x and y axis in Figure 6.9.

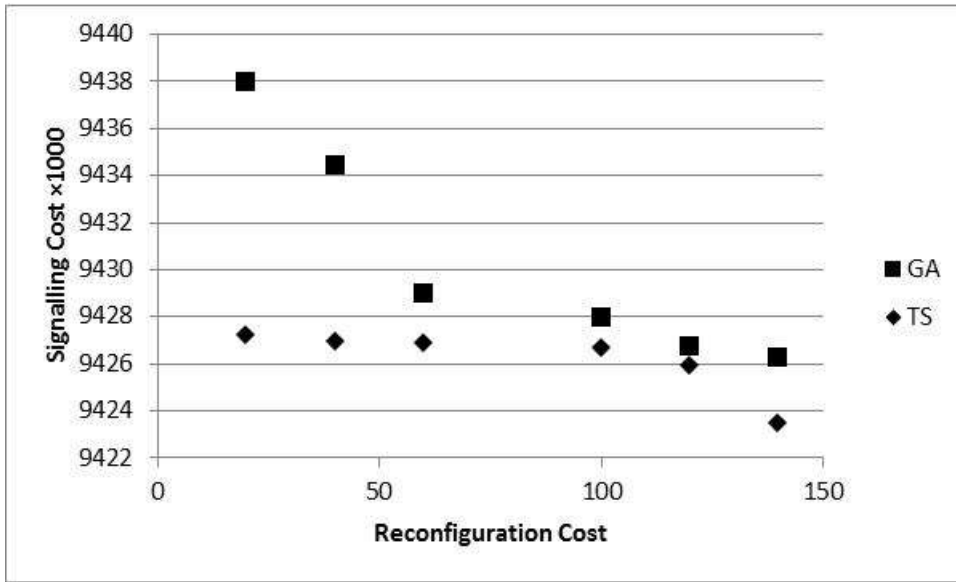


Figure 6.9 Pareto-optimal solutions in Network 3

Table 6.7 Signalling overhead generated in TS and GA associated with the fixed reconfiguration costs in Network 3 (* 1000)

<i>Reconfiguration cost</i>	<i>20</i>	<i>40</i>	<i>60</i>	<i>100</i>	<i>120</i>	<i>140</i>
TS	9427.19053	9426.9004	9426.87145	9426.6755	9425.912	9423.490
GA	9437.976	9434.444	9428.987	9427.917	9426.747	9,426

6.4.4 Reconfiguration cost

When there is an improvement in the signalling overhead the Reconfiguration cost serves as a trade-off. For the same number of iterations the average reconfiguration cost is computed in both approaches as shown in Figure 6.8. The reconfiguration cost generated in the TS approach is far less than that generated in the GA. This can be explained by the fact that in the GA crossover phase two points are randomly chosen and the values between these two points of the two parent chromosomes are swapped to generate new offspring. If these two points are not close then the reconfiguration cost is high. While in the TS approach only one swap is executed at a time until the diversification phase is reached where the least swapped cells are swapped to take the search to unexplored regions.

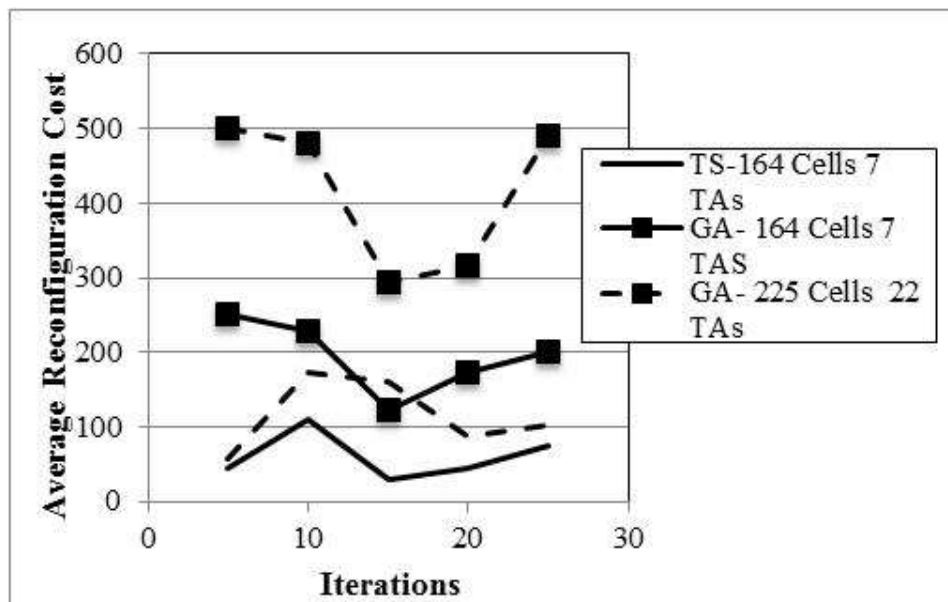


Figure 6.10 Average reconfiguration cost obtained in both TS and GA

6.5 Complexity of the algorithms

Table 6.8 represents the running time in seconds of the GA and TS. It is observed in all the different network scenarios the TS is slower than the GA. This is because of the different memory mechanisms that are used to produce close to optimality solutions.

Table 6.8 Running time in seconds of the TS and GA.

	164	225	1000
TS	8.07	10	183
GA	0.82	2	155

The cost of the GA is $O(IPN) * \text{fitness function } (O(N^2)) = O(IPN^3)$ where I is the iteration limit, P is the POPSIZE and N is the size of the chromosome. The cost of the TS in the best case scenario is $O(N^2) * \text{fitness function } (O(N^2)) = O(N^4)$, however in the worst case the diversification phase is executed and it costs $O(N^2)$ resulting in a total cost of $O(N^4) + O(N^2) = O(N^4)$.

CHAPTER 7

CONCLUSION

In LTE, the location of UEs and their mobility patterns change over time, hence TA design must be altered to adapt to such changes. In this paper we have used the TS heuristic to solve the TAs reconfiguration problem. In our solution, TS starts with an initial solution then to come up with new solutions, cells in different TAs are swapped. The fitness of every solution is based on the signalling overhead resulting from paging and location update. The main goal is to obtain a solution with a reduced signalling overhead. We have used a tenure matrix to manage tabu moves just by checking its indices instead of going over the list. Visited non-feasible solutions are penalized by increasing their tabu time period in order to prevent, in the next iterations, swaps that do not result in better solutions. Our algorithm defines an aspiration criterion to allow a tabu move to be executed only if it provides a better solution than the current best solution. It also uses a short memory structure to tolerate the transition to less good solutions to prohibit local optima and prevent the risk of cycles. In addition, the algorithm defines a diversification criterion in long memory structure to make it possible, after a certain number of iterations resulting in non-feasible solutions, to restart with a new initial solution that can be generated by swapping the least frequently permuted cells before switching back to short memory structure. We have implemented the proposed algorithm as well a Genetic Algorithm (GA) based approach [14] with minor changes. The tracking area reconfiguration model was presented

as a pareto-optimal problem i.e. a bi-objective problem in which it is impossible to improve one objective function without worsening the other. The IPM provides the exact pareto-optimal solutions and the proposed TS approach provides closer to optimality solutions than the GA for both small and large-scale networks in feasible time.

As a future work and in order to improve the running time of the proposed TS approach the problem can be solved in parallel. The number of cells assigned to a TA can be limited by adding this constraint to the system model. However the number of cells per TA must be chosen carefully in order to produce good results. This will require the addition of a memory mechanism.

REFERENCES

- [1] Cox, Christopher. An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications. John Wiley & Sons, 2012.
- [2] Mulligan, Catherine. EPC and 4G Packet Networks: Driving the Mobile Broadband Revolution. Academic Press, 2012.
- [3] Chadchan, S. M., and C. B. Akki. "3gpp lte/sae: An overview." International Journal of Computer and Electrical Engineering 2.5 (2010): 806-814.
- [4] Nossenson, Ronit. "Long-term evolution network architecture." Microwaves, Communications, Antennas and Electronics Systems, 2009. COMCAS 2009. IEEE International Conference on. IEEE, 2009.
- [5] "The LTE Network Architecture." Alcatel.
- [6] Liou, Ren-Huang, Yi-Bing Lin, and Shang-Chih Tsai. "An investigation on LTE mobility management." Mobile Computing, IEEE Transactions on 12.1 (2013): 166-176.
- [7] Lim, Jaechan, and Daehyoung Hong. "Mobility and Handover Management for Heterogeneous Networks in LTE-Advanced." Wireless personal communications 72.4 (2013): 2901-2912.
- [8] Yu, Yifan, and Daqing Gu. "The Cost Efficient Location Management in the LTE Picocell/Macrocell Network." (2013): 1-4.
- [9] Roa, V. Srinivasa . "Interoperable UE Handovers in LTE."
- [10] Lin, Yi - Bing, Ren - Huang Liou, and Chun - Ting Chang. "A dynamic paging scheme for long - term evolution mobility management." Wireless Communications and Mobile Computing (2013).
- [11] Yixue Lei; Yongsheng Zhang, "Enhanced mobility state detection based mobility optimization for FEMTO cells in LTE and LTE- Advanced networks," Communication Technology and Application (ICCTA 2011), IEEE International Conference on, pp.341,345, 14-16 Oct. 2011
- [12] Fu, Huai-Lei, Phone Lin, and Yi-Bing Lin. "Reducing signaling overhead for femtocell/macrocell networks." Mobile Computing, IEEE Transactions on 12.8 (2013): 1587-1597.
- [13] Nawaz, Mohsin. "Exploiting Tracking Area List Concept in LTE Networks." (2013).

- [14] Modarres Razavi, Sara, et al. "Performance and cost trade-off in Tracking Area reconfiguration: A Pareto-optimization approach." *Computer Networks* 56.1 (2012): 157-168.
- [15] 3GPP 12.3.0 DynaReport 2013.
- [16] Kang, Hyung-Woo, Hyon-Goo Kang, and Seok-Joo Koh. "Optimization of TAC configuration in mobile communication systems: A tabu search approach." *Advanced Communication Technology (ICACT), 2014 16th International Conference on. IEEE, 2014.*
- [17] Pierre, Samuel, and Fabien HouéTo. "A tabu search approach for assigning cells to switches in cellular mobile networks." *Computer Communications* 25.5 (2002): 464-477.
- [18] Diallo, Mamadou M., Samuel Pierre, and Ronald Beaubrun. "A tabu search approach for assigning node Bs to switches in UMTS networks." *Wireless Communications, IEEE Transactions on* 9.4 (2010): 1350-1359.
- [19] 3GPP Long Term Evolution (LTE) (: GPRS Tunneling Protocol (GTP) in LTE) <http://4g-lte-world.blogspot.com/2013/03/gprs-tunneling-protocol-gtp-in-lte.html>
- [20] Toril, M., Luna-Ramirez, S., & Wille, V. (2013). Automatic Replanning of Tracking Areas in Cellular Networks. *Vehicular Technology, IEEE Transactions on*, 62(5), 2005-2013
- [21] L. Davis : "Handbook of Genetic Algorithms", edited by Lawrence Davis, Van Nostrand Reinhold, New York, 1991
- [22] 3GPP TS 23.40, 1 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 12), December 2013.
- [23] Gurobi 3.0.1. ReferenceManual. 2010.
- [24] F.Glover, Tabu Search – Part I, *ORSA Journal on Computing* 1 (3) (1989) 190-206.
- [25] T.R. Andel and A. Yasinac, *On the credibility of manet simulations*, IEEE Computer Society Press (Los Alamitos, CA, USA), vol. 39, IEEE Computer Society Press, 2006, pp. 48-54.