UC Irvine
UC Irvine Electronic Theses and Dissertations

Title
Optimal Capacity Augmentation of Cellular Mobile Networks

Permalink
https://escholarship.org/uc/item/6q05056d

Author
Albanna, Amr Kamal

Publication Date
2018

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Optimal Capacity Augmentation of Cellular Mobile Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical Engineering and Computer Science

by

Amr Albanna

Dissertation Committee:
Professor Homayoun Yousefi'zadeh, Advisor
Professor Hamid Jafarkhani, Chair
Professor Ahmed M. Eltawil

2018

© 2018 Amr Albanna

DEDICATION

To my beloved Wife
To the memory of my Father and my Mother,
To my little angels Doha and Maryam

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Homayoun Yousefi'zadeh whose guidance and support helped me overcome challenges in all aspects of my research at UC Irvine. I am glad I was able to work with Dr. Homayoun who combines understanding of technology, industry applications, and networks with academic know-how. This unique combination helped me extensively in utilizing state-of-the-art algorithms in creating highly efficient new solutions that produced great results. I would also like to thank Prof. Ahmed M. Eltawil for the opportunity to join the PhD program at UCI and Prof. Fadi Kurdahi who supported me in getting to the right faculty members who were interested in my work. In addition, I would like to thank Prof. Hamid Jafarkhani for dedicating his time to be in my committee.

I acknowledge the support of my colleagues and friends, especially, Dr. Wael ElSharkasy, Dr. Muhammed Khairy, Dr. Amr Hussien, Dr. Elsayed Ahmed, and Eng. Mohammed Fouda. Further, I would like to thank all my other colleagues and friends for making all those years at UC Irvine a great and enjoyable experience.

Most importantly, I would like to dedicate this work to my late father Prof. Kamal Albanna and my mother Fawzeya Elsayed who always encouraged me to pursue my PhD degree. Finally, this work would not have been finished without the constant support of my beloved wife, Manal Galal. Her moral and physical support was always a driving force to get into the program after being in the industry for 15 years and throughout my PhD studies. Finally, I express my gratitude to my daughters, Doha and Maryam, for sacrificing their fun time to allow me finish my work.

CURRICULUM VITAE

Amr Albanna

EDUCATION

Doctor of Philosophy in Electrical and Computer Engineering **2018**
University of California, Irvine *Irvine, CA*

Thesis title: *Optimal Capacity Augmentation of Cellular Mobile Networks*

M.Sc. in Electrical Engineering **2001**
Ain Shams University *Cairo, Egypt*

Thesis title: *GSM Frequency Planning Techniques using Artificial Intelligence*

B.Sc. in Electrical Engineering **1995**
Ain Shams University *Cairo, Egypt*

INDUSTRY EXPERIENCE

Founder and Vice President **Nov 2010–Now**
Omega Wireless, Irvine, CA.

Senior Engineering Manager **Jan 2005 – Oct 2010**

T-Mobile USA, Los Angeles, CA.

Principal Engineer **Apr 2000–December 2004**

AT&T Wireless, Cerritos, CA.

Systems Engineering Manager **Aug 1998–Apr 2000**
Motorola, Cairo, Egypt.

RF Engineer **Mar 1996–Aug 1998**
Alcatel, Cascais, Portugal.

Network Engineer **Sep 1995–Mar 1996**
ITI, Cairo, Egypt.

Honors

- ◇ Peak Achievement Award T-Mobile USA 2006.
- ◇ Best Manager Award T-Mobile USA 2006.
- ◇ Best Network Annual Improvement Award T-Mobile USA 2006.
- ◇ Best Team Innovation Award T-Mobile USA 2007.

Amr Albanna, Homayoun Yousefi'zadeh,"Optimal Capacity Augmentation of LTE Networks," Submitted to IEEE Transactions on Networking, November 2017.

Amr Albanna, Homayoun Yousefi'zadeh,"Learning-Constrained Enhancement of Cellular Networks Capacity," IEEE Transactions on Mobile Computing, Accepted March 2018.

ABSTRACT OF THE DISSERTATION

Optimal Capacity Augmentation of Cellular Mobile Networks

By

Amr Albanna

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Irvine, 2018

Professor Homayoun Yousefi'zadeh, Advisor

Every year, network operators spend hundreds of millions of dollars to improve cellular capacities. Capacity improvements typically aim at adding carriers, frequencies, bands, radios, distributed antenna systems, small cells, and cell towers. In many cases, user traffic and network load associated with special events complicate the issues of tracking, congestion, and degrading quality of service. Often times, operators handle special events by dedicating human resources to manually solve them instead of deploying automated solutions. In this thesis, we first use a pair of supervised learning approaches to model Universal Mobile Telecommunication System (UMTS) cellular network capacity measured in terms of total number of users carried and then predict breakpoints of cellular towers as a function of network traffic loading. Similarly, we utilize a supervised deep learning technique to predict the Long Term Evolution (LTE) network loading of connected users and then dynamically predict the congestion threshold of each cellular tower under offered load

Next, we formulate an optimization problem to maximize UMTS network capacity subject to constraints of user quality and predicted breakpoints. For LTE, we use the predicted congestion thresholds together with quality constrains to fine-tune cellular network operating parameters leading to minimizing overall network congestion.

We investigate a few algorithmic alternatives including Simulated Annealing (SA), Hill

Climbing,Regression, and Genetic Algorithm (GA). We propose a novel variant of simulated annealing referred to as Block Coordinated Descent Simulated Annealing (BCDSA) to solve the problem. Our performance measurements show that BCDSA offers dramatically improved algorithmic success rate and best characteristics in utility, runtime, and confidence range measures compared to alternative solutions for both problems in UMTS and LTE. It is observed that BCDSA is up to an order of magnitude faster than other algorithms and offers success rates twice better than other algorithms in finding best solutions.

Chapter 1

Introduction

1.1 Motivation

Maximizing network capacity and minimizing the number of blocked users or users serviced by congested cellular towers given an offered load and a minimum level of acceptable user quality is a major challenge in the operation of cellular networks. As a result every year, network operators spend hundreds of millions of dollars to improve cellular capacities. Capacity improvements typically aim at adding carriers, frequencies, bands, radios, distributed antenna systems, small cells, and cell towers. In many cases, user traffic and network load associated with special events complicate the issues of tracking, congestion, and degrading quality of service. Often times, operators handle special events by dedicating human resources to manually solve them instead of deploying automated solutions.

We felt the critical need for a solution that helps network operators cope with network congestion, improves customer satisfaction and reduces churn. This solution has to be accurate and fast to respond to network RF and loading conditions and reassign network unused capacity to areas of congestion. We believe that the best way to do that would be to employ cutting edge techniques in artificial intelligence , machine learning, and optimization to achieve the above goal.

1.2 Thesis Contribution

The main contribution of this thesis are threefold.

First, we introduce a deep learning system and identify the right network counters as inputs to the system allowing to accurately predict the congestion thresholds of individual UMTS and LTE cellular towers even of those towers that never congested before.

Next, we formulate and efficiently solve optimization problems aiming at minimizing the congestion of a cluster of cells subject to UE quality constraints.

Third, we compare the results of our proposed BCDSA algorithm and a few other solutions including Hill Climbing, Simulated Annealing and Genetic Algorithm in order to evaluate performance, utility, runtime, and success rate of each technique.

1.3 Organization

The rest of the thesis is organized as follows. An Introduction to Cellular Networks is presented in Chapter 2. Chapter 3 discusses Learning-Based Breakpoint Modeling using various techniques in both UMTS and LTE, with focus on Deep Learning. Chapter 4 presents the problem description, formulation and solution approach for each of the UMTS and LTE networks presenting various optimization techniques like hill climbing, genetic algorithm and BCDSA. In Chapter 5 the results from various optimization techniques are presented and compared side by side. Finally conclusions and summary are drawn in Chapter 6.

Chapter 2

Introduction to Cellular Networks

In this chapter we provide a brief introduction to cellular network technologies, mainly Universal Mobile Telecommunication Systems (UMTS), also referred to as 3G and Long Term Evolution (LTE) also referred to as 4G. We will explore various network elements, functionalities, capacity challenges, typical capacity solutions, quality measures, control parameters and operating environment. We also present a table of notations that we will use throughout the rest of the thesis.

2.1 Introduction to UMTS Network

Maximizing the number of served users while maintaining an acceptable level of Quality of Service (QoS) is always a serious challenge in the operation of cellular networks. Considerable effort has been excerpted by standards groups of 3GPP as well as equipment vendors research teams, chipmakers, operations research groups, and network operators to handle this challenge. The capacity challenge in cellular networks stems from various sources depending on technology and access techniques.

In broadband systems such as CDMA and WCDMA, the main factors affecting capacity are power, interference, and processing hardware elements. The real challenge in such systems is to control downlink and uplink power consumption in order to a) ensure base stations do not run out of power when feeding downlink power amplifiers, and b) prevent noise rise in uplinks causing access failures. Considerable research has been conducted on power control algorithms to address the challenge [23, 36].

Such efforts have mostly focused on mobile handsets, Base Transceiver Stations (BTS) and UMTS base stations (NodeB), Base Station Controllers (BSC), and Radio Network Controllers (RNC) software that make power control decisions. However, most of these algorithms focus on the behavior of individual elements and the associated impact on quality and capacity of those individual elements. In reality, network operation concerns itself to make sure a cluster of sites serving a specific geographical area such as a large urban downtown are performing well when delivering capacity under proper quality to mobile users. It is well understood that traffic has to be offloaded from more congested cells to less congested cells in order to optimize the operation of a cluster of cells. Traffic offloading can be achieved in several ways such as changing the Common control Pilot CHannel (CPiCH) power of a cell i referred to as Ω_i, tilt of antennas, azimuth of cells, and handover thresholds as detailed in [15, 29]. In reality, such changes may be effective for static network configurations relying on manual changes made by network engineers while monitoring and assessing the impact. Further, some changes such as changing azimuth or tilt of the antennas take effect slowly but yet are very costly. They also require proper knowledge of the location of users such that the change does not have a negative impact on user coverage. To that end, some research was conducted using changes to Ω_i power and tilt [23], [36] while others utilized Cell Individual Offset (CIO) of a cell i [5] referred to as Φ_i to redistribute traffic.

The need to address the above-mentioned capacity challenge in cellular networks is the main driver for an automatic solution capable of dynamically handling traffic distribution according to demand. Accordingly, the main contributions of this thesis are predicting capacity limits and breakpoints of cellular towers, and providing a dynamic automated solution that significantly improves capacity. First, we apply a multi-layer perceptron deep learning technique that can utilize real network measurement data to model capacity limits and predict breakpoints of cellular towers. Second, we formulate an optimization problem with the objective of maximizing the overall capacity of a collection of cell towers covering an area of interest through traffic offloading and subject to constraints associated with the above mentioned cell capacity modeling results as well as minimum quality thresholds. Third, we propose solving this optimization problem using a number of Simulated Annealing (SA) techniques among which there is a novel variant referred to as BCDSA. BCDSA is inspired by Block Coordinated Descent (BCD) [41] and Accelerated Coordinated Descent (ACD) [19] methods. Our proposed BCDSA technique shows significant improvements in the quality of results while offering attractive time complexities compared to standard SA techniques.

2.2 UMTS Operating Environment

In order to apply our proposed modeling techniques and optimization of capacity algorithms, we use a cellular network cluster illustrated in Fig. 2.1. Our choice represents a typical major US city downtown area. The cluster is comprised of ten sites with each site having three cellular towers covering hundred and twenty degrees and presented by arrows pointing at three different directions. For a given operation scenario, cells in red represent congested cells while cells in black represent non-congested cells. Various measurements and Key Performance Indicators (KPIs) are collected to be used for analysis. These measurements will be detailed in Section 3.1, but they mainly cover aspects of traffic loading, total number of users served, blocked users (if any), power utilization in downlinks, and noise rise in uplinks. The values of each cell tower measurements are taken from an operator's UMTS

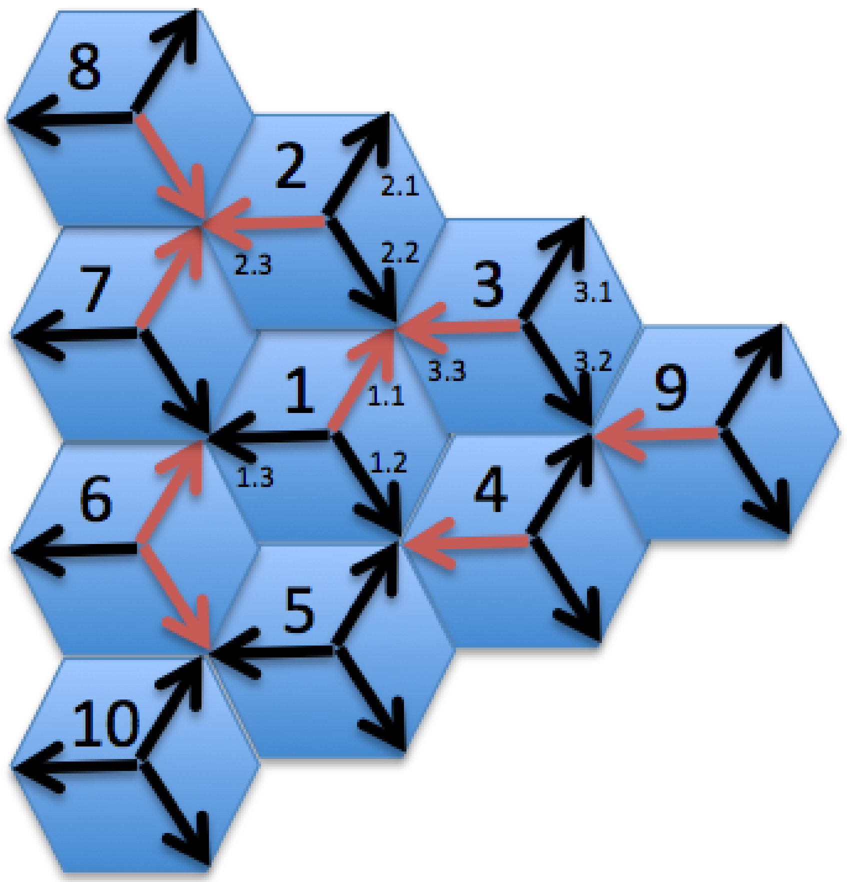


Figure 2.1: A typical downtown cellular network cluster comprised of ten sites with each site having three towers.

network in downtown Los Angeles. As observed, some of the cells in this cellular network shown in red are congested while some of their immediate neighbors are not. This scenario helps us examine various congestion scenarios and understand the best and worst scenarios of traffic offloading.

In order to mitigate cellular network congestion, additional capacity is added using one of the three typical approaches below.

- ◇ Adding more sites has the best impact but may not be possible due to location availability, zoning, and leasing. Further, adding a site typically costs hundreds of thousands of dollars and may take up to 24 months to complete.
- ◇ Adding radios operating over a new frequency within the existing band could theoretically double the capacity of operating over a single frequency. However, the cost of acquiring new licensed spectrum assuming availability is typically in the range of millions of dollars with deployment cycles of up to 24 months.
- ◇ Shifting traffic from congested cells to neighboring cells that do not carry as much traffic within the same site or nearby sites represents the most economical and timely alternative. However, it requires addressing a number of challenges described below.

With respect to the third alternative, shifting traffic can be done in two ways. First, lowering CPICH power Ω_i of a cell i results in reallocating a portion of the pilot power to carry additional traffic users. In addition, such approach shrinks the footprint of the cell resulting in shifting some of the users on the boundary of a congested cell to the neighboring cells. Second, changing CIO Φ_i of a cell i allows for adding an artificial margin to a neighboring cell signal level during handover calculations. This results in making the handover algorithm conclude that a neighboring cell signal is stronger than what it really is. Hence, it results in shifting some of the users on the handover boundaries to the neighboring cell. Since this parameter is set on a cell by cell basis, it can be set differently for each neighboring cell in order to control offloading to those neighboring cells. However, caution has to be exercised to control total offloading from all congested neighboring cells into an individual cell.

Based on data obtained from cellular operators, procedures aimed at manual parameter tuning could offer an additional capacity of up to 5%. The latter is subject to significant overhead accrued due to continuously and manually monitoring network traffic and making changes on a timely manner when congestion occurs. In addition, offloading traffic in isolation can potentially result in congesting neighboring cells and deteriorating link qualities of shifted users thereby dropping their calls [15]. Hence, traffic offloading has to be done in a cluster setting rather than in isolation in order to identify optimal operating points for the collection of cell towers as oppose to individual cells.

2.3 Introduction to LTE Network

Due to exponential growth of LTE traffic, mobile operators are spending hundreds of millions of dollars improving their cellular infrastructure. Different capacity improvement and congestion mitigation approaches include spending major capital to acquire new spectrum, building new macro sites to add bandwidth, and building small cells as well as in-building solutions. These approaches have proven effective in certain cases but are expensive and not always practical when facing challenges associated with dynamic capacity demands.

In the absence practically viable systematic optimization approaches, mobile operators exercise manual fine-tuning of cellular network parameters in order to alleviate cellular congestion. However, the results are trivially suboptimal compared to systematic optimization approaches. In this thesis, we introduce systematic optimization approaches to minimize the congestion of LTE networks.

2.4 LTE Operating Environment

The challenge associated with such task in LTE networks is better understood by explaining how resources are allocated to users. Under LTE standard, each cellular tower has a fixed number of Physical Resource Blocks (PRBs) defined in time and frequency. Utilization of each PRB is independent of utilization of other PRBs within the same cell without causing interfere. When a user requests a certain type of service or Enhanced Radio Access Bearer (ERAB), the LTE scheduler at a cell-site will allocate a certain number of PRBs depending on the type of service, i.e., guaranteed bit rate versus non-guaranteed bit rate, required bandwidth, required latency, and most importantly the maximum throughput that can be carried. This throughput associated with each PRB mainly depends on the maximum allowable modulation scheme ranging from QPSK at the lowest level to 16QAM and up to 64QAM. The maximum allowable modulation depends on the Signal to Interference and Noise Ratio (SINR) experienced by a given user for that PRB. For example, a user requesting video streaming while experiencing excellent RF conditions and hence high SINRs will be able to use high modulation schemes such as 64QAM per each PRB assigned and will hence require a small number of PRBs to satisfy its requested ERAB. On the other hand, a user experiencing sub-par RF conditions and hence poor SINRs will only be able to utilize low modulation schemes such as QPSK hence requiring a much larger number of PRBs than the previous user in order to satisfy a similar video streaming quality [2]. Table 2.1 captures the relationship among LTE Channel Quality Indicator (CQI), modulation, coding rate, spectral efficiency, achievable throughput per PRB, and SINR.

Aside from the utilization of techniques such as Multi Input Multi Output (MIMO) and Inter-Cell Interference Coordination (ICIC) [2] to mitigate sub-par RF conditions and improve SINR, the physical limitation on the number of available PRBs still presents a challenge that

Table 2.1: LTE Channel Quality Indicator (CQI), modulation, coding rate, spectral efficiency, achievable throughput per PRB, and SINR.

CQI Index	Modulation Type	Code Rate (x 1024)	Spectral Efficiency (bps/Hz)	DL TP per PRB (kbps)	DL SINR (idB)
1	QPSK	78	0.1523	19.1898	-7.28
2	QPSK	120	0.2344	29.5344	-4.78
3	QPSK	193	0.377	47.502	-2.04
4	QPSK	308	0.6016	75.8016	0.66
5	QPSK	449	0.877	110.502	2.84
6	QPSK	602	1.1758	148.1508	4.73
7	16QAM	378	1.4766	186.0516	6.38
8	16QAM	490	1.9141	241.1766	8.78
9	16QAM	616	2.4063	303.1938	11.49
10	64QAM	466	2.7305	344.043	13.27
11	64QAM	567	3.3223	418.6098	16.52
12	64QAM	666	3.9023	491.6898	19.71
13	64QAM	772	4.5234	569.9484	23.12
14	64QAM	873	5.1152	644.5152	26.37
15	64QAM	948	5.5547	699.8922	28.79

has to be addressed in heavily loaded scenarios of operations. Depending on the bandwidth of an LTE channel, each cell offers a fixed number of PRBs. For example, a 5MHz and a 10MHz LTE channel offer no more than 25 and 50 PRBs. When the demand for PRBs is higher than what a cell can offer, adverse impacts on User Equipment (UEs) connected to the cell may be imposed. The impacts range from degrading the speed of existing connections, denying incoming handover requests, or even dropping calls. Since LTE systems only support hard handovers in which a UE is only connected to only one cell tower at a time and all cellular towers operate on the same frequency, a UE remains connected to its original cellular tower if denied a handover request. As such, it can experience severe quality degradation and eventually a call drop [2]. In order to mitigate the issue, most operators attempt at keeping per cell PRB utilization under a congestion threshold of 80%. Cells exceeding the congestion threshold usually trigger augmentation mechanisms such as carrier additions or bandwidth expansions.

In an effort to keep that limit, it is critical to manage traffic amongst various cells where traffic from highly loaded cells is offloaded to lightly loaded cells serving the same area. This traffic offload can be achieved in several manners, i.e., by changing the footprint of cells, shifting cell boundaries, and changing tilts as well as azimuths of cells. However, implementing physical changes is time consuming and more suited for static or slowly changing environments as oppose to fast changing dynamic environments. Alternatively, we propose changing the power of a cell i referred to as $\wp_i$ and handover margin of a cell i referred to as $\hbar_i$ in order to control the serving area of cells and redistribute traffic as needed. We note that these parameters can be changed instantly in the field in response to dynamic changes in traffic distributions in order to offload traffic from congested cells to neighboring cells without congesting the neighboring cells and without degrading the quality of the UEs on the edge of congested cells that end up shifting to a neighboring cell.

2.5 Notations

Table 2.2 shows the common notations that have been used through the thesis for both UMTS and LTE networks. But, due to the differences between the networks, tables 2.3 and 2.4 show the notations for the specific notations of UMTS and LTE networks, respectively.

Table 2.2: The common notations that used for both networks.

I	Set of all UMTS cells within the cluster
N	The number of cells within set I
i	Cell index
$\underline{x}$	Vector of elements $(x_1, \cdots, x_N)$
ξ	Freeze count measure of the used algorithm
ξ_{max}	Maximum freeze count measure of the used algorithms
m	No. of search attempts in the used algorithm
T	Temperature of the used algorithm
T_i	Initial temperature of the used algorithm
T_f	Final temperature of the used algorithm
a	Cooling factor of the used algorithm
ρ	Multiplier of N controlling the number of iterations at each temperature point of the used algorithm
σ	Number of times the temperature will be cooled down in the used algorithm
B	Boltzman constant
R	Random number derived from uniform distribution $U\ [0, 1]$
$\mathcal{U}$	Unit step function

Table 2.3: UMTS notations.

C_Υ	Total capacity measured as total number of users carried
C_i	Maximum capacity of cell i identified by learning algorithm
Ω_i	Common Control Pilot CHannel (CPICH) power of cell i
Φ_i	Cell Individual Offset (CIO) of cell i
x_i	Ordered pair setting (Ω_i, Φ_i) for cell i
c_i	Current measured traffic carried by cell i
c_i^Ω	Traffic offload of cell i due to reducing CPiCH power
Ψ_i	Traffic offload due to border shift after power reduction of cell i
$c_{i,j}^\Phi$	Traffic offload from cell i due to reducing CIO in cell j
$\eta_{i,j}$	Overlap percentage between cell i and its neighbor j
Γ_i	Traffic offload from cell i to cell j after changing Φ_j
q_i	Quality of cell i
Q	Minimum allowed quality of a cell
γ_i	Voice or circuit switched access failures of cell i
τ_i	Voice traffic loading of cell i in Erlang
μ_i	Carried data volume of cell i in MB
α_i	Radio Resource Connection Circuit Switched (RRC-CS) or voice access attempts of cell i
ψ_i	Radio Resource Connection Packet Switched (RRC-PS) attempts of cell i
β_i	Downlink transmit power of cell i or TX-PWR
θ_i	Uplink Received Signal Strength Indicator (RSSI) of cell i
λ_i	Average transmit power per user β_i/τ_i of cell i
ν_i	Adjusted downlink received signal strength at the edge of cell i
Π_i	Total path loss of cell i
δ_i	Penalty of violating cell i quality and capacity constraints
$\tilde{C}_\Upsilon$	Penalty-augmented C_Υ due to violating all per cell quality and capacity constraints
ϕ	Reduction ratio of energy per bit divided by noise (E_b/N_t)

Table 2.4: LTE notations.

$\wp_i$	Power of cell i
$\hbar_i$	Handover margin of cell i
x_i	Ordered pair setting $(\wp_i, \hbar_i)$ for cell i
λ_i	Average connected UEs to cell i
$\lambda_i^{\wp}$	UE offload of cell i due to power change
$\eta_{i,j}$	Overlap percentage between cell i and its neighbor j
q_i	Quality of service experienced by a UE connected to cell i
Q	Minimum acceptable quality of a UE
γ_i	Received SINR of a UE connected to cell i
δ	Penalty of violating load preservation constraint
$\lambda_{i,j}^{\hbar}$	UE offload from cell i to j due to handover margin change
Λ_i	Average number of UEs connected to cell i utilizing a PRB congestion threshold of 80%
Λ_L	Overall cluster load measured as total number of average UEs connected to all cells of set I
Λ_Υ	Total congestion of cluster measured as $\sum_{i \in I}(\lambda_i - \Lambda_i)$
$\tilde{\Lambda}_\Upsilon$	Penalty-augmented Λ_Υ due to violating quality constraints
n	Initial population count for GA algorithm
χ	Fixed integer multiplier for GA algorithm depending on
	the number of decision variables and their variation ranges
ϵ	Small number used in the stoppage criterion of GA algorithm

Chapter 3

Learning-Based Breakpoint Modeling

In this chapter we will explain various network counters and Key Performance Indicators (KPIs) that are commonly available in cellular network for both UMTS and LTE. We then explain how congestion happens for both technologies, and what defines these breakpoints. We then explore various options to predicting these breakpoints utilizing various approaches like regression and deep learning neural networks with various network sizes and depths.

3.1 UMTS Problem Modeling

This section presents our modeling approaches of network capacity and how breakpoints can be predicted for all cells even those that have not experienced congestion before. We are modeling particular network KPIs based on certain input measurements and traffic loading values with the objective of predicting future values of KPIs as the result of increased network traffic loads. The KPI of interest is circuit switched access and voice access failures of cell i denoted by γ_i. The input measurements include voice traffic loading τ_i in Erlang, carried data volume μ_i in MB, RRC-CS-Attempts α_i, RRC-PS-Attempts ψ_i, transmit power β_i, and RSSI θ_i. A detailed description of these KPIs can be found in [23]. These KPIs and measurements were collected from a major operator's database of cell-by-cell measurements for a period of four weeks starting on October 16, 2013 and ending on November 14, 2013.

The goal is to detect the breakpoint of each cell (sector) based on the observed behavior of similar cells and using a learning system. Fig. 3.1 illustrates such behavior, i.e., plots of γ_i as a function of α_i for a number of cells (sectors). Each cell is presented with a different colored line in the figure. As shown, cells will have no failures until reaching a certain threshold of RRC-CS-Attempts α_i at which point they exhibit failures. The breakpoint of each cell is unique and depends on a number of inputs. The purpose of utilizing a learning algorithm is then to capture the effects of such inputs. It is seen in Fig. 3.1 that some cells start rejecting additional call attempts causing access failures as early as 200 attempts, but other cells can take up to 500 attempts before having access failures. In the next two subsections, we explore a couple of learning approaches to predict the breakpoints of the underlying cells.

3.1.1 Machine Learning

In our first learning approach, we characterize the breakage behavior of cell i using a softplus function described below.

$$\gamma_i = \log[1 + \exp(\alpha_i - C_i)] \tag{3.1}$$

Hence given α_i, identifying γ_i is equivalent to identifying C_i. In essence, C_i is identified as the value of α_i at which γ_i departs from a value of zero associated with access failures due to high traffic loads. Then, we compare the measured breakpoint C_i to other measurements in order to identify possible correlations. The measurements of interest include the ratios TX-PWR /Erlang (User) β_i/τ_i, TX-PWR /RRC-CS-Attempts β_i/α_i, RSSI /Erlang (User) θ_i/τ_i, and RSSI /RRC-CS-Attempts θ_i/α_i. As seen in the curves of Fig. 3.2, some of the

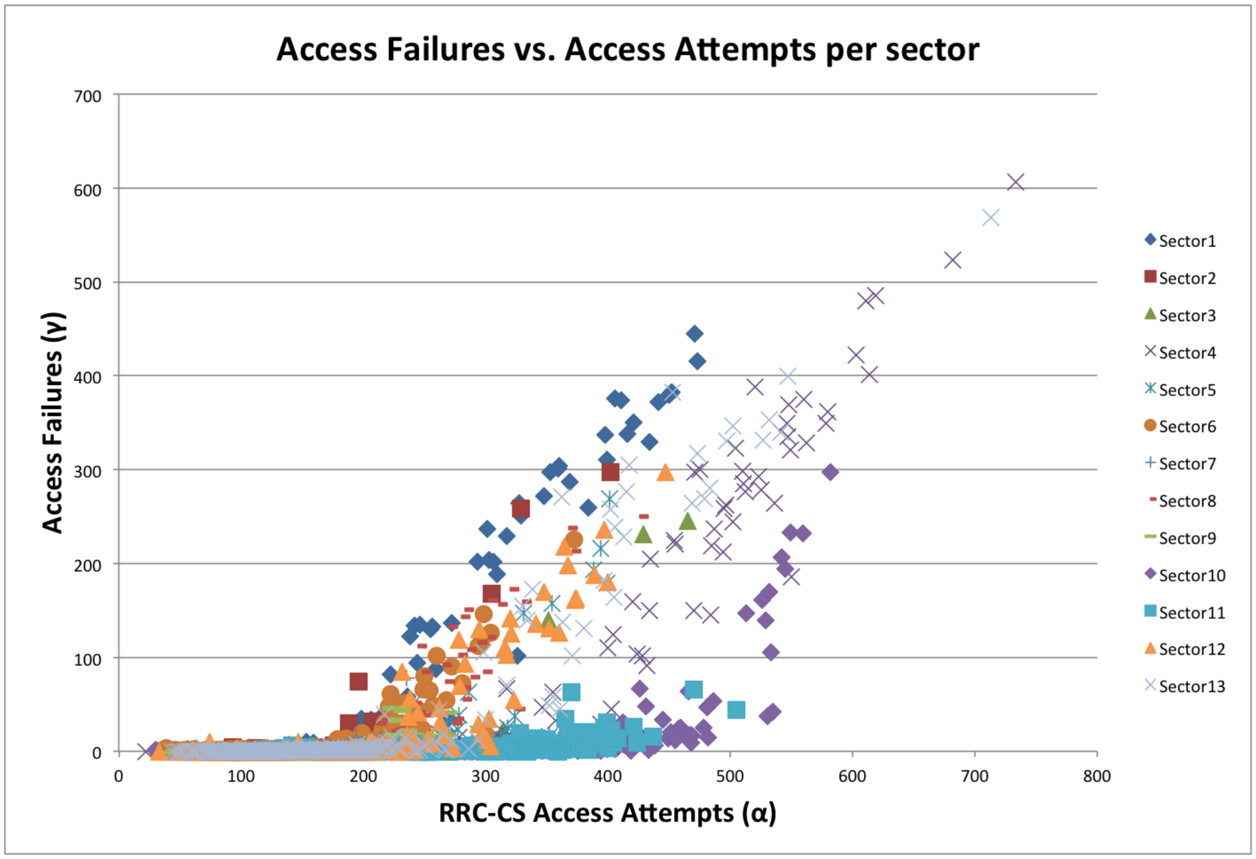


Figure 3.1: Sample drawings of CS access failures γ_i versus RRC access attempts α_i illustrating breakpoints vary.

measured parameters seem to be closely correlated with the observed breakpoints, especially, the ratio TX-PWR / Erlang (User) β_i/τ_i. Interestingly, dependencies are inherent features of operation in UMTS networks as oppose to the modeling approach.

In Eq. (3.2) below, we express the maximum capacity of cell i identified by the learning algorithm C_i as a function of measured quantities where $\underline{w}_i^T$ is the vector of regression coefficients [14] and the vector $\underline{G}_i$ is identified as a function of its arguments.

$$C_i = \underline{w}_i^T \cdot \underline{G}(\frac{\beta_i}{\tau_i}, \frac{\beta_i}{\alpha_i}, \frac{\theta_i}{\tau_i}, \frac{\theta_i}{\alpha_i}) \tag{3.2}$$

This is explained considering the fact that as the ratio β_i/τ_i is increased, the power of the power amplifier of NodeB in UMTS runs out quicker causing access failures for new users attempting to access the cell. The ratio β_i/τ_i is a unique per cell characteristic reflecting the average amount of power needed by users due to the unique configuration of the cell. For example, cells serving users close by have relatively low transmit powers allowing them to serve a larger number of users before running out of resources. To the contrary, cells serving users that are far away or inside buildings will need to have higher powers in order to reach those users and hence run out of power resources much faster. We use Machine Learning (ML) and in particular linear regression curve fitting to model these curves.

The goal of ML is to minimize the Mean Square Error (MSE) or Root Mean Square Error (RMSE) defined below between predicted values $\hat{C}_i$ and measured values C_i of the training samples by calculating the coefficient vector $\underline{w}_i^T$ [10].

$$RMSE = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\hat{C}_i - C_i\right)^2\right]^{\frac{1}{2}} \tag{3.3}$$

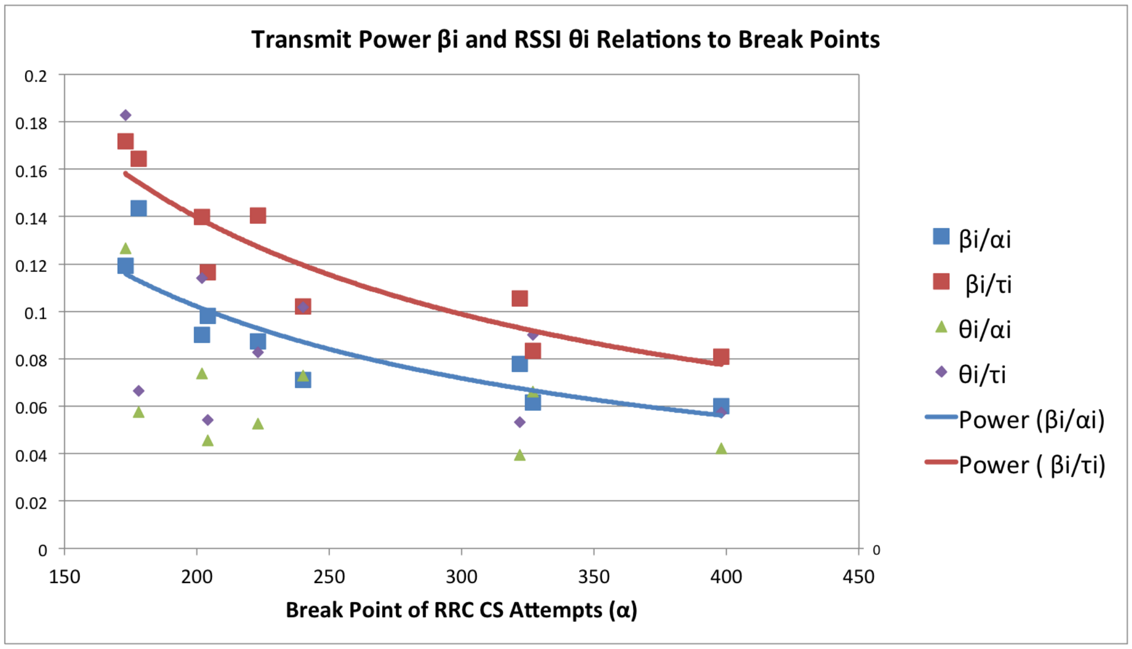


Figure 3.2: The relationship between breakpoints and various values of β_i/α_i, β_i/τ_i, θ_i/α_i and θ_i/τ_i.

It is easy to model cells that have reached breakpoints before by detecting those breakpoints as a function of loading. The real challenge in cellular networks is to accurately predict the breakpoints of cells that have never reached those points before. In our work here, we manage to find the factors that characterize the behavior of a specific cell and hence its unique breakpoint. This is mainly what we explained earlier as power used per user in the downlink β_i/τ_i as well as noise rise per user θ_i/τ_i.

To examine how accurately we are able to predict these breakpoints, we build our model using 9 cells as the training set with actual breakpoints. From that, we apply the model to a test set of 4 cells, but the data used is for samples taken well before reaching the breakpoint. This is equivalent to applying models to cells that never reached breakpoints before except that in this case, we do have the actual breakpoints of these 4 cells and are able to compare predicted and measured breakpoints.

The results are shown in Table 3.1. We see error ranges of 3% to 9% with the exception of one cell at a 15% error which is due to special events impacting the loading of that cell.

Table 3.1: Actual vs predicted breakpoints using regression.

Cell	Actual BP	Predicted BP	Error
Cell 3	278	304.83	9.65%
Cell 6	174	168.32	-3.26%
Cell 9	182	153.92	-15.42%
Cell 12	221	213.46	-3.40%

3.1.2 Multi-Layer Perceptron Deep Learning

In this subsection, we describe our alternative approach to modeling γ_i and how it relates to other collected measurements mentioned previously. The main reason for introducing the alternative modeling approach of this subsection is to reduce modeling error and avoid relying on the trial and error approach of choosing inputs and patterns as needed in the ML approach of the previous subsection. It is also important to note that the alternative modeling approach of this subsection is able to model γ_i as a function of α_i including the breakpoint as oppose to the previous modeling approach only identifying the breakpoint. Then considering γ_i as a function of α_i in Eq. (3.1), C_i is identified as the value of α_i at which γ_i departs from zero.

3.1.2.1 Learning Approach

Multi-Layer Perceptron Deep (MLPD) learning presented in [22, 21] has proven to be very effective in solving complex learning problems, especially, pattern recognition [39]. In our learning approach, we use supervised MLPD learning [12, 18] to model cellular networks' behavior and predict KPIs as the result of traffic loading increase.

The promise of MLPD modeling approach is to replace the analytical difficulties encountered in other modeling approaches with a straightforward computational learning algorithm [20]. The proposed modeling approach simply takes advantage of a fixed structure nonlinear system with a well defined analytical model capable of predicting KPIs based on measurements. The fixed, fully connected, feedforward perceptron learning structure utilized for the task of modeling in our study consists of an input layer with up to eight processing elements, two to four hidden layers with twenty processing elements in each layer, and an output layer with one processing element. In each iteration of learning, the current input is propagated in the forward direction through hidden layers to generate an output. The output error is then propagated in the reverse direction to the input layer in order to adjust weighting functions between every pair of processing elements in adjacent layers. The process is repeated until reaching an acceptable threshold of output error. We refer the reader to [42] for the details of learning.

We note that the accuracy of learning typically depends on the complexity of perceptron structure, i.e., the number of layers and processing elements per layer, training, verification, and testing algorithms. Accuracy is also traded against complexity and runtime. Fig. 3.3 includes illustrations of how RMSE decreases as the number of processing elements per layer increases.

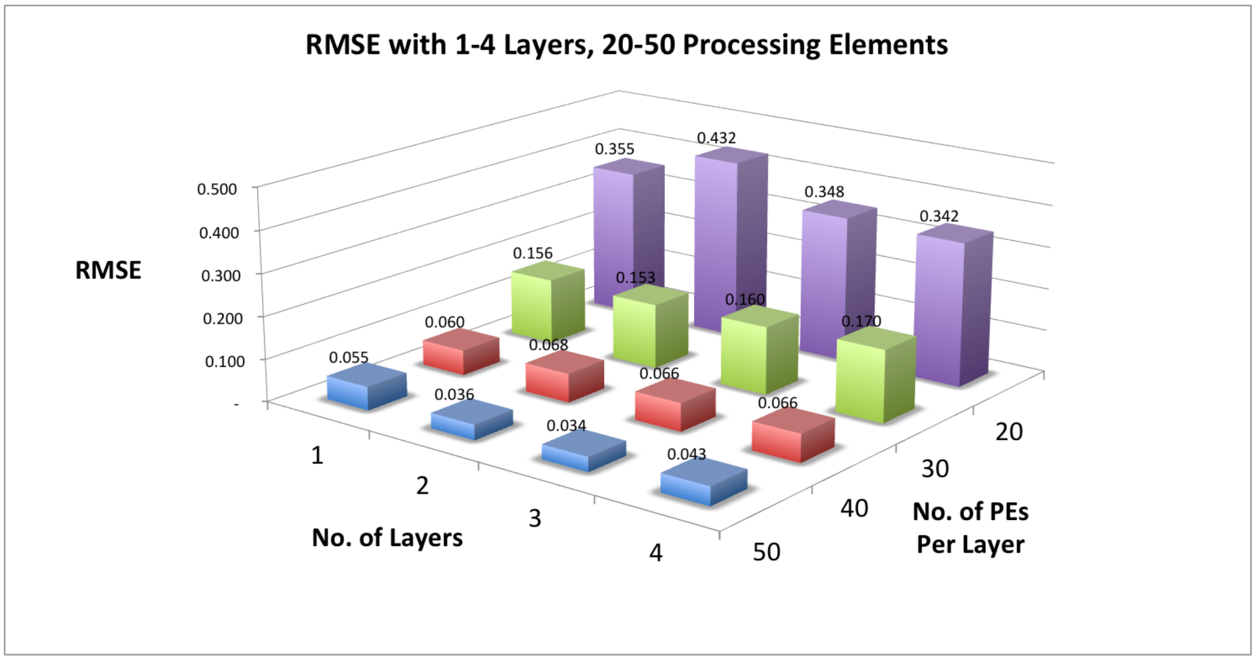


Figure 3.3: An illustration of how RMSE decreases as the number of processing elements per layer increases.

Attempting at investigating the impact of the number of layers and processing elements per layer, we found out that having a larger number of input samples in the training set is the most critical factor in improving RMSE error results measured on verification and test sets [4]. However, that comes at the significant cost of increasing runtimes by up to an order of magnitude. Further, increasing the number of hidden layers beyond 4 has a positive impact on improving the RMSE error in the range of $[20\%, 30\%]$ in some configurations. However, the impact is small compared to an impact of up to 90% associated with changing the number of processing elements. Another critical aspect in using MLPD learning is the choice of the back-propagation algorithm. We tried a number of algorithms as discussed in [9] namely, a) traingdm, a gradient descent with momentum back-propagation, b) traingdx, a gradient descent with momentum and adaptive learning rate backpropagation, c) trainbfg, a Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton backpropagation [21, 1], and d) trainlm Levenberg-Marquardt algorithm. Similar to quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix [18, 7]. Having explored the results of various back propagation techniques, we make use of Levenberg-Marquardt back propagation learning scheme producing the best results.

3.1.2.2 Cluster Modeling

Here, we focus on modeling the cluster of cells using training samples from the measurement data and aiming at modeling cell behaviors with the least modeling error.

The MLPD network utilized in this subsection consists of an input layer with eight processing elements, four hidden layers each with twenty processing elements, and one output layer with a single processing element. The eight inputs are per cell measurement data of τ_i, μ_i, α_i, ψ_i, β_i, θ_i, β_i/τ_i, and β_i/α_i while the output is γ_i.

This allows MLPD learning to not only capture the combined dynamics of all cells within the cluster set but also to consider correlations in the behavior of similar cells when modeling the breakpoint of each cell within the cluster set. It is worth mentioning here that we also experimented with two, four, and six processing elements at the input layer of MLPD, but the results of average and minimum RMSE were best considering the computing overhead when we used the eight inputs listed above. In the latter case, the learning achieved average and minimum RMSE values of 4.4% and 2.75%, respectively.

In the above effort, we also had a major challenge in which transmit power β_i presented combined measurements of both voice and data services in the UMTS network under study. In UMTS, voice users have a higher priority than data users but both of them share the same power amplifier levels reported as β_i. In order to address the issue, we used an ML regression analysis similar to what was explained in the previous section and available measurements of τ_i, μ_i, ψ_i, α_i , β_i to estimate the power of voice and data separately. We then scaled the measured values of β_i according to our regression results of voice and data. As a result, we were able to significantly improve the accuracy of our MLPD learning algorithm. As shown in Table. 3.2, the model augmented by regression shows average and minimum RMSE values of 2.9% and 2%, respectively.

Table 3.2: Isolating transmit power of voice and data greatly improves RMSE of the predicted breakpoints.

Inputs	Average RMSE %	Minimum RMSE %
MLPD	4.4%	2.75%
MLPD & Regression	2.9%	2%

3.2 LTE Problem Modeling

In this section, we present our approach to learning the congestion threshold of each LTE cell i in a cluster of cellular towers as a function of the average number of user connected to that cell. Fig 3.4 shows sample drawings of actual LTE PRB utilization as a function of average connected users collected from a major mobile operator data over one month in downtown Los Angeles. Inspecting the graphs, it is evident how each cell/sector has its own PRB utilization characteristics under different loading levels of average connected users. For example, it is observed that Sector 4 has a high PRB utilization at a low average connected users. On the other hand, Sector 3 has a much lower PRB utilization for similar loading values of average connected users. Hence, Sector 3 reaches the 80% congestion threshold of PRB utilization at a much higher number of average connected users around 150 users, while Sector 4 reaches the same threshold at around 74 users. Considering the 80% congestion threshold of PRB Utilization, we can claim that Sector 3 is able to carry a larger number of users than Sector 4 before it breaks. The question we are trying to answer next is how to predict the value of Λ_i representing the average number of connected UEs λ_i crossing the PRB utilization congestion threshold of 80%, for each cell i based on its unique characteristics.

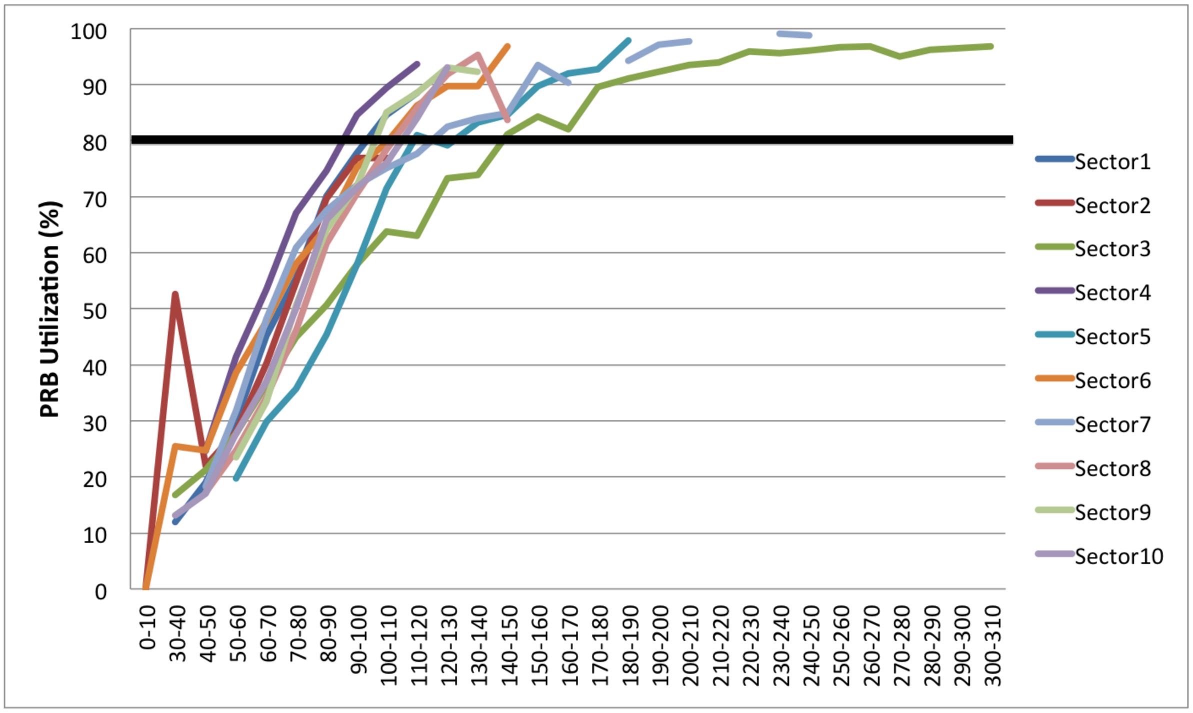


Figure 3.4: Actual sample drawings of LTE PRB utilization as a function of average connected UEs.

3.2.1 Deep Learning

Similar to our approach in UMTS, we use a Multi-Layer Perceptron Deep (MLPD) learning approach to accurately predict the congestion threshold of individual LTE cells in a cluster of cell towers. The latter is equivalent to identifying the value of Λ_i for each cell i, i.e., the average number of connected UEs λ_i crossing the PRB utilization congestion threshold of 80%. The fixed, fully connected, feedforward perceptron learning structure utilized for the task of LTE congestion threshold modeling in our study consists of an input layer with twenty two processing elements to accept twenty two LTE input counters. In order to strike the balance between accuracy and complexity, we experiment with two to four hidden layers each layer containing ten to twenty processing elements. The structure has an output layer with one processing element predicting the value of Λ_i for cell i. Fig. 3.5 illustrates the MLPD structure used for the task of learning. In each iteration of learning, we propagate

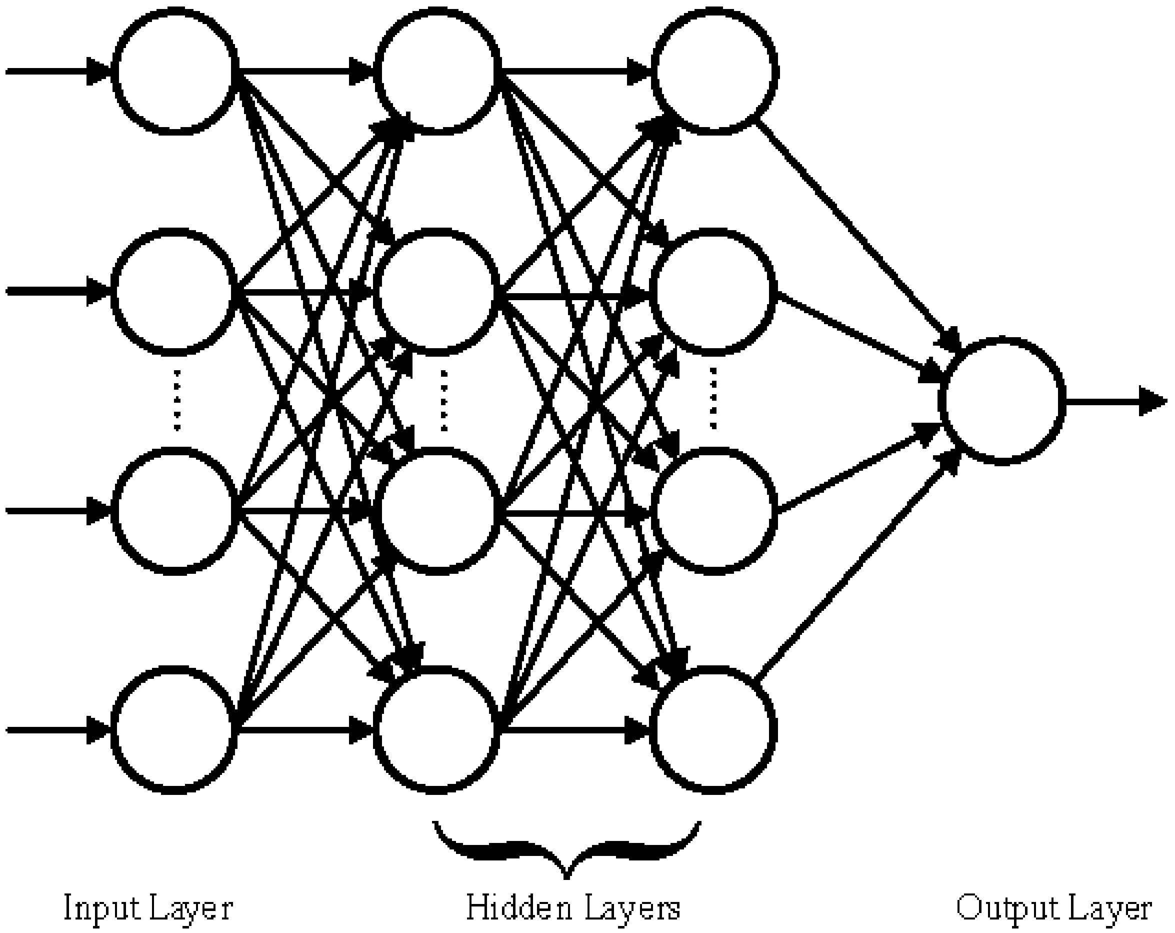


Figure 3.5: Multi layer perceptron deep learning structure used for learning congestion thresholds of individual LTE cellular towers.

all counters associated with a sample input in the forward direction from the input through hidden layers to generate an output. The output value is compared to the measured output from the counters and an output error is calculated. The output error is then propagated in the reverse direction to the input layer in order to adjust weighting functions between every pair of processing elements in adjacent layers. The process is repeated until reaching an acceptable threshold of output error. For evaluating the error, we use Root Mean Square

Error (RMSE) calculated between the measured PRB utilization from the collected counters and predicted by MLPD.

Fig. 3.6 provides illustrations of RMSE variations as a function of the number of processing elements and layers. We utilize 20 MLPD runs in each configuration to get average, maximum, and minimum RMSE for that configuration. Generally speaking, we note that the minimum and average RMSE decrease for higher number of layers and processing elements. Additionally, the runtime of each experiment is a function of the number of layers

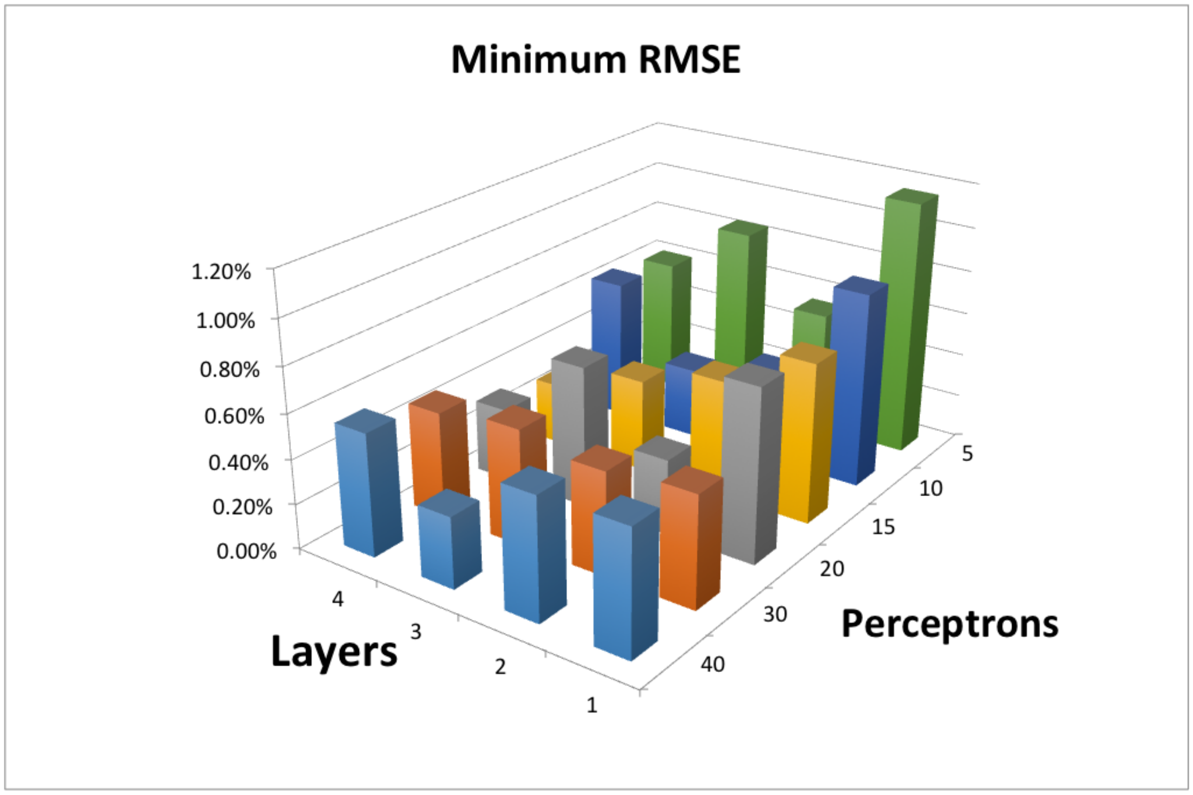


Figure 3.6: An illustration of minimum RMSE of MLPD prediction as a function of number of layers and perceptrons.

and perceptrons. While most configurations run in the order of few hundred seconds in our experiments, those with more than three hidden layers and more than thirty perceptrons could very well take more than a thousand seconds using our computing platform. Fig. 3.7 reports measurements of average runtimes as a function of the number of layers and perceptrons. Looking at addressing the trade off between runtime and RMSE, we choose to

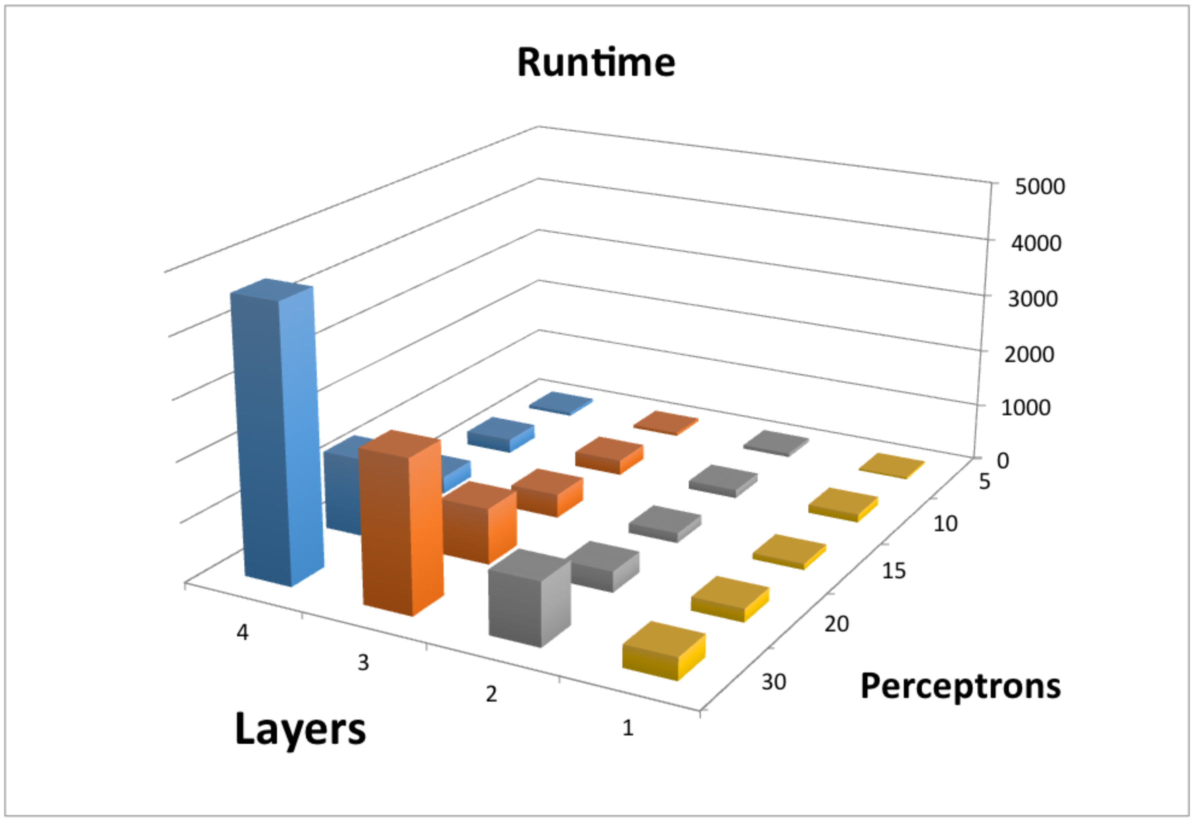


Figure 3.7: Measurements of average runtime as a function of number of layers and perceptrons.

use an MLPD structure containing two hidden layers with twenty perceptrons per hidden layer offering an average RMSE of approximately 0.34% and an average runtime of about 352 seconds to complete 10 runs. The latter ensures finding a good solution with a low value of RMSE.

3.2.2 Input Counters to Learning

One of the critical factors in generating accurately predicted results is the choice of input parameters, i.e., LTE counters. In this section, we explore the impact on modeling from a group of available LTE counters. The goal is to utilize inputs that are most closely related to PRB utilization of cells. In order to identify the best input counters, we experimented in multiple phases with various counters and KPIs from the real network of a major US carrier collected in hourly intervals over one week. Among the set of input data, some of these KPIs are average and peak connected UEs, Quality of service Class Identifier (QCI), modulation scheme used, average and peak throughputs of UEs as well as cells uplink SINR, CQI, spectral efficiency, average Receive reference Signal on Reference Power (RSRP), and Reference Signal Received Quality (RSRQ). In the first phase of learning, average active UEs, average connected UEs, and peak connected UEs of a single cell were used to predict PRB utilization of that cell subsequently introducing an RMSE of 34%. Adding call Attempts, average and peak number of ERABs, and total Voice over LTE (VoLTE) calls resulted in an RMSE of about 36%. In the next phase of learning, adding traffic measures of VoLTE in Erlangs and data volume in Megabytes improved RMSE to 22%. It is important to note that this phase added the actual voice and data loading of connected UEs on individual cells. The following phase added QCI as presented in Table 3.3 identifying the type of service requested and subsequently reduced the RMSE to 20%. Next, the distribution of modulation schemes was added in an effort to consider the RF conditions and that managed to bring the RMSE down to 9%. It has to be noted that better RF conditions allow for using higher modulation schemes thereby allocating a smaller number of PRBs to serve the same UE. Modulation types included Quadrature Phase Shift Keying Modulation (QPSK), 16 Quadrature Amplitude Modulation (16QAM), and 64 Quadrature Amplitude Modulation (64QAM).

In the last phase in which the breakthrough in reducing RMSE happened, average cell throughput, peak cell throughput , average UE throughput, and average cell spectral efficiency were added. The latter resulted in reducing the RMSE to less than 0.5%. This can be explained realizing the fact that the throughput counters are the best indicators of link qualities and speeds associated with PRB usage.

Table 3.3: QCI and related services.

QCI	Type	Priority	Delay	Pkt Loss %	Services
1	GBR	2	100ms	10^{-2}	Conversational voice
2	GBR	4	150ms	10^{-3}	Conversational video (live)
3	GBR	3	50ms	10^{-3}	Real-time gaming
4	GBR	5	300ms	10^{-6}	Non-conversational video (buffered streaming)
5	non-GBR	1	100ms	10^{-6}	IMS signaling
6	non-GBR	6	300ms	10^{-6}	Video (buffered streaming) TCP-based (www, email)
7	non-GBR	7	100ms	10^{-3}	Voice, video (live streaming), interactive gaming
8	non-GBR	8	300ms	10^{-6}	Video (buffered streaming) TCP-based (www, email)
9	non-GBR	9	300ms	10^{-6}	Video (buffered streaming) TCP-based (www, email).

Chapter 4

Optimizing Network Parameters

In this chapter we explore various network parameters and settings that could help us solve the capacity and congestion problem by shifting traffic from congested to non congested cells. We calculate impact on various cells' traffic carried and quality of connected users. We then formulate an optimization problem that is customized to UMTS and LTE. We then explore various optimization solutions to maximize network capacity. This needs to be achieved while adhering to constraints of minimum Quality Of Service QOS offered to users, and maximum allowed users on a particular cell, which we predicted from the previous chapter. Some of the techniques that will be explored are Constrained Simulated Annealing CSA, Hill Climbing, Block Coordinated Descent Simulated Annealing BCDSA, and Genetic Algorithm.

4.1 UMTS Network Parameters Optimization

Having modeled the breakpoints of the individual cells within a cluster of cellular towers, we now aim at offloading the load of cells operating close to their breakpoints in order to maximize the capacity of cellular network cluster. In this section, we describe the problem in hand, offer the formulation of an optimization problem aiming at maximizing the capacity of the cluster, and present three different alternatives to solve the problem. The alternatives include the standard constrained SA, the hill climbing SA, and a novel variant of SA inspired by block coordinated descent optimization to which we refer to as BCDSA.

4.1.1 Problem Description

We start by noting that we are trying to maximize the capacity of a cluster of UMTS cell towers such that it can carry more users without facing access failures. In UMTS, the radius and hence coverage of a cell i is predominantly controlled by its CPiCH power referred to as Ω_i. This channel carries both signaling and control messaging. Increasing CPiCH power increases cell radius, while decreasing it decreases cell radius. Decreasing CPiCH also results in freeing up a portion of the power allocated to the amplifier of NodeB, which in turn can be used to carry user traffic instead of signaling. Additionally, the CIO of a cell i referred to as Φ_i controls its coverage at its boundary allowing to expand or shrink cell foot print and thereby serving more or less user traffic.

As discussed earlier, our approach calls for a) reducing Ω_i power of a congested cell i in order to shift traffic to its neighbors and also allocate more power to its own user traffic than signaling traffic, and b) changing the CIO handover threshold Φ_j of a neighboring cell in order to control handover to it without changing the power of cell i. We note that both changes result in shifting existing users on cell edges to be served by neighboring cells at slightly lower quality than the quality experienced when connected to the original cell. The quality experienced by a user is typically represented by E_b/N_t directly mapped to Signal to Interference Noise Ratio (SINR).

In Fig. 4.1, we are illustrating the received signal strength at a mobile user as the user moves from the vicinity of cell tower A to that of cellular tower B. The x-axis is the distance of the user from cell tower A to which the user is initially connected, while the y-axis is the user's received signal strength. In Fig. 4.1a, the blue line labeled A shows that the user's received signal strength from cell A decreases as the distance increases, i.e., as the user travels away from cell A. The green line labeled B shows the user's received signal strength from cell B increases as the distance from cell A increases, i.e., as the user travels toward cell B. The intersection point of blue and green lines represents the initial boundary distance point at which the user is handed over from cell A to cell B. The red line labeled A' shows reducing the value of CPiCH power Ω_A by a sample value of 3dB shrinks the footprint of cell A from r_A to $r_{A'}$. The reduction in power shifts the intersection point to the left causing the handover to occur at a shorter distance from cell A where red line and green line cross. Similarly, Fig. 4.1b shows the increase in the footprint of cell B from r_B to $r_{B'}$ as the result of increasing the value of CIO Φ_B. Increasing Φ_B by a sample value of 3dB shifts the intersection point of the blue line labeled A and the green line labeled B to the left and causes the handover point to occur at a shorter distance from cell A where the blue line and the red line labeled B' cross.

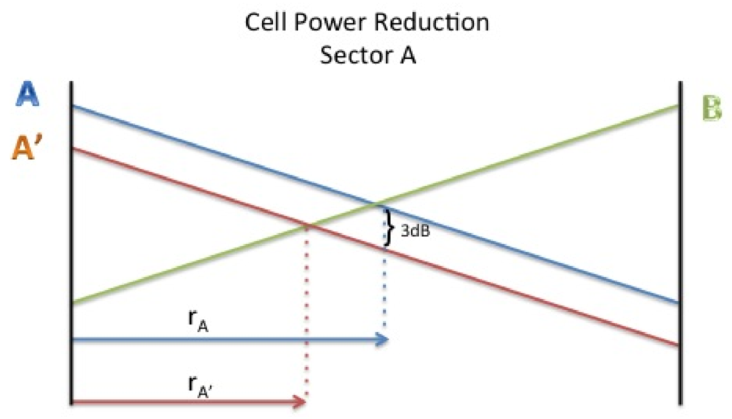
4.1.2 Problem Formulation

We now formulate our optimization problem.

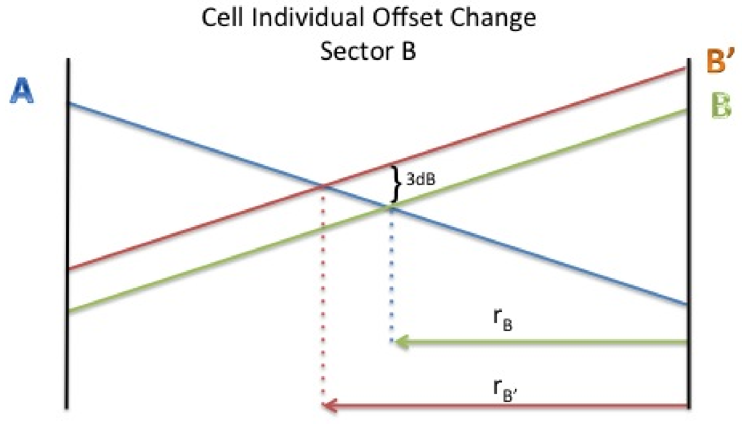
$$\max_{\forall\, \Omega_i, \Phi_i} \quad C_\Upsilon = \sum\nolimits_{i \in I} \big[c_i + c_i^\Omega + \sum_{\substack{j \in I \\ i \neq j}} c_{i,j}^\Phi\big] \tag{4.1}$$

$$\text{S.T.:} \quad \big[c_i + c_i^\Omega + \sum_{\substack{j \in I \\ i \neq j}} c_{i,j}^\Phi\big] \leq C_i, \quad \forall i \in I \tag{4.2}$$

$$q_i \geq Q, \quad \forall i \in I \tag{4.3}$$



(a)



(b)

Figure 4.1: The impact of changing a) CPICH power Ω_A for cell A and b) CIO Φ_B for cell B on reducing cell coverage radius.

Our formulation attempts at maximizing the total cluster capacity C_Υ by controlling transmit power Ω_i and CIO Φ_i on a cell-by-cell basis. The optimization utility function is subject to two constraints. First, the total capacity carried by cell i has to be lower than its maximum available capacity C_i in order to avoid congestion. Second, the quality of cell i denoted by q_i has to meet a minimum acceptable quality threshold of Q explained shortly. The total traffic capacity C_Υ in Eq. (4.17) is the summation of three terms associated with all individual cells. These terms for cell i are the current traffic carried by cell i, the change in cell traffic capacity associated with changing power c_i^Ω, and the sum of changes in cell traffic capacity associated with offloading users from cell i to neighboring cells j after changing CIO values of cell j, $c^{\Phi}_{i,j}$.

We note that the optimization problem formulated above represents a nonlinear programming problem with a total of $2N$ decision variables Ω_i and Φ_i where $i \in \{1, \cdots, N\}$ and decision variables assume values from discrete sets. Hence, the solution to the problem is not necessarily introducing a trivial utility value of $C_{\Upsilon} = \sum_i C_i$ due to discrete values of decision variables and also constraint (4.19). In what follows, we provide a mathematical analysis defining individual terms of the optimization problem.

The change in cell traffic capacity associated with c^{Ω}_i is illustrated by Eq. (4.20) and comprises of the sum of two terms. Both terms can help alleviate the congestion of cell i due to traffic overload.

$$c^{\Omega}_i = \frac{\Delta\Omega_i}{\lambda_i} + \Psi_i(c_i, \Delta\Omega_i) \tag{4.4}$$

The first term $\frac{\Delta\Omega_i}{\lambda_i}$ captures the lowering CPICH power Ω_i of cell i which in turn results in reallocating a portion of the pilot power to carry additional traffic users. It is the ratio of extra power saved from $\Delta\Omega_i$ divided by the average transmit power per user equipment λ_i for cell i which is calculated from collected measurement data. The second term Ψ_i represents traffic offload to the neighboring cells as the result of shrinking the footprint of cell i after changing Ω_i.

In order to calculate Ψ_i, we need to estimate traffic offload associated with the reduction in cell coverage due to cell boundary changes. First, we note that the total path loss Π_i experienced at the boundary of cell i is expressed as a function of Ω_i and ν_i, the adjusted downlink received signal strength at the edge of cell i . Accordingly, the path loss Π_i is expressed below.

$$\Pi_i = \Omega_i - \nu_i \tag{4.5}$$

In a real cell tower, typical values of ν_i have to be at a minimum level of -116dBm after taking into account antenna gains and cable losses.

Next, we apply Hata propagation model presented in [23] to express path loss Π_i as a function of distance from cell tower i.

$$\Pi_i = K_1 + K_2 \log r_i \tag{4.6}$$

In the equation above, K_1 and K_2 are constants depending on the area and morphology, and r_i is the distance from cell tower i. Typical values of K_1 and K_2 for urban environments are -35dB and -40dB/decade [16], respectively. For suburban environments, the values are about -20dB and -30dB/decade, and for rural environments, the values are about -10dB and -20dB/decade, respectively. In our study, we mainly focus on urban environments. Based on the argument above, the distance r_i can be found by comparing the right hand sides of Eq. (4.5) and Eq. (4.6) as shown below.

$$r_i = 10^{[(\Omega_i - \nu_i - K_1)/K_2]} \tag{4.7}$$

It is observed from Eq. (4.7) that reducing Ω_i results in reducing the cell footprint for a fixed value of ν_i. Assuming traffic is homogeneously distributed in the serving area [13], a reduction in served traffic proportional to the reduction in area is resulted. Thus, we can calculate Ψ_i as a function of c_i and $\Delta\Omega_i$ as shown below.

$$\begin{aligned}\Psi_i(c_i, \Delta\Omega_i) &= \Big[1 - \Big(\frac{r_{A'}}{r_A}\Big)^2\Big] c_i \\ &= \Big[1 - \Big(\frac{10^{[(\Omega_i - \Delta\Omega_i - \nu_i - K_1)/K_2]}}{10^{[(\Omega_i - \nu_i - K_1)/K_2]}}\Big)^2\Big] c_i \\ &= \Big[1 - \Big(10^{\frac{-\Delta\Omega_i}{K_2}}\Big)^2\Big] c_i\end{aligned}$$

In an urban environment with a typical cell radius of 1000m, the radii are found to be r_A associated with ($\Omega_i = 33$) and $r_{A'}$ associated with ($\Omega_i = 30$) for node A as shown in Fig. 4.1a . This results in a reduction of about 29% in the serving area and the same percentage reduction in served traffic.

Next, we express $c^{\Phi}_{i,j}$ as a function of $\Gamma_i(c_i, \Delta\Phi_j)$ the amount of traffic offload of cell i to its neighbor j and the area overlap percentage $\eta_{i,j}$ between cells i and j.

$$c^{\Phi}_{i,j} = \eta_{i,j}\ \Gamma_i(c_i, \Delta\Phi_j) \tag{4.8}$$

While the overlap percentage can be calculated from handover statistics on a cell pair basis, $\eta_{i,j}$ is practically set to 40% for front facing neighbors and 10% for co-site neighbors. To understand the definitions of front facing and co-site neighbors, note that in Fig. 2.1 cell 1.1 has front facing neighbors 2.2 and 3.3, and co-site neighbors 1.2 and 1.3.

Similar to $\Psi_i(c_i, \Delta\Omega_i)$, the function $\Gamma_i(c_i, \Delta\Phi_j)$ representing traffic offload from cell i to cell j after changing Φ_j as shown below.

$$\Gamma_i(c_i, \Delta\Phi_j) = \Big[1 - \Big(10^{\frac{-\Delta\Phi_j}{K_2}}\Big)^2\Big] c_i \tag{4.9}$$

Next, we discuss quality constraints. We note that the average quality q_i of cell i after applying new settings is presented as shown below.

$$q_i = \min_j \quad q_{i,j} \tag{4.10}$$

The impact to quality is mainly associated with the shift of cell boundaries due to $\Delta\Omega_i$, $\Delta\Phi_j$, or the sum of them combined. The combined effect results in shifting users at the edge of cell i to a neighboring cell j where they are served by a weaker signal and with a degraded quality. This shift is calculated for each serving cell i and each of its neighbors j. We choose the worst quality value $q_{i,j}$ to present the quality of cell i guaranteed to be not lower than a minimum allowed quality level of Q.

In order to express $q_{i,j}$ as a function of $\Delta\Omega_i$ and $\Delta\Phi_j$, we choose E_b/N_t energy per bit divided by noise total after de-spreading as the quality metric [5]. When reducing the serving cell i power Ω_i, say by 3 dB, the boundary of cell i shrinks forcing the users outside that boundary to be served at a lower quality by a neighboring cell. In the typical environment of our study, the users at the boundary of the serving cell typically experience a reduction factor of about $\phi = 1.5$ in E_b/N_t. Hence, the variations in quality of a user shifted from cell i to a neighboring cell j is expressed as shown below.

$$\Delta(E_b/N_t)_{i,j} = \phi \, . \, (\Delta\Omega_i + \Delta\Phi_j) \tag{4.11}$$

Consequently, the quality impact is captured as shown below.

$$q_{i,j} = (E_b/N_t)_{i,j} - \Delta(E_b/N_t)_{i,j} \tag{4.12}$$

At the end of this subsection, we note that certain reference values have to be selected since we are looking at calculating function variations. In a typical environment of our study, users at a cell boundary experience a reference E_b/N_t value of 10dB. Further, a minimum E_b/N_t value of 7dB is needed in order to support basic voice and data services for covered users [8]. Therefore, Q is set to 7dB.

4.1.3 Solution Approach

In this subsection, we propose a number of algorithms to solve the problem of the previous section. Considering the fact that our formulated problem is a nonlinear optimization problem in which decision variables assume discrete values, we propose to solve our problem using a number of variants of the simulated annealing algorithm after adding a set of penalty terms δ_i to the objective function [25, 37, 6]. Penalty terms are added in order to enforce maximum per cell capacity and quality constraints . The penalty-augmented objective function is then defined below.

$$\tilde{C}_{\Upsilon} = \sum_{i \in I} [c_i + c_i^{\Omega} + \sum_{\substack{j \in I \\ i \neq j}} c_{i,j}^{\Phi} + \delta_i] \tag{4.13}$$

In Eq. (4.25),

$$\delta_i = -[10000\, \mathcal{U}(Q - q_i) + (c_i - C_i)\, \mathcal{U}(c_i - C_i)] \tag{4.14}$$

where $\mathcal{U}(.)$ is the unit step function defined below.

$$\mathcal{U}(y) = \begin{cases} 1, & \text{if } y > 0 \\ 0, & \text{Otherwise} \end{cases} \tag{4.15}$$

It has to be noted that δ_i is a weighted penalty factor applying a constant hard penalty of 10000 for violating the quality constraint (4.19) of cell i and a linear soft penalty proportional to the difference of $c_i - C_i$ for violating the capacity constraint (4.18) of cell i.

4.1.3.1 Constrained Simulated Annealing (CSA)

In the optimization context, an SA algorithm seeks to emulate the annealing process in which a solid material already heated up to high temperatures is allowed to slowly cool until it crystallizes. As the temperature is reduced, the energy of the material decreases until a state of minimum energy is achieved. An SA algorithm begins at high temperature values where input values are allowed to have a great range of variations. As the algorithm progresses, temperature is allowed to fall while restricting input variations. This often leads the algorithm to improve its current solution similar to the actual annealing process. As long as temperature is being decreased, input variations typically lead to successively improved solutions and eventually reaching an optimal set of input values when temperature is close to zero [43, 38, 28].

In our maximization problem, a change in the configuration of the system at temperature T is acceptable if the objective function is increased ($\Delta\tilde{C}_\Upsilon > 0$) or otherwise ($\Delta\tilde{C}_\Upsilon < 0$) may be accepted if the Boltzmann condition below is met [24].

$$\exp\left(\Delta\tilde{C}_\Upsilon / BT\right) \; > \; R \tag{4.16}$$

In the inequality above, R is a random number derived from the uniform distribution $U\,[0, 1]$, T is the temperature, and B is the Boltzmann constant set to one. Additionally, the cooling factor follows a geometric distribution in which the new temperature is the product of the previous temperature and a number smaller than 1 [40, 17]. From (4.16), it is apparent that the probability of accepting non-improving changes depends on both the temperature which is the control parameter and the change in the objective function.

In our problem, we are trying to maximize the traffic carried by the cluster while neither exceeding the maximum traffic allowed by each cell C_i nor degrading the quality below the minimum quality threshold Q. The maximum traffic allowed C_i is predicted by the learning algorithms of Section 3.1, while the minimum quality Q allowed for individual cell is specified based on certain Radio Access Bearers (RABs) requirements.

Our standard SA solution to the problem is illustrated in Algorithm 1. In the algorithm, $\tilde{C}_{\Upsilon}(\underline{x})$ is the total penalty-augmented carried traffic and $\underline{x} = (x_1, x_2 ....., x_N)$ is the set of parameter pairs of individual cells with $x_i = (\Delta\Omega_i, \Delta\Phi_i)$.

The worst case time complexity of the standard SA algorithm is in the order of $\mathcal{O}(\sigma\rho N)$ considering its nested while loops. The number of iterations in the outer loop is set to $\sigma = \frac{\log T_f - \log T_i}{\log a}$ following the number of temperature points from geometric distribution and the number of iterations in the inner loop is set to ρN where ρ is a fixed integer multiplier and N is the number of cellular towers.

4.1.3.2 SA with Hill Climbing

In this subsection, we evaluate the performance of an alternative of the SA algorithm. Referred to as SA with hill climbing, this alternative attempts at evaluating m potential solutions at each step and choosing the best of those solutions before deciding to move to the next step. The additional attempts significantly increase the chances of improving the cost function compared to CSA at the cost of a much higher time complexity as discussed in [34] and evidenced by our results. The worst case time complexity of this algorithm is in the order of $\mathcal{O}(m\sigma\rho N)$ where m is the number of search attempts at each step.

Algorithm 1: CSA()

```
Form penalty-augmented objective function $\tilde{C}_\Upsilon(\underline{x})$
      where $\underline{x} = (x_1, x_2....., x_N), x_i = (\Delta\Omega_i, \Delta\Phi_i)$
Set initial values $\underline{x}[0]$ and $T = T_i$
Set $K = \rho N$ and final value $T_f$
Set cooling factor $a$ in interval $[0, 1]$
$While$ $(T > T_f)$ /* Temperature Bound */
   Set $k = 0$
   $While$ $(k \leq K)$ /* Iteration Bound */
      Choose a random cell $i$, random $\Delta\Omega_i$, and $\Delta\Phi_i$
      $x_i = (\Omega_i - \Delta\Omega_i, \Phi_i + \Delta\Phi_i)$
      $\Delta\tilde{C}_\Upsilon = \tilde{C}_\Upsilon(\underline{x}[k]) - \tilde{C}_\Upsilon(\underline{x}[k-1])$
      $if$ $\Delta\tilde{C}_\Upsilon > 0$
          Accept new solution: $\tilde{C}^*_\Upsilon = \tilde{C}_\Upsilon, \underline{x}^* = \underline{x}$
      $elseif$ $\Delta\tilde{C}_\Upsilon < 0$
         Generate a random number $R$ in interval $[0, 1]$
         $if$ $\exp[\Delta\tilde{C}_\Upsilon/T] > R$
             Accept new solution: $\tilde{C}^*_\Upsilon = \tilde{C}_\Upsilon, \underline{x}^* = \underline{x}$
      $end$ $if/else$
      $k = k + 1$
   $End$ $/*\{While(k < K)\}*/$
   $T = a * T$
$End$ $/*\{While(T > T_f)\}*/$
```

4.1.3.3 Block Coordinated Descent Simulated Annealing

In this subsection, we describe a third solution alternative viewed as the thesis's algorithmic contribution. Our proposed alternative is inspired by the block coordinated descent optimization techniques [41, 19, 27]. Referred to as the BCDSA algorithm, the main idea of this algorithmic variation is to apply the SA algorithm to a partitioned set of decision variables, i.e., optimizing one set while keeping the other set fixed, then optimizing the other set while keeping the first set fixed, and alternating between the two sets.

Compared to traditional nonlinear optimization algorithms, BCDSA offers few major advantages. First and just like SA, BCDSA is able to identify solutions in the vicinity of global optimal point and not get caught up in local optimum points. Second, it can improve the runtimes of SA and other evolutionary optimization algorithms such as genetic algorithms. Third, it can improve the quality of solution measured by best solution value, average solution value, and percentage of solutions in the vicinity of the best solution.

In our problem, there are two per cell decision variables, namely, $\Delta\Omega_i$ and $\Delta\Phi_i$. Hence, the optimization alternates between sets of $\Delta\Omega_i$ values and $\Delta\Phi_i$ values, i.e., the algorithm finds the optimal values of $\Delta\Omega_i$ with fixed values of $\Delta\Phi_i$ for all values of $i \in I$ after selecting a random cell at a time. When the objective function is no longer changing after few iterations as measured by a freeze factor ξ, the algorithm switches between the two sets of decision variables and finds the optimal values of $\Delta\Phi_i$ with fixed values of $\Delta\Omega_i$ for all values of $i \in I$ after selecting a random cell at a time. The algorithm then continues alternating between two sets of decision variables until reaching the optimal point or the maximum number of iterations. The BCDSA algorithm is explained in Algorithm 2.

With respect to convergence, we conjecture that the BCDSA algorithm converges to a local optimal point in the vicinity of the global optimal solution of the problem formulated in Section 4.2.2. To support our claim, we note that [11] proves the convergence of the SA

Algorithm 2: BCDSA()

```
Form penalty-augmented objective function $\tilde{C}_\Upsilon(\underline{x})$
    where $\underline{x} = (x_1, x_2....., x_N), x_i = (\Delta\Omega_i, \Delta\Phi_i)$
Set initial values $\underline{x}[0]$ and $T = T_i$
Set $K = \rho N$ and final value $T_f$
Set cooling factor $a$ in interval $[0, 1]$
Define max freeze factor $\xi_{max}$
$\forall i,$ Optimize $\Delta\Omega_i$ but freeze $\Delta\Phi_i$
$While$ $(T > T_f)$ /* Temperature Bound */
  Set $k = 0,\ \xi = 0$
  $While$ $(k \leq K)$ /* Iteration Bound */
    Choose a random cell $i$
    $if$ Optimizing $\Delta\Omega_i$
      $x_i = (\Omega_i - \Delta\Omega_i, \Phi_i)$
    $elseif$ Optimizing $\Delta\Phi_i$
      $x_i = (\Omega_i, \Phi_i + \Delta\Phi_i)$
    $end\ if/else$
    $\Delta\tilde{C}_\Upsilon = \tilde{C}_\Upsilon(\underline{x}[k]) - \tilde{C}_\Upsilon(\underline{x}[k-1])$
    $if$ $\Delta\tilde{C}_\Upsilon > 0$
      Accept the new solution: $\tilde{C}^*_\Upsilon = \tilde{C}_\Upsilon, \underline{x}^* = \underline{x}$
    $elseif$ $\Delta\tilde{C}_\Upsilon < 0$
      Generate a random number $R$ in interval $[0, 1]$
      $if$ $\exp[\Delta\tilde{C}_\Upsilon/T] > R$
         Accept the new solution: $\tilde{C}^*_\Upsilon = \tilde{C}_\Upsilon, \underline{x}^* = \underline{x}$
    $end\ if/else$
    $if$ $\tilde{C}_\Upsilon[k] = \tilde{C}_\Upsilon[k-1]$ /* $\tilde{C}_\Upsilon$ is not changing! */
      $\xi = \xi + 1$
    $else$
      $\xi = 0$
    $end\ if/else$
    $k = k + 1$
    $if$ $(\xi > \xi_{max})$ /* Switch decision variables */
      $if$ Optimizing $\Delta\Omega_i$
        $\forall i,$ Optimize $\Delta\Phi_i$ but freeze $\Delta\Omega_i$
      $elseif$ Optimizing $\Delta\Phi_i$
        $\forall i,$ Optimize $\Delta\Omega_i$ but freeze $\Delta\Phi_i$
      $end\ if/else$
      $\xi = 0$
    $end$
  $End$ $/*\{While\ (k < K)\}*/$
  $T = a * T$
$End$ $/*\{While\ (T > T_f)\}*/$
```

algorithm to a local optimal point in the vicinity of the global optimal point for proper choices of parameters. Further, BCD algorithms are known to converge to stationary points if the Lagrangian function formed by the objective and the nonlinear constraint functions is convex or under milder conditions quasiconvex and hemivariate [26, 35, 3]. The BCDSA algorithm is primarily an SA algorithm augmented by BCD techniques and hence our choices of parameters warrant its convergence to a local optimal point. The effect of BCD augmentation is in essence improving its average speed and robustness of convergence. We cannot mathematically prove the convergence of the BCDSA algorithm to the global optimal point neither can we prove the average speed of convergence of the BCDSA algorithm is better than that of SA. However, we have consistently observed through extensive simulations that BCDSA robustly converges to the vicinity of the global optimal solution, identified by exhaustive search, in higher speeds and much better confidence intervals than standard SA and SA with hill climbing.

The worst case time complexity of the BCDSA algorithm is in the order of $\mathcal{O}(\sigma\rho N)$ which is identical to that of standard SA. However, BCDSA has a better average time complexity compared to other SA alternatives. Further, it has much better success rates in converging to the vicinity of global optimal solutions than other SA alternatives. The results of next section provide evidence to our claims.

4.2 LTE Network Parameters Optimization

In the previous chapter, we were able to predict the congestion threshold Λ_i representing the 80% congestion threshold of PRB utilization of each cell i within a cluster of cell towers. In this section, we utilize the predicted congestion thresholds in the formulation of an optimization problem aiming to minimize the overall congestion of a cluster of cellular towers by means of shifting traffic from congested cells to their non-congested neighboring cells.

Shifting LTE traffic can be done in two ways. First, adjusting LTE cell power $\wp_i$ of a cell i results in shrinking the footprint of the cell hence shifting UEs on the border to the neighboring cells. Second, artificially changing the handover margin $\hbar_j$ of a neighboring LTE cell j results in making it look stronger thereby triggering an earlier handover. The latter also effectively shrinks the footprint of cell i and shifts border UEs from cell i to cell j. However, traffic offloading has to be controlled to assure the volume of shifted traffic to a neighboring cell keeps the overall load of that neighboring cell below its threshold of congestion.

Hence, our problem aims at identifying the optimal settings of the operating parameters of each cell power $\wp_i$ and handover margin $\hbar_i$ in order to minimize the congestion of the cluster of cell towers as the result of shifting traffic from congested cells to their non congested neighbors. This is achieved subject to satisfying two constraints associated with the minimum acceptable quality experienced by a UE connected to a cell tower and the maximum allowed PRB utilization of the set of cell towers.

4.2.1 Problem Description

We are trying to minimize congestion, measured by the total number of UEs serviced by cell towers operating beyond their 80% PRB utilization. While our algorithm can operate with any choice of congestion threshold, a value of 80% utilization is the typical choice of mobile operators preventing various performance issues such as handover failures and call drops.

In LTE networks, cell power $\wp_i$ is a key factor to determining the cell footprint. Decreasing cell power reduces the footprint of a cell while increasing it increases the footprint. Additionally, the handover margin $\hbar_i$ determines the cell boundary in comparison to the neighboring cells. The latter can also be used to decrease or increase the serving area of a cell. Hence, changing the setting of both of these parameters results in decreasing or increasing the serving area and hence the number of UEs connected to an LTE cell.

Our approach calls for a) reducing $\wp_i$ power of a congested cell i in order to shrink its footprint and hence shift traffic to its neighbors, and b) changing the handover margin $\hbar_j$ of a neighboring cell j in order to increase the footprint of cell j. We note that both changes result in shifting existing connected UEs on cell i edges to be served by its neighboring cells at a slightly lower quality than the quality experienced when connected to the original cell i. The quality experienced by a UE connected to cell i is typically represented by SINR denoted as q_i.

Fig. 4.1 can be used to explain the impact of changing both cell power $\wp_A$ of cell A and the handover margin $\hbar_B$ of cell B on reducing cell A footprint. In Fig. 4.1a The reduction in cell power shifts the intersection point to the left causing the handover to occur at a shorter distance from cell A where red line and green line cross. This means that the cell radius of cell A and hence footprint has shrunk and UEs have been shifted to cell B. Similarly, Fig. 4.1b shows the increase in the footprint of cell B from r_B to $r_{B'}$ as the result of increasing the value of handover margin $\hbar_B$. Increasing $\hbar_B$ by a sample value of 3dB shifts the original intersection point of the blue line labeled A and the green line labeled B to the left and causes the handover point to occur at a shorter distance from cell A where the blue line labeled A and the red line labeled B' cross. Again, this means that the radius of cell A and hence its footprint have shrunk and UEs have been shifted to cell B.

4.2.2 Problem Formulation

We can now formulate our optimization problem as shown below in which $[x]^+ = \max(x, 0)$.

$$\min_{\forall\, \wp_i, \hbar_i} \quad \Lambda_\Upsilon = \sum\nolimits_{i \in I} \Big[\big(\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \big) - \Lambda_i \Big]^+ \tag{4.17}$$

$$\text{S.T.} \qquad \sum\nolimits_{i \in I} \Big[\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \Big] \;=\; \Lambda_L \tag{4.18}$$

$$q_i \geq Q, \quad \forall i \in I \tag{4.19}$$

The formulation attempts at minimizing Λ_Υ the total cluster congestion by changing power $\wp_i$ and handover threshold $\hbar_i$ on a cell-by-cell basis. The optimization cost function is subject to two constraints. First, the total number of UEs connected to all cells has to sum up to the total load of the cluster, Λ_L. This constraint in essence guarantees the preservation of load within the cluster. Second, the quality experienced by a UE connected to cell i denoted by q_i has to meet a minimum acceptable quality threshold of Q explained shortly. The total traffic congestion Λ_Υ in Eq. (4.17) is the difference of the summation of three terms and predicted congestion threshold associated with all individual cells. These terms for cell i are the current UEs λ_i connected to cell i, the change in connected UEs associated with changing power $\lambda_i^{\wp}$, and the sum of changes in connected UEs associated with offloading users from cell i to neighboring cells j after changing handover threshold values of cell j, $\lambda_{i,j}^{\hbar}$. Finally, Λ_i represents the predicted congestion threshold of cell i. The optimization problem formulated above represents a nonlinear programming problem with a total of $2N$ decision variables $\wp_i$ and $\hbar_i$ where $i \in \{1, \cdots, N\}$ and decision variables assume values from discrete sets. Hence, the solution to the problem is not necessarily introducing a trivial utility value of $\Lambda_\Upsilon = \sum_i \Lambda_i$ due to discrete values of decision variables and also constraint (4.19).

Now, we provide a mathematical analysis defining individual terms of the optimization problem. The change in connected UEs associated with $\lambda_i^{\wp}$ represents traffic offload to the neighboring cells as the result of shrinking the footprint of cell i after changing $\wp_i$. The value of $\lambda_i^{\wp}$ is derived utilizing Hata propagation model [23, 2] and assuming traffic is homogeneously distributed in the serving area [13, 30].

$$\lambda_i^{\wp} = \lambda_i \left[1 - \left(10^{\frac{-\Delta \wp_i}{H}} \right)^2 \right] \tag{4.20}$$

In the equation above, H is a constant with typical values of -40, -30, and -20 dB/decade for urban, suburban, and rural environments, respectively.

Similarly, $\lambda_{i,j}^{\hbar}$ is expressed as a function of the traffic offload of cell i to its neighbor j and the area overlap percentage $\eta_{i,j}$ between cells i and j.

$$\lambda_{i,j}^{\hbar} = \eta_{i,j} \, \lambda_i \left[1 - \left(10^{\frac{-\Delta \hbar_j}{H}} \right)^2 \right] \tag{4.21}$$

While the overlap percentage can be calculated from handover statistics on a cell pair basis, $\eta_{i,j}$ is set separately for front facing and co-site neighbors described in section 5.2.

Next, we discuss quality constraints. We note that the average quality q_i of cell i after applying new settings is presented as shown below.

$$q_i = \min_j \quad q_{i,j} \tag{4.22}$$

The impact to quality is mainly associated with the shift of cell boundaries due to $\Delta\wp_i$, $\Delta\hbar_j$, or the sum of them combined. The combined effect results in shifting users at the edge of cell i to a neighboring cell j where they are served by a weaker signal and with a degraded quality. This shift is calculated for each serving cell i and each of its neighbors j. We choose the worst quality value $q_{i,j}$ to present the quality of cell i guaranteed to be not lower than a minimum allowed quality level of Q.

In order to express $q_{i,j}$ as a function of $\Delta\wp_i$ and $\Delta\hbar_j$, we choose SINR of a UE connected to cell i and denoted by γ_i as the quality metric presented in [2, 32]. When reducing the serving cell i power $\wp_i$, by 3 dB, the boundary of cell i shrinks forcing the UEs at $r_{A'}$ to be served at a lower quality by a neighboring cell. In the environment of our study, the UEs at the boundary of the serving cell typically experience a reduction of q_i equivalent to the reduction in power $\wp_i$ and handover margin $\hbar_j$. Hence, the variations in quality of a UE shifted from cell i to a neighboring cell j is expressed as shown below.

$$\Delta\gamma_{i,j} = \Delta\wp_i + \Delta\hbar_j \tag{4.23}$$

Consequently, the quality impact is captured as shown below.

$$q_{i,j} = \gamma_{i,j} - \Delta\gamma_{i,j} \tag{4.24}$$

At the end of this subsection, we note that certain reference values have to be selected since we are looking at calculating function variations. In a typical LTE environment of our study, UEs at a cell boundary experience a reference SINR value of zero dB. Further, a minimum SINR value of -3dB is needed in order to support a minimum modulation scheme of QPSK for covered UEs [2, 33]. Therefore, Q is set to -3dB.

4.2.3 Solution Approach

In this subsection, we propose two optimization algorithms to solve the problem of the previous section. We propose to solve it using BCDSA and GA after adding a set of penalty terms $\mathcal{U}_i$ one per cell i and δ to the objective function [25, 37, 6]. Penalty terms are added in order to enforce quality constraints. The penalty-augmented objective function is then defined below.

$$\tilde{\Lambda}_{\Upsilon} = \sum_{i \in I} \Big\{ \big[\big(\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \big) - \Lambda_i \big]^+ + 10^6 * \mathcal{U}_i \Big\} + 10^6 * \delta \tag{4.25}$$

In Eq. (4.25),

$$\delta = \begin{cases} 1, & \text{if } \big(\sum_{i \in I} \big[\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \big] \neq \Lambda_L \big) \\ 0, & \text{Otherwise} \end{cases} \tag{4.26}$$

and

$$\mathcal{U}_i = \begin{cases} 1, & \text{if } (q_i < Q) \\ 0, & \text{Otherwise} \end{cases} \tag{4.27}$$

It has to be noted that δ is a weighted penalty factor applying a constant large hard penalty, set to 10^6, for violating the load preservation constraint in Eq. (4.18). Numerically, the load preservation constraint in Eq. (4.18) is met by balancing the offloading of connected UEs from a congested cell to its neighbors. Further, $\mathcal{U}_i$ is a weighted penalty factor applying a constant large hard penalty, set to 10^6, for violating the quality constraint in Eq. (4.19) of cell i.

4.2.3.1 Block Coordinated Descent Simulated Annealing

Our first proposed algorithm to solve the problem is Block Coordinated Descent Simulated Annealing algorithm that have been used to solve the UMTS problem in the previous section. We have experimentally observed that BCDSA has a better average time complexity and a much better success rate in converging to the vicinity of global optimal solutions compared to other SA alternatives such as standard SA or SA with hill climbing. In our problem, there are two per cell decision variables, namely, $\Delta\wp_i$ and $\Delta\hbar_i$. Accordingly, the partitioning strategy splits the decision variables to two sets, namely the set of $\Delta\wp_i$ and the set of $\Delta\hbar_i$ values. We used the same BCDSA algorithm, illustrated in Algorithm 3 to minimize Λ_Υ , by replacing $\tilde{C}_\Upsilon, \Delta\Omega_i, and \Delta\Phi_i$ by $\tilde{\Lambda}, \Delta\wp_i$, and $\Delta\hbar_i$, respectively.

We conjecture that the BCDSA algorithm converges to a local optimal point in the vicinity of the global optimal solution of the problem formulated in Section 4.2.2. To support our claim, we note that [11] proves the convergence of the SA algorithm to a local optimal point in the vicinity of the global optimal point for proper choices of parameters. Further, BCD algorithms are known to converge to stationary points if the Lagrangian function formed by the objective and the nonlinear constraint functions is convex or under milder conditions quasiconvex and hemivariate [35, 3]. The BCDSA algorithm is primarily an SA algorithm augmented by BCD techniques and hence our choices of parameters warrant its convergence to a local optimal point. The effect of BCD augmentation is in essence improving its average speed and robustness of convergence.

The worst case time complexity of the BCDSA algorithm is in the order of $\mathcal{O}(\sigma\rho N)$ considering its nested while loops. The number of iterations in the outer loop is set to $\sigma = \frac{\log T_f - \log T_i}{\log a}$ where T_i, T_f, and a are the initial temperature, final temperature, and cooling factor of BCDSA algorithm. The number of iterations in the inner loop is set to ρN where ρ is a fixed integer multiplier and N is the number of cellular towers.

Algorithm 3: BCDSA()

```
Form penalty-augmented objective function $\tilde{\Lambda}_\Upsilon(\underline{x})$
      where $\underline{x} = (x_1, x_2....., x_N), x_i = (\Delta\wp_i, \Delta\hbar_i)$
Set initial values $\underline{x}[0]$ and $T = T_i$
and $\underline{x}[0]$
Set $K = \rho N$ and final value $T_f$
Set cooling factor $a$ in interval $[0, 1]$
Define max freeze factor $\xi_{max}$
$\forall i,$ Optimize $\Delta\wp_i$ but freeze $\Delta\hbar_i$
$While$ $(T > T_f)$ /* Temperature Bound */
   Set $k = 0,\ \xi = 0$
   $While$ $(k \leq K)$ /* Iteration Bound */
      Choose a random cell $i$
      $if$ Optimizing $\Delta\wp_i$
         $x_i = (\wp_i - \Delta\wp_i, \hbar_i)$
      $elseif$ Optimizing $\Delta\hbar_i$
         $x_i = (\wp_i, \hbar_i + \Delta\hbar_i)$
      $end\ if/else$
      $\Delta\tilde{\Lambda}_\Upsilon = \tilde{\Lambda}_\Upsilon(\underline{x}[k]) - \tilde{\Lambda}_\Upsilon(\underline{x}[k-1])$
      $if$ $\Delta\tilde{\Lambda}_\Upsilon < 0$
         Accept the new solution: $\tilde{\Lambda}^*_\Upsilon = \tilde{\Lambda}_\Upsilon, \underline{x}^* = \underline{x}$
      $elseif$ $\Delta\tilde{\Lambda}_\Upsilon > 0$
         Generate a random number $R$ in interval $[0, 1]$
         $if$ $\exp[\Delta\tilde{\Lambda}_\Upsilon/T] > R$
            Accept the new solution: $\tilde{\Lambda}^*_\Upsilon = \tilde{\Lambda}_\Upsilon, \underline{x}^* = \underline{x}$
      $end\ if/else$
      $if$ $\tilde{\Lambda}_\Upsilon[k] = \tilde{\Lambda}_\Upsilon[k-1]$ /* $\tilde{\Lambda}_\Upsilon$ is not changing! */
         $\xi = \xi + 1$
      $else$
         $\xi = 0$
      $end\ if/else$
      $k = k + 1$
      $if$ $(\xi > \xi_{max})$ /* Switch decision variables $*/$
         $if$ Optimizing $\Delta\wp_i$
            $\forall i,$ Optimize $\Delta\hbar_i$ but freeze $\Delta\wp_i$
         $elseif$ Optimizing $\Delta\hbar_i$
            $\forall i,$ Optimize $\Delta\wp_i$ but freeze $\Delta\hbar_i$
         $end\ if/else$
         $\xi = 0$
      $end$
   $End$ $/*\{While\ (k < K)\}*/$
   $T = a * T$
$End$ $/*\{While\ (T > T_f)\}*/$
```

4.2.3.2 Genetic Algorithm

Depending heavily on randomness thereby allowing it to explore vast solution spaces, GA is an evolutionary algorithm widely used in optimization domain, especially, for solving nonlinear large solution space problems [31]. It is known to identify near-global optimal points without getting trapped in local optima. A flowchart of the general operation of GA is provided in Fig. 4.2 in which an initial population solution is iteratively evolved utilizing three types of operators, namely, Selection, Crossover, and Mutation until reaching a final population solution.

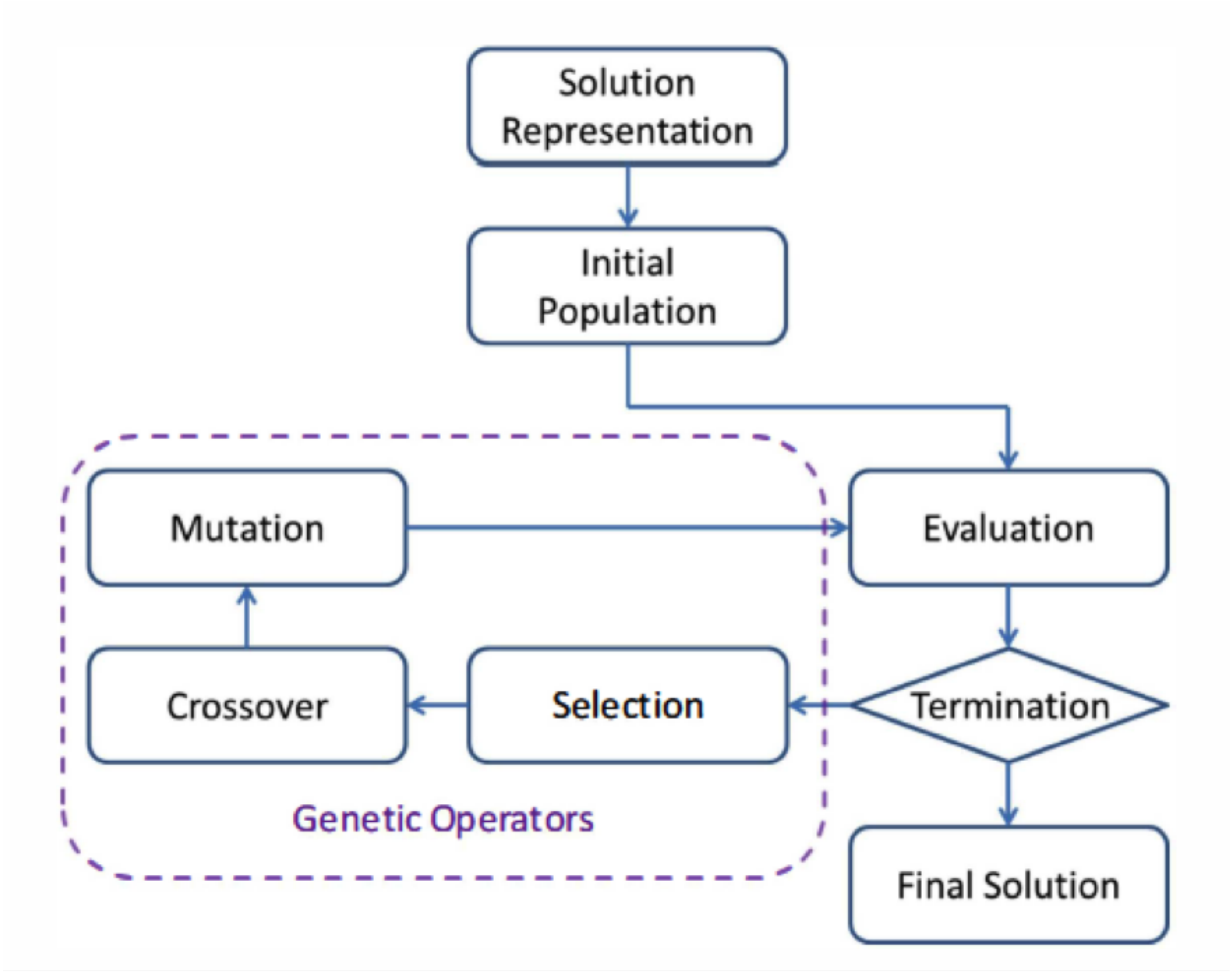


Figure 4.2: A flowchart of GA operation.

The GA algorithm applied to solve our problem is illustrated in Algorithm 3. In applying GA to our problem, chromosomes are sets as vectors of genes. The number of genes is set to 60 presenting power $\wp_i$ and handover margin $\hbar_i$ of a total of 30 cells. The range of these parameters is $[0, 3]$ in order to enforce allowable degradation bounds of quality constraints of LTE systems as captured by Eq. 4.25.

Starting from an initial population, operators are applied to evolve the population. Applying Selection (aka Elite) operator results in choosing a certain percentage of top ranked chromosomes with the lowest cost values as chromosomes of the next generation. In our solution, we set the percentage value to 2%. As illustrated by Fig. 4.3, Crossover operator is used to select a pair of chromosomes in order to create offsprings. The new population is ranked again in order to keep its top chromosomes and discard the rest. The last operator used is Mutation. As illustrated by Fig. 4.4, a random chromosome is chosen and the value of a number of its genes are changed to new values. This operator allows the GA algorithm to jump to unexplored areas of the solution space that may have never been explored by other operators or would have taken a much longer time to converge to. Hence, it could help the algorithm escape local optima. Similar to the case of other operators, chromosomes with lowest cost values are kept in the population count and the rest are discarded after applying Mutation operator.

The worst case time complexity of the GA algorithm is in the order of $\mathcal{O}(n\chi N)$ where n is the GA initial population count, χ is a fixed integer multiplier depending on the number of decision variables and their ranges, and N is the number of cellular towers.

Algorithm 3: GA(*Topology*, *Thresholds*)

```
Set real number multiplier $\chi$
Set population size $\kappa = \chi * N$
Set chromosomes $\underline{x}_j = (\Delta\wp_1, \Delta\hbar_1, \cdots, \Delta\wp_N, \Delta\hbar_N)_j$
      with $j \in \{1, \cdots, \kappa\}$ and genes $(\Delta\wp_i)_j, (\Delta\hbar_i)_j$
Set initial population matrix $P[1] = (\underline{x}_1, \cdots, \underline{x}_\kappa)^T$
Form penalty-augmented objective function $\tilde{\Lambda}_\Upsilon(\underline{x}_j) with\, j \in \{1, \cdots, \kappa\}$
Set $g = 1$, $\xi = 0$, and $\xi_{max} = 10$
$While$ (g< $MaxGen$) {  /* Gen. # Bound */
   /* Form elite, crossover, mutation pools */
  $For$ ($j = 1$ $to$ $\kappa$) {
     Rank chromosomes in population $P[g]$ according to values of $\tilde{\Lambda}_\Upsilon(\underline{x}_j)$
     Form Elite Pool (EP) from lowest 2% of ranked values in $P[g]$
     Randomly assign 80% of the remaining chromosomes in $P[g]$ to Crossover Pool (CP)
     Assign the remaining 18% chromosomes to Mutation Pool (MP)
  }
  /* Begin creating the new generation $P[g+1]$ */
     Assign all chromosomes in EP to $P[g+1]$
     $While$ (CP is not empty) {
        Randomly select chromosomes $\mathcal{C}_1$, $\mathcal{C}_2$ from CP
        Cross over genes from chromosomes $\mathcal{C}_1$ and $\mathcal{C}_2$
        Save resulting chromosomes into $P[g+1]$
        Remove $\mathcal{C}_1$ and $\mathcal{C}_2$ from CP
     }
     $While$ (MP is not empty) {
        Randomly select chromosome $\mathcal{C}$ from MP and a gene $\psi$ from $\mathcal{C}$
        Randomly change the value of $\psi$
        Save resulting chromosome into $P[g+1]$
        Remove $\mathcal{C}$ from MP
     }
  /* End creating the new generation $P[g+1]$ */
  $if$ $\frac{(\min\tilde{\Lambda}_\Upsilon \text{ in } P[g]) - (\min\tilde{\Lambda}_\Upsilon \text{ in } P[g+1])}{\min\tilde{\Lambda}_\Upsilon \text{ in } P[g]} < \epsilon$       /* $\min\tilde{\Lambda}_\Upsilon$ is not changing! */
     $\xi = \xi + 1$
  $else$
     $\xi = 0$
  $end\ if/else$
  $if$ ($\xi > \xi_{max}$), $then$ break
  g=g+1
  P[g]=P[g+1]
} $/*$ $While$ (g< $MaxGen$) $*/$
Report best solution: $\tilde{\Lambda}^*_\Upsilon = \tilde{\Lambda}_\Upsilon(\underline{x}_j)$, $\underline{x}^* = \underline{x}_j$ in $P[g]$
```

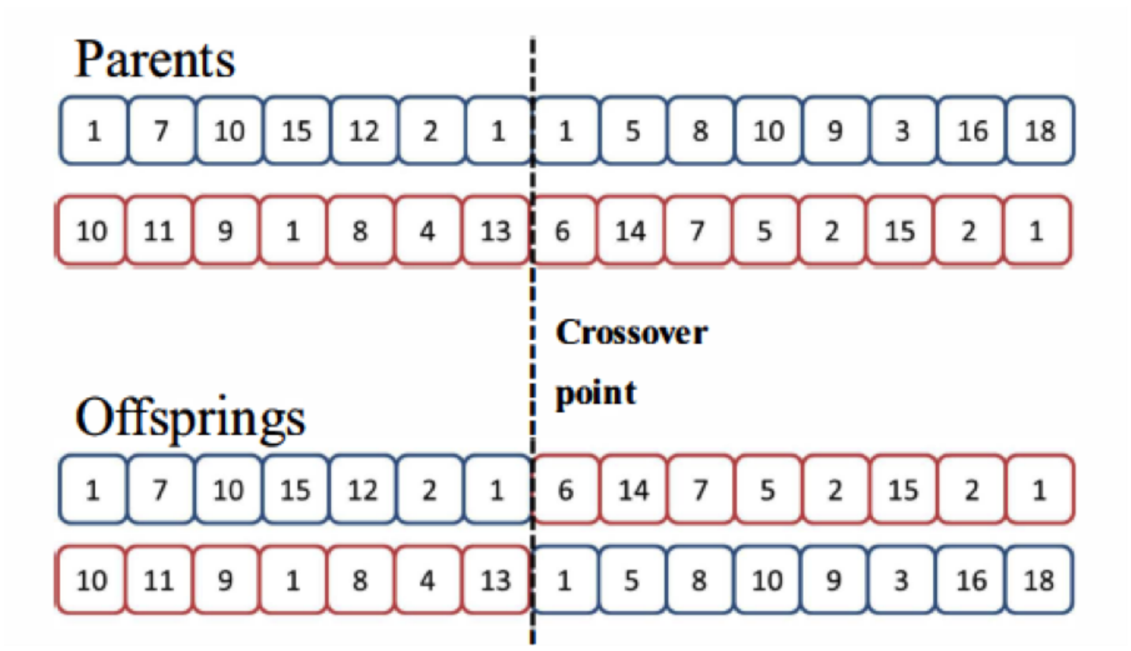


Figure 4.3: An illustration of Crossover operator in GA.

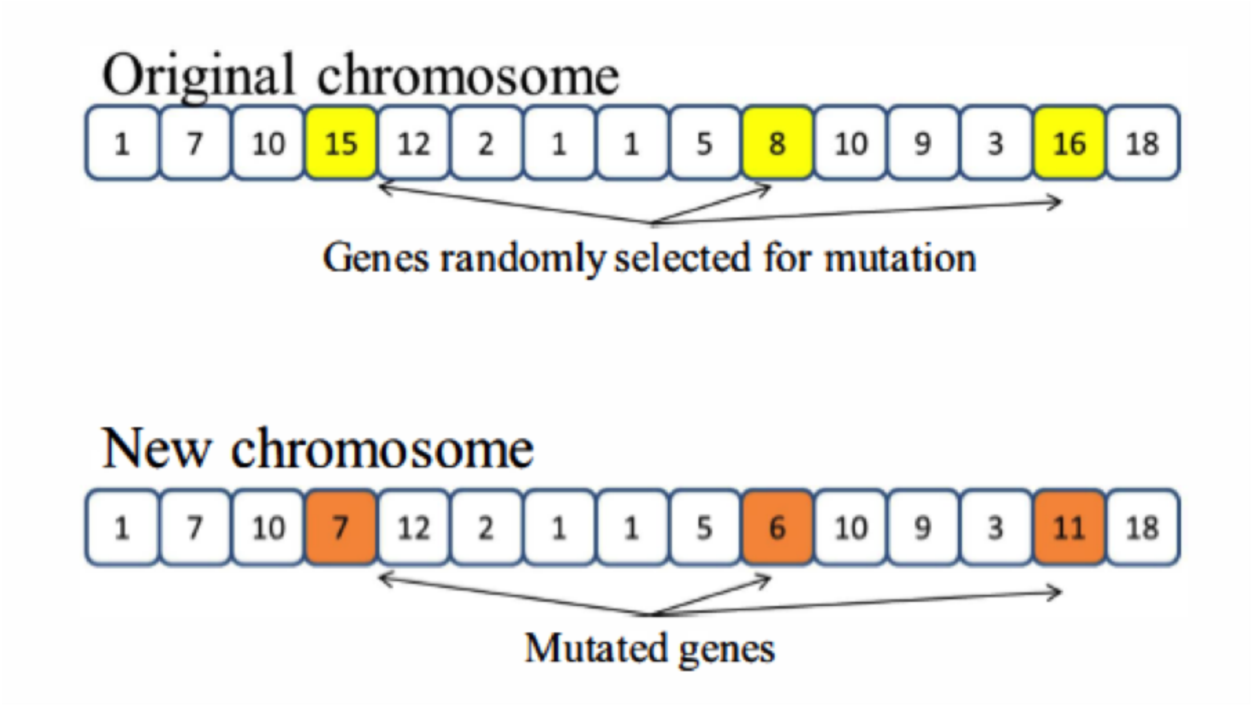


Figure 4.4: An illustration of the Mutation operator of genetic algorithm.

Chapter 5

Experimental Results

In this chapter we detail the assumptions of the problem, parameters changes suggested and calculations of impact to network as a result of these changes on UMTS and LTE. We then show the results of various optimization algorithms like Simulated Annealing SA, Hill Climbing, Block Coordinated Descent Simulated Annealing BCDSA, and Genetic Algorithm GA applied to both UMTS and LTE optimization problems. We also present detailed comparison of various algorithm parameters like initial temperature and cooling factors for SA, freeze values for BCDSA, and populations size and initial population values for GA optimization. We compare all these algorithms based on the average best solution that can be achieved, time needed to reach this solution, and probability of finding a solution within 1% of the best solution achieved for the network.

5.1 UMTS Results

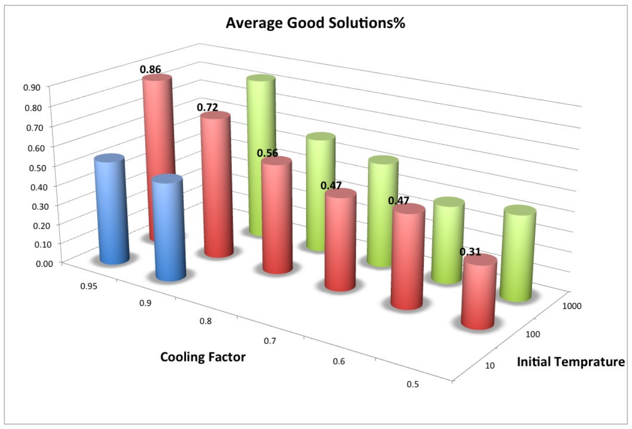
In this section, we first describe our assumptions, parameter settings of our experiments, and error handling associated with learning for UMTS network. Then, we report comparison results of the algorithms discussed in Section 4.1.

5.1.1 Simulation Assumptions, Parameter Settings, and Error Handling

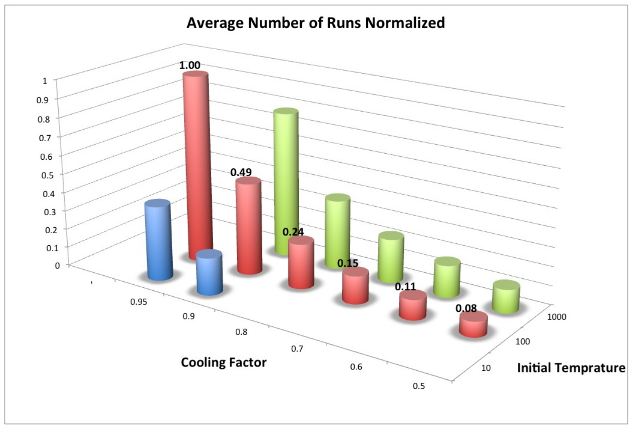
We make several assumptions as described below. We assume only a number of but not all cells are congested and further congested cells have at least a neighbor that is not congested. Further, the cellular network is operating in an urban environment with propagation loss coefficients $K_1 = -35$dB and $K_2 = -40$dB/decade. For simplicity, we assume 40% handoff from a cell to its two facing neighbors and 10% to its co-site neighbors. Further, a reduction in Ω_i or increase in Φ_j results in 1.5 times reduction in E_b/N_t for border users based on the selected urban environment and the typical inter site distance. Traffic is homogeneously distributed in the serving area and hence reduction in traffic served is at a rate similar to reduction in serving area. The range of variations of both $\Delta\Omega_i$ and $\Delta\Phi_i$ is $[0, 6]$dB with a granularity of 0.2dB. We assume that border users are being served with an E_b/N_t value of about 10dB and a minimum acceptable value no smaller than 7dB [15]. The baseline for total traffic carried by all cells in the cluster in our scenario(s) of interest is measured to be 7937 Erlangs without parameter optimization.

It is critical to choose SA parameters in order to control the time complexity of the solution while achieving good utility measures. In order to find the best values of the parameters, i.e., the initial temperature T_i and the cooling factor a, we run a number of experiments the results of which are shown in Fig. 5.1. We observe that small values of T_i result in short runtimes but poor utility measures. On the other hand, large values of T_i result in longer runtimes but better utilities. Consistent with the findings of [24], our results show that the best value of T_i is in the same order as the value of change in the utility function. For example, if we expect the utility function to change in the range of 10 users, then T_i has to be set to a value close to 10.

Our results further show that choosing a value of a in the range of $[0.5, 0.8]$ offers relatively quick convergence but poor values of the utility. To the contrary, values in the range of $[0.95, 0.99]$ result in lower speeds of convergence but good utilities. Hence, we choose a value of $a = 0.9$ as our best practical finding addressing the tradeoff between runtime and utility performance.



(a) Percentage of average good solutions.



(b) Normalized average number of runs.

Figure 5.1: SA results showing an initial temperature of 100 and a cooling factor of 0.9 address the tradeoff between utility reliability and runtimes.

Last but not least, our measurements have shown prediction RMSE errors in the range of $[2\%, 3\%]$ for the maximum available capacity C_i of cell i using deep learning. In consideration of the error, we apply a safety margin of 3% to the predicted cell maximum available capacity C_i when running optimization. This ensures that non-congested cells accepting offloaded traffic do not exceed their maximum capacities and access failures associated with the prediction of breakpoints are eliminated.

5.1.2 Algorithmic Comparison Results

In conducting simulations, we run each scenario 20 times. Then, we compare the results obtained from various algorithms. In comparing theses results, we look at different aspects of performance in terms of a) cost measured as the algorithmic runtime, and b) quality measured as the best solution value, average solution value, and percentage of solutions that are within 1% of the best solution. This last parameter reflects the success rate of an algorithm in finding good solutions.

In the figures of this section, the label *SA Alternating* represents a solution in which we switch the optimization parameters from CPICH to CIO in every step.

Fig. 5.2 provides a comparison of mean, maximum, and minimum values of the traffic carried by different algorithms of Section 4.1 within the cluster of our study with an initial traffic capacity of 7937 Erlangs. Looking at the results of Fig. 5.2, we observe that the highest solution value at 8320 Erlangs is generated by the SA hill climbing algorithm with $m = 8$. However, the average solution value for this algorithm seems to be much lower at 8156 Erlang only. The best algorithm in terms of the average solutions value is BCDSA with a freeze parameter setting of $\xi = 20$ at 8211 Erlangs. This algorithm is found to generate the second maximum carried traffic value at 8264 Erlangs, almost tied with the SA hill climbing algorithm with a parameter setting of $m = 4$. Comparing this last algorithm with the standard SA algorithm at an average value of 8127 Erlangs and best value of 8224 Erlangs shows the improvement achieved by the BCDSA algorithm.

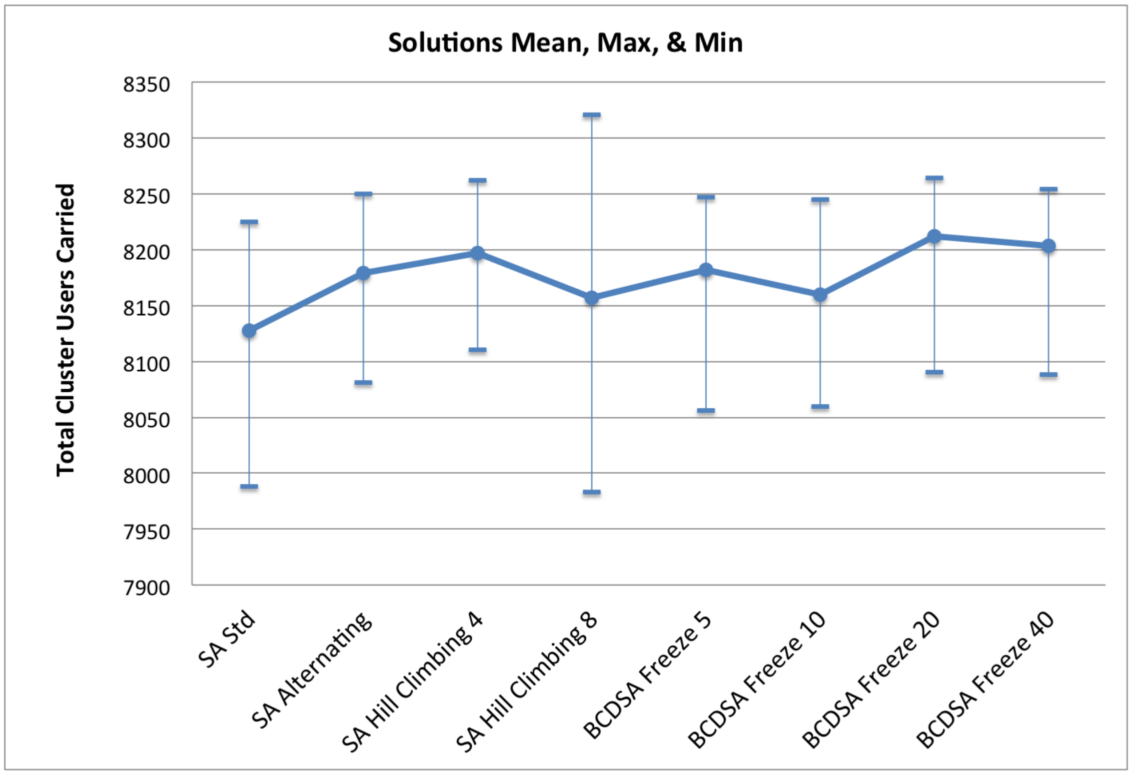


Figure 5.2: A comparison of mean, maximum, and minimum values of the traffic carried by different algorithms within the cluster of our study.

Fig. 5.3 illustrates the convergence behavior captured by the average number of changes per temperature point and overall runtimes of various algorithms. The data in the figure shows that almost all algorithms are close in the average number of 1980 changes for each temperature points. These results also show that the BCDSA algorithm with a parameter setting of $\xi = 5$ has the lowest runtime at 4.76 seconds and in general BCDSA runtimes are shorter than the 5.42 seconds needed for the standard SA algorithm. Hence, BCDSA is up to 12% faster than the standard SA algorithm. The reason is mainly due to the fact that BCDSA escapes local minima faster than the standard SA using its parameter switching mechanism. The results also show that using SA hill climbing with $m = 4$ or 8 results in having runtimes of four to seven times higher than typical runtimes of the other two algorithms.

Finally, Fig. 5.4 offers a comparison of the highest number of users carried and success

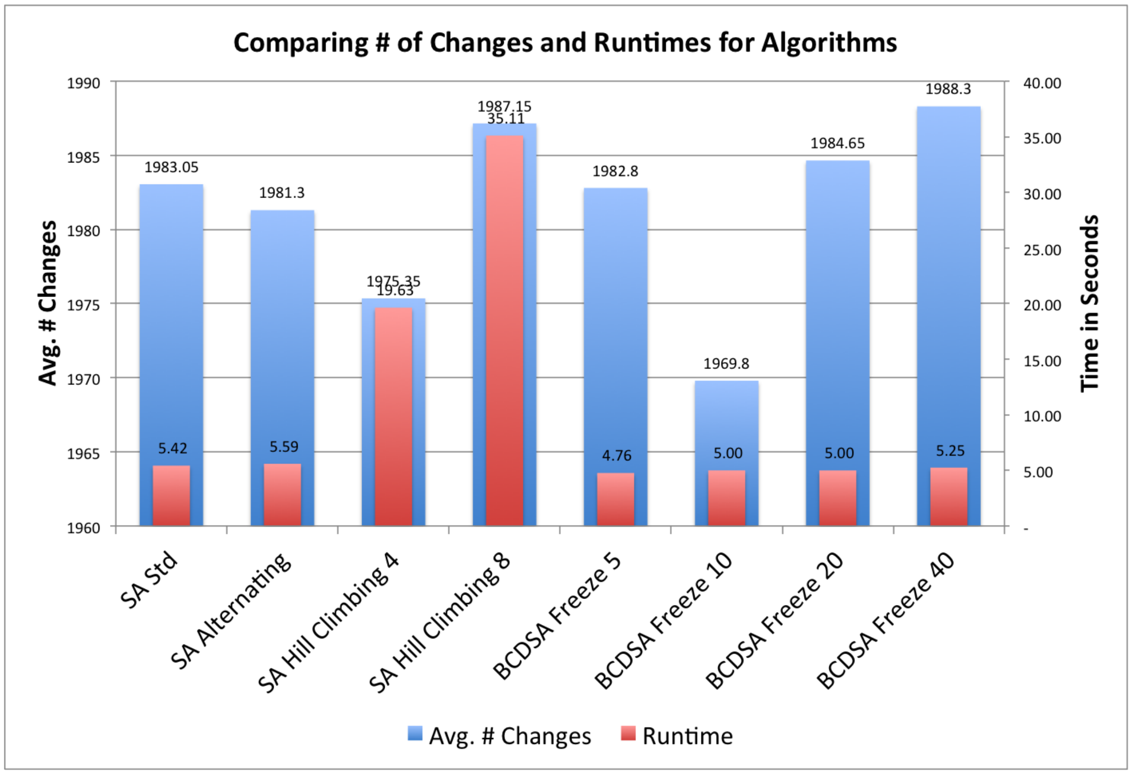


Figure 5.3: A comparison of average number of changes per temperature point and runtimes of different algorithms.

rate percentage for different algorithms within the cluster of our study. The results show that BCDSA with a parameter setting of $\xi = 20$ has the highest percentage of success rate at 95%, i.e., 95% of the solutions generated by this algorithm are within 1% of the best solution value of 8264 Erlangs. Comparing this to standard the SA algorithm at about 45% success rate shows the dramatic improvement reached using our novel BCDSA algorithm. In addition, the BCDSA algorithm with a parameter setting of $\xi = 20$ has the highest average traffic amongst all algorithms at 8264 Erlangs. While we observe the same success rate for BCDSA with a parameter setting of $\xi = 40$, the latter has a lower average traffic at 8254 Erlangs.

From the combined results above, we can see that the BCDSA algorithm with a parameter setting of $\xi = 20$ has the highest success rate at 95%, the second highest user traffic at 8264 Erlangs compared to the highest user traffic solution reached by SA hill climbing with $m = 8$

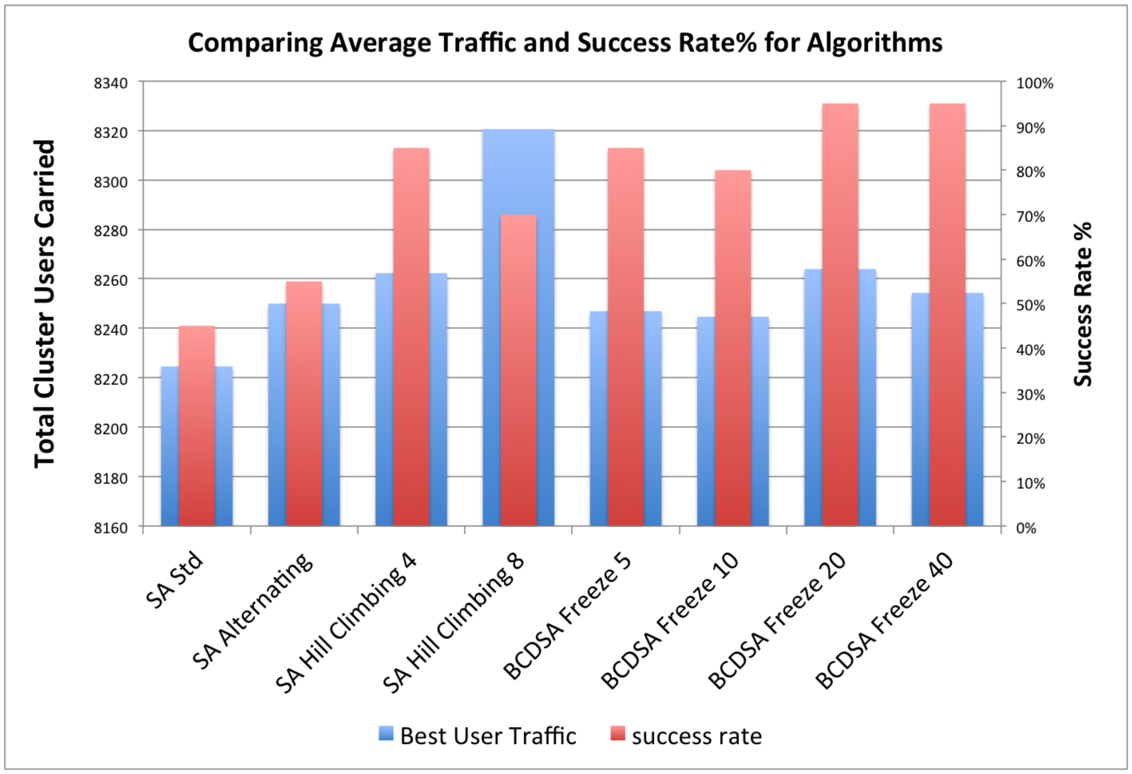


Figure 5.4: A comparison of the highest number of users carried and success rate percentage for different algorithms.

at 8320 Erlangs, and the second lowest average runtime at 5 seconds compared to BCDSA with a parameter setting of $\xi = 5$ at 4.75 seconds.

5.2 LTE Results

In this section, we first describe our experimental settings for the LTE network and then report comparison results of the algorithms discussed in Section 4.2.

5.2.1 Experimental Settings

In our experiments, we use a sample cluster of LTE cellular towers in a dense urban downtown of a US city as depicted in Fig. 2.1. The cluster has ten sites, with each site having three sectors or cells and each cell presented with an arrow. In this scenario, red arrows represent congested cells while black arrows represent non-congested cells. With the exception of boundary cells, each cell has 2 front facing and 2 co-site neighbors. To understand the definitions of front facing and co-site neighbors, note that in Fig. 2.1 cell 1.1 has front facing neighbors 2.2 and 3.3, and co-site neighbors 1.2 and 1.3. As shown by the figure, not all cells are congested and further congested cells have at least a neighbor that is not congested. In addition, the cellular network serves an urban environment with a propagation loss coefficient of $H = -40$dB/decade. Furthermore, 40% handoff from a cell to its two facing neighbors and 10% to its co-site neighbors are assumed.

A reduction in $\wp_i$ or increase in $\hbar_j$ results in a similar reduction in SINR for border users based on the selected urban environment and the typical inter site distance. Traffic is homogeneously distributed in the serving area and hence reduction in traffic served is at a rate similar to reduction in serving area. The range of variations of both $\Delta\wp_i$ and $\Delta\hbar_i$ is $[0, 3]$dB with a granularity of 0.1dB. Border users are being served with an SINR value of 0dB and a minimum acceptable value no smaller than -3dB [2]. The latter is the minimum value of SINR needed to achieve QPSK coding and throughput as presented in Table 2.1. In our scenario(s) of interest, the baseline value of Λ_Υ, i.e., total average load of connected UEs in the cluster, is measured as 2836 users.

It is critical to choose BCDSA parameters to control the time complexity of the solution while achieving good utility results. In order to identify the best values of the initial temperature T_i and the cooling factor a, we run a number of scenarios. We observe that low values of T_i result in shorter runtimes but poor utility measures. On the other hand, high values of T_i result in longer runtimes but better utilities. Our results are consistent with the findings of [24], showing that the best value of T_i is in the same order as the value of change in the utility function. Our results also show that choosing a value of the cooling factor a in the range of $[0.5, 0.8]$ offers relatively faster convergence but poor values of the utility.

On the other hand, values in the range of $[0.95, 0.99]$ result in slower convergence but good utilities. Hence, we choose a value of $a = 0.9$ as our best practical finding addressing the tradeoff between runtime and utility performance. As for BCDSA, we experimented with various values for the freeze parameter ξ. Results showed that the BCDSA algorithm with a parameter setting of $\xi = 20$ best addresses the tradeoff between success rate and runtimes.

Last but not least, measurements have shown deep learning prediction RMSE errors in the range of $[0.5\%, 1\%]$ for the congestion threshold of Λ_i of cell i associated with 80% PRB utilization. In consideration of the error, a safety margin of 3% is applied to the predicted value of Λ_i when running optimization. This ensures that non-congested cells accepting offloaded traffic do not exceed their congestion threshold as a result of prediction error.

In conducting simulations, we run each GA experiment 10 times for each value of initial population count starting from 10 and ending at 200 chromosomes. Considering the fact that BCDSA is an order of magnitude faster than GA, we run each BCDSA experiment 100 times. The purpose of running multiple iterations of each algorithm is to measure the best and average total traffic values and also to measure the consistency of algorithms in finding good solutions. A solution is considered good if its congestion value is within 1% of the best congestion solution obtained using that algorithm.

5.2.2 Comparison Results

In this section, we compare the results obtained from both algorithms in the previous section. In comparing the results, different aspects of performance are viewed in terms of *i*) cost measured as the algorithmic runtime, *ii*) improvement measured as best and average congestion reduction values, and *iii*) success rate measured as the percentage of good solutions, i.e., the number of solutions within 1% of the best solution. This last parameter measures the consistency of an algorithm in finding good solutions.

First, we attempt at identifying the best selection of GA parameters in our experiments and then compare the results of BCDSA and GA. The scenarios of interest for GA include the following configurations in which

a) all $\wp$ and $\hbar$ are initialized with a value of 0;

b) all $\wp$ and $\hbar$ are initialized with values of 1;

c) $\wp$ and $\hbar$ are initialized with values of 0 and 1 respectively;

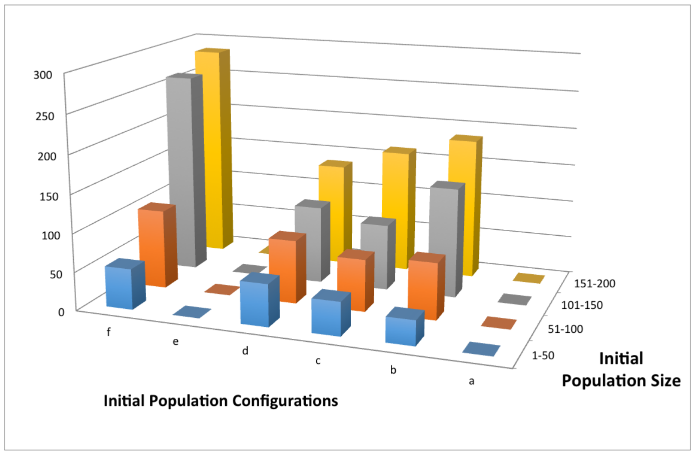
d) $\wp$ and $\hbar$ are initialized with values of 0 and 2 respectively;

e) $\wp$ and $\hbar$ are initialized with values of 0 and 3 respectively; and

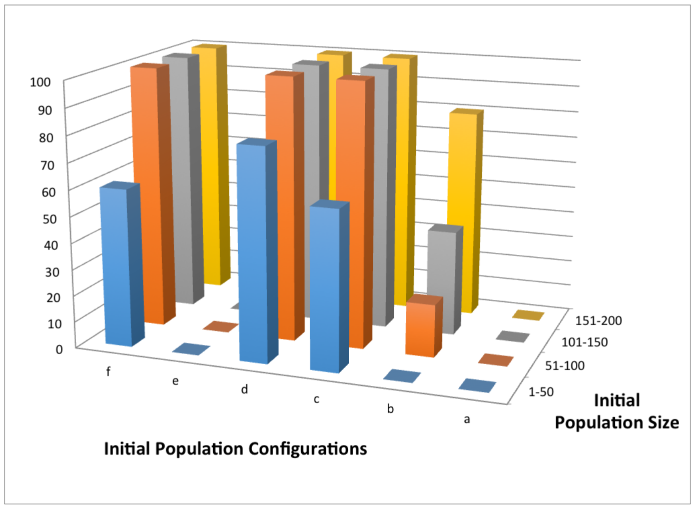
f) $\wp$ and $\hbar$ are assigned random values in the range [0,3].

A comparison of average runtimes and success rates for various GA configurations above is presented in Fig. 5.5. The reported results reflect averages calculated over 10 runs. As can be seen in this figure, configurations (a) and (e) do not converge to any good solutions. We chose configurations (c), (d), and (f) with an initial population of 100 chromosomes as they offer the lowest runtimes while achieving near perfect success rates.

Fig. 5.6 compares the results of various scenarios of GA with BCDSA in terms of success rate percentage and runtime associated with 10 runs. It shows that the success rate of most scenarios of GA after 10 runs is nearly 100% guaranteeing to reach a solution that is within 1% of the best solution in 10 attempts. Reviewing the results of BCDSA, a 10 run success rate of 63% is observed implying that the algorithm finds a good solution within 1% of the best solution 63 times in 100 runs. All GA algorithms record runtimes in the range of 60 to 70 seconds for 10 runs. Comparing these numbers to the runtimes of BCDSA averaging to 5.7 seconds for 10 runs, it is concluded that BCDSA is over one order of magnitude faster than GA. The excellent runtime efficiency advantage of BCDA over GA is hence traded off



(a)



(b)

Figure 5.5: A comparison of (a) average runtimes and (b) success rates for various GA configurations.

against its relatively lower success rates.

The difference in runtimes and success rates can be intuitively explained based on the understanding of how each algorithm works. On one hand, the GA algorithms creates multiple solutions in the population and attempts at optimizing them using crossover and mutations to reach a global optimum. Hence, the chance of getting to that global optimum is higher since it approaches the solution from various directions. This leads to a higher success rate. However, it takes longer to process all these solutions. On the other hand, BCDSA only attempts at navigating its way to the global optimum. Hence, it offers a much lower pro-

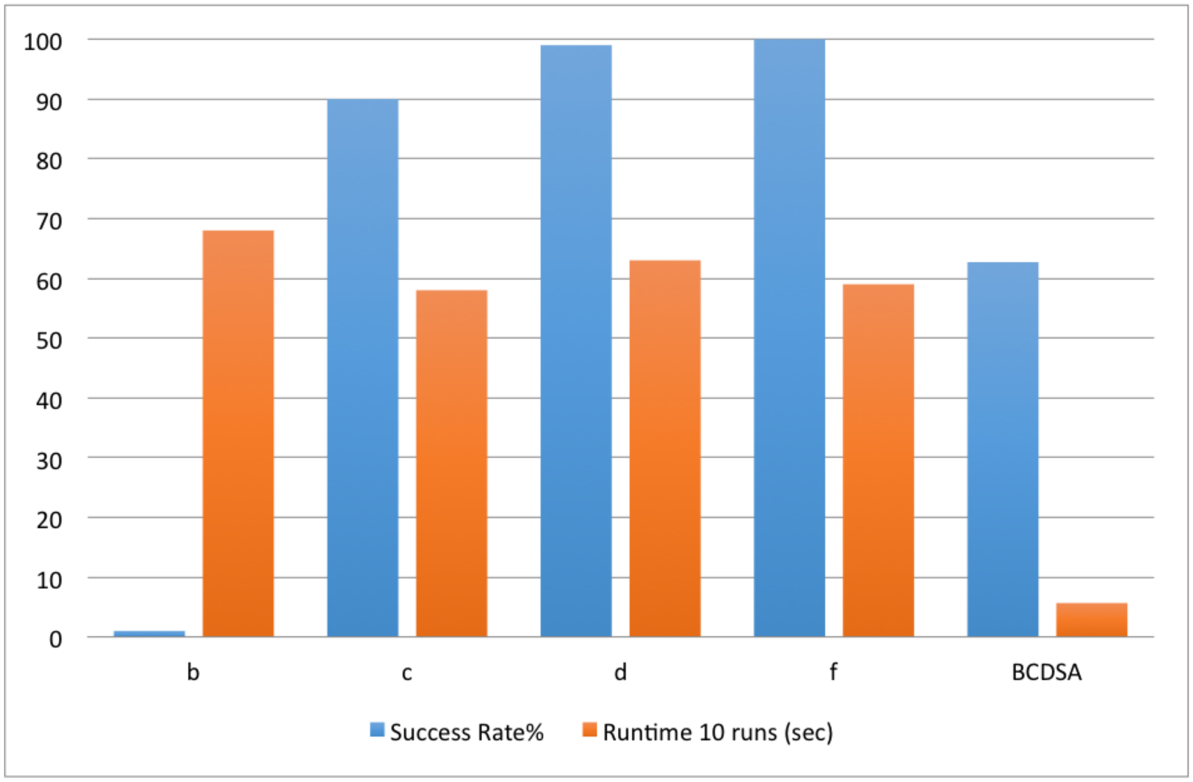


Figure 5.6: A comparison of runtimes (in seconds) and success rates (in percents) for various scenarios of GA and BCDSA calculated over 10 runs.

cessing time than that of GA. However, there is a higher chance of getting trapped in a local minimum and missing the global minimum since BCDSA approaches the global minimum from only one direction. A good analogy to this would be hiring 100 amateur hikers to find the mountain summit, versus hiring one professional hiker to navigate around the terrain and find that summit.

Utilizing a maximum runtime of 60 seconds, Fig. 5.7 compares average and best congestion reduction in various scenarios of GA and BCDSA from a baseline congestion of 506 connected UEs as predicted by deep learning. It is noted that the remaining congestion is the difference between the baseline value of 506 and what is shown in the graph. It is seen that the best and average congestion reduction solutions are very similar comparing most scenarios except scenario (b) in which all parameters are initialized with values of 1. The average case of congestion is not shown in the latter case because the average value increases the baseline congestion of 506 instead of decreasing it. As seen by the results, the total volume of congested traffic is reduced from 506 to 302 representing 40.3% overall congestion reduction within the cluster.

In comparing the performance of the two algorithms, it is observed that GA has significantly higher runtimes than BCDSA. Optimal solutions usually result in $\hbar_i$ variations in the range of 1 to 3 dB and $\wp_i$ variations close to zero. It is also observed that initializing the population with random values of $\wp$ and $\hbar$ usually results in longer convergence times and lower total traffic volumes.

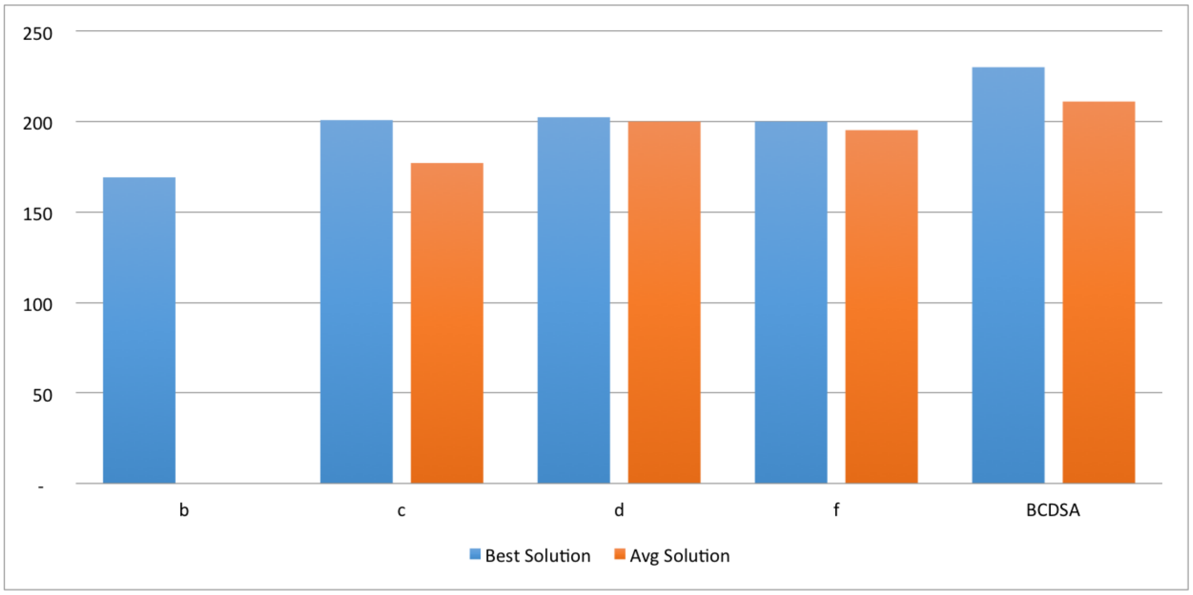


Figure 5.7: A comparison of average and best congestion reduction in various scenarios of GA and BCDSA.

In conclusion we believe that BCDSA offers a better overall algorithm considering runtimes, solutions results and success rate, especially in deploying for real networks and keeping our optimization running in near-real time manner to cope with network traffic changes.

Chapter 6

Conclusion and Future Work

6.1 Conclusion and Summary

In this thesis, we introduced a pair of modeling approaches to predict the capacity limits of UMTS cellular networks . Our approaches utilized machine learning regression and also multi-layer perceptron deep learning. The models were trained using existing measurements collected over one month in downtown Los Angeles. We were able to predict breakpoints of cellular towers with good accuracies within the cluster of our study as the quantities of interest. This allowed us to understand the maximum loading limits of each cell within the cellular network of our study. Relying on our learning results, we formulated an optimization problem aiming at maximizing the capacity of the cluster of cellular towers. Our problem formulation used the CPICH power and CIO of individual cells as decision variables in order to efficiently redistribute traffic from congested cell towers to non-congested cell towers under the constraints of maximum load and minimum quality. We, then, offered three simulated annealing variants for solving the problem. Our third alternative was a novel solution approach referred to as block coordinated descent simulated annealing offering dramatically improved algorithmic success rate, excellent utility, runtime, and confidence interval characteristics compared to other solution alternatives.

In addition, we introduced a modeling approach to predict the PRB utilization of LTE cellular networks based on counter measurements collected over one month in downtown Los Angeles. Our approach utilized a multi-layer perceptron deep learning model. After training our model using existing measurements, we were able to predict congestion thresholds of cellular towers with high accuracies. This allowed us to understand the loading limits of each cell tower within the cellular network of our study leading to reaching a congestion threshold of 80% associated with PRB utilization. Relying on our learning results, we formulated an optimization problem aiming at minimizing the congestion of a cluster of LTE cellular towers. Our problem formulation used cell power and handover margin of individual cells as decision variables in order to efficiently redistribute traffic from congested cell towers to neighboring non-congested cell towers under the constraints of connected UE loads and minimum qualities. We then offered two optimization algorithms, namely, Block Coordinated Descent Simulated Annealing (BCDSA) and Genetic Algorithm (GA) to solve this problem. We compared the results from BCDSA and GA demonstrating that GA offered higher success rates in finding optimal solutions while BCDSA had an order of magnitude lower runtimes with reasonable success rates.

6.2 Future Work

In this work we focused on capacity in terms of connected users in UMTS and PRB utilization in LTE and we based our optimization and capacity offload on these two parameters. Depending on the maturity of network, operators may need to also evaluate capacity form a signaling stand point like Physical Downlink Control Channel (PDCCH) utilization. This channel may reach high utilization and even congestion possibly before PRB Utilization depending on network usage patterns. In this case we may need to use this as another learning output as well as optimization goal.

Also we assumed in this work that each serving cell has one value for Cell Individual Offset (CIO). Some equipment manufacturers allow one CIO value for each serving - neighbor relation. Hence our optimization and solution space could grow exponentially, from 30 values to 900 values, which in turn could make optimization using GA a much complicated problem compared to BCDSA.

Bibliography

[1] M. Apostolopoulou, D. Sotiropoulos, I. Livieris, and P. Pintelas. A memoryless bfgs neural network training algorithm. In *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, pages 216–221, June 2009.

[2] M. S. Ayman ElNashar, Mohamed A. El-Saidny. *DESIGN, DEPLOYMENT AND PERFORMANCE OF 4G-LTE NETWORKS*. John Wiley & Sons, 2014.

[3] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, January 2013.

[4] X. bin Li and X.-L. Yu. Influence of sample size on prediction of animal phenotype value using back-propagation artificial neural network with variable hidden neurons. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4, Dec 2009.

[5] A. G. K. P. M. Christophe Chevallier, Christopher Brunner and K. R. Baker. *WCDMA Deployment Handbook Planning and Optimization Aspects*. John Wiley & Sons, 2006.

[6] Y. Cui, K. Xu, J. Wu, Z. Yu, and Y. Zhao. Multi-constrained routing based on simulated annealing. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 3, pages 1718–1722 vol.3, May 2003.

[7] S. Fahlman. *An Empirical Study of Learning Speed in Back- Propagation Networks*. Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.

[8] M. Garcia-Lozano, S. Ruiz, and J. Olmos. Umts optimum cell load balancing for inhomogeneous traffic patterns. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 2, pages 909–913 Vol. 2, Sept 2004.

[9] V. Garg and R. Bansal. Comparison of neural network back propagation algorithms for early detection of sleep disorders. In *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*, pages 71–75, March 2015.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] B. Hajek. Cooling shcedules for optimal annealing. *Operations Research*, May 1988.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation, 2/E.* Precntice Hall, 1988.

[13] A. F. C. Hurtado. Umts capacity simulation study, master of science in telematics thesis, October 2005.

[14] Y. B. Ian Goodfellow and A. Courville. *Deep Learning.* 2016. Book in preparation for MIT Press.

[15] T. N. Jaana Laiho, Achim Wacker. *Radio Network Planning and Optimisation for UMTS.* John Wiley & Sons, 2006.

[16] C. Johnson. *RADIO ACCESS NETWORKS FOR UMTS PRINCIPLES AND PRACTICE.* John Wiley & Sons, 2008.

[17] Y. Kim and M. Lee. Scheduling multi-channel and multi-timeslot in time constrained wireless sensor networks via simulated annealing and particle swarm optimization. *Communications Magazine, IEEE*, 52(1):122–129, January 2014.

[18] P. Kuang, W.-N. Cao, and Q. Wu. Preview on structures and algorithms of deep learning. In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on*, pages 176–179, Dec 2014.

[19] Y. T. Lee and A. Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 147–156, Oct 2013.

[20] S. A. P. M. Minsky. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, Expanded Edition, 1988.

[21] S. McLoone, V. Asirvadam, and G. Irwin. A memory optimal bfgs neural network training algorithm. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 1, pages 513–518, 2002.

[22] S. McLoone, M. Brown, G. Irwin, and G. Lightbody. A hybrid linear/nonlinear training algorithm for feedforward neural networks. *Neural Networks, IEEE Transactions on*, 9(4):669–684, Jul 1998.

[23] A. H. A. Meciej J Nawrocki, Mischa Dohler. *Understanding UMTS Radio Network Modeling, Planning and Automated Optimisation.* John Wiley & Sons, 2006.

[24] S. Mohammadi, C. Shang, Z. Ouhib, T. Leventouri, and G. Kalantzis. A computational study on different penalty approaches for constrained optimization in radiation therapy treatment planning with a simulated annealing algorithm. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, pages 1–6, June 2015.

[25] I. Necoara. A random coordinate descent method for large-scale resource allocation problems. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 4474–4479, Dec 2012.

[26] J. Ortega and W. Rheinboldt. *Iterative Solutions of Nonlinear Equations in Several Variables*. Academic Press, New York, NY, 1970.

[27] R. Qi and S. Zhou. Simulated annealing partitioning: An algorithm for optimizing grouping in cancer data. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 281–286, Dec 2013.

[28] H. Singh, A. Isaacs, T. Ray, and W. Smith. A simulated annealing algorithm for constrained multi-objective optimization. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1655–1662, June 2008.

[29] I. Siomina, P. Varbrand, and D. Yuan. Automated optimization of service coverage and base station antenna configuration in umts networks. *Wireless Communications, IEEE*, 13(6):16–25, Dec 2006.

[30] I. Siomina and S. Wanstedt. The impact of qos support on the end user satisfaction in lte networks with mixed traffic. In *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, Sept 2008.

[31] S.N.Sivanandam and . S.N.Deepa. *Introduction to Genetic Algorithms*. Springer, 2008.

[32] L. Song and J. Shen. *Evolved Cellular Network Planning and Optimization for UMTS and LTE*. Taylor and Francis Group, LLC, 2011.

[33] M. B. Stefania Sesia, Issam Toufik. *LTE - The UMTS Long Term Evolution*. John Wiley & Sons, 2011.

[34] E.-G. Talbi and T. Muntean. Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem. In *System Sciences, 1993, Proceeding of the Twenty-Sixth Hawaii International Conference on*, volume ii, pages 565–573 vol.2, Jan 1993.

[35] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, June 2001.

[36] U. Turke and M. Koonert. Advanced site configuration techniques for automatic umts radio network design. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 3, pages 1960–1964 Vol. 3, May 2005.

[37] B. Wah, Y. Chen, and A. Wan. Constrained global optimization by constraint partitioning and simulated annealing. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 265–274, Nov 2006.

[38] B. Wah and T. Wang. Constrained simulated annealing with applications in nonlinear continuous constrained global optimization. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 381–388, 1999.

[39] M. Wu and L. Chen. Image recognition based on deep learning. In *Chinese Automation Congress (CAC), 2015*, pages 542–546, Nov 2015.

[40] X. Yao, X. Liu, G. Hu, and F. Qian. Link delay inference based on simulated annealing constrained optimization method. In *Wireless, Mobile and Multimedia Networks, 2006 IET International Conference on*, pages 1–4, Nov 2006.

[41] H. Yousefi'zadeh, A. Habibi, X. Li, H. Jafarkhani, and C. Bauer. A statistical study of loss-delay tradeoff for red queues. *Communications, IEEE Transactions on*, 60(7):1966–1974, July 2012.

[42] H. Yousefi'zadeh and E. Jonckheere. Dynamic neural-based buffer management for queuing systems with self-similar characteristics. *Neural Networks, IEEE Transactions on*, 16(5):1163–1173, Sept 2005.

[43] R. ZeinEldin. An improved simulated annealing approach for solving the constrained optimization problems. In *Informatics and Systems (INFOS), 2012 8th International Conference on*, pages BIO–27–BIO–31, May 2012.