Congestion Minimization of LTE Networks: A Deep Learning Approach

Amr Albanna and Homayoun Yousefi'zadeh, *Senior Member, IEEE*

***Abstract*— Reducing the number of 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 LTE networks. In this paper, we utilize a supervised Deep Learning (DL) technique to predict the LTE and LTE-A loading of connected users and then dynamically predict the congestion threshold of each cellular tower under offered load. We then use the predicted congestion thresholds together with quality constraints to fine-tune cellular network operating parameters leading to minimizing overall network congestion. We propose two sets of optimization algorithms to solve our formulated congestion optimization problem. Those are, namely, a variant of Simulated Annealing (SA) algorithm to which we refer as Block Coordinated Descent Simulated Annealing (BCDSA) and Genetic Algorithm (GA). We first compare the performance of integrated DL-BCDSA and DL-GA algorithms and then show that our integrated DL-BCDSA can outperform existing state-of-the-art commercial self organizing tool already deployed in actual cellular networks.**

***Index Terms*— LTE network congestion, LTE-A network congestion, deep learning, block coordinated descent, simulated annealing, genetic algorithm.**

I. Introduction

DUE to exponential growth of LTE and LTE-A traffic, mobile operators are spending a significant percentage of their operating budget on augmenting their cellular infrastructure. Different 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 mitigation of congestion. In the context of 3GPP standards including Release 9 [1], joint implementation of Mobility Load Balancing (MLB) and Mobility Robustness Optimization (MRO) have been proposed. MLB is defined as a function in which congested cells can offload their excess users to other cells with spare resources. MLB includes load reporting between eNodeBs to exchange information about load level and available capacity. MRO is a solution for automatic detection and correction of errors in mobility configuration. In Release 9, the focus is on errors causing Radio Link Failures (RLFs) due to late or early handover, or handover to an incorrect cell.

This paper focuses on systematic minimization of the congestion of LTE and LTE-A networks. The subject material of this paper is considered to be a more powerful alternative to MLB in which a dataset associated with multiple eNodeBs within a cluster of interest, as opposed to limited neighboring eNodeB sets, are processed together in order to offer load balancing results that are optimized for the overall cluster of interest. Just like standard MLB, the use of MRO is complimentary to the proposed scheme of this paper.

The challenge associated with congestion minimization in LTE/LTE-A networks is better understood by explaining how resources are allocated to users. Under LTE standard and LTE-A extensions, each cellular tower has a fixed number of Physical Resource Blocks (PRBs) defined in time and frequency. 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 request. The throughput associated with each PRB mainly depends on the maximum allowable modulation order ranging from QPSK to 16QAM, 64QAM, and up to 256QAM. The maximum allowable modulation order depends on a number of factors related to channel conditions and correlation experienced by a given user for that PRB. Among such factors, Signal to Interference and Noise Ratio (SINR), the number of multiplexing channels also known as rank, and transmission mode can be mentioned. In one operating scenario, a user requesting video streaming while experiencing excellent channel conditions 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. In another operating scenario, another user experiencing sub-par channel conditions will only be able to utilize QPSK modulation scheme hence requiring a larger number of PRBs in order to satisfy a similar video streaming quality [2]. Noting that SINR versus CQI mapping depends on modem manufacturer and transmission modes [3]–[5], Table I shows the specifications of the indices of LTE/LTE-A Channel Quality Indicator (CQI).

Although Multi Input Multi Output (MIMO) and Inter-Cell Interference Coordination (ICIC) techniques such as [2] are utilized to mitigate sub-par RF conditions, cell breathing techniques such as [6], [7] are used to alleviate cell loads, and load balancing techniques [8], [9] are utilized to balance

Manuscript received August 26, 2019; revised November 2, 2019 and December 8, 2019; accepted December 12, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. F. Chiasserini. *(Corresponding author: Homayoun Yousefi'zadeh.)*

The authors are with the Department of EECS, University of California at Irvine, Irvine, CA 92697 USA, and also with the Center for Pervasive Communications and Computing (CPCC), University of California at Irvine, Irvine, CA 92697 USA (e-mail: aalbanna@uci.edu; hyousefi@uci.edu).

Digital Object Identifier 10.1109/TNET.2019.2960266

TABLE I
SPECIFICATIONS OF LTE/LTE-A CHANNEL QUALITY INDICATOR (CQI) INDICES [3]–[5]

CQI Index	Modulation Type	Code Rate (x 1024)	Spectral Efficiency (bps/Hz)	DNLNK PRB TPT (kbps)	DNLNK SINR (idB)
1	QPSK	78	0.1523	19.1898	-7.28
2	QPSK	193	0.377	47.502	-2.04
3	QPSK	308	0.6016	75.8016	0.66
4	QPSK	449	0.877	110.502	2.84
5	QPSK	602	1.1758	148.1508	4.73
6	16QAM	490	1.9141	241.1766	8.78
7	16QAM	616	2.4063	303.1938	11.49
8	64QAM	466	2.7305	344.043	13.27
9	64QAM	567	3.3223	418.6098	16.52
10	64QAM	772	4.5234	569.9484	23.12
11	64QAM	873	5.1152	644.5152	26.37
12	64QAM	948	5.5547	699.8922	28.79
13	256QAM	772	6.03	759.0465	29.31
14	256QAM	948	7.39	933.0435	32.39
15	256QAM	975	7.61	959.0543	34.81

cell loads, the physical limitation on the number of available PRBs still presents a challenge that 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 per slot. 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, 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 mobile operators attempt at keeping per cell PRB utilization under a congestion threshold set to 80% industry wide. Cells exceeding their 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 opposed 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 its serving area and redistribute traffic as needed. We note that these parameters can be changed dynamically in the field in response to shifts in traffic distributions thereby offloading traffic from congested cells to neighboring cells without congesting the neighboring cells and without degrading the quality of the shifted UEs below a threshold of acceptance.

This paper focuses on systematically finding optimal power and handover settings of a cluster of LTE cell towers minimizing overall cluster congestion in the presence of user quality constraints. First, we propose a deep learning scheme that identifies the breakpoint of each cell associated with the average number of connected users at which PRB utilization will reach a congestion threshold of 80%. Next, we formulate an optimization problem aiming at minimizing the congestion of a cluster of cell towers using per cell power and handover parameters as decision variables. We solve the problem using two optimization techniques, namely, BCDSA and GA.

The main contributions of this paper are four fold. 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 LTE cellular towers. Next, we formulate and efficiently solve an optimization problem aiming at minimizing the congestion of a cluster of cells subject to UE quality constraints. Third, we compare the results of our proposed BCDSA and GA solutions in order to evaluate performance, utility, runtime, and success rate of each technique. Finally, we show how the use of our integrated DL-BCDSA approach can outperform a state-of-the-art commercial Self Organizing Network (SON) tool [10] in an actual cellular network.

Table II provides a listing of notations used in this paper.

The rest of the paper is organized as follows. In Section II, we introduce our learning approach to predict congestion threshold of LTE cellular towers. In Section III, we describe the formulation of our optimization problem aiming at traffic offloading of congested cells along with the solution to the problem using both BCDSA and GA algorithms. Experimental results are reported in Section IV. Finally, Section V contains the conclusion of our work.

II. LEARNING-BASED BREAKPOINT 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. 1 shows sample drawings of actual LTE PRB utilization as a function of average connected users collected from a major mobile operator data at a snapshot of time 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 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. Therefore, we can claim that Sector 3 is able to carry a larger number of users than Sector 4 before it becomes congested. The question we are trying to answer next is how to predict the value of Λ_i representing the average number of connected UEs corresponding with the PRB utilization of 80%, for each cell i based on its unique characteristics.

A. Deep Learning

We use a multi-layer perceptron DL approach [11], [12] with supervised learning [13], [14] to accurately predict the congestion threshold of individual LTE cells in a cluster of

TABLE II
THE TABLE OF NOTATIONS

I	Set of all LTE cells within cluster
i	LTE cell index within set I
N	No. of cells within set I
$\wp_i$	Power of cell i
$\Delta\wp_i$	Change in the power of cell i
$\hbar_i$	Handover margin of cell i
$\Delta\hbar_i$	Change in the handover margin of cell i
x_i	Ordered pair setting $(\wp_i, \hbar_i)$ for cell i
$\underline{x}$	Vector of elements x_i where $i \in \{1, \cdots, N\}$
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
$\Delta\gamma_{i,j}$	Quality variation of a UE shifted from cell i to cell j
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
ξ	Freeze count measure of simulated annealing algorithm
ξ_{max}	Maximum freeze count measure of BCDSA and GA algorithms
m	No. of search attempts in BCDSA algorithm
T	Temperature of BCDSA algorithm
T_i	Initial temperature of BCDSA algorithm
T_f	Final temperature of BCDSA algorithm
a	Cooling factor of BCDSA algorithm
ρ	Multiplier of N controlling the number of iterations at each temperature point of BCDSA algorithm
σ	Number of times the temperature will be cooled down in BCDSA algorithm
B	Boltzman constant
κ	Initial population count of GA algorithm
χ	Fixed integer multiplier for GA algorithm depending on the number of decision variables and their variation ranges
R	Random number derived from uniform distribution U $[0, 1]$
ϵ	Small number used in the stoppage criterion of GA algorithm
$\mathcal{U}$	Unit step function
τ	Experimental repetition period (minutes)
Σ	No. of experimental iterations
W	Experimental data collection period (weeks)
D	No. of experimental datasets collected
H	Hata propagation model constant
r_ζ	Horizontal distance of a UE from cell ζ
Φ	Optimization penalty coefficient

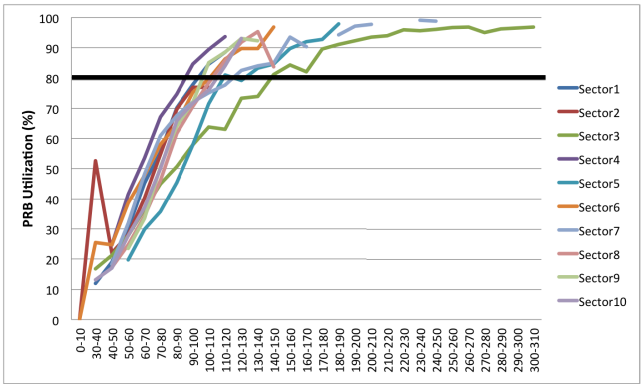


Fig. 1. Actual sample drawings of LTE PRB utilization as a function of average connected UEs at a snapshot of time.

cell towers. That is to identifying the value of Λ_i for each cell i, i.e., the average number of connected UEs crossing the PRB congestion threshold of 80%.

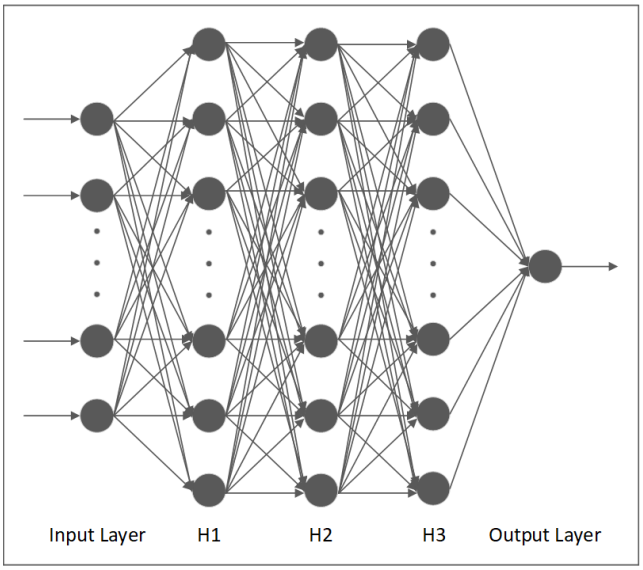


Fig. 2. Multi layer perceptron deep learning structure used for learning congestion thresholds of individual LTE cellular towers.

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 26 processing elements to accept 26 LTE input counters. We explain the choices of input counters in the next subsection. In order to strike a 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. 2 illustrates the multi-layer perceptron DL structure.

In each iteration of learning, we propagate 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 multi-layer perceptron DL.

With regards to the choice of learning, we tried a number of algorithms as discussed in [15] namely, a) GDM, a gradient descent with momentum back-propagation, b) GDX, a gradient descent with momentum and adaptive learning rate backpropagation, c) Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton backpropagation [12], [16], and d) Levenberg-Marquardt (LM) [17] algorithm. Similar to quasi-Newton methods, the LM algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. Having explored the results of various back propagation techniques, we make use of LM back propagation learning scheme producing the best results.

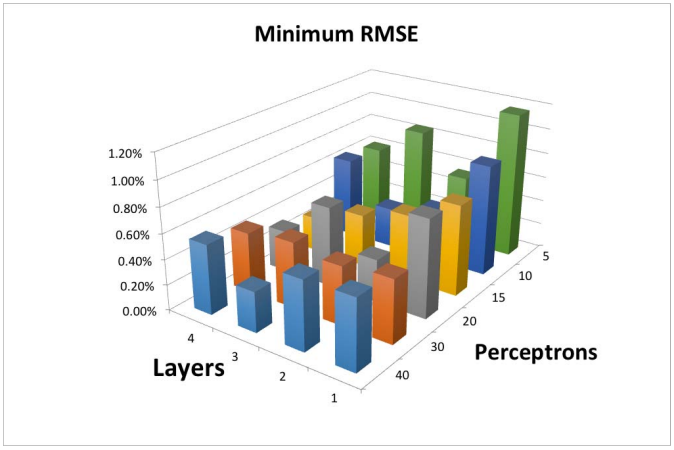


Fig. 3. An illustration of minimum RMSE of DL prediction as a function of number of layers and perceptrons.

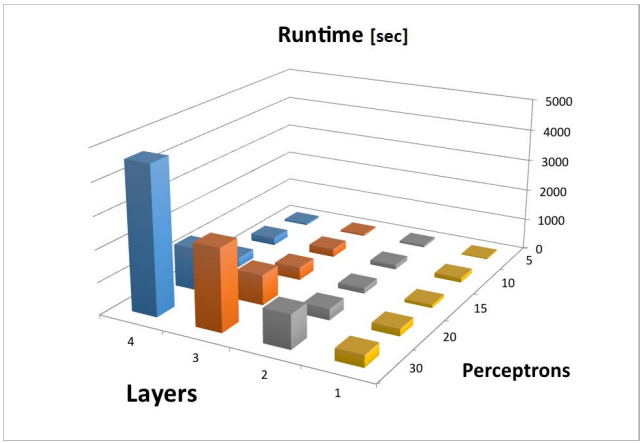


Fig. 4. Measurements of average runtime as a function of number of layers and perceptrons.

We note that the accuracy of learning is traded against complexity and runtime. It depends, among other factors, on the complexity of perceptron structure, training, verification, and testing algorithms. We use a mix of 70% training, 15% verification, and 15% independent testing. Fig. 3 provides illustrations of RMSE variations as a function of the number of processing elements and layers. We utilize 20 multi-layer perceptron DL 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 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 consisting of a Virtual Machine to which an i7 processor with 2 Cores, 4 Threads, 8GB of RAM, and 256GB of SSD were allocated.

Fig. 4 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 use a DL 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.

B. Input Counters to Learning

One of the critical factors in generating accurately predicted results is the choice of input parameters, i.e., LTE counters. 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 conducted experiments in multiple phases with various counters and KPIs from the real network of a major US carrier collected in 15 minute intervals over one week. In the first phase, we used average active UEs, average connected UEs, and peak connected UEs of a single cell to predict PRB utilization of that cell subsequently introducing an RMSE of 34%. Next, we used data collected from all cells to predict PRB utilization of the collection of cells. In subsequent phases, we added call attempts, average and peak number of ERABs, total Voice over LTE (VoLTE), traffic measures of VoLTE in Erlangs and data volume in Megabytes, QCI as presented in Table III [18], modulation type, average cell throughput, peak cell throughput, average UE throughput, and average cell spectral efficiency. Conducting learning with the collection of KPIs above, resulted in reducing the RMSE to less than 0.5%.

TABLE III
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 signalling
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).

III. CONGESTION MINIMIZATION

In the previous section, we predicted the congestion threshold Λ_i representing 80% PRB utilization for each cell i within a cluster of cell towers. In this section, we utilize those 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.

We note that shifting LTE traffic can be done in two ways. First, reducing LTE cell power $\wp_i$ of a cell i results in shrinking the footprint of the cell hence shifting UEs on its border to the neighboring cells. Second, artificially increasing 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 each individual cell tower.

A. Problem Description

We are trying to minimize congestion measured by the number of total 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 in determining the cell footprint. Decreasing cell power reduces the footprint of a cell while increasing it has the opposite effect. Additionally, the handover margin $\hbar_i$ determines a cell boundary in comparison to its 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) increasing 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 the edges of cell i 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. 5 illustrates the received signal strength at a UE as it moves from the vicinity of cell tower A to that of tower B. The x-axis is the distance of the UE from cell tower A to which the UE is initially connected, while the y-axis is the UE's received power. In Fig. 5a, the blue line labeled A shows that the UE's received signal strength from cell A decreases as the distance increases, i.e., as the UE travels away from cell A. The green line labeled B shows the UE's received signal strength from cell B increases as the distance increases, i.e., as the UE travels toward cell B. The intersection point of blue and green lines represents the initial boundary distance point at which the UE is handed over from cell A to cell B. The red line labeled A' shows reducing the value of cell power $\wp_A$ by a sample value of 3dB shrinks the footprint of cell A from r_A to $r_{A'}$. 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.

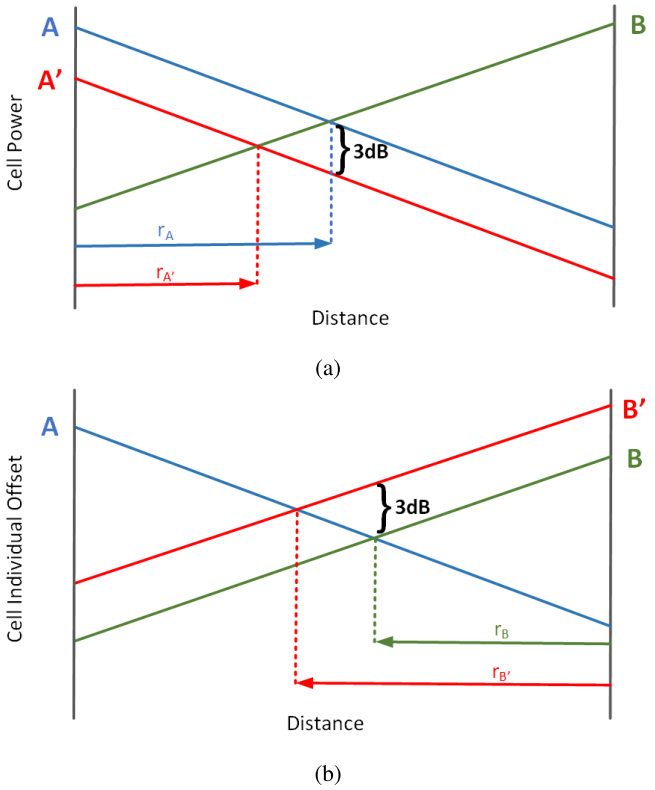


Fig. 5. The impact of changing a) cell power $\wp_A$ of cell A and b) handover margin $\hbar_B$ of cell B on reducing cell A footprint.

Similarly, Fig. 5b 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.

B. Problem Formulation

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

$$\min_{\forall\, \wp_i, \hbar_i} \ \Lambda_\Upsilon = \sum\nolimits_{i \in I} \Big[\big(\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \big) - \Lambda_i \Big]^+ \quad (1)$$

$$\text{S.T.} \ \sum\nolimits_{i \in I} \Big[\lambda_i - \lambda_i^{\wp} - \sum_{\substack{j \in I \\ i \neq j}} \lambda_{i,j}^{\hbar} \Big] \ = \ \Lambda_L \quad (2)$$

$$q_i \geq Q, \quad \forall i \in I \quad (3)$$

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 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. (1) is the summation of expressions taken over all cells. For each cell, the expression consists of the difference of three terms and predicted congestion threshold of that cell. 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 cannot be a trivial utility value of $\Lambda_\Upsilon = \sum_i \Lambda_i$ due to discrete values of decision variables and constraint (3).

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 [2], [19] and assuming traffic is homogeneously distributed in the serving area [20].

$$\lambda_i^\wp = \lambda_i \left[1 - \left(10^{\frac{-\Delta\wp_i}{H}}\right)^2\right] \tag{4}$$

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.

$$\lambda_{i,j}^\hbar = \eta_{i,j}\ \lambda_i \left[1 - \left(10^{\frac{-\Delta\hbar_j}{H}}\right)^2\right] \tag{5}$$

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 IV.

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.

$$q_i = \min_j \ \ q_{i,j} \tag{6}$$

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 $q_{i,j}$ that presents the quality of cell i guaranteed not to be 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], [21]. When reducing the serving cell i power $\wp_i$, say 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.

$$\Delta\gamma_{i,j} = \Delta\wp_i + \Delta\hbar_j \tag{7}$$

Consequently, the quality impact is captured as shown below.

$$q_{i,j} = \gamma_{i,j} - \Delta\gamma_{i,j} \tag{8}$$

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], [22]. Therefore, Q is set to -3dB.

C. Solutions Description

In this subsection, we propose two optimization algorithms to solve the problem of the previous section. First and in order to enforce quality constraints, we add a set of penalty terms $\mathcal{U}_i$ one per cell i and δ to the objective function [23]–[25]. Then, we use BCDSA and GA to solve the problem. The penalty-augmented objective function is defined below.

$$\tilde{\Lambda}_\Upsilon = \sum\nolimits_{i\in I} \Big\{ \big[\big(\lambda_i - \lambda_i^\wp - \sum_{\substack{j\in I \\ i\neq j}} \lambda_{i,j}^\hbar\big) - \Lambda_i \big]^+ + \Phi * \mathcal{U}_i \Big\} + \Phi * \delta \tag{9}$$

In Eq. (9), Φ is set to 10^6,

$$\delta = \begin{cases} 1, & \text{if } \big(\sum_{i\in I} \big[\lambda_i - \lambda_i^\wp - \sum\limits_{\substack{j\in I \\ i\neq j}} \lambda_{i,j}^\hbar\big] \ \neq\ \Lambda_L\ \big) \\ 0, & \text{Otherwise} \end{cases} \tag{10}$$

and

$$\mathcal{U}_i = \begin{cases} 1, & \text{if } (q_i < Q) \\ 0, & \text{Otherwise} \end{cases} \tag{11}$$

It has to be noted that δ is a weighted penalty factor applying a constant large hard penalty Φ, set to 10^6, for violating the load preservation constraint in Eq. (2). Numerically, the load preservation constraint in Eq. (2) 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 Φ, set to 10^6, for violating the quality constraint in Eq. (3) of cell i.

1) Block Coordinated Descent Simulated Annealing: Inspired by the block coordinated descent optimization techniques [26]–[28], our first proposed algorithm modifies the standard SA algorithm in an attempt to address the tradeoff between accuracy and complexity. Referred to as BCDSA algorithm, this algorithmic variation applies the SA algorithm to a partitioned set of decision variables. That is to say, it optimizes one set of decision variables while keeping the other set fixed, then optimizes the other set while keeping the first set fixed, and alternates between the two sets. Alternating between two sets of decision variables occurs if the cost function does not change after few iterations of one set measured by a freeze factor ξ. 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 of $\Delta\wp_i$ and $\Delta\hbar_i$ values. The BCDSA algorithm is illustrated in Algorithm 1.

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 III-B. To support our claim, we note that [29] 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 quasiconvex and hemivariate conditions [30], [31]. 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 $\sigma = (\log T_f - \log T_i)/\log a$ where T_i, T_f, and a are the initial temperature, final temperature, and cooling factor. 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.

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 [32]. 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. 6 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.

The GA algorithm as applied to solve our problem is illustrated in Algorithm 2. 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. 9.

Algorithm 1 BCDSA(*Topology,Thresholds*)

Form penalty-augmented objective function $\tilde{\Lambda}_\Upsilon(\underline{x})$
  where $x = (x_1, x_2 ....., x_N), x_i = (\Delta\wp_i, \Delta\hbar_i)$
Set initial values $\underline{x}[0]$ and $T = T_i$
Set $K = \rho N$ and final value T_f
Set cooling factor a in interval $[0, 1]$
Define max freeze factor ξ_{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* $(T > T_f)$ } */

Starting from an initial population, GA 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. 7, 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. 8, a random chromosome is chosen and the value of a number of its genes

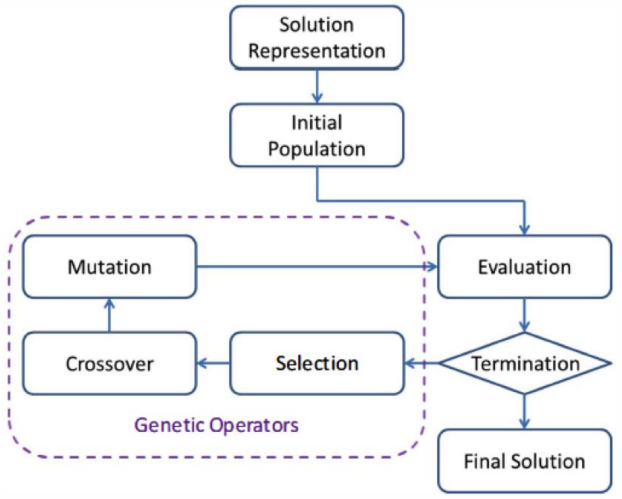


Fig. 6. A flowchart of GA operation.

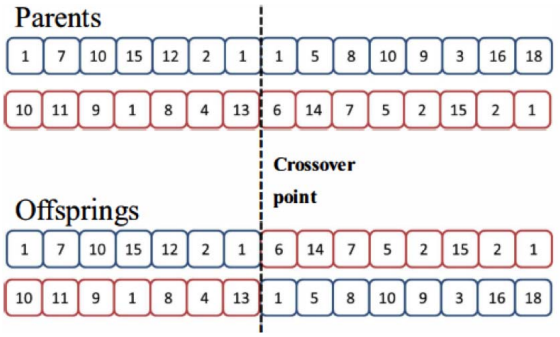


Fig. 7. An illustration of Crossover operator in GA.

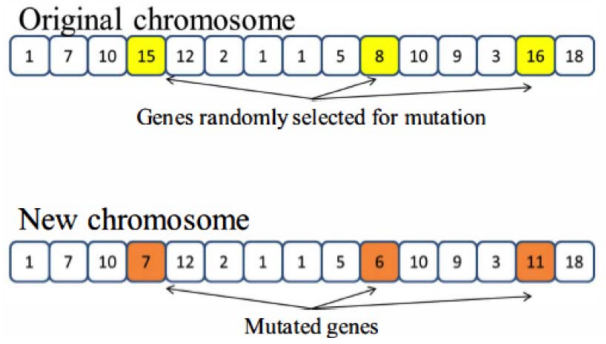


Fig. 8. An illustration of the Mutation operator of genetic algorithm.

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 2 GA*(Topology,Thresholds)*

Set real number multiplier χ
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 ξ_{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 ψ from $\mathcal{C}$
      Randomly change the value of ψ
      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 $P[g]$

IV. Performance Evaluation

We open this section by making the following statements about data gathering process.

- The data needed for performing *real-time operation* is extracted from eNodeBs in the form of real-time reports. Each eNodeB report contains multiple cell towers and a

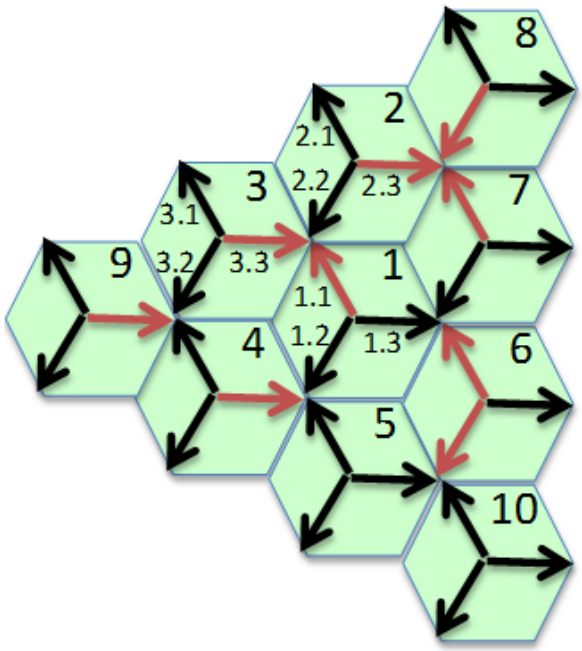


Fig. 9. A cluster of cellular towers in greater Los Angeles area comprised of ten sites with each site having three cellular towers.

collection of 80 per cell KPIs. Reports are refreshed and can be pulled within intervals of 15 minutes.

- The data content included in eNodeB reports is stored for later processing and analysis in a database tool deployed by the mobile operator. The typical delay in populating the database data is 60 minutes.

A typical operating scenario relies on the data gathering process above. In such scenario, the learning algorithm is run followed by the optimization algorithm in a repeating cycle. The frequency of running the learning algorithm followed by the optimization algorithm is set to once every τ minutes. The value of τ is typically in the range of 15 to 60 minutes depending on the operating requirements of a mobile operator. The first iteration starts by importing the first D measurement datasets covering a period of the first W weeks. Noting datasets are collected once every τ minutes, the value of D is calculated as $D = (W \times 7 \times 24 \times 60)/\tau$. The learning algorithm is then run using these datasets in order to predict congestion thresholds. Next, the predicted thresholds are fed into the optimization algorithm of choice. The next iteration repeats the steps above after deleting the oldest dataset and importing dataset $D + 1$. Experimentally, we set the number of repeating iterations to Σ.

A. A Comparison of Integrated DL-BCDSA and DL-GA

This subsection compares the performance of integrated DL-BCDSA and DL-GA algorithms using a first cluster located in greater Los Angeles area and depicted in Fig. 9. The cluster has ten sites with each site having three sectors or cells and each cell presented with an arrow. Red arrows represent congested cells while black arrows represent non-congested cells. With the exception of the 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. 9 cell 1.1 has front facing neighbors 2.2 and 3.3, and co-site neighbors 1.2 and 1.3. As shown, not all cells are congested and further congested cells have at least a non-congested neighbor.

1) Experimental Settings: The experimental data is extracted from the database tool above along with settings $W = 2$, $\tau = 15$, and $D = 1344$ in a typical operating scenario. We collected a total of $2D = 2688$ datasets associated with the traffic profile of the first cluster of our study over a continuous period of $2W = 4$ weeks between October 2018 and November 2018.

The cellular network serves an urban environment with a propagation loss coefficient of $H = -40$dB/decade. Furthermore, handover percentages from a cell to its facing and co-site neighbors are averaged at 40% and 10%, respectively. Handover percentages are calculated as the ratio of the number of handovers to a particular neighbor over total handovers.

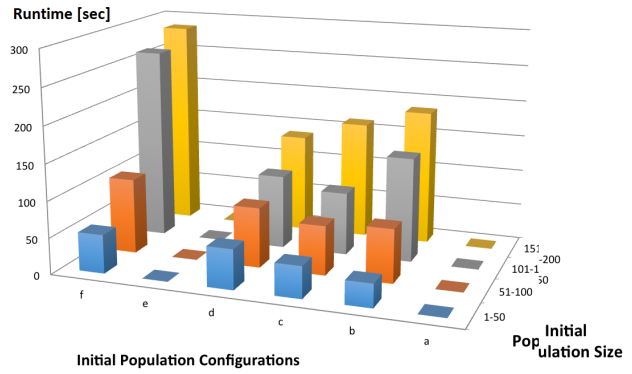
A reduction in $\wp_i$ or increase in $\hbar_j$ results in a similar reduction in SINR for border users based on the selected urban environment. Traffic is assumed to be homogeneously distributed in the serving area and hence traffic reduction is at a rate similar to reduction in serving area. The range of variations of both $\Delta\wp_i$ and $\Delta\hbar_i$ is $[0, 3]$dB. 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 for the the mobile operator environment and approximating the values presented in Table I.

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. Looking at the results, we observe that setting $a = 0.9$ along with a value of T_i in the same order as the value of change in the utility function best addresses utility-runtime tradeoff. As for ξ, results show that a choice of $\xi = 20$ best addresses the tradeoff between success rate and runtimes.

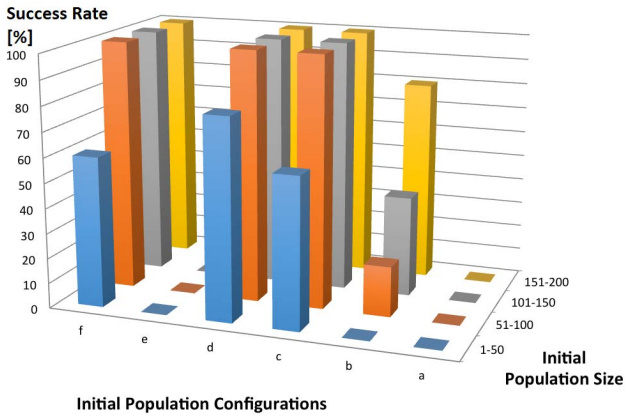
Last but not least, measurements show RMSE errors of DL prediction are in the range of $[0.5\%, 1\%]$ for a congestion threshold of 80% PRB utilization. In consideration of the error, a safety margin of 3% is applied to the predicted value of Λ_i when running optimization. This ensures that non-congested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

In conducting simulations leading to the results reported by Fig. 10, Fig. 11, and Fig. 12, we use a subset of collected data for comparing the performance of BCDSA and GA algorithms. The initial value of Λ_Υ, i.e., total average load of connected UEs in the cluster, is measured as 2836 users.

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



(a)



(b)

Fig. 10. A comparison of (a) average runtimes and (b) success rates for various GA configurations.

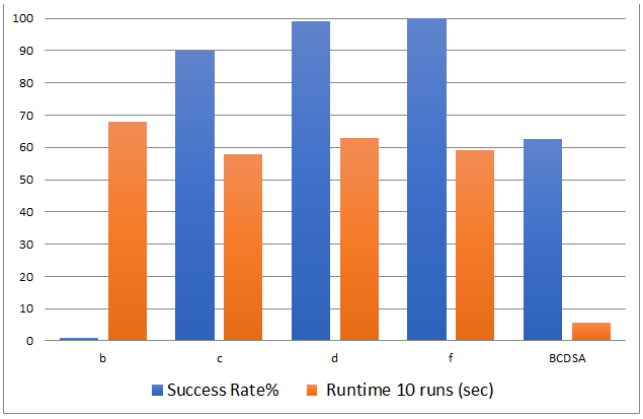


Fig. 11. A comparison of runtimes and success rates for various scenarios of GA and BCDSA calculated over 10 runs.

good solutions. A solution is considered good if its congestion value is within 1% of the best congestion solution obtained using that algorithm.

2) Comparison Results: We compare different aspects of performance 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

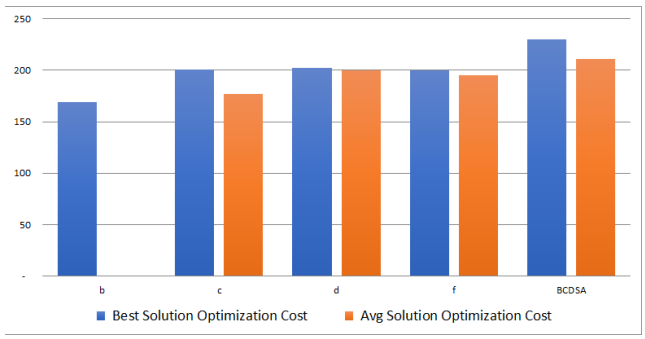


Fig. 12. A comparison of average and best congestion reduction in various scenarios of GA and BCDSA.

the percentage of good solutions, i.e., the number of solutions within 1% of the best solution.

First, we attempt at identifying the best selection of GA parameters in our experiments and then compare the results of BCDSA and GA both fed with the predictions of DL. 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 of [0,3].

A comparison of average runtimes and success rates for various GA configurations is presented in Fig. 10. 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. 11 compares the results 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 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 algorithm creates multiple solutions in the population and attempts at globally optimizing them using crossover and mutation operators. Hence a higher success rate is achieved at the cost of longer runtime since GA approaches the solution from various directions. On the other hand, BCDSA attempts at navigating its way to the global optimum using a partitioned SA approach. Hence, it offers a much lower processing time at the cost of having

TABLE IV
A COMPARISON OF EFFECTIVE CONVERGENCE TIMES OF BCDSA AND VARIANTS OF GA USING $\Sigma = 1344$ ITERATIONS

Algorithm	ECT (sec)
BCDSA	9
GA Alternative f	59
GA alternative d	64
GA Alternative c	64
GA Alternative b	64800

a higher chance of missing the global minimum. A good comparison analogy 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 cap of 60 seconds on the runtime, Fig. 12 compares average and best congestion cost reduction in various scenarios of GA and BCDSA from an initial congestion of 506 connected UEs as predicted by DL. It is noted that the remaining congestion is the difference between the initial value of 506 and what is shown in the graph. It is seen that the best and average congestion reduction solutions are 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 initial 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.

Next, we define the Effective Convergence Time (ECT) of an algorithm as the ratio of its average runtime over its success rate. Hence, ECT measures the robustness of convergence performance of an algorithm by factoring in the effect of its success rate into its runtime. Noting that smaller ECT values reflect better results, we measure and report the ECT of different algorithmic flavors utilizing the full set of $2D = 2688$ datasets described above. Table IV reports the results associated with running a total of $\Sigma = 1344$ iterations. As shown in the table, BCDSA algorithm has the best results. Alternatives f, d, and c of the GA algorithm report the next best results although the closest associated ECT is over 6 times higher than that of BCDSA. The results imply that the use of BCDSA is preferred over GA because it can be run more times to effectively compensate against its lower success rate.

B. A Real-Time Comparison of DL-BCDSA and a Baseline SON

This subsection compares the real-time performance of integrated DL-BCDSA and a commercial SON tool in a second cluster of cell towers located within greater Los Angeles area. The target cluster includes 84 cell towers with a mix of 75 macro cells and 9 small cells. Instead of using averages of 40% and 10%, the percentages of handover values to front facing and co-site neighbors are calculated from the actual reported KPIs.

1) Experimental Settings: Setting $W = 2$, $\tau = 60$, $D = 336$, and $\Sigma = 45$ in a typical operating scenario, we collected a total of 504 datasets associated with the traffic profile of the second cluster of our study over a continuous period of three weeks in May 2019. While the data needed for initial learning is extracted from the database tool using the data within the first two weeks, the data needed for optimization is extracted from real-time eNodeB reports during the third week.

The experiment was conducted during weekdays of those three weeks, starting Monday the 6-th of May of 2019 and ending on Friday the 24-th of May 2019. The selected dates represent a period during which network traffic load was consistently comparable and no seasonality of traffic change were reported. During weekdays, the experiments were started at 9:00am and concluded at 5:00pm. One set of changes were applied per hour utilizing a real-time dataset collected from the cell towers at the time of change. No changes were applied outside the hours of operations or during weekends.

We compare our results against a baseline formed by a state-of-the-art commercially-available self organizing network tool from Ericsson [10]. Referred to as SON, the tool offers an Automated Mobility Optimization (AMO) capability among its optimization features. AMO addresses issues with early handovers, late handovers, handovers to wrong cells, and handover oscillations. The baseline SON tool was deployed within the target cluster of our experiments with AMO feature turned on. We chose BCDSA as the choice of our optimization algorithm considering its convergence and speed characteristics as reported in the previous subsection. Accordingly, we label our results as DL-BCDSA. We use the best parameter settings of DL-BCDSA as reported in the Section IV-A.

We left the SON tool on during the first two weeks of our experimentation period, i.e., Monday the 6-th through Friday the 17-th. However, we turned it off when applying DL-BCDSA in the third week. At 9:00am of every weekday of the third week, we fed the DL with the latest $D = 336$ datasets associated with the most recent past 2 weeks. In subsequent iterations of the day, we added the next dataset to the mix after deleting the oldest dataset and repeated the same steps. A total of 9 real-time iterations were applied during the period of 9:00am to 5:00pm on every weekday of the third week.

2) Comparison Results: In the figures of this subsection, the labels SON-W1, SON-W2, and DL-BCDSA-W3 are associated with the results of SON tool in week 1, SON tool in week 2, and DL-BCDSA in week 3, respectively. All reported results are for the period of 9:00m to 5:00pm.

Fig. 13 shows a comparison of cluster excess users. Excess users are defined as connected UEs served by a cluster operating over 80% utilization. Note that the UEs served by a congested cell below the threshold of 80% utilization are not counted in the measure of excess users. As observed in the figure, cluster excess users reduced in the range of $[17\%, 27\%]$ when applying DL-BCDSA.

Fig. 14 shows a comparison of cluster PRB utilization. As observed in the figure, PRB utilization of the cluster reduced in the range of $[7\%, 10\%]$ when applying DL-BCDSA.

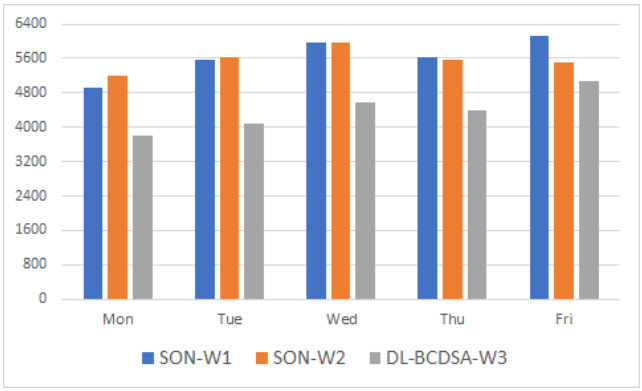


Fig. 13. A comparison of excess users between a commercial SON tool and DL-BCDSA over a period of 3 weeks.

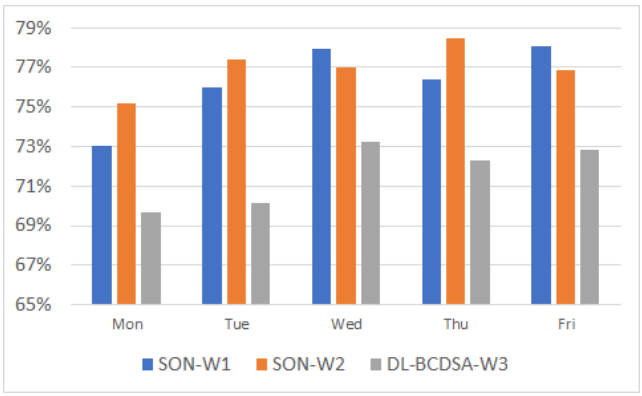


Fig. 14. A comparison of PRB utilization between a commercial SON tool and DL-BCDSA over a period of 3 weeks.

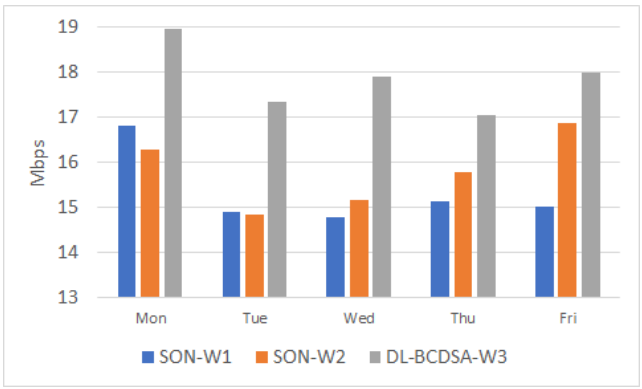


Fig. 15. A comparison of throughput between a commercial SON tool and DL-BCDSA over a period of 3 weeks.

Similarly, Fig. 15 illustrates a comparison of cluster throughput. As observed in the figure, cluster throughput increased in the range of $[14\%, 21\%]$ when applying DL-BCDSA.

Finally, Fig. 16 displays a comparison of cluster's average RSSI as a good indicator of signal strength. As observed in the figure, applying DL-BCDSA caused no significant change, i.e., less than 0.2% in average RSSI.

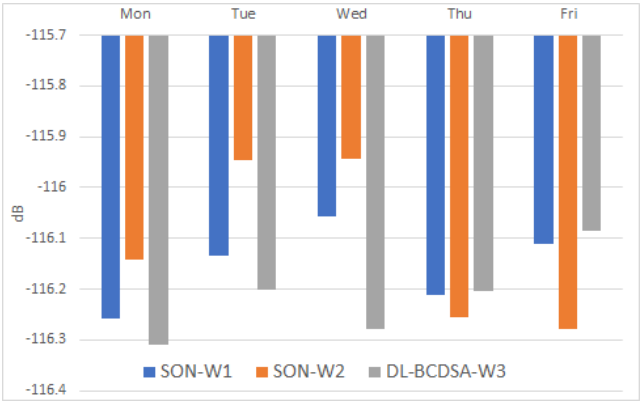


Fig. 16. A comparison of RSSI between a commercial SON tool and DL-BCDSA over a period of 3 weeks.

V. Conclusion

In this paper, we introduced an integrated Deep Learning (DL) and optimization approach resulting in minimizing the congestion of clusters of LTE/LTE-A cellular towers. The DL algorithm utilized measurements collected from a cluster of LTE cellular towers to accurately predict congestion thresholds of those cellular towers within a cluster of interest. Relying on DL results, our optimization algorithms, namely Block Coordinated Descent Simulated Annealing (BCDSA) and Genetic Algorithm (GA), then aimed at minimizing the congestion of the same cluster. We compared the results of 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. We also compared effective convergence times of different algorithms showing that BCDSA offers the best combined robustness and runtimes. Utilizing a large hybrid cluster of macro sites and small cells in greater Los Angeles area, we then reported the advantages of applying our integrated DL-BCDSA algorithm in real-time compared to a state-of-the-art commercial SON tool.

References

[1] *Self-Configuring and Self-Optimizing Network (SON) Use Cases and Solutions (Release 9)*, document TR 36.902, 3GPP, v9.3.1, Mar. 2011.

[2] A. ElNashar and M. A. El-Saidny, *Design, Deployment, and Performance of 4G-LTE Networks*. Hoboken, NJ, USA: Wiley, 2014.

[3] S. Rathi, N. Malik, N. Chahal, and S. Malik, "Throughput for TDD and FDD 4 G LTE systems," *Int. J. Innov. Technol. Exploring Eng.*, May 2014. [Online]. Available: https://docplayer.net/11419153-Throughput-for-tdd-and-fdd-4-g-lte-systems.html

[4] M. Kawser, N. I. B. Hamid, M. Hasan, A. Alam, and M. Rahman, "Downlink SNR to CQI mapping for different multiple antenna techniques in LTE," *Int. J. Inf. Electron. Eng.*, vol. 2, pp. 756–760, Sep. 2012, doi: 10.7763/IJIEE.2012.V2.201.

[5] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures*, document TS 36.213, 3GPP, Oct. 2010.

[6] A. Jalali, "On cell breathing in CDMA networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 1998, pp. 985–988.

[7] J. Perez-Romero, O. Sallent, R. Agusti, N. Garcia, L. Wang, and H. Aghvami, "Network-controlled cell-breathing for capacity improvement in heterogeneous CDMA/TDMA scenarios," in *Proc. IEEE WCNC*, Apr. 2006, pp. 36–41.

[8] A. Sang, X. Wang, M. Madihian, and R. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in *Proc. ACM MobiCom*, 2004.

[9] C. Franco and J. de Marca, "Load balancing in self-organized heterogeneous LTE networks: A statistical learning approach," in *Proc. IEEE Latin-Amer. Conf. Commun.*, Nov. 2015, pp. 1–5.
[10] Ericsson. *SON Optimization Manager*. Accessed: Dec. 20, 2019. [Online]. Available: https://www.ericsson.com/en/portfolio/digital-services/automated-network-operations/network-management/son-optimization-manager
[11] S. McLoone, M. D. Brown, G. Irwin, and G. Lightbody, "A hybrid linear/nonlinear training algorithm for feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 669–684, Jul. 1998.
[12] S. McLoone, V. Asirvadam, and G. Irwin, "A memory optimal BFGS neural network training algorithm," in *Proc. IJCNN*, 2002.
[13] S. Haykin, *Neural Networks: A Comprehensive Foundation, 2/E*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.
[14] P. Kuang, W.-N. Cao, and Q. Wu, "Preview on structures and algorithms of deep learning," in *Proc. Wavelet Active Media Technol. Inf. Process.*, Dec. 2014.
[15] V. Garg and R. Bansal, "Comparison of neural network back propagation algorithms for early detection of sleep disorders," in *Proc. ICACEA*, Mar. 2015.
[16] M. Apostolopoulou, D. Sotiropoulos, I. Livieris, and P. Pintelas, "A memoryless BFGS neural network training algorithm," in *Proc. IEEE Ind. Inform.*, Jun. 2009.
[17] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963.
[18] *Policy and Charging Control Architecture*, document TS 23.203, 3GPP, Dec. 2014.
[19] M. Nawrocki and M. Dohler, *Understanding UMTS Radio Network Modeling, Planning and Automated Optimisation*. Hoboken, NJ, USA: Wiley, 2006.
[20] I. Siomina and S. Wanstedt, "The impact of QoS support on the end user satisfaction in LTE networks with mixed traffic," in *Proc. IEEE PIMRC*, Sep. 2008, pp. 1–5.
[21] L. Song and J. Shen, *Evolved Cellular Network Planning and Optimization for UMTS and LTE*. New York, NY, USA: Taylor & Francis, 2011.
[22] S. Sesia and I. Toufik, *LTE—The UMTS Long Term Evolution*. Hoboken, NJ, USA: Wiley, 2011.
[23] I. Necoara, "A random coordinate descent method for large-scale resource allocation problems," in *Proc. IEEE 51st Annu. Conf. Decis. Control (CDC)*, Dec. 2012, pp. 4474–4479.
[24] B. Wah, Y. Chen, and A. Wan, "Constrained global optimization by constraint partitioning and simulated annealing," in *Proc. 18th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2006, pp. 265–274.
[25] Y. Cui, K. Xu, J. Wu, Z. Yu, and Y. Zhao, "Multi-constrained routing based on simulated annealing," in *Proc. IEEE ICC*, May 2003, pp. 1718–1722.
[26] H. Yousefi'zadeh, A. Habibi, X. Li, H. Jafarkhani, and C. Bauer, "A statistical study of loss-delay tradeoff for red queues," *IEEE Trans. Commun.*, vol. 60, no. 7, pp. 1966–1974, Jul. 2012.
[27] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2013, pp. 147–156.
[28] R. Qi and S. Zhou, "Simulated annealing partitioning: An algorithm for optimizing grouping in cancer data," in *Proc. IEEE 13th Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2013, pp. 281–286.
[29] B. Hajek, "Cooling schedules for optimal annealing," *Oper. Res.*, vol. 13, no. 2, May 1988. [Online]. Available: http://web.mit.edu/6.435/www/Hajek88.pdf
[30] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, pp. 475–494, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1017501703105#citeas, doi: 10.1023/A:1017501703105.
[31] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, pp. 2037–2060, Jan. 2013. [Online]. Available: https://www.math.ucdavis.edu/~sqma/MAT258A_Files/Beck-CD-2013.pdf
[32] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms*. Springer, 2008. [Online]. Available: https://www.springer.com/us/book/9783540731894?gclid=CjwKCAiA3abwBRBqEiwAKwICA8gVCgDi6QBQCNc0a03DCpgfq_QJk6-XD-XhTeNl-swGJYzSd7SnKRoC260QAvD_BwE

Amr Albanna received the B.S. and M.S. degrees from Ain Shams University, Cairo, Egypt, in 1995 and 2000, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, University of California at Irvine, in 2018. In the recent past, he was the Senior Engineering Manager at a Tier 1 Wireless Carrier and the VP of Engineering at Omega Wireless engineering firm. He is currently the President of CellOnyx, Inc. His research interests include cellular networks, machine leaning, and optimization.

Homayoun Yousefi'zadeh received the E.E.E. and Ph.D. degrees from the Department of EE-Systems, USC, in 1995 and 1997, respectively. He is currently the CEO of CellOnyx, Inc., and an Adjunct Professor with the Department of EECS, University of California at Irvine. In the recent past, he was a Consulting Chief Technologist at Boeing Company and the CTO of TierFleet. He has published more than 70 scholarly reviewed articles, authored more than 20 design articles associated with deployed industry products, and is the inventor of several U.S. patents. He was a recipient of multiple best paper, faculty, and engineering excellence awards. He is/was with the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE COMMUNICATIONS LETTERS, the *IEEE Wireless Communications Magazine*, the IEEE Journal of Selected Topics in Signal Processing, and *Journal of Communications and Networks*.