A mechanism for 5G MIMO performance optimization and evaluation

Dimosthenis Tzanakos, Foivos Allayiotis, Vasileios Kokkinos, Christos Bouras
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
tzanakos@ceid.upatras.gr, st1056636@ceid.upatras.gr, kokkinos@cti.gr, bouras@cti.gr

***Abstract*— MIMO (Multiple-Input Multiple-Output) technology has become a fundamental technique in current wireless communications, both in cellular and Wi-Fi networks. Fifth generation networks (5G) are expected to support extremely high data rates and more reliable services. The performance of 5G networks can be significantly improved with MIMO; yet the integration of optimization algorithms can further improve its performance. The purpose of this paper is to study an optimization technique for Channel State Prediction in a MIMO environment. We will examine a specific interconnection between 5G and optimization algorithms, to produce a feasible Channel State Prediction mechanism. For our experiments, we use a Dataset produced by the DeepMIMO simulator, which sets up all necessary parameters for our experiment. On this dataset we apply the Hungarian Algorithm to optimize the total throughput and based on the results we conclude that such mechanism is feasible and required for the introduction of 5G Networks in day-to-day tasks.**

Keywords—5G, MIMO, optimization, throughput, channel state information

I. INTRODUCTION

Worldwide, research into the new generation of mobile networks, called 5G, has been launched for years. Architectures provided by the International Telecommunication Union (ITU) in [1] have made it possible to integrate Machine Learning (ML) related mechanisms with optimization techniques to cooperate with network systems. With this kind of research on 5G technologies, efforts are made to improve the performance of existing technologies, by blending different areas of expertise. In our work, we will blend 5G technology with optimization algorithms, to produce a mechanism for Channel State Prediction.

The term Optimization is mainly related to techniques optimizing or even maximizing a problem, given specific constraints and problem parameters [2]. Due to the ever-increasing complexity of 5G Networks, and the increase of devices that will be connected to such networks, optimization is essential to achieve the 5G Key Performance Indicators (KPI). By integrating such algorithms into our experiment, we automatize every possible procedure, improve every aspect of the technology, and accelerate its introduction in the scientific area. The data employed in our approach constitude the input that will be fed to our optimization algorithm. It will then provide us with the necessary optimum output, without knowing the values or footprint. The idea above describes a common optimization algorithm, the Hungarian Algorithm [3].

The idea of applying optimization algorithms in 5G networks has been studied and assessed in previous research works. In [4], a mechanism is proposed, using a Deep Learning algorithm. In detail, a Channel State Prediction Mechanism uses a Deep Learning Algorithm to distribute the services to the users according to their throughput. We believe that a simpler optimization model, will provide a mechanism that requires lesser time complexity than a Deep Learning model and manages to reach similar size footprint. The authors of [5] have concluded that a generic algorithm (GA) can reach the similar accuracy of an exhaustive algorithm (EA) but without the large time complexity. Their idea is tested, through a series of experiments and produces a Channel State Prediction Mechanism limited to Video Application Services (VASes). We expand their idea and introduce a mechanism that could be used in 5G Networks for Calls, Music and Gaming as well.

On the other hand, optimization algorithms could be applied in other aspects of 5G networks. For example, in [6], a Generic Algorithm (GA) is used to find the optimal places of Base Stations (BSs), in several dense area networks, so the energy consumption is minimized. The BSs are placed in such way, so as to serve as many users as possible with the least number of BSs. The results show that a Real-coded Generic Algorithm (RGA) can find the optimal places of BSs to ensure that energy consumption is at a minimum. In [7] the authors, believe that an optimization algorithm would help the system performance of 5G networks, by reducing the frequency of call drop rate (CDR) and long interruption time (IT), when a user is moving. They concluded that an optimization algorithm, could make handover (HO) seamless, where HO is defined as the process of establishing a new radio link connection from the source to the target BS.

For our research we are going to develop an environment that incorporated MIMO technology. Using specific parameters, we get an appropriate dataset, that we then use it as input to our optimization algorithm**.** We will step on a MIMO Dataset based on accurate ray-tracing data from [8] and [9]. From there we will apply our optimization algorithm to produce the result. Our mechanism ensures that the total sum of throughput, that each user receives, is in the end maximized and does so in the best time possible.

The rest of the paper is organized as follows. In Section II we analyze our problem, its necessary features, and the optimization aspect. In Section III, we examine the optimization model, the method used in the experiments, the parameters and conditions contained in its inputs and the results, while we also describe the proposed mechanism. Furthermore, in Section IV we describe the testbed and examine all the necessary parameters presented in the experiments. The performance evaluation is presented in Section V with the use of plots and figures. Finally, the

ISBN 978-3-903176-42-3

conclusions and some possible future steps are briefly described in Section VI.

II. System model

In this section, we describe the adopted MIMO system and the network side of our problem.

We use the communication setup introduced in [9] (DeepMIMO Dataset), where a group of users are served by multiple antennas simultaneously. Our system model could easily be illustrated as a normal MIMO environment with many Base Stations (BSs). We assume that multiple users receive signals from multiple antennas and that the Base stations are equipped with several (M) antennas.

With this example environment we generate data that will serve as input for our optimization algorithm. The data are based on specific parameters set by the creators of [9] and [10]. The dataset is open source and available to be used by anyone. It contains data from three different scenarios, but the one that interests us is the one where signals are being produces by stationary BSs. The parameters, that were set by the original creators of the dataset, are divided into two categories. Firstly, the parameters that the system contains as standard information for its channels. The parameters for every Base Station and for every channel are the Angles of Departure (AoD) from the base station $\varphi_{az}^{b,u}$, $\varphi_{el}^{b,u,}$, the Angles of Arrival (AoA) at the user, $\theta_{az}^{b,u}$, $\theta_{el}^{b,u}$, the power received by the user, $P_l^{b,u}$, the phase of the path $\theta_l^{b,u}$ and the delay of the path $\tau_l^{b,u}$. Secondly the parameters that have a geometrical meaning in our environment, such as the number of Base stations (BS) , the number of antennas at each Base Station (M), the antenna spacing (d) , the bandwidth of the system (B), the total of the OFDM subcarriers K used for the channels' calculations and the number of the paths of the channels (L).

Fig. 1. A 2D example of the system, representing the real life MIMO environment with two main streets, buildings, 18 Base stations and the users along the streets. [10]

Using these data and considering the parameters, we calculate the Channel State Information.We firstly construct the MxN channel vector $h_k^{b,u}$ for every pair of BS and user that the researcher activates. The calculations are based on the number of subcarriers, using all the data mentioned above and which are necessary for these calculations [8], and $h_k^{b,u}$ is then generated from (1):

$$h_k^{b,u} = \sum_{l=1}^{L} \sqrt{\frac{\rho_l}{K}}\, e^{j\left(\theta_l^{b,u} + \frac{2\pi k}{K}\tau_l^{b,u}B\right)} \alpha\left(\varphi_{az}^{b,u}, \varphi_{el}^{b,u}\right) \quad (1)$$

After processing these data and using them accordingly, we generate the received signal results. With the knowledge of the channel vector between the BS and the user, which is the Channel State Information for these two parameters, the signal received by the BS and more specifically at the subcarrier K is then calculated, as in:

$$y_k = \sum_{n=1}^{N} h_{k,n}^{T}\, x_{k,n} + v_k \quad (2)$$

where $x_{k,n}$ is the transmit power (for example a transmit power of 30dBm) and where we also added the noise v_k at the subcarrier K.

The Dataset that we will use, created in [9], is defined by a ray-tracing scenario, based on works of [10], and it holds information concerning the signals, the bandwidth, the Angle of Arrival and Departure (AoA, AoD), etc. On this Dataset we apply a ML algorithm, to extract the Spectral Efficiency dataset as output. After being processed, the data are then passed and sorted into a matrix. Finally, they are categorized according to the service (Call, Music, Video, Gaming). More precisely, we will use these limits as a classification in the dataset that we obtain. The dataset itself contains the throughput information of the channel for each user (lines of the matrix), according to the codebook created by the Base Stations (rows of the matrix) [11]. Considering that every line of the matrix is a user and that the columns are the properties of the user, which are predefined by a codebook, we make assumptions and predict the throughput state of each user and the recommended service for the user.

In summary we firstly, use a supervised learning model that links the number of OFDM subcarriers in the Base Stations that we have chosen to activate with the beamforming vector created by each one of them. With that beamforming algorithm, we manage to gather every information necessary for the channel so that we can continue with the calculation of the output for our experiments. This output is the rate of the Channel for every pair of BS and users and it is also referred as the achievable rate of the channel. Its content is measured in bps/Hz and we calculate it as follows; for every BS n the achievable rate of the channel $R_n(p)$, with a beamforming vector fp, is given by:

$$R_n^{(p)} = \frac{1}{|K|}\sum_{k\in|K|} \log_2\left(1 + SNR\left|f_p^T h_k^{n,u}\right|^2\right) \quad (3)$$

Then, using the achievable rate of the channel, we calculate the throughput of the channel. This process excludes the Bandwidth from the Dataset that we have until that point. The Dataset with the achievable rate of the Channel contains Information with the form of bps/Hz. This content shows the spectral efficiency of the channel. Spectral efficiency most commonly is expressed as "bits per second per hertz," or bits/s/Hz. It can also be defined as the data rate of the channel divided by the bandwidth of the channel in Hz.

Spectral efficiency = channel data rate in bps / Channel Bandwidth in Hz

In our example, the bandwidth of the channel is 0.5 GHz, and it can support a data rate of 6 Mbps. In that case the spectrum efficiency of the channel can be calculated as follows:

Spectral efficiency= $6 \times 10^6 / 5 \times 10^8 = 12 \times 10^{-2}$ bits/second/Hz

The information we need for the maximization side of our experiments is bps. Then, we evaluate the quality of the channel and propose different services for it. The services are based on known throughput limits that are set worldwide and of course are dependent of the channel itself.

Based on the service we want to support in our system, the more throughput our users require. For our experiments, we only collect the speed requirements for 4 types of services: music, video, gaming, and calls. Each of these services have different speed requirements. For example, a call needs less bps than a song, which need less bps than a video. Then of course, online gaming consumes more bps than all the other services. Then again, the number of bps that each service requires depends on many factors, such as the number of services used at the same time or the quality of service that the user asks for. To make our experiments simpler and easier to understand, we will simply take into consideration the minimum recommended speed for every service. We will think of the services as one independent from the other. We collect the recommended speed for the 4 services as followed:

TABLE I: THE THROUGHPUT RECOMMENDATIONS FOR INTERNET SERVICES

Services and Throughput	
Service	***Recommended Throughput in bps***
Call(VoIP)	100 kbps
Music	160 kbps
Video	500 kbps
Gaming	6000 kbps

All of the above equations and the overall Dataset is applied to the algorithms we will discuss in detail in the next Section, where we analyze our mechanism, and the overall idea behind our research.

III. PROPOSED MECHANISM DESCRIPTION

To better understand our experiment from the mobile networks side, it is very important to explain the mechanism that we used to produce all the necessary parameter values of our system. To do that, it is better to present the pseudocode hidden behind the lines of code that produce, line by line, our dataset and all the content of the simulated environment. This simulation is a good example of a real environment and it shows how every part of the system, such as the antennas or the subcarriers, would interact with all the other parts of the system, such as the user or the other antennas.

The proposed system model follows a specific process step by step. It is highly adaptable at any stage of the process and is also intertwined with our system parameters and the dataset that we introduce as input. The process is as follows:

Algorithm 1 Proposed scheme:

```
1: Set system parameters
2: Run ray-tracing simulator Wireless InSite by Remcom
3: Get channel calculations
4: Run Machine Learning model
5: Get users' throughput
6: Run Hungarian Algorithm
7: For i = 1: total_num_users      % Define ML classes
8:     if user_throughput(i) < 100 kbps
9:             class_user(i) = 1
10:    else if 100 kbps <= user_throughput(i) < 160 kbps
11:            class_user(i) = 2
12:    else if 160 kbps <= user_throughput(i) < 500 kbps
13:            class_user(i) = 3
14:    else
15:            class_user(i) = 4
16:    end     % if statement
17: end        % for loop
```

We first create the virtual environment and system characteristics as proposed by [9]. We determine all the necessary parameters that will define our 5G MIMO system. We reproduce all the inputs and scripts in MATLAB, and we export various results and channel datasets. The dataset that we will use on the first level of output processing is the Channel State Information dataset. That Dataset is used by a Machine Learning algorithm, which gives us a more detailed channel calculations like the spectral efficiency of the channel for specific conditions. Next, we prepare the spectral efficiency Dataset to get the throughput of the channel for the user and for the same specific conditions. We then run the Optimization model which studies our system and provides us with maximized solutions based on the Dataset and the throughput criteria that we defined. What we just described can also be depicted with a simple flowchart (Fig. 3).

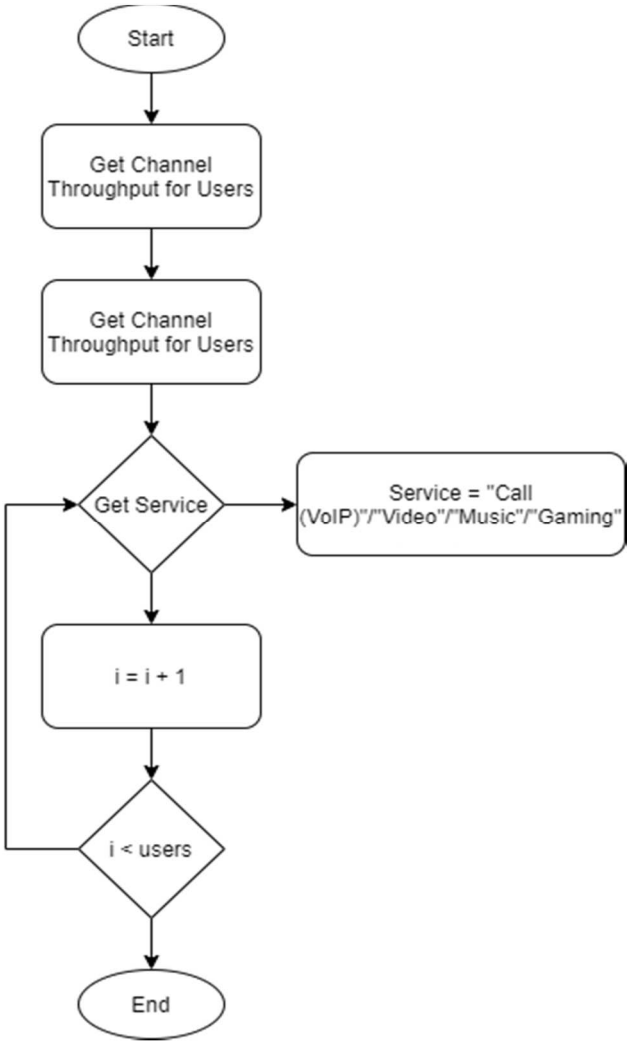

Fig. 2. The process of the throughput-information creation and of the maximization.

In our scenario, we advocate using the Hungarian Algorithm because it is a general-purpose algorithm for solving assignment problems, providing us with the maximum and/or minimum results at the shortest time complexity. The Hungarian Algorithm primarily aims to reduce the total output. In our case, we want to increase the total throughput. To achieve this, we minimize the amount of throughput we would normally lose. Thus, the first two steps have been added to the generic Hungarian Algorithm to obtain the desired output. It's a very useful model for our system requirements, as we have a lot of characteristics that primarily slow down our system. With the steps of the algorithm presented below, we accelerate our system's results and maximize the desired output. The algorithm that we implement is as follows:

Algorithm 2 Hungarian Algorithm:

1: Find maximum value of the matrix
2: Subtract all elements from maxValue
3: Find row minimum
4: Subtract row min from all values in that row
5: Find column minimum
6: Subtract column min from all values in that column
7: Cover all zeros with lines
8: If num_lines < users
9: Find min and subtract from elements with no line
10: Add min to elements with two lines
11: Go to line 7
12: Make the final assignment

The algorithm above is applied to the outputted Dataset of the ML model introduced in [9] and discussed in detail above. The algorithm can be summarized in words as such: Firstly, we find the maximum value of the matrix and subtract from that value all the elements. We then find the minimum from every row and subtract it from all the elements of that row. We perform the same action to the columns. Then we try to find an optimal assignment. We take each row and search for a zero. If there are more than one zero in that row, we leave it, and we continue. If there is only one zero, we make the assignment and delete all other zeros in the column where the zero was found. We continue until no more rows are left. The same process happens for the columns. If there is only one zero, our code makes the assignment and deletes all other zeros in the row where the zero was found. We continue until no more columns are left. If each row has an assignment, we move forward. If not, we repeat the assignment process, until no more assignments are possible. If an optimal assignment was found, the Algorithm stops.

If an optimal assignment has not been found, we follow the procedure below:

- Tick an unassigned row. If there is a zero in that row, tick the corresponding column. Stop when no more ticking is possible
- Look up all the ticked columns if there is an assignment tick the corresponding row. Stop when no more ticking is possible
- Run lines through unticked rows and ticked columns
- Find the minimum from the elements that have no lines passing through them
- Subtract the minimum from the elements that have no lines passing through them. Add the minimum to all elements that have two lines passing through them. Leave the rest unchanged
- Test for an optimal assignment and repeat the process
- Make the final assignment if there is one

The Hungarian Algorithm produces the desired result in $O(n^3)$ time. Initially it worked only on square matrices, but later versions have modified the algorithm to work with any size of matrices [12]. In [13], James Munkres had determined that the algorithm is strongly polynomial.

IV. TESTBED DESCRIPTION

For creating a MIMO-compliant maximization dataset, we used the MATLAB tool to simulate a real-world environment with specific parameters. The parameters that we can change depending on our needs are: Active BSs, Active users, Number of BS antennas, Antenna spacing, System bandwidth, OFDM parameters and Number of channel paths.

By setting the parameters as shown in Table II, we firstly get the Channel State Information. Then we use that information to get the content that we want for our experiments.

TABLE II: THE DATASET PARAMETERS WE USED FOR OUR SYSTEM.

Dataset parameters and Values	
Dataset parameters	***Values***
active_BS	[3, 10]
active_user_first and active_user_last	1600 and 1620
num_ant_x, num_ant_y, and num_ant_z	[1, 32, 8]
ant_spacing	0.5
bandwidth	0.5
num_OFDM, OFDM_sampling_factor, OFDM_limit	[1024, 1, 64]
num_paths	3

For creating a MIMO-compliant maximization dataset, we used the MATLAB tool to simulate a real-world environment with specific parameters. The parameters that we can change depending on our needs are:

- Active BSs (active_BS in the MATLAB code): The first parameter is the number of the Base Stations that we want to use and activate in our experiments. By having active and inactive Base Stations in our code, we minimize the Dataset size and in the same time maximize the performance of our technique and algorithm. In case we want for example to assess the interaction between the mobile users and the Base Stations 3 and 10, we set active_BS=[3,10].
- Active users (active_user_first and active_user_last in the MATLAB code): In the same context as in Active BSs, we define our active users, meaning the ones that want to use our network at that specific period of time. We do that by naming the first and last row of the users

that need the networks services at that time. For example, by setting active_user_first=1600 and active_user_last=1620, we activate the users between the rows 1600 and 1620.

- Number of BS antennas (num_ant_x, num_ant_y, and num_ant_z in the MATLAB code): In our example, we have a total of 18 Base Stations in an area depicted in Fig.1. With a uniform array, our Base Stations expand in the x, y and z axes. That means that if our system and our Base Stations have a 16 × 16 uniform planar array (UPA) along the street, we set num_ant_x=1, num_ant_y=16, and num_ant_z=16.
- Antenna spacing (ant_spacing in the MATLAB code): The distance between two antennas of a Base Station is relative to the system's wavelength. In our experiments, we set ant_spacing=0.5 , as in a half-wavelength antenna spacing system.
- System bandwidth (bandwidth in the MATLAB code): Our system requires a GHz type of data transfer capacity. A network with the specifications of our experiments would need a 500MHz bandwidth. Thus, we set bandwidth=0.5.
- OFDM parameters (num_OFDM, OFDM_sampling_factor, and OFDM_limit in the MATLAB code): The antennas contained in our system's BSs have a specific number of subcarriers, the number of which we can control. By activating only some of the subcarriers, we reduce, as mentioned previously, the Dataset's size. With the OFDM_limit parameter, we set a limit on the samples that we want to consider for our experiments. With OFDM_sampling_factor we define exactly that sample group of subcarriers which we take into consideration for our calculations. In our example, meaning an OFDM system with 1024 subcarriers, we set num_OFDM = 1024, OFDM_sampling_factor = 1, and OFDM_limit = 64 because we calculate the channels only at the first 64 subcarriers,
- Number of channel paths (num_paths in the MATLAB code): For each interaction between a mobile user and a Base Station, our system calculates a total of 25 possible paths, each one with different received power and data capacity. In many applications, we consider only the strongest path, or a number of the strongest paths. We set num_paths=3 for the calculation of the 3 strongest channel paths calculated in the system.

V. Performance evaluation

In this section, we give describe our experiment, and an explanation of the results with plots and figures.

To begin with the production of our last dataset, we select the beam detection scenario we are interested in, from the available scenarios provided by Remcom Wireless InSite [8]. In our case, a real mobile phone system. This ray tracing scenario contains the appropriate data describing the physical characteristics of the environment we are studying, as described in IV, and in the form of MATLAB code entry (base stations, users, buildings, materials). We prepare the environment on which the simulations will be made, giving to the code the appropriate parameters for the experiments. We give the values of Table II in the corresponding parameters in the code file, we run the first level of experiments and we get the Channel State Information Dataset. Then, after generating the dataset with the CSI content, we use it as input to the 2nd part of the simulation, which is to use Machine Learning techniques to produce the dataset with the throughput information we want for the final part of the experiments. This step generates all the useful files that contain useful information to describe the channel and the user-base station relationship. In addition to useful, for our case, files, such as the codebook, which determines the order in which the information is placed in the final dataset for the throughput of the user, and the Deep Learning input, which is the processed dataset with complex information for the throughput.

The DL output contains the spectral efficiency information we have mentioned above and which we use to find the final dataset with the throughput for the channel from the user's side. It has a size where U is the number of users and C is the number of columns given by the codebook. We then divide by the channel wavelength to produce a matrix of the same size, but with throughput (bps) information. We believe that an optimization algorithm helps us achieve our goal in a relatively small amount of time.

So, for the optimization application that we recommend, we have the following:

- We take the output of the Deep Learning and remove the wavelength from the dataset, dividing each cell by the wavelength of the system.
- Each line of the generated dataset is a user (entity for the optimization algorithm).
- We run the proposed optimization algorithm on the output matrix.
- The final dataset is composed of as many rows as users and four columns. The first column is the user's throughput given from a specific subcarrier. The second column holds the number of the subcarrier that provides the throughput, while the third column has the service the throughput falls upon. The final column contains the recommended throughput the user needs to carry on his service with no problem.

With this method, we will receive a maximized solution with the proposed internet service. We give as an example four services, but a future implementation could include a wider range of throughput. In the graphs, below we can see the results when we run three different scenarios of algorithms used to generate the final output. The first two algorithms are basic forms of the Hungarian Algorithm The third algorithm, instead, provides the optimized implementation of the Hungarian Algorithm. We choose this approach to show that optimization algorithms, will always favor others, both in results and in time complexity. The algorithms are as follows:

- Algorithm 1 uses basic "for" loops and "ifs" to scan the input matrix for the maximum value. It then deletes the user and subcarrier from which the max value came from. It continues iteratively until all the users are satisfied.
- Algorithm 2 uses the built-in functions for retrieving the maximum value in linear indices, and it transforms them into row and column values. The returned row and column represent the user to whom the Throughput belongs to and the subcarrier from which

the maximum Throughput is originated. The returned user and subcarrier are deleted. It also continues iteratively until all the users are satisfied.

- Algorithm 3 is the implementation of the proposed Hungarian Algorithm. The Algorithm runs the built-in function "matchpairs()" which finds the best assignment for all users to achieve the maximum total throughput.

In Fig.3, we show the different aforementioned scenarios (marked as S in the figure, where S1 corresponds to Scenario 1, S2 to Scenario 2 and S3 to scenario 3). The number, under each bar corresponds to the number of users we employed to our system. Finally, AL1 corresponds to Algorithm 1, AL2 to Algorithm 2, while HA corresponds to the Hungarian Algorithm. As we can see, the proposed algorithm greatly impacts the total throughput. In all cases, the total throughput per number of users is greater than both the other algorithms that use a very basic structure. This is achieved as the Hungarian Algorithm is designed to solve assignment problems, in such a way that in the end the desired output, in our case the throughput, is maximized. The users are served by the nearest subcarrier, as the closer a user is to an antenna the larger the generated Throughput is. Furthermore, the other two algorithms eliminate a subcarrier without taking into account the other users. More specifically, the deleted subcarrier may provide the highest throughput to a single user, but if the deleted subcarrier covered another user equally well, the total throughput could have been higher if the deleted subcarrier had been assigned elsewhere. The other two algorithms do not take such cases into account.

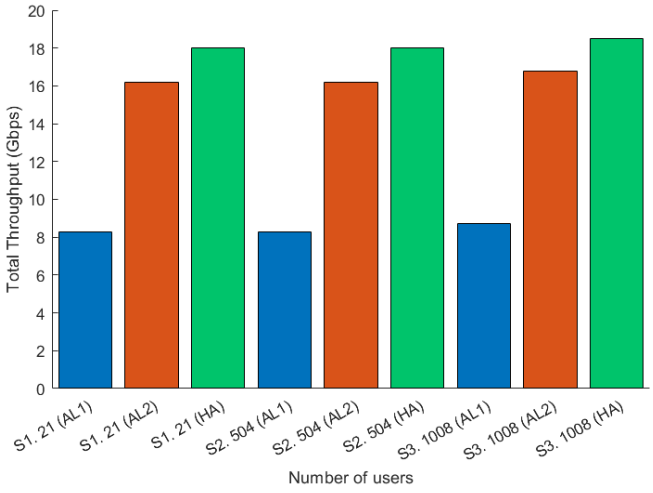

Fig. 3. The effects of each algorithm on the throughput of the users.

Fig.4 shows that our proposed algorithm produces the final output in the best time possible. Even though, the time for a small number of users is slightly greater than the other two algorithms, with several users, around 500 and 1000, it produces the results in less time than both algorithms. It takes longer than the other two algorithms in cases where the users are fewer as it is not profitable to run this complex algorithm for such a small number of samples. While in cases of increasing number of users we can say that it handles a larger volume of data better than the other two algorithms, as it consists of much fewer iterations and no nested iterations. It also takes advantage of MATLAB's built in functions, which are designed to execute at the best possible time, but also makes better use of recursion. Another thing to factor in, is the fact that the Hungarian Algorithm only requires to be executed once and it also skips the process of deleting the user who requests Throughput, and the subcarrier who provides it.

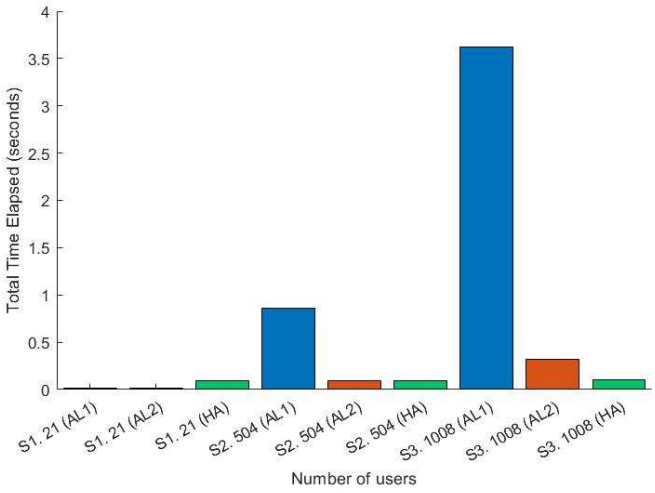

Fig. 4. The time complexity of each algorithm used in the experiments per number of users.

VI. Conclusions and Future Work

In this paper we presented a method for creating a dataset for a 5G mobile network and maximizing the results each time, depending on the needs of the system and of the users. This dataset has a feature that can be used further in optimization techniques, to automate and accelerate the mobile network. After generating the dataset with the desired information, the next step is to convert it to the appropriate format as an input to an optimization algorithm. To this dataset we add the minimum throughput required to perform a particular service on the Internet, such as Call over IP or playing a video on YouTube. Then, the optimization algorithm takes this dataset as input and accurately selects the service that the user can support at that time and under the specific conditions.

By introducing optimization methods into the system, a comprehensible and sustainable process is achieved that solves various problems, such as finding speed limits for the services provided by the internet. The results show that the proposed model, with the appropriate input parameters, effectively adapts to the ever-changing environment, while at the same time providing a robust mobile network system.

The results of this work encourage various future research, such as the use of other forms of information to predict and maximize network quality and the exploration of other network simulation scenarios in multi-user environments and multiple antennas.

Finally, a future extension of our work could be done in the field of Machine Learning in selecting a more appropriate technical classification. There are many such techniques that can be applied and evaluated based on meeting the real needs of users and databases. Examples are linear classifiers, Perceptron's sensors, support vector machines, decision trees, neural networks, Bayes networks, etc.

References

[1] F. Wilhelmi, S. Barrachina-Muñoz, B. Bellalta, C. Cano, A. Jonsson, and V. Ram Ov, "A Flexible Machine-Learning-Aware Architecture for Future WLANs", IEEE Communications Magazine • March 2020 IEEE Communications Magazine 58(3):25-31.

[2] S. Amaran, N. V. Sahinidis, B. Sharda, B. “Sharda Simulation optimization: a review of algorithms and applications”, 4OR-Q J Oper Res 12, 301–333 (2014).

[3] H. W. Kuhn, "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, 2: 83–97, 1955.

[4] M.-P. Bui, N.-S. Vo, T.-V. Truong, T.-H. Nguyen, N. V. Nguyen, and C. Yin, “Genetic Algorithms for Multi-tier Caching and Resource Sharing Optimized Video Streaming in 5G Ultra-dense Networks”.

[5] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, and V. Gopalakrishnan, “On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges”, IEEE Communications Magazine • March 2020.

[6] R. Sachan, T. J. Choi, C. W. Ahn, “A Genetic Algorithm with Location Intelligence Method for Energy Optimization in 5G Wireless Networks”, Hindawi Publishing Corporation, Discrete Dynamics in Nature and Society, 2016.

[7] A. Alhammadi, M. Roslee, M. Y. Alias, I. Shayea, S. Alraih and K. S. Mohamed, "Auto Tuning Self-Optimization Algorithm for Mobility Management in LTE-A and 5G HetNets," in IEEE Access, vol. 8, pp. 294-304, 2020, doi: 10.1109/ACCESS.2019.2961186.

[8] Remcom, “Wireless insite,” http://www.remcom.com/wireless-insite.

[9] DeepMIMO Dataset. [Online]. Available: http://www.DeepMIMO.net

[10] A. Alkhateeb, “DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications”, Proc. of Information Theory and Applications Workshop (ITA), Feb., 2019.

[11] C.-K. Wen, W.-T. Shih, and S. Jin, “Deep learning for massive mimo csi feedback,” IEEE Wireless Communications Letters, 2018.

[12] H. W. Kuhn, "Variants of the Hungarian method for assignment problems", Naval Research Logistics Quarterly, 3: 253–258, 1956.

[13] J. Munkres, "Algorithms for the Assignment and Transportation Problems", Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38, 1957 March.