A Machine Learning Approach to User Assignment in 5G Networks

Michael Kouris[1], Vasileios Kokkinos[1], Apostolos Gkamas[2], Philippos Pouyioutas[3], Christos Bouras[1]
[1]*Computer Engineering and Informatics Department, University of Patras, Patras, Greece*
[2]*University Ecclesiastical, Academy of Vella, Ioannina, Greece2*
[3]*Computer Science Department, University of Nicosia, Nicosia, Cyprus*
mkouris@ceid.upatras.gr; kokkinos@cti.gr; gkamas@aeavellas.gr; pouyioutas.p@unic.ac.cy; bouras@upatras.gr

Abstract—**With the advancement of wireless networks the data needs of the wireless internet have become so great, and the use of 4G and 5G so ubiquitous, that challenges arise in the availability and distribution of resources. In this paper we examine an application of Machine Learning and subsequently Neural Networks to offer a solution to this problem. The, now more than ever, profound availability of processing power greatly empowers and makes the deployment of these tools easier than ever, even in problems where in the past their application would not be feasible. The use of machine learning techniques to distribute resources in wireless networks is investigated and contrasted to a traditional algorithmic method. Scenarios are also entertained in which such approaches might be applicable.**

Keywords—5G, MIMO, Deep Learning, Neural Networks, User Assignment

I. Introduction

With the rapid advancement of 4th and now 5th generation mobile networks and their widespread use by the general public we have more congested wireless networks than ever before. Having the above in mind, the international scientific community has been exploring mechanisms and optimization techniques to alleviate any potential issues. The current architectures provided by the International Telecommunication Union (ITU) allow the deployment of said tools and techniques. Some of these proposals include the usage of Multiple-Input Multiple-Output (MIMO) and Downlink and Uplink Decoupling (DUDe) but also of the implementation of Machine Learning (ML) to better improve the functionality of said networks. The ongoing development and research around MIMO and adjacent techniques and technologies allows us to better take advantage of the available radio resources. Most research is based around the effective usage of spectral resources along with heterogenous networks [1].

Similarly, the ever-increasing availability of computing power has complemented a rise in the usage of Deep Learning and neural networks to create solutions in areas where traditional algorithmic approaches fail or are too costly to develop. These data structures essentially can be trained to perform algorithmic-like tasks with decent generalization [2].

Naturally, the intersection of 5G-MIMO and Machine Learning and/or Deep Learning has already been broached by other related works such as [3] in which it is used as a tool to predict the Channel State and in effect act as an optimization agent. Additionally, in paper [4] ML is used as an all-around solution for both traffic optimization and revenue maximization (or energy minimization). Another proposal by Guan et al. [5] uses reinforcement learning in an effort to allocate various types of channel resources more efficiently with no concern for the computing costs. This differs significantly from what was attempted in this paper, where the main objective is the decrease of computational complexity with short amounts of training for our neural networks and with a somewhat portable solution.

In addition to the above works, in [6] a cascade of Neural Networks (NNs) is used, one to approximate the bandwidth allocation and one to output the transmit power required for the Quality of service (QoS) needs of the users. This work also differs from our approach, which aims to be more economical in computing power and in reducing delays, while simultaneously it highlights the importance of working towards minimizing the energy requirements as well as maintaining adequate bandwidth for the users based on their needs.

In this paper a novel application of ML in the problem of user assignment is investigated. The user assignment problem refers to the most favorable assignment of a number of users to a number of antenna cells in a way that maximizes total throughput. This is essentially a solved problem as variations of the Hungarian Algorithm already solve this in an optimal way in regards to total throughput. However, this is a very computationally expensive proposition as the algorithm is highly iterative and exhibits polynomial time complexity of $O(n^3)$ and sometimes even worse. This constitutes a problem with the low-power hardware that is often used in Base Stations (BSs). The novelty of this approach lies in the usage of Neural Networks not for the solving of problems in which traditional algorithmic approaches fail but rather as a means of reducing computational complexity.

The paper is laid out as follows. In Section II the tools used to simulate a simple 5G wireless network are explained as well as some of the parameters and options chosen to create the datasets. Section III examines the method by which our networks are trained and contrasted against a traditional algorithmic approach, along with some practices to ensure the integrity of the testing. In Section IV the performance results are evaluated and examined with the help of plots and figures. Finally, Section V briefly outlines the conclusions that one can draw from these results as well as some ideas for future work and improvements both on the models used and some of the tools.

II. System Model Simulation

This section includes the description of the MIMO simulation system as it pertains to the network side of the problem.

The simulation tool used in [7] offers a variety of scenarios, as well as customization options in regards to number of BSs, number of antennae cells per BS, their position, signal strength, number of users etc. In addition, this tool simulates the obstruction and interference caused by buildings and other objects which might hinder any individual cells' signal strength to the users. There is a variety of scenarios built-in the tool, differing on the building layout and some other factors, in Fig. 1 we can see the scenario that was picked for this experiment.

We use REMCOM's Wireless InSite which features reflections, transmissions and diffractions of the Electromagnetic signals (EM) along with atmospheric absorption and diffuse scattering for a more realistic propagation of EM radio resources. What is more it is specifically designed to emulate MIMO antennae used in 5G networks along with some of the most common techniques like beamforming and spatial multiplexing in order to predict various channel metrics like signal-to-interference & noise ratio (SINR) as well as bit error rate (BER).

In order to calculate these effects some parameters are used that are initialized by the creators of the application such as the Angles of Departure (AoD) from the BS, the Angle of Arrival (AoA) to the user, the power received, the phase of the path and finally the delay. Other parameters that are set by the 'user' include the number of BSs, number of users (and by extension User Equipments, UEs), the location of the users and BSs, the number of antennae per BS as well as the bandwidth of the system. Using this data and the parameters the Channel State Information (CSI) is calculated. First a MxN vector of the channel ${h_k}^{b,u}$ is calculated, where M and N are the number of users and antenna cells respectively.

The calculations are based on the number of subcarriers using all the data that were mentioned above and as such we can calculate ${h_k}^{b,u}$ as:

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

From the processing of this data, the results of the recipient signal are created. Knowing the vector of the channel between BS and the User (the CSI) for these 2 parameters the signal received from the BS and specifically in subcarrier K is calculated as follows:

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

where $x_{k,n}$ is the transmission strength (in dBm) and the addition of noise v_k to the subcarrier K. The sum of the data that will be used are set by a raytracing scenario; an ML algorithm is applied to this data to calculate the spectral efficiency. Upon calculation the data is stored and sorted on a matrix form. This contains the information concerning the channel for every user (lines on the matrix) according to the code book created by the BS (rows on the matrix).

The achievable rate of the channel ${R_n}^{(p)}$ is calculated as follows:

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

Having calculated the achievable rate of the channel one can calculate the efficiency of the channel. This process isolates the bandwidth from the dataset used up until that point. This content shows the spectral efficiency of the channel, usually expressed as 'bits per second per hertz' or bit/s/Hz.

Some of the parameters that had to be set manually include

- Number of BSs: 2
- Number of cells per BS: 500
- Maximum number of users: 1000
- Antenna spacing: 0.5

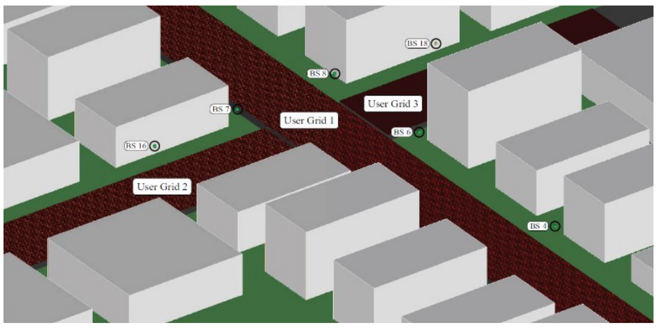

Fig. 1. The topology simulated by the DeepMIMO tool.

III. Mechanism Description

The testing mechanism builds upon the works of [1] and is built to evaluate the performance and limitations of a ML-based approach for user assignments as opposed to a more traditional algorithmic approach. Firstly, the dataset on which the NNs will be trained is created. The process used follows the Algorithm below:

Algorithm

```
1: Set System Parameters
2: Set number of Users
3: Execute Simulation using DeepMIMO Dataset Generator
4: Receive relevant channel information, namely throughput
   of users to every BS
5: Execute Hungarian Algorithm
6: Save the results of the Hungarian Algorithm and repeat
   steps 2-6 as needed, hundreds/thousands of times
7: Feed 90% of the unsolved problems and their solution
   (results of step 6) to a feedforward Neural Network
8: [net, tr] = train(net,input,desiredM','useGPU','yes');
9: Solve the remaining 10% of problems both by the now
   trained Neural Network as well as the Hungarian
   Algorithm
10: y=net(input_test);
11: [Minput,uR,uC] = matchpairs(DL_output,
    costUnmatched, 'max');
12: Compare runtimes and cumulative throughput afforded to
    users
```

- A number of users is decided and they are placed into the simulation suite of DeepMIMO.
- The simulation then calculates the achievable throughput of every user in relation to every antenna cell of each BS.
- The assignment problem is then optimally solved in polynomial time O(n^3) through an improved version of the Hungarian algorithm.
- The data is saved and the optimal antenna-user pairs are identified in regards to their relative location in this specific geographic topology.
- Repeat for a great variety of user numbers and locations, always keeping the BSs stationary in the same positions and the geographic topology constant.

Each iteration of this process generates a matrix sized MxN (Number_of_users*Number_of_Antennae) which in each position has the signal strength/throughput (the two are analogous in this context) for the current user with all the antennae as well as an identically sized extremely sparse matrix which only denotes the optimal coupling of antennae cells with users for a maximum total throughput.

These datasets, they are then fed into a feedforward NN for training. This, in essence, allows the NNs to understand the topology.

Having created the datasets, 90% of them are fed into our feedforward NNs along with the correct assignment as indicated by the Hungarian algorithm to begin the process of supervised learning. The remaining 10% of the datasets are used to evaluate the NNs performance in examples they have not been trained on, a very important factor to avoid overfitting.

The NNs used are Feedforward Neural Nets [8], meaning the information only moves forward through the hidden nodes and doesn’t loop. In addition, the training function used was Scaled Conjugate Descent (SCD) as it performs very well in highly parallel tasks and is shown to be significantly faster than simple backpropagation and other conjugate gradient methods [9], [10].

A few different networks architectures were used in conjunction with a manual softmax layer. Specifically, 5 networks of differing numbers of neurons and hidden layers were tested to examine both their suitability for our problem but also to observe if different helpful properties could be observed in any of them. Before the datasets were fed into the Neural Nets they were converted into binary scale so as to denote the BS-User pairings in the matrices. Due to storage limitations a few hundred examples were created. Each example had a 1000x1000 matrix denoting the relative signal strength of each user to each individual antenna cell, the solved pairing by the Hungarian algorithm for each such matrix which was also a 1000x1000 extremely sparse binary matrix as mentioned above as well as some other relevant information concerning the channels.

Of the 5 networks tried, as seen in Table I, and in the interest of brevity, we will examine some of the training data from Network 3 and whose architecture can be seen in Fig. 2, in which we can observe 2 hidden layers of 100 neurons each and one output layer.

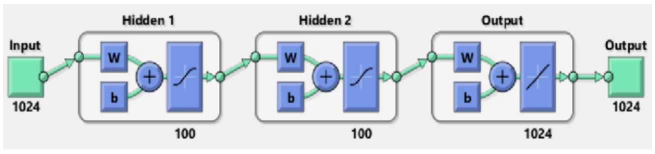

Fig. 2. Network 3 architecture.

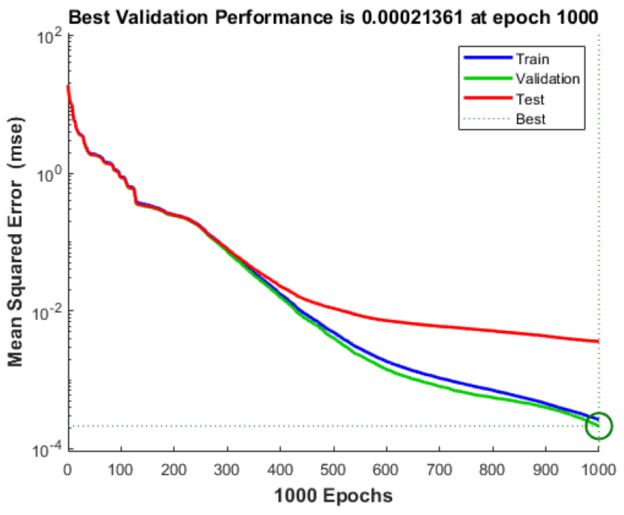

Fig. 3. Mean Squared Error of Training Net 3.

In Fig. 3 we can see one of the validation performance graphs of the training of Network 3. Their function is to measure the Mean Square Error (MSE) between observations and predictions as well as their performance in the validation sample. In general this error is reduced as the epochs progress. Delayed increase usually indicates an overfitting problem which occurs when the examples are too similar which leads to a weakness to generalize. There is protection against this issue as after 6 continuous increases of the MSE the training halts. As long as the error continues to decrease the training continues and receives asymptotically better accuracy. For implementation reasons networks were halted at 1000 epochs of training.

The lack of any continuous increases that was observed in the training of all of the networks shows that no overfitting has occurred which in turn indicates that these models could have increased performance with further training or more data, for practical implementation reasons this was not attempted.

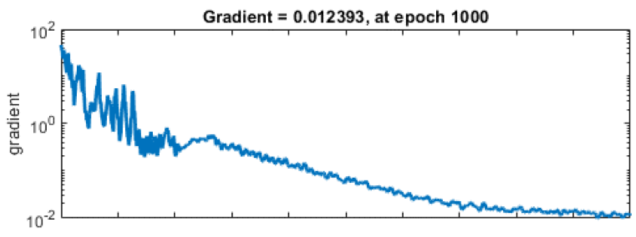

Fig. 4. Training state plot of Net 3.

Training state plots provide us with the value of the backpropagation gradient in each epoch in a logarithmic scale (Fig. 4). Effectively this indicates the speed at which the model converges to the local minima of the activation functions. Validation fails indicate overfitting or overtraining. As expected, the model converges quickly in the beginning and levels off. This is a strong indicator that further increases in accuracy would require more data rather than more training time.

In the following Table we can see a brief overview of the different variations of the Feedforward Neural Networks that were used.

TABLE I. NETWORK CHARACTERISTICS

Nets	Nr of Layers	Nr of Neurons	Nr of weights
Net 1	1	100	205924
Net 2	1	1024	2099200
Net 3	2	200	216024
Net 4	3	250	169874
Net 5	3	350	236174

IV. PERFORMANCE EVALUATION

Different architectures of NNs were tried, with different number of neurons as well as differing number of hidden layers. The computational cost of training these networks differs and in general linearly increases with the number of neurons and hidden layers. The comparison between them in this regard is not judged important as it is a one-time cost that is only relevant during the first creation of the network.

The two main metrics that are examined are the following:

- Efficiency, the total sum of the throughput that is achieved by users with their specific BS pairing which can be seen in Fig. 5.
- The runtime of the NN as an expression of the computational complexity as contrasting to the runtime of the Hungarian algorithm. An important factor here is the usage of CPU (for both) so that comparisons are applicable. The results can be seen in Fig. 6.

Important to note that runtime was used rather than calculating the computational complexity as it is not feasible to perform said calculation on NNs.

Firstly the NNs are compared to each other and to the Hungarian algorithm in both efficiency and runtime followed by a more in depth look at the NNs and their architectures.

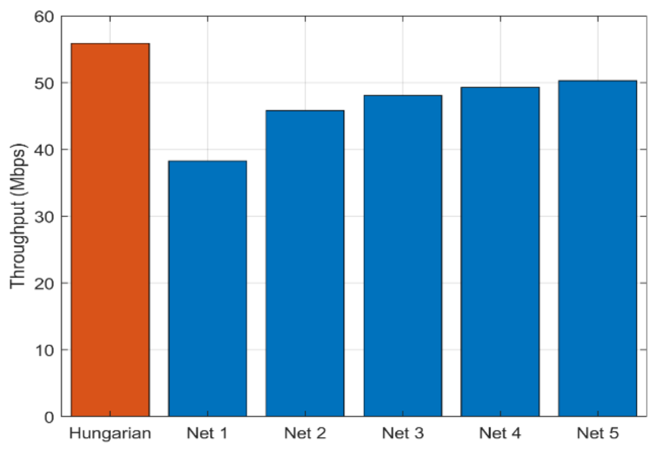

Fig. 5. Average throughput achieved per user.

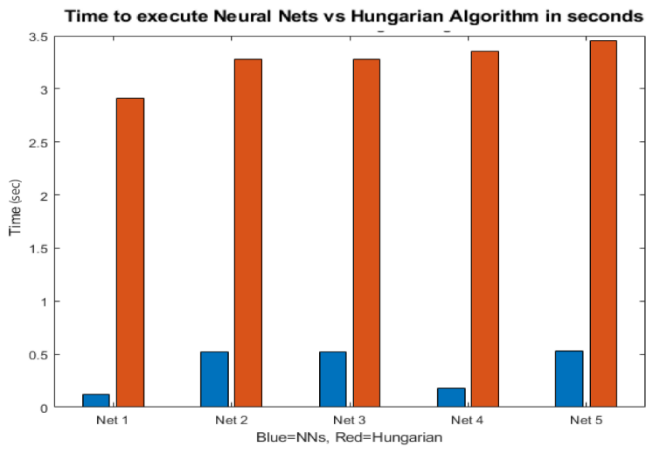

Fig. 6. Runtime of Neural Networks.

As expected, a decrease in efficiency is observed, as outlined previously. This holds especially true in the more rudimentary Networks such as 1 & 2, it is theorized that this is the result of the significantly fewer weights leading into a weakness to learn all the required information. After a certain point, further adding hidden layers and neurons in the networks seem to result in diminishing returns. The worst performing Net 1 achieves 82% of the total throughput achieved by the Hungarian Algorithm and the best performing, Net 5 achieves 90%. It is assumed that a further and more thorough training with more examples would increase said performances.

The difference in runtime as an analogue to computational complexity, however, is staggering. 10-fold decreases in runtime and more were observed. This was the main objective of this work. It is also assumed that these differences will only magnify as more users and BSs are added and the computational cost of the Hungarian algorithm dramatically increases. What is more these are the times needed by a powerful desktop CPU, whereas BSs will be using low power (and performance) processing units which would only exacerbate the runtimes. It is worth noting the difference in runtime between the Hungarian algorithm in identical problems (which means identical operations), this occurs because at different times the computer has other tasks in the background which create small fluctuation in how long an identical process will need on the same hardware. This is treated as noise/error for the purposes of this paper. This is important as especially with the shorter runtimes of the Nets this error margin overwhelms the ability to meaningfully compare the networks against one another.

V. CONCLUSIONS AND FUTURE WORK

It is evident from the results that the process outlined in this paper works to drastically reduce computational complexity in the user assignment problem. However, a not insignificant loss of total throughput was also observed. The above, in conjunction with the fact that no training validation graphs at any point showed an increase in the MSE but were rather halted upon reaching the required epoch number indicates that there is room for more robust training of the networks either in training time or in the form of more data. This is a relatively straightforward problem that essentially requires more computing power as well as storage, which was the real limiting factor in this attempt.

At the same time this implies that one could even work to further optimize some of the simpler networks, like Net 1 which would both make it more accurate but at the same time maintaining what was the least computationally expensive option. It would also be interesting to examine the performance of such an approach is significantly more massive numbers of users and more complicated geographical topologies to examine whether the Nets are capable of adapting.

In today's hugely populated wireless networks user assignment done by traditional methods can be very computationally expensive, especially when one considers that the movement of users in the network and their exiting or entering it will necessitate these calculations being ran many times a minute. Drastically reducing the time these require, even at some loss of total throughput, will possibly yield effectively better service and a greater effective throughput for the end user. This of course is hard to quantify in practice and beyond the abilities of the used simulation technique which assumes users that do not move in space. This necessitates the creation and implementations of even more holistic and realistic simulation tools that do everything the current ones do and in addition feature moving users, handshake protocols for when they move to different BSs and the whole spectrum of possibilities one might encounter in a real large scale wireless network.

Acknowledgment

This research has been co-financed by the Hellenic Foundation for Research & Innovation (H.F.R.I) through the H.F.R.I.'s Research Projects to Support Faculty Members & Researchers (project code: 02440).

References

[1] L. Zhao , H. Zhao, K. Zheng, and W. Xiang, 2018. "Massive MIMO in 5G networks: selected applications." Switzerland: Springer International Publishing.

[2] S. Shalev-Shwartz and S. Ben-David, 2014. "Understanding machine learning: From theory to algorithms." Cambridge University Press.

[3] D. Tzanakos, F. Allayiotis, V. Kokkinos and C. Bouras, "A mechanism for 5G MIMO performance optimization and evaluation," 2021 13th IFIP Wireless and Mobile Networking Conference (WMNC), 2021, pp. 48-54, doi: 10.23919/WMNC53478.2021.9619012.

[4] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore and X. Costa-Pérez, "A Machine Learning Approach to 5G Infrastructure Market Optimization," in IEEE Transactions on Mobile Computing, vol. 19, no. 3, pp. 498-512, 1 March 2020, doi: 10.1109/TMC.2019.2896950.deepMIMO dataset generator

[5] M. Guan, Z. Wu, Y. Cui, et al. "An intelligent wireless channel allocation in HAPS 5G communication system based on reinforcement learning." J Wireless Com Network 2019, 138 (2019). https://doi.org/10.1186/s13638-019-1463-8

[6] R. Dong, C. She, W. Hardjawana, Y. Li and B. Vucetic, "Deep Learning for Radio Resource Allocation With Diverse Quality-of-Service Requirements in 5G," in IEEE Transactions on Wireless Communications, vol. 20, no. 4, pp. 2309-2324, April 2021, doi: 10.1109/TWC.2020.3041319.

[7] A. Alkhateeb "DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications" arXiv preprint arXiv:1902.06435 Proc. of Information Theory and Applications Workshop (ITA), Feb., 2019.

[8] J. Schmidhuber "Deep learning in neural networks: An Overview", Neural Networks, vol 61, 2015, pp. 85-117

[9] M.F. Moller, 1993. "A scaled conjugate gradient algorithm for fast supervised learning." Neural Networks, 6 (4), p. 525-533. doi.org/10.1016/S0893 6080(05)80056-5

[10] H. Braun, M. Riedmiller, , Rprop: "A fast and robust backpropagation learning strategy." Proc. of the ACNN, 1993.