ITU-ML5G-PS-013: Improving the capacity of IEEE 802.11 WLANs through Machine Learning

Paola Soto[1,2], David Góez[2], Natalia Gaviria[2], and Miguel Camelo[1]

[1]University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science, Antwerp, Belgium
[2]Universidad de Antioquia, Department of Telecommunications Engineering, Medellín, Colombia

I. Introduction

According to Cisco, in the most recent Annual Internet Report [1], the global number of IEEE 802.11 (Wi-Fi) hotspots will suffer a fourfold increase by 2023, which means that from 169 million hotspots in 2018, we will have around 628 million public Wi-Fi hotspots by 2023. Moreover, by that date, 27.4% of Wireless Local Area Network (WLAN) endpoints will be equipped with 802.11ax (Wi-Fi 6), while 66.8% will be equipped with 802.11ac (Wi-Fi 5). Also, as in the last reports, the average number of devices and connections per household and capita increases even faster than the population and Internet users.

Part of the success in Wi-Fi adoption is the promised higher speeds and the improved average throughput per user. Channel Bonding (CB) is a technique introduced in 802.11n (Wi-Fi 4) and further developed in 802.11ac and 802.11ax that improves the channel capacity by bonding multiple frequency channels in a given transmission. Consequently, the throughput per user can be potentially improved as well as the spectrum usage efficiency.

In such highly dense scenarios of 2023, where the user density per m^2 is high, the number of deployed Access Points (APs) is also high to cover the demand. Therefore, to coordinate and optimize the spectrum access is of paramount importance. Specifically for CB, inter-AP interference mitigation plays a vital role as neighboring APs might overlap, avoiding other APs to access one or more frequency channels. This situation can cause drastic performance deterioration and increase the packet error rate [2].

One of the main findings from previous works [3], [4] is that the system performance can be benefited by having different CB policies that dynamically select the appropriate channel bandwidth based on current spectrum usage. By using analytical models, the previously mentioned works show that always selecting the widest available bandwidth is counterproductive in the long term. Moreover, given the problem's combinatorial nature, such analytical models are not scalable nor tractable for bigger instances of the problem.

To explore further solutions of the problem, the wireless networking research group of the Universitat Pompeu Fabra (UPF) proposes the application of Machine Learning (ML) to predict the throughput of dense deployments of WLANs. For achieving this, data extracted from simulated deployments is used. The generated data uses different random parameters, including channel allocation, location of nodes, and number of Stations (STAs) per AP. The performance prediction can be used to optimize the planning phase of a given deployment or improve the performance during the operation of a WLAN. All of this framed under the International Telecommunication Union (ITU) Artificial Intelligence (AI)/ML in Fifth Generation (5G) Challenge that seeks to solve relevant problems in 5G using AI/ML.

II. Motivation of the considered approach

Deep Learning (DL) approaches have taken advantage of the myriad of data sources to solve complex problems where analytical models and traditional ML can not scale or produce (near) optimal solutions. They have been proved to be successful in learning (in a supervised or unsupervised manner) from examples [5]–[7]. Such an abundance of data makes DL powerful on different aspects of network optimization as they avoid making *a priori* assumptions, typically found on analytical models. However, DLs face challenges when learning from structured data, as they require a simpler representation of these data. By structured data, we mean data that has clear and defined relationships within data-points. Those relationships are lost in getting a simpler representation, and the obtained result often lacks explainability.

Structured data can be represented as graphs, where the graph topology implicitly contains the relationships that we are mentioning. Recently, Graph Neural Networks (GNNs) [8] have been proposed as neural networks that operate on graphs intending to achieve relational reasoning and combinatorial generalization, i.e., to compose new tools from small building blocks. GNNs have also been successfully applied to combinatorial optimization problems [9], [10]. Therefore they are a reasonable choice for CB since CB is a problem with a combinatorial action space in dense deployments. Moreover, the relationships between STAs and APs (connectivity, interference, etc.) can be captured via a graph representation.

Specifically within the frame of the challenge, two properties of the generated dataset emphasize the use of GNNs. First, the data can be easily expressed as graphs. Second, approaches as Convolutional Neural Networks (CNNs) cannot be directly applied as each deployment presented a variable size of devices (APs are fixed per scenario, but the connected STAs are a random variable). On the contrary, GNNs exploits

the graphs' topological information, independently of how many nodes the graph has, by aggregating neighboring nodes' information.

Lastly, GNNs are proposed as a solution of multiple network optimization processes [11], [12] given its permutation equivariance property and their ability to generalize to large-scale problems.

III. Description of the proposed approach

The provided dataset for the challenge was generated by Komondor [13]. Komondor is a wireless network simulator for next-generation WLANs developed at UPF. The dataset consists of two types of files: the input node files, which contain the definition of the network deployment, and the output files, containing the simulator's output.

One of the first problems tackled during this challenge was selecting the relevant information for data-driven approaches. This information must be taken from both types of files and correlated with the objective of predicting the throughput of a given AP. Additionally, this data had to be appropriately formatted for later processing at the GNN approach. The next two subsections explain how we solve this issue and how the data is converted into a graph.

A. Definition of a graph

The basic entity in approaches like GNNs is a graph. A graph can be defined as a 2-tuple $G = (V, E)$ with V representing the set of nodes with cardinality N and E representing the set of edges with cardinality L. Each node has associated a set of features, represented by a matrix of the form $N \times D$ where D is the number of input features of each node. Similarly, each edge has associated a set of features represented by a matrix of the form $L \times R$ where R is the number of input features per edge. The graph's topology is represented by an adjacency matrix of the form $2 \times (s_k, d_k)_{k=1:L}$ where s_k is the index of the source node, and d_k is the destination node's index.

Regarding the dataset, each deployment is considered a directed graph where STAs and APs are the graph's nodes. Most node features are defined in the input node files, while others are defined in the output files. Although the organizers provided 23 node features in the input node files, at most 8 of those are relevant for data-driven approaches. Moreover, we defined two types of nodes present in the dataset, each having its related features; if the node is not related to a feature, that feature is set to zero for that node. Table I defines the node and edge features used for training.

On the other hand, the edges, in this case, are wireless links, which are defined based on the wireless interactions that are present in the dataset. As with the nodes, we also considered two types of edges based on the type of interaction, AP-AP interactions, represented in the interference map, and AP-STA interactions, represented in the Received Signal Strength Indicator (RSSI) values. We defined an additional feature based on the source-destination distance.

B. Model

Up to this date, there are different flavors of GNNs specific to the problems they aim to solve. However, there are only a few frameworks that try to integrate different models into one. Authors in [14] propose a framework where the basic computation unit is a Graph Network Block (GNB). A GNB contains three update functions and three aggregation functions where the computation is done from edge to nodes, to global parameters. So first, the edge's features are updated and aggregated into the node features, then the node features are updated having into account the vicinity within a predefined depth/range, and lastly, the global parameters are updated according to the state of the nodes. Moreover, various other architectures can be composed in terms of GNBs, depending on the available information as input to the functions.

The implementation of the GNBs is referenced as a MetaLayer in PyTorch Geometric [15], a geometric deep learning extension library for PyTorch. Figure 1 represents the architecture of one MetaLayer. We defined an edge model that uses two dense layers using REctified Linear Unit (ReLU) as an activation function in a typical MultiLayer Perceptron (MLP) configuration. A node model is also defined by two MLPs, one for aggregating the edge features into the node features and the second to update nodes' state based on their neighbors. Note that following the formal definition of a GNB in [14], global parameters were not used. Our implementation is available in GitHub[1].

IV. Results and Insights

This section summarizes the obtained results and insights during different stages of the first phase of the challenge. We also propose some improvements to the considered approach as future work at the end of the section.

A. Pre-processing

Even though the features of Table I are clearly defined, not all the features are used. For instance, we used the *WLAN Code* and the *Node Code* to relate APs and STAs belonging to the same WLAN. We also noticed that *Primary Channel, Minimum, and Maximum Channel Allowed* define a configuration of wireless channels. Therefore, these features are combined into one categorical variable representing the set of channels a given node uses. We defined six possible channel configurations within the dataset, as shown in Fig. 2.

A summary of the main findings during data exploration is given below.

- C4 and C5 configurations are mostly used across different training scenarios, as shown in Fig. 3.
- All STAs are within a range of $10\ \mathrm{m}$ of distance.
- The mean throughput is around $12.8\mathrm{Mbps}$. Therefore, the throughput of the APs is considered as outliers for a ML approach.

[1] https://github.com/kemets/ituml

TABLE I
NODE AND EDGE FEATURES USED FOR TRAINING.

Node Features			
Feature	**Definition**	**AP/STA feature**	**Type**
Node Type	Wireless node type, 0=AP, 1=STA	Both	Categorical
Node Code	Unique identifier for a node within a deployment	Both	Categorical
WLAN Code	Wireless network to which the wireless node is connected	Both	Categorical
x(m)	x-coordinate of the wireless node	Both	Scalar
y(m)	y-coordinate of the wireless node	Both	Scalar
Primary Channel	Primary wireless channel of the node	Both	Scalar
Minimum Allowed Channel	Minimum allowed wireless channel of the node	Both	Scalar
Maximum Allowed Channel	Maximun allowed wireless channel of the node	Both	Scalar
SINR	Signal to Interference plus Noise Ratio	STA	Scalar
Airtime	percentage of time each AP occupies each of the assigned channels	AP	Scalar
Edge Features			
Edge Type	Wireless edge type, 0=STA, 1=AP	Both	Categorical
Distance	Euclidean distance between source and destination	Both	Scalar
RSSI	Received Signal Strength Indicator	STA	Scalar
Interference map	Inter-AP interference sensed from evey AP	AP	Scalar

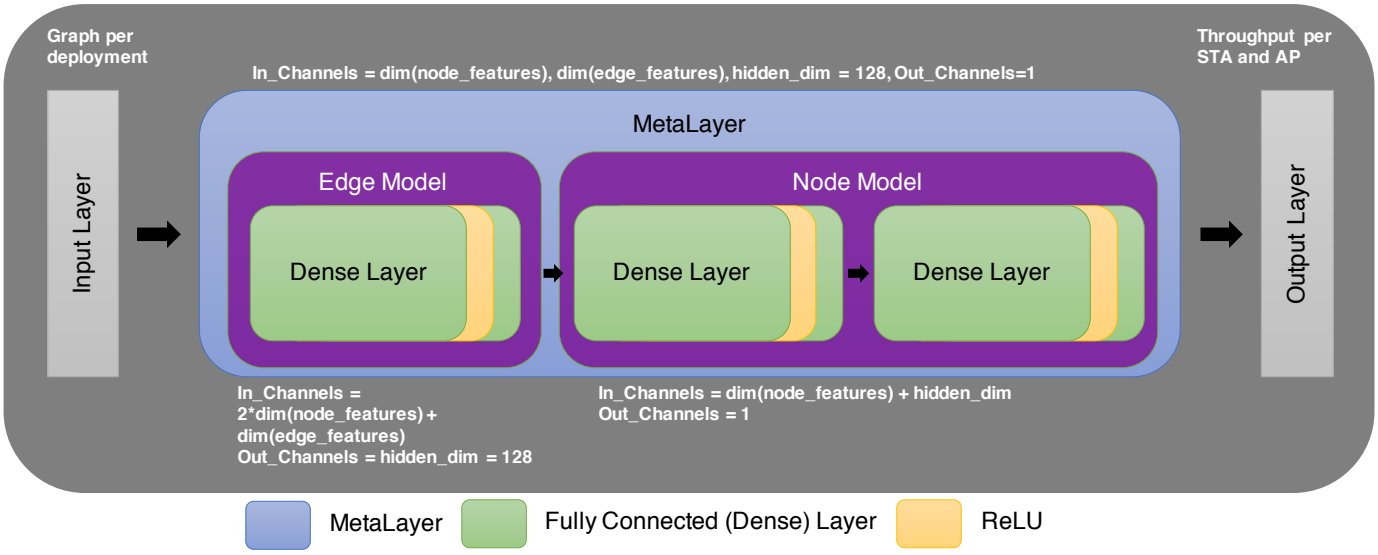

Fig. 1. Architecture of the MetaLayer

C0	**0**	1	2	3	4	5	6	7
C1	0	1	**2**	3	4	5	6	7
C2	0	1	2	3	**4**	5	6	7
C3	0	1	2	3	4	5	**6**	7
C4	**0**	1	2	3	4	5	6	7
C5	**0**	1	2	3	4	5	6	7

Fig. 2. Defined wireless channel configurations

- The different scenarios introduce a given amount of interference within WLAN, as shown in the spatial distribution in Fig. 4

B. Training

The model described in previous sections was trained on the provided dataset (80% for training and 20% for validation). Firstly, we defined the loss function as the Root Mean-Squared Error (RMSE) obtained across all the predictions compared to each type of deployment's actual result. For doing this, the throughput of APs and STAs is taken into account in the loss function.

However, initial results showed that the model mostly focused on correctly predicting the throughput of the APs, given that the error is minimized on large values. Therefore, we proposed a masked loss provided that APs throughput should be equal to the sum of the associated STAs throughput. In other words, the network does not need to learn to predict the AP throughput. We incorporate this into the training procedure

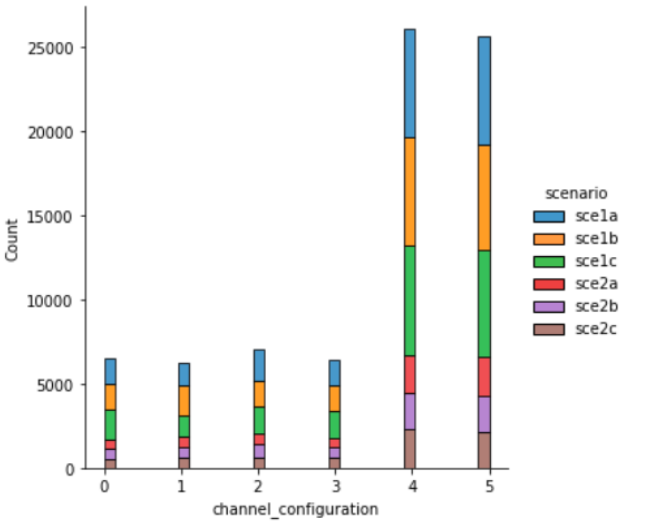

Fig. 3. Distribution of the channel configurations across the training scenarios

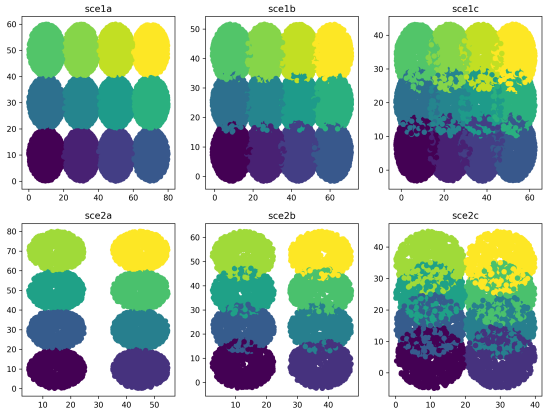

Fig. 4. Spatial distribution per training scenario

by masking the AP contributions to the loss function, i.e., only the STAs will contribute to the loss.

Fig. 5 shows the obtained results. The *train_loss* metric is the masked loss, while the *valid_loss* is the loss function when the model predicts both AP and STA throughput in the validation dataset. The *score* metric is calculated in a post-step as the sum of the predicted throughput of the associated STAs. As shown in the figure, the train, validation, and score loss rapidly decrease during the training loop's first epochs. After that, all the metrics slowly decrease for almost all the remaining epochs.

For the submission to the challenge, we selected the model with a lower *score* metric. The selected model obtained a score of 4.2 during validation.

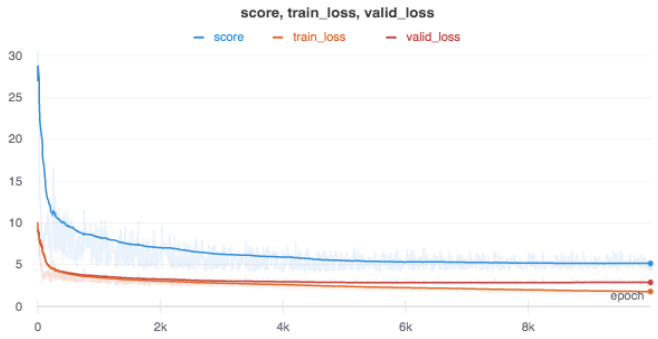

Fig. 5. Training loss

TABLE II
MEAN RMSE PER TEST SCENARIO

Scenario	Mean RMSE
test_sce1	26.97
test_sce2	18.94
test_sce3	12.76
test_sce4	8.73

C. Testing

The testing dataset introduced several changes regarding the training dataset, including map size, deployment size, and each deployment density. Our proposed model was compared with the correct values given by the simulator. Results in Table II show the mean RMSE across all the deployments of a given test scenario. As can be seen, our model performs better as the test scenario is more similar to the train scenario.

D. Problems and possible improvements

As shown in the last subsection, there is an indication that our model is over-fitting, which is reflected in the high Mean RMSE in *test_sce1* and *test_sce2*. In our case, we believe that over-fitting reduces the capabilities of our approach to generalize to unseen scenarios, e.g., *test_sce1* and *test_sce2* use a lower number of devices compared with the training data set. However, authors in [16] pointed out that Convolutional GNNs requires a higher amount of labeled data. Therefore, we propose data augmentation for tackling this issue.

Data augmentation can be considered by generating more deployments but also by performing variations on the provided dataset. For instance, by varying the position of the nodes of a deployment or by removing some of the WLANs in the deployments.

Another improvement direction is to consider convolutional layers within the model. Graph Convolutional Networks (GCNs) [17] are implemented in PyTorch. However, they are node-focused, which do not take into account the edge features. Fortunately, approaches like the one in [18] shed light on a self-made implementation.

V. CONCLUSION

Future wireless networks will be highly dense and will require higher performance than what we have today. Channel

Bonding (CB) is a technique that allows to bond two or more wireless channels for increasing the performance. However, inter-Access Point (AP) interference mitigation is one critical point in highly dense wireless networks as it has been found that CB might be counterproductive in such scenarios.

To explore different solutions to the problem, the wireless networking research group of the Universitat Pompeu Fabra (UPF) proposes the application of Machine Learning (ML) to predict the throughput that an AP would obtain in a dense deployment.

This document summarizes our participation in the ITU AI/ML in 5G challenge. We have provided the motivations for using Graph Neural Networks (GNNs) in graph-based problems that exhibit combinatorial behavior. Additionally, we showed the insights and the results of our model in different stages of the challenge, namely, pre-processing, training, and testing. According to the obtained results, we also propose some improvements that could be translated to a better prediction of the throughput.

Finally, we would like to thank the ITU and the UPF for organizing this exciting challenge that allowed us to gain hands-on experience on relevant topics to build and design future wireless technologies.

ACKNOWLEDGEMENT

Team ATARI wants to sincerely thank the meaningful contributions and insights of Kevin Mets and Prof. Steven Latré from the University of Antwerp and Prof. Juan Felipe Botero and Prof. Luis Alejandro Fletscher from the University of Antioquia.

REFERENCES

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," 2020. [Online]. Available: http://www.cisco.com/go/vni

[2] M. Park, "Ieee 802.11 ac: Dynamic bandwidth channel access," in *2011 IEEE international conference on communications (ICC)*. IEEE, 2011, pp. 1–5.

[3] S. Barrachina-Muñoz, F. Wilhelmi, and B. Bellalta, "To overlap or not to overlap: Enabling channel bonding in high-density wlans," *Computer Networks*, vol. 152, pp. 40–53, 2019.

[4] S.-s. Lee, T. Kim, S. Lee, K. Kim, Y. H. Kim, and N. Golmie, "Dynamic channel bonding algorithm for densely deployed 802.11 ac networks," *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8517–8531, 2019.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[9] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[10] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[11] H.-G. Kim, S. Park, S. Lange, D. Lee, D. Heo, H. Choi, J.-H. Yoo, and J. W.-K. Hong, "Graph neural network-based virtual network function management," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2020, pp. 13–18.

[12] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *arXiv preprint arXiv:2007.07632*, 2020.

[13] S. Barrachina-Munoz, F. Wilhelmi, I. Selinis, and B. Bellalta, "Komondor: A wireless network simulator for next-generation high-density wlans," in *2019 Wireless Days (WD)*. IEEE, 2019, pp. 1–8.

[14] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[15] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[16] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," *arXiv preprint arXiv:1801.07606*, 2018.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] Z. Chen. Graph convolutional networks for graphs with multi-dimensionally weighted edges. [Online]. Available: https://cims.nyu.edu/~chenzh/files/GCN_with_edge_weights.pdf