GROWS – Improving Decentralized Resource Allocation in Wireless Networks through Graph Neural Networks

Martín Randall
Facultad de Ingeniería, Universidad de la República
Austrian Institute of Technology
Uruguay/Austria
mrandall@fing.edu.uy

Pablo Belzarena
Facultad de Ingeniería, Universidad de la República
Uruguay
belza@fing.edu.uy

Federico Larroca
Facultad de Ingeniería, Universidad de la República
Uruguay
flarroca@fing.edu.uy

Pedro Casas
Austrian Institute of Technology
Austria
pedro.casas@ait.ac.at

ABSTRACT

Wireless networks have progressed exponentially over the last decade, and modern wireless networking is today a complex to manage tangle, serving an ever-growing number of end-devices through a plethora of technologies. The broad range of use cases supported by wireless networking requires the conception of smarter resource allocation approaches, which make the most of the scarce wireless resources. We address the problem of user association (UA) in wireless systems. We consider a particularly challenging setup for UA, represented by modern ad-hoc networks such as FANETS, where connectivity is provided by a group of unmanned aerial vehicles (UAVs). We introduce GROWS, a Deep Reinforcement Learning (DRL) driven approach to efficiently connect wireless users to the network, leveraging Graph Neural Networks (GNNs) to better model the function of expected rewards. While GROWS is not tied to any specific wireless technology, the decentralized nature of FANETS and the lack of a pre-existing infrastructure makes a perfect case study. We show that GROWS learns UA policies for FANETS which largely outperform currently used association heuristics, realizing up to 20% higher throughput utility while reducing user rejection by more than 90%, and that these policies are robust to concept drifts in the expected load of traffic, maintaining performance improvements for previously unseen traffic loads.

CCS CONCEPTS

• **Networks → Wireless access networks**; • **Computing methodologies → Reinforcement learning**.

KEYWORDS

User Association; Wireless Networks; FANETS; Graph Neural Networks; Deep Reinforcement Learning

1 INTRODUCTION

The exponential growth of wireless connectivity in the past decades has led to vast research in the general problem of Resource Allocation (RA) in wireless networks. Next Generation Wireless (NGW) networks would need to accommodate to the ever-growing number of connected devices (e.g., IoT) and the surge of novel, resource-demanding applications, motivating a renewed interest in finding better, more efficient User Association (UA) policies [10, 19]. While the focus of NGW networking research is mainly on mobile 5G and beyond networks, the surge of Unmanned Aerial Vehicles (UAV) in a broad variety of applications has brought back their usage as a complementary or substitute ad-hoc communication infrastructure, in the form of the so-called Flying Ad-hoc NETworks (FANETs) [18]. FANETs offer advanced features such as high mobility, fast deployment, self-configuration, low cost, scalability, and others, which make them a potential extension to the 5G network itself, specially to handle emergency situations and unplanned deployments. Within this context, the UA problem consists of the conception of association policies indicating the connectivity provider – e.g., UAVs, base stations – an incoming user should connect to so as to maximize a global system utility function, typically throughput related. The optimal association policy is usually intractable, and although simple heuristics yield good results for simple, low traffic scenarios, they are far from optimal under congestion.

The UA problem can be posed as a sequential decision-making one, hence we consider it from a Deep Reinforcement Learning (DRL) perspective. The RL paradigm is particularly suited for decision making problems, where an agent wants to find an optimal policy regarding some system reward. As the problem is NP-hard and the number of states quickly grows to become infeasible, we approximate the value function by a Graph Neural Network (GNN). Graphs portray a very expressive representation for systems composed of connected nodes, which are typical in networking. GNNs combine the expressiveness of graph representations with the learning ability of neural networks to advance learning-driven tasks. Due to their ability to exploit data structure and to generalize to unseen scenarios, GNNs have gained much attention in several fields (e.g., chemistry, natural language processing, networking). A particularly interesting property of GNNs for the proposed UA problem is their distributed nature: GNNs are amenable to a distributed implementation, turning them suitable for a variety of applications where central nodes cannot be relied on. For a FANET deployment, the distributed nature of GNNs is key to the algorithm's implementation. Indeed, the biggest advantage of graph-based learning for RA and UA tasks comes in scenarios where no central entity

may be assumed to exist, and base stations can only exchange information with their neighbors. Actually, all deployments where the infrastructure is provided by an ad-hoc network fall into this category. The main contribution of our work is on the modelling of the UA problem using graph representations and a reinforcement learning framework. We propose GROWS, a decentralized decision making framework in which agents are able to serve users through a general enough system-representation, which enables learning UA policies that properly perform in different use cases. In this paper we provide a series of synthetic results on UA for FANETS using GROWS, evidencing that smart and adaptable UA policies can perform significantly better than classical heuristics. In particular, we show that GROWS can serve up to 50% more users, achieving an increase on utility of over 20%. We additionally confirm the robustness and adaptive capabilities of GROWS, evaluating its functioning under varying traffic load.

2 GNN CONTEXT & RELATED WORK

2.1 Graph Neural Networks 101

GNNs are pretty novel as learning model. In a nutshell, a GNN consists of a cascade of layers, each of which applies a *graph filter* followed by an activation function. Consider that each node in the graph has an associated feature vector $\mathbf{x}_i \in \mathbb{R}^d$ (for $i = 1, \ldots, N$), which may regarded as the input features. Making the analogy to discrete-time convolution, a first-order convolutional layer for a GNN may be obtained as follows [16]:

$$\mathbf{x}'_i = \sigma\left(\mathbf{\Theta}^T \sum_{j \in \mathcal{N}_i \cup \{i\}} S_{j,i}\mathbf{x}_j\right), \tag{1}$$

where $\mathbf{x}'_i \in \mathbb{R}^{d'}$ is the output of the layer, $\sigma(\cdot)$ is a point-wise non-linearity (e.g. the ReLU function), $\mathbf{\Theta} \in \mathbb{R}^{d \times d'}$ is the learnable parameter of this layer, $\mathcal{N}_i$ is the set of neighbors of node i, and $S_{i,j}$ is the i, j entry of matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$, the so-called Graph Shift Operator (GSO). This is a matrix representation of the graph, which should respect its sparsity – i.e., $S_{i,j} \neq 0$ whenever there is an edge between nodes i and j. The adjacency matrix of the graph, its Laplacian, or their normalized versions are all valid GSOs.

Note that in (1), each node needs to linearly combine the vectors of its neighbors only. As we concatenate K such layers, the output of the GNN, which consists of the final vector representation of node i or its node *embedding*, depends on its neighbors up to K hops away. This observation implies that a GNN may be implemented in a fully-distributed way, as long as an edge in the graph means that the corresponding pair of nodes can communicate.

We may be more general and build higher-order filters. Let us stack all nodes' vectors $\mathbf{x}_i$ into matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, which is called a graph signal. The matrix product $\mathbf{SX} = \mathbf{Y}$ results in another graph signal, corresponding to the first-order convolution we used in (1), albeit without parameter $\mathbf{\Theta}$, which we will include shortly. By writing $\mathbf{S}^K\mathbf{X} = \mathbf{S}(\mathbf{S}^{K-1}\mathbf{X})$ we may see that this way we aggregate the information K hops away. Again, although it requires K rounds of information exchange, this operation may be performed without intervention of a central entity.

Finally, a general graph convolution is defined simply as a weighted sum of these K signals, i.e., $\sum_k \mathbf{S}^k\mathbf{X}h_k$, where scalars h_k are the taps of the filter. In this context, parameter $\mathbf{\Theta}$ in (1) is interpreted as a filter bank. That is to say, by considering a $d \times d'$ matrix $\mathbf{H}_k$ instead of the scalar taps, a single-layer GNN (or graph perceptron) is obtained by applying the pointwise non-linear function $\sigma(\cdot)$ to this convolution [8, 14]:

$$\mathbf{X}' = \sigma\left(\sum_{k=0}^{K-1} \mathbf{S}^k\mathbf{X}\mathbf{H}_k\right), \tag{2}$$

whereas a deep GNN is constructed by concatenating several perceptrons.

2.2 GNNs for User Association

Although GNNs are a recent paradigm [20, 24], they have already proved their usefulness to address diverse RA problems [13, 22, 23]. For complete reviews of GNN methods and applications, we refer the reader to [28, 32**?**]. Examples of GNNs applied to solving NP-hard or RA problems include combinatorial optimization [15], measurements [5], network virtualization [25], optical networks [1], and more.

Regarding UA for a UAV networking deployment, most of the existing work focus on the problem of where to position the UAVs to, for instance, maximize coverage [3, 29]. In our case study, we assume that this deployment phase has already taken place, and focus on how to associate users to each UAV base station. In this case, previous work proposes mostly a centralized scheme, or assumes complete knowledge of all nodes [12, 18, 21]. A recent approach to UAV deployment with GNNs [17] puts the emphasis on position optimization for interference mitigation. The closest work to ours [6] elaborates on a distributed approach, based on multi-agent learning. However, and different from GROWS, each base-station may only use a single subchannel per time-slot, meaning only one user is served, imposing a strong restriction in the practice.

3 THE GROWS ALGORITHM

3.1 System Model

Let us consider a set of N *Base Stations* (BSs) – we use the general term BS to refer to a connectivity provider, either a UAV in FANETS, a fixed BS in 5G, etc. Each BS has a limited set of frequency resources, referred to as resource blocks (RB). We assume that BSs have an internal assignation policy they follow for their associated users, such as distributing its available resources equally among connected users.

Like in many decision problems, we consider that time is slotted. At each time interval t a user may or may not arrive, following some probability distribution. Following the decentralization requirement, we look for a UA solution which considers only local k-hop information exchanges, and take into consideration the increase of graph complexity related to the augmentation of the action/state size. We therefore keep both state and actions simple – for scaling purposes, yet expressive enough to learn, specially to take full advantage of the graph representations. In the event of an arrival, one of the k-th BSs with strongest RSSI to the user, and with available resources, serves the newcomer.

Following a Reinforcement Learning (RL) formulation [26], we need to define the tuple formed by (s, a, T, r, γ). We refer to Figure 1 for a diagram visualization of the different components within

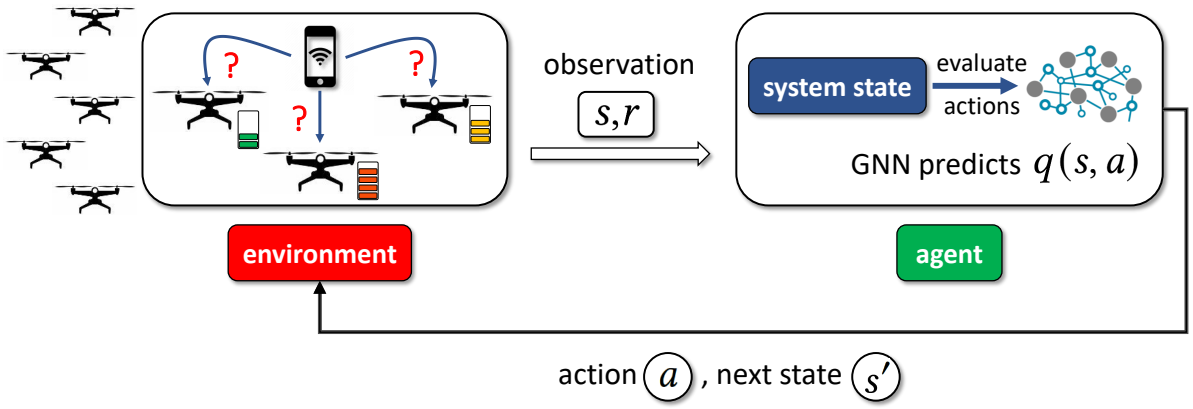


Figure 1: System model in GROWS. We consider at most one arrival at each time step. The combination of past decisions and the present arrival constitutes the state of the system. The choice of which BS associates with the currently arrived user is the action. After executing an action, a new state is observed and a reward is obtained.

the RL formulation in GROWS.

① **The system's state** s is defined as the aggregation of the BS's states and the user's state. For each BS, the state is composed of a representation of the present system's state – number of associated users and mean utility, and the new user's characteristics – RSSI to the BSs and traffic demand.

② **The action** a consists of selecting one of the BSs. Note that not every time step involves an action; they may or may not occur. To have a well defined Markov process, we include the decision-making in the state, setting the demand to 0 for the time steps when no user arrives. In this case there is no action to be taken, only updating the system's state.

③ **Transitions** T occur as depicted in Figure 1: the BS's descriptors are updated to include the action effect – a +1 increase on the number of users associated and a new mean utility, and the time effect – a -1 decrease on the number of users associated if a user's demand has been satisfied. Each time step updates the new user characteristics in the event of a new arrival. Transitions are deterministic over the BS's features given the action a and the state s, but stochastic for the new user features.

④ **The reward** r is defined as the instantaneous utility of executing an action for a given state. Following the literature [2, 4], and as a means to promote fairness in the distribution of resources, we take as utility/reward the log-sum of the throughput over the users, $r = \sum_N \sum_i \log(1 + th(u_{i,n}))$, where $u_{i,n}$ represents the i-th user associated with BS n.

In RL formulations, the **discount factor** γ is defined as a means to estimate the expected discounted cumulative reward, which is the value the RL algorithm tries to maximize:

$$R_t = \Sigma_t^{\infty} \gamma^t r(t)$$

The expected discounted cumulative reward is optimized by updating a policy (π) through one of Bellman's equations. In GROWS we take the action-value function for policy π, defined as [26]:

$$q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\Sigma_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right],$$

and use the update rule given by the optimality equation for the state-action value function:

$$q^{\pi}(s,a) = r + \gamma q^{\pi}(s', \pi(s'))$$

To form the graph representation, we take at each decision time a sub-graph of the FANET's interconnected UAVs, where nodes correspond to those BSs having (i) a signal strength with the newly arrived user above a certain threshold (to allow potential association), and (ii) available resources to serve the user, and edges represent their inter-connections. The state of each node is given by the state of its related BS, and the user's characteristics with respect to that BS.

3.2 GROWS Algorithm Design

The GROWS algorithm is based on the classic Double DQN reinforcement learning algorithm [11], and its integration with the GNN is inspired on previous work [1, 15]. The goal of the GNN is to learn how to best approximate the q-function, an estimation of the value-action function for the RL problem. The *actor-critic* role in RL is to stabilize convergence of the algorithm. It is important to notice that training and execution are done separately: once the GNN is trained, prediction of the q-function according to the state and possible actions is done instantly.

We take a *LocalGNN* [9] as the GNN model, which represents an implementation of the popular Graph Convolutional Network (GCN) introduced in section 2. An important virtue of the GCN is that information aggregation is local for each node – and extended to the K-hops neighborhood, which means that the GCN output for each node can run locally, enabling scalability for the proposed algorithm.

4 UA IN FANETS WITH GROWS

We consider a wireless system in which users arrive and have to connect to a certain BS. As we want to evaluate GROWS behavior and performance in a FANETs deployment use case, we build a synthetic scenario for evaluations. As explained before, the goal is to learn a UA policy which maximizes the log-sum of the users'

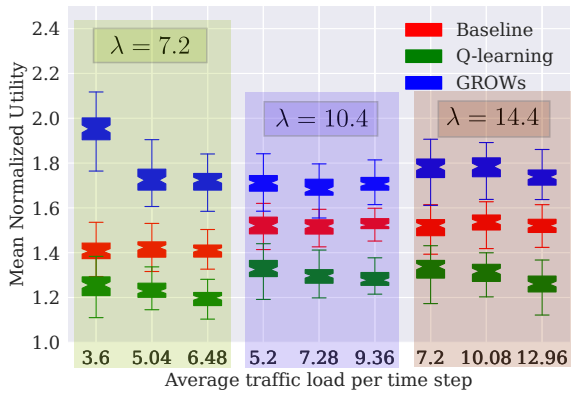


Figure 2: Mean normalized utility per episode over different expected traffic loads and arrival rates. GROWS clearly outperforms both the baseline and the traditional q-learning algorithm.

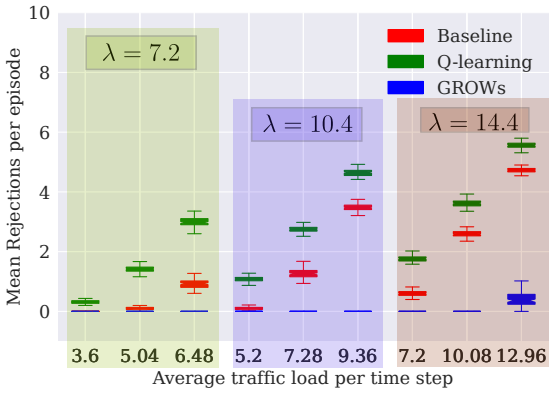


Figure 3: Mean rejections per episode for different traffic loads. For the same expected traffic demand λ, user rejections increase with higher user arrivals (0.5, 0.7 and 0.9), except for GROWS, which is able to accept almost all users at every episode.

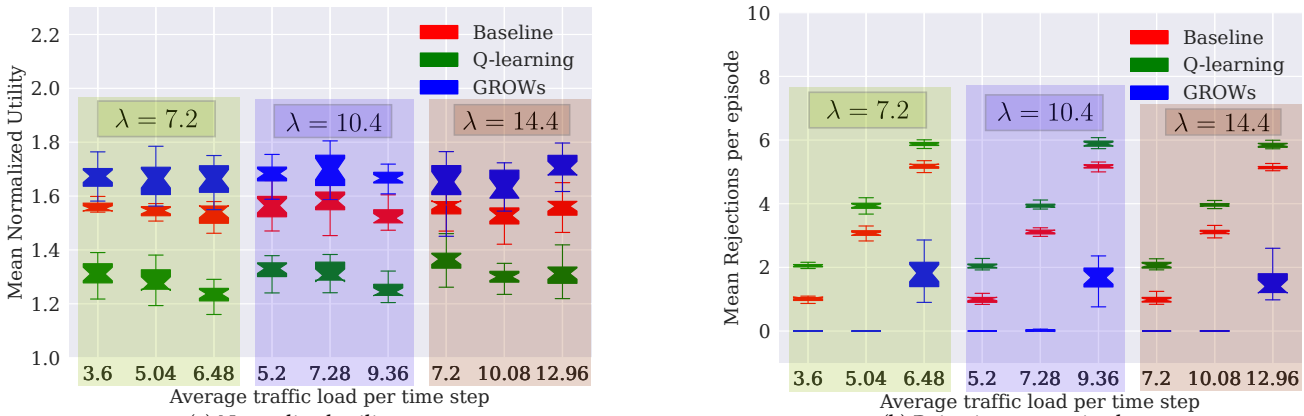
throughput, potentially improving the overall system utility as compared to certain baseline policy. As stated, the optimal policy is intractable: the problem is NP-hard and the number of states grows too fast to run a search over all possible decisions. For this reason, UA in current mobile networks is generally done through a simple heuristic, consisting of selecting the BS with the highest signal strength – we refer to this UA policy as an *argmax* policy. Even if simple, this strategy achieves reasonably good results, and is usually considered among baselines in previous work [6, 27, 30, 31]. We compare GROWS against this simple *argmax* policy, representing the currently followed strategy – we refer to it as the *Baseline*. We note that this Baseline provides very close to optimal results in non-congested scenarios, but fails in more stringent scenarios where users are massed around one of the base stations.

We also compare GROWS against a pure RL approach, based on *q-learning*. Naturally, this comparison can only be done for small scenarios, as the q-learning approach rapidly becomes infeasible due to the explosion in the number of states, which can only be handled by DRL. Still, the comparison against q-learning serves to evidence the advantage of using a deep model to estimate the q action-value function.

Regarding parametrization of the experiments and testing conditions, the following list describes the different components. As an additional contribution, and for the sake of reproducibility, we share GROWS implementation, see https://gitlab.fing.edu.uy/mrandall/grows. We consider four BSs randomly located on a plane, with the condition of being each of them connected to at least another BS – i.e., all UAVs have at least one connection to another UAV. We estimate the RSSI to each other and to arriving users by using the standard Friis equation with shadowing (setting the loss coefficient in 3). We consider the shared frequencies and modulations of 4G and 5G, meaning a 20MHz bandwidth with 15KHz subcarriers over a 3GHz carrier, without MIMO and with FDD. Possible modulations are selected depending on the RSSI: QPSK (-85 to -95 dB), 16-QAM (-75 to -85 dB), 64-QAM (-65 to -75 dB), and 256-QAM (more than -65 dB). At each step t, users arrive with probability p, which can vary in $[0.5, 0.7, 0.9]$. Users are uniformly located in the plane, with the condition to have at least one valid signal with a BS – e.g., at least QPSK with a BS. Users have three possible down link demand values to be satisfied (for the sake of simplicity, we take units in bits and not bytes): *low* (2 Mbits), *medium* (10 Mbits), and *high* (20 Mbits). These demand values are taken following a probability distribution, with which we can impose either heavier or lighter demands. We call λ the expected demand of a user arriving, which can vary in $[7.2, 10.4, 14.4]$. The user's throughput is determined by using the Friis equation as stated above, finding the user's RSSI with the BS and then using the best suitable modulation.

Episodes consist of 100 time steps, and we use an epsilon-greedy exploration/exploitation policy. We do exploration during the first 15,000 episodes, and an aggressive exploitation on the last 5,000 episodes, following an acute exponential decay. Hyper-parameters are calibrated through grid search, and we found that a learning rate of 5.10^{-4} and batch size of 32 are suitable. We take $\gamma = 0.5$ as discount factor.

Ⓐ **GROWS performance for different traffic loads:** to study the functioning of GROWS under different traffic loads, we compare results for the three different increasing arrival rates – $p = [0.5, 0.7, 0.9]$, and for the three different demand probability distributions – $\lambda = [7.2, 10.4, 14.4]$. Figure 2 reports the obtained results in terms of utility for nine different UA policies, learned for different *traffic loads* – the x-axis corresponds to $p \times \lambda$, where the cumulative reward is averaged every 50 episodes over the 5,000 exploitation episodes – resulting in 100 values, reported as box-plots. For the sake of better visualization and interpretation of results, all utility results are normalized to a random UA policy, where users are assigned to BSs in a random manner. This means that a value of 1 in the normalized utility is equivalent to a random UA policy. Figure 3 additionally reports the mean number of rejections per episode. GROWS is not only able to improve utility as compared to the other strategies, but most importantly, it does so while realizing a close to null rejection rate, even in highly congested scenarios.



(a) Normalized utility.

(b) Rejections per episode.

Figure 4: UA performance for unseen scenarios with higher traffic load, taking $\lambda = 16.2$. Note that the average traffic load values (i.e., the x-axis) correspond to the traffic loads used during training – i.e., $p = [0.5, 0.7, 0.9]$ and $\lambda = [7.2, 10.4, 14.4]$. The actually tested traffic loads correspond to $p = [0.5, 0.7, 0.9]$ and $\lambda = 16.2$. GROWS outperforms the other strategies in utility, accepting almost every user (50/50, 70/70, 88/90).

The steep curve for each λ value follows the user arrival rate increase: with $p = [0.5, 0.7, 0.9]$ we have a mean number of arrivals of $[50, 70, 90]$ per episode. GROWS realizes an increase of 10% to 20% over the baseline utility, accepting almost all users – less than one user-rejection per episode, while the baseline has rejection rates over 5% of the total users.

Figure 2 also reveals that utility is mostly dependent on the expected traffic load λ. Once users arrive and get connected to a BS, utility remains rather stable, but as traffic load increases, connections last longer and new arrivals are rejected if no resources are available. Following this reasoning, arrival rates p will have a great impact in the number of rejections per episodes, which is clearly observed in Figure 3. Indeed, for the same expected load, the rejection rate increases directly with the increase of arrival rates. GROWS UA policy counterbalances this performance degradation through a smarter assignment of available resources, resulting in an almost negligible user rejection rate.

Ⓑ GROWS performance for unseen traffic loads: an important characteristic for an AI/ML driven system is being able to cope with co-called Concept Drifts (CDs) in the analyzed data - i.e., modifications in the properties of the underlying probability distribution. A CD can manifest itself as a shift in the mean, an increase or decrease in the variance, or even as complete data modifications. For the specific problem of UA through learning, a desirable property is being able to handle a significantly higher load of traffic and users, not seen at training time, where the standard heuristics such as *argmax* would saturate (cf. Figure 3). For example, traffic load could surge in the event of flash-crowds, during emergency situations, or even due to major social events requiring higher connectivity and resources – e.g., the traffic load generated at the Super Bowl [7].

To analyze how GROWS behaves in the event of a strong surge of traffic load, we explore its application to higher traffic loads not observed at training time. More specifically, we take the nine UA policies learned for the different values of p and λ (cf. Figures 2 and 3), and apply them to a significantly higher user demand, taking $\lambda = 16.2$. For reference, this value represents a traffic demand surge of $\times 2.25$, $\times 1.56$, and $\times 1.13$ for the UA policies learned with the original λ values (7.2, 10.4, and 14.4). Figure 4 reports the obtained results in terms of (a) utility and (b) rejections – note that the x-axis indicates the traffic loads used in training – i.e., $p = [0.5, 0.7, 0.9]$ and $\lambda = [7.2, 10.4, 14.4]$, whereas the actually tested traffic loads correspond to $p = [0.5, 0.7, 0.9]$ and $\lambda = 16.2$.

GROWS is able to properly adapt to the new unseen traffic load, maintaining utility close to previous results and clearly outperforming the other strategies. Most importantly, in the event of a surge in traffic load, GROWS rejection rates are still negligible, whereas they significantly increase for both *argmax* and *q-learning*. We conclude that GROWS is able to avoid bottlenecks, even when confronted to unexpectedly high traffic loads, which certainly represents the most desirable feature for any RA/UA strategy. Finally, results bring to light the good generalization properties of the GNN model, which can correctly approximate the q-value function over unseen states.

5 CONCLUSIONS

We have presented our work in the field of user association, an open topic with renewed interest due to the needs of 5G and beyond technologies, as well as emerging wireless networks such as FANETs. We proposed a deep reinforcement learning formulation of the UA problem, using a GNN to estimate the q-value function, which gives birth to the GROWS algorithm: a decentralized and scalable solution for user association in wireless systems. We have presented results for several synthetic scenarios simulating FANETS wireless networks, where GROWS outperforms nowadays policies as well as other q-learning approaches, realizing significantly higher throughput utility while drastically reducing user rejection. We have presented initial evidence suggesting that GROWS can generalize to unseen scenarios and adapt to traffic variations, keeping a high throughput utility as well as low user rejection rates, even

when confronted to unexpectedly high surges in traffic demand. The integration of users' mobility and handovers into GROWS is part of our ongoing work. Finally, while the focus of GROWS is into UA and wireless networks, the proposed concept and system are applicable to other RA problems in different domains, suggesting further research leveraging DRL and GNNs for better and more efficient usage of resources.

ACKNOWLEDGEMENTS

This work is partially funded by the Agencia Nacional de Investigación e Innovación (ANII) project *Artificial Intelligence for 5G Networks* (FMV_1_2019_1_155700), as well as supported by the Austrian FFG ICT-of-the-Future DynAISEC project 887504 (*Adaptive AI/ML for Dynamic Cybersecurity Systems*). Martín Randall's PhD is supported by a scholarship granted by the Agencia Nacional de Investigación e Innovación (ANII).

REFERENCES

[1] Paul Almasan et al. 2019. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *arXiv preprint arXiv:1910.07421* (2019).
[2] T. Bu, L. Li, and R. Ramjee. [n. d.]. Generalized Proportional Fair Scheduling in Third Generation Wireless Data Networks. In *INFOCOM 2006.*
[3] Ursula Challita et al. 2019. Interference Management for Cellular-Connected UAVs: A Deep Reinforcement Learning Approach. *IEEE Transactions on Wireless Communications* (2019).
[4] Jingdi Chen et al. 2021. Bringing Fairness to Actor-Critic Reinforcement Learning for Network Utility Optimization. In *INFOCOM 2021.*
[5] Miles Cranmer et al. 2021. Unsupervised resource allocation with graph neural networks. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning.*
[6] Jingjing Cui et al. 2020. Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks. *IEEE Transactions on Wireless Communications* (2020).
[7] Jeffrey Erman et al. 2013. Understanding the Super Sized Traffic of the Super Bowl. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13).*
[8] Fernando Gama et al. 2019. Convolutional Neural Network Architectures for Signals Supported on Graphs. *IEEE Transactions on Signal Processing* (2019).
[9] Fernando Gama et al. 2019. Convolutional Neural Network Architectures for Signals Supported on Graphs. *IEEE Transactions on Signal Processing* (2019).
[10] Xiaohu Ge et al. 2014. 5G wireless backhaul networks: challenges and research advances. *IEEE network* (2014).
[11] Hado Hasselt. 2010. Double Q-learning. *Advances in neural information processing systems* 23 (2010).
[12] Samira Hayat et al. 2016. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys & Tutorials* (2016).
[13] Ziyan He et al. 2020. Resource allocation based on graph neural networks in vehicular communications. In *GLOBECOM.*
[14] Elvin Isufi et al. 2021. EdgeNets:Edge Varying Graph Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
[15] Elias Khalil et al. 2017. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems* (2017).
[16] Thomas N. Kipf et al. 2017. Semi-supervised classification with graph convolutional networks. *5th ICLR* (2017).
[17] Pei Li et al. 2022. Graph neural network-based scheduling for multi-UAV-enabled communications in D2D networks. *Digital Communications and Networks* (2022).
[18] Mohammad Mozaffari et al. 2019. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Communications Surveys & Tutorials* (2019).
[19] Hawar Ramazanali et al. 2016. Survey of user association in 5G HetNets. In *2016 8th IEEE LATINCOM.*
[20] Franco Scarselli et al. 2008. The graph neural network model. *IEEE transactions on neural networks* (2008).
[21] Hazim Shakhatreh et al. 2019. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* (2019).
[22] Yifei Shen et al. 2019. A graph neural network approach for scalable wireless power control. In *2019 IEEE Globecom Workshops (GC Wkshps).*
[23] Yifei Shen et al. 2020. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis. *IEEE Journal on Selected Areas in Communications* (2020).
[24] José Suárez-Varela et al. 2021. Graph Neural Networks for Communication Networks: Context, Use Cases and Opportunities. *arXiv preprint arXiv:2112.14792* (2021).
[25] Penghao Sun et al. 2020. Combining deep reinforcement learning with graph neural networks for optimal VNF placement. *IEEE Communications Letters* (2020).
[26] Richard S Sutton et al. 2018. *Reinforcement learning: An introduction.* MIT press.
[27] Ning Wang et al. 2016. Joint downlink cell association and bandwidth allocation for wireless backhauling in two-tier HetNets with large-scale antenna arrays. *IEEE Transactions on Wireless Communications* (2016).
[28] Zonghan Wu et al. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).
[29] Qianqian Zhang et al. 2018. Machine Learning for Predictive On-Demand Deployment of Uavs for Wireless Communications. In *GLOBECOM.*
[30] Yalin Zhang et al. 2020. Deep learning based user association in heterogeneous wireless networks. *IEEE Access* (2020).
[31] Nan Zhao et al. 2019. Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks. *IEEE Transactions on Wireless Communications* (2019).
[32] Jie Zhou et al. 2020. Graph neural networks: A review of methods and applications. *AI Open* (2020).