DELAY-ORIENTED DISTRIBUTED SCHEDULING USING GRAPH NEURAL NETWORKS

Zhongyuan Zhao⋆, Gunjan Verma†, Ananthram Swami†, and Santiago Segarra⋆

⋆Rice University, USA   †US Army's DEVCOM Army Research Laboratory, USA

ABSTRACT

In wireless multi-hop networks, delay is an important metric for many applications. However, the max-weight scheduling algorithms in the literature typically focus on instantaneous optimality, in which the schedule is selected by solving a maximum weighted independent set (MWIS) problem on the interference graph at each time slot. These myopic policies perform poorly in delay-oriented scheduling, in which the dependency between the current backlogs of the network and the schedule of the previous time slot needs to be considered. To address this issue, we propose a delay-oriented distributed scheduler based on graph convolutional networks (GCNs). In a nutshell, a trainable GCN module generates node embeddings that capture the network topology as well as multi-step lookahead backlogs, before calling a distributed greedy MWIS solver. In small- to medium-sized wireless networks with heterogeneous transmit power, where a few central links have many interfering neighbors, our proposed distributed scheduler can outperform the myopic schedulers based on greedy and instantaneously optimal MWIS solvers, with good generalizability across graph models and minimal increase in communication complexity.

Index Terms— Maximum weighted independent set, graph neural networks, distributed scheduling, latency.

1. INTRODUCTION

Wireless multi-hop networks are fundamental to modern wireless communications, including military communications, wireless backhaul for 5G and beyond, and Internet of Things (IoT) [1–4]. One challenge of wireless multi-hop networks is distributed resource allocation, such as link scheduling, without the help of infrastructure. Specifically, link scheduling determines which links should transmit and when should they transmit, along with other relevant parameters [5, 6]. In this paper, we focus on link scheduling in wireless networks with time-slotted orthogonal multiple access, in which a time slot comprises a scheduling phase followed by a transmission phase [7, 8]. The optimal scheduling problem in wireless multi-hop networks is typically formulated as solving a maximum weighted independent set (MWIS) problem on a *conflict graph* [5–19], in which a vertex represents a link in the wireless network, an edge captures the interference relationship between two links, and the vertex weight is the utility of the corresponding link. The scheduling scheme contains two main parts: 1) a per-link utility function to
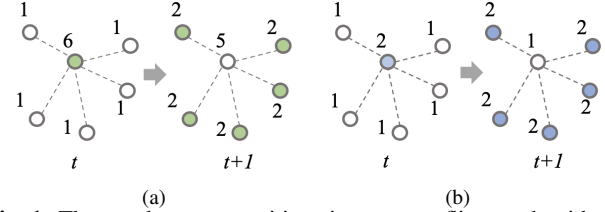
Fig. 1: The steady-state transitions in a toy conflict graph with star topology, constant arrival rate of 1, link rate of 2, utility function given by the queue length, and initial state of all empty queues. The colored nodes are scheduled by (a) an optimal MWIS scheduler and (b) a greedy scheduler. The latter has shorter average delay.

evaluate the importance of a link relative to the scheduling objective, and 2) an approximate (and possibly distributed) solver for the associated MWIS problem, which is known to be NP-hard [16, 17].

Although the MWIS-formulated schedulers seek to maximize the total throughput or utility [5–11], they usually have poor delay performance [20]. Indeed, the memoryless nature of the queue-based utility functions [5–11] makes it difficult to optimize network metrics that depend on serial decisions such as delay. As an example, consider a conflict graph with star topology for a wireless network with 6 links (represented as the vertices of the graph) as in Fig. 1. Furthermore, consider an initial state of all empty queues, a constant arrival rate of 1 and a link rate of 2 (both in packets per time slot) for each link, and set the per-link utility as the queue length so that links with longer queues are preferentially scheduled. Since the star has two maximal independent sets (one is the central node, the second is all peripheral nodes), the optimal MWIS scheduler will schedule the set that achieves the maximum sum of queue lengths, leading to the alternation of two network states as in Fig. 1(a). The greedy scheduler builds the schedule iteratively by adding one node at a time starting with the one with the largest queue and subsequently remove its neighbors to avoid collisions, leading to the alternation of the two network states in Fig. 1(b). The example reveals that the average queue length under the optimal MWIS scheduling is $\bar{q}_{opt} = 2.17$, which is greater than that of the greedy scheduler, $\bar{q}_{Gr} = 1.5$. Consequently, for unsaturated network traffic, the optimal MWIS solver has poorer delay performance than a simple greedy heuristic.

Existing approaches to reduce the delay of scheduling include setting delay as a constraint of optimization [21, 22], replacing the queue length in the conventional utility functions [5–11] with metrics of delay (e.g., virtual queues of congestion [20], sojourn time [23], age-of-information [24]), and machine learning-based solutions [25–28]. Conventional approaches [20–24] generally do not fully consider network topology. In contrast, machine learning-based approaches incorporate topological information as part of the network state either implicitly through, e.g., a multi-layer perceptron (MLP) tied to a specific topology [27], or explicitly through graph embedding [26] and graph neural networks [28]. Schedules are generated either by algorithmic frameworks based on the outputs of neural networks, such as utility [25] and bias [27], or directly

by MLP-based binary classifiers [26]. However, these machine learning-based approaches are either limited to centralized scheduling [25, 26], continuous-valued problems [28] (link scheduling is discrete-valued), or have poor scalability [27].

In this paper, we depart from existing approaches and propose a scheme based on graph convolutional networks (GCNs) [29] and inspired by deep Q learning (DQL), denoted as GCN-DQL, which can improve the delay performance of wireless networks by generating per-link utilities that are aware of the network state and topology. Our scheduler is composed of a GCN followed by the application of a non-differentiable distributed local greedy solver (LGS) [10], similar to the architecture adopted in [18, 19]. However, we propose a different training approach to capture future returns of the trained scheduling policy. More specifically, we introduce a reward signal to evaluate the relative performance of the trained policy compared to that of the vanilla LGS in a K-step lookahead scheduling based on the current state (which includes the backlogs and capacities of links) and topology of the wireless network. Although our method relies on centralized training, it can be deployed in a fully distributed manner thanks to the distributed nature of the GCN and LGS.

Contribution. The contributions of this paper are twofold: 1) we propose the first GCN-based distributed scheduler with temporal lookahead capabilities, and 2) through numerical experiments, we demonstrate the superior performance of the proposed method as well as its generalizability over different topologies.

2. SYSTEM MODEL AND PROBLEM STATEMENT

Consider a wireless multi-hop network, where an (undirected) link (i, j) implies that user i and user j can communicate with each other. A flow describes the stream of packets from a source node to a destination node. A flow may pass through multiple links determined by a routing scheme. In each link, there is a queuing system q for packets of all the flows as well as exogenous arrivals. We denote by $q(v)$ the sum of bidirectional queues for a link v.

The scheduling algorithm works on the *conflict graph*, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, which is defined as follows: a vertex $v \in \mathcal{V}$ represents a link in the wireless network, and the presence of an undirected edge $e = (v_a, v_b) \in \mathcal{E}$ captures the interference relationship between links $v_a, v_b \in \mathcal{V}$. The interference relationship in the system is considered to follow a physical distance model [17]. Two links interfere with each other if their incident users are within a certain distance such that their simultaneous transmission will cause the outage probability to exceed a prescribed level, or they share the same user with only one radio interface. For the rest of this paper, we focus on the conflict graph $\mathcal{G}$, which we assume to be known; see, e.g., [30] for its estimation. In principle, the interference zone of each link (hence $\mathcal{G}$) depends on the transmit power and antenna directivity of the corresponding users. To simplify the analysis and avoid this dependence, we consider the scenario in which all the users transmit at power levels that do not vary with time.

An independent (vertex) set in a graph is a set of nodes such that no two nodes in the set are neighbors of each other. From the definition of $\mathcal{G}$, only wireless links that form independent sets in $\mathcal{G}$ can communicate simultaneously in time and frequency under orthogonal access. Our link scheduling aims to minimize the average communication delays over a long time horizon. More precisely, we describe the *network state* at time t by the tuple $(\mathcal{G}(t), \mathbf{q}(t), \mathbf{r}(t))$ consisting of the conflict graph $\mathcal{G}(t)$ (potentially changing over time), queue lengths $\mathbf{q}(t)$, and link rates $\mathbf{r}(t)$. If we denote by $\mathcal{C}$ the space of all functions that go from network states into independent sets of

the graph, we can formally define our problem as follows.

Problem 1. *For a time horizon of interest T, we want to solve for the delay-optimal scheduler given by*

$$c^* = \underset{c \in \mathcal{C}}{\operatorname{argmin}} \ \mathbb{E} \left(\frac{1}{T+1} \sum_{t=0}^{T} \frac{\|\mathbf{q}(t)\|_1}{|\mathcal{V}(t)|} \right) \tag{1a}$$

$$s.t. \quad \hat{\boldsymbol{v}}(t) = c(\mathcal{G}(t), \mathbf{q}(t), \mathbf{r}(t)), \tag{1b}$$

$$q_v(t+1) = \begin{cases} q_v(t) + a_v(t) & \text{if } v \notin \hat{\boldsymbol{v}}(t), \\ q_v(t) + a_v(t) - \min(r_v(t), q_v(t)) & \text{if } v \in \hat{\boldsymbol{v}}(t), \end{cases} \tag{1c}$$

where both constraints hold for every time $t = 0, \dots, T$ and the second constraint holds for all $v \in \mathcal{V}$.

To better understand Problem 1, first notice that in constraint (1b) we are defining the set of vertices $\hat{\boldsymbol{v}}(t)$ to be scheduled at every time t. Since $c \in \mathcal{C}$, these sets of vertices are guaranteed to be independent sets, thus, feasible scheduling choices. Constraint (1c) updates the queues at every vertex accordingly. To be precise, if a node v has not been scheduled, its queue at the next time point is given by the previous queue plus any arrivals $a_v(t)$. On the other hand, if a node was scheduled then we need to subtract the packets that were sent, which equals the minimum between the queue length at that vertex and the rate achievable. Among all possible scheduling functions in $\mathcal{C}$ we seek to find the one that minimizes the objective in (1a), which computes the average (over time and over vertices) queue length. Notice that the queue lengths ultimately depend on the (unknown) link rates and arrival rates. Thus, we model queues as random variables and we seek to minimize their expected value with respect to the random distributions of arrivals and link rates. It should be noted that Problem 1 is exclusively focused on the optimal scheduler (implemented at the link layer) and, thus, we have no decision over the arrival rates (possible defined by a routing algorithm implemented at the network layer).

Finding an exact solution to Problem 1 is extremely challenging. Notice that even in the single-step case ($T = 0$) and in the absence of any randomness, selecting the optimal (weighted) independent set is known to be NP-hard [16, 17]. Moreover, the optimization in (1) is more challenging than a vanilla MWIS problem for at least three reasons: i) We are optimizing over the set of functions $\mathcal{C}$ that goes from network states to independent sets, ii) Our objective depends on random arrivals and link rates, and iii) Our objective depends on multiple scheduling instances. In the next section we present our solution to Problem 1, which addresses the aforementioned challenges.

3. DELAY-ORIENTED SCHEDULING WITH GCN

To address the three challenges in solving Problem 1, we respectively employ three strategies: i) Defining a parameterization of a subspace of $\mathcal{C}$ and optimizing over those parameters, ii) Training from collected historic data whose arrivals and link rates follow the distributions of interest, and iii) Incorporating a lookahead reward in our training that penalizes greedy scheduling choices.

To ensure that the output of our scheduler is a valid independent set, our approach consists of two steps: a trainable GCN [29] that learns per-link utilities $\mathbf{u}_{\text{GCN}}$ followed by the application of classical LGS [10]. More precisely, if we omit t for notational simplicity and denote by $\hat{v} = \text{LGS}(\mathcal{G}, \mathbf{u})$ the nodes selected by LGS for a vector $\mathbf{u}$ of per-link utilities, then our solution to Problem 1 is of the form

$$\hat{\boldsymbol{v}}_{\text{GCN}} = c(\mathcal{G}, \mathbf{q}, \mathbf{r}) = \text{LGS}(\mathcal{G}, \Psi_{\mathcal{G}}(\mathbf{S}; \boldsymbol{\omega})). \tag{2}$$
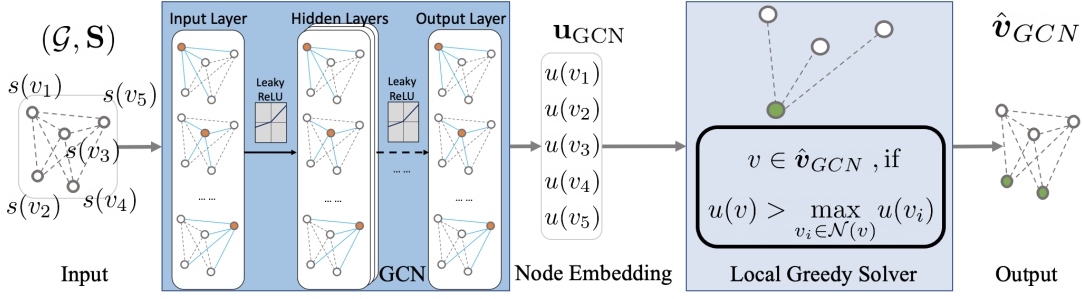
Fig. 2: Architecture of GCN-based distributed scheduler [18, 19]. First, the conflict graph $\mathcal{G}$ and node features $\mathbf{S}$ are encoded into the scalar embeddings $\mathbf{u}_{\text{GCN}}$ via a GCN. Then, the node embeddings $\mathbf{u}_{\text{GCN}}$ are fed into a distributed local greedy solver to generate the solution $\hat{\boldsymbol{v}}_{\text{GCN}}$.

In (2), $\Psi_{\mathcal{G}}$ is an L-layered GCN defined on the conflict graph $\mathcal{G}$, $\mathbf{S}$ is a matrix collecting the features for all $v \in \mathcal{V}$, e.g., $\mathbf{S} = [\mathbf{q}, \mathbf{r}]$, and $\boldsymbol{\omega}$ is the collection of trainable parameters of the GCN. The downstream architecture of our solution is illustrated in Fig. 2. At the system level, the GCN observes the state and topology of the network and generates per-link utilities as the expected returns of scheduling each link, based on which the LGS selects an independent set that seeks to maximize the total return as a schedule.

Formally, by defining the output of an intermediate lth layer of the GCN as $\mathbf{X}^l \in \mathbb{R}^{|\mathcal{V}| \times g_l}$ with $\mathbf{X}^0 = \mathbf{S}$ and $\mathbf{u}_{\text{GCN}} = \mathbf{X}^L$, we have that the expression for the lth layer of the GCN:

$$\mathbf{X}^l = \sigma\left(\mathbf{X}^{l-1}\boldsymbol{\Theta}_0^l + \boldsymbol{\mathcal{L}}\mathbf{X}^{l-1}\boldsymbol{\Theta}_1^l\right), l \in \{1, \ldots, L\}, \quad (3)$$

where $\boldsymbol{\mathcal{L}}$ is the normalized Laplacian of $\mathcal{G}$, $\boldsymbol{\Theta}_0^l, \boldsymbol{\Theta}_1^l \in \mathbb{R}^{g_{l-1} \times g_l}$ are trainable parameters, and $\sigma(\cdot)$ is the activation function. The activation functions of the input and hidden layers are selected as leaky ReLUs whereas a linear activation is used for the output layer. The output dimension is configured as $g_L = 1$, so that $\mathbf{u}_{\text{GCN}}$ is a vector. The trainable parameters $\boldsymbol{\omega}$ in (2) correspond to the collection of $\boldsymbol{\Theta}_0^l$ and $\boldsymbol{\Theta}_1^l$ for all L layers.

The output $\mathbf{u}_{\text{GCN}}$ of the GCN is used as the per-link utilities in LGS. In general, the LGS algorithm builds an estimate $\hat{\boldsymbol{v}}_{\text{Gr}}$ by iteratively adding vertices with the largest utility in their neighborhoods to the solution set, and then excluding them and their neighbors from the residual graph:

$$\hat{\boldsymbol{v}}_{\text{Gr}} \leftarrow \hat{\boldsymbol{v}}_{\text{Gr}} \cup \left\{v \mid u(v) > \max_{v_i \in \mathcal{N}(v)} u(v_i), \text{ for all } v \in \mathcal{V}'\right\}, \quad (4a)$$

$$\mathcal{G}'(\mathcal{V}', \mathcal{E}') \leftarrow \mathcal{G}'(\mathcal{V}', \mathcal{E}') \setminus (\hat{\boldsymbol{v}}_{\text{Gr}} \cup \mathcal{N}(\hat{\boldsymbol{v}}_{\text{Gr}})), \quad (4b)$$

where $\mathcal{N}(\cdot)$ represents all vertices that are neighbors to a vertex or vertex set, and initially $\hat{\boldsymbol{v}}_{\text{Gr}} = \emptyset$ and $\mathcal{G}' = \mathcal{G}$. The algorithm terminates when $\mathcal{G}'$ is empty, and outputs $\hat{\boldsymbol{v}}_{\text{Gr}}$ as the solution, which is guaranteed to be an independent set by (4b). In the case of a tie in (4a), the link with a larger initially-assigned identification number wins without additional information exchanges.

The local communication complexity (defined as the rounds of local exchanges between a node and its neighborhood) of the proposed scheduler is $\mathcal{O}(L + \log|\mathcal{V}|)$, where $\mathcal{O}(\log|\mathcal{V}|)$ is the average local complexity of LGS [10]. With (3) and (4), $\hat{\boldsymbol{v}}_{\text{GCN}}$ can be computed in a distributed manner, where the local computational and communication costs can be controlled by modifying the number of layers L in the GCN. Importantly, the logarithmic local communication complexity is a key aspect to promote scalability.

It should be noted that the GCN in (3) can generalize to networks of various sizes and topologies, and be deployed in a distributed manner. Since $\boldsymbol{\mathcal{L}}$ is a local operator on $\mathcal{G}$, $u_{\text{GCN}}(v)$ can be
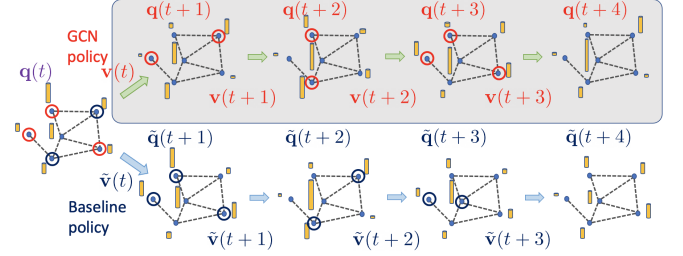
Fig. 3: In a 4-step lookahead scheduling, the GCN and baseline policies are executed for 4 steps from the network state at t, under identical environmental (arrivals and link rates) conditions. Whether the queues attained by the GCN improve upon the baseline is used to inform the quality of the scheduling decision at time t.

computed locally at v by neighborhood aggregation with L rounds of local exchanges between v and its neighbors [29]. Furthermore, the application of LGS ensures the scheduled links $\hat{\boldsymbol{v}}_{\text{GCN}}$ in (2) form a valid independent set for *any choice of trainable parameters $\boldsymbol{\omega}$*. However, we can train $\boldsymbol{\omega}$ to minimize the objective in Problem 1.

To train the parameters $\boldsymbol{\omega}$ in the GCN, we first create an interactive virtual network environment, where the conflict graph $\mathcal{G}$, packet arrivals $\mathbf{a}$, and link rates $\mathbf{r}$, are drawn from (stationary) distributions of interest. Next, experience tuples $(\mathcal{G}(t), \mathbf{S}(t), \hat{\boldsymbol{v}}_{GCN}(t), \boldsymbol{\rho}(t))$, for $t \in \{0, \ldots, T\}$ are collected from the GCN-based scheduler interacting with the virtual environment. The return vector $\boldsymbol{\rho}(t)$ captures the relative performance of the GCN under training with respect to a baseline policy in a K-step lookahead scheduling from network state of t, as illustrated in Fig. 3. As a baseline, we use LGS based on the per-link utility given by $\mathbf{u}(t) = \mathbf{q}(t) \odot \mathbf{r}(t)$ [10], where $\odot$ denotes an element-wise product. Formally, $\boldsymbol{\rho}(t)$ is given by

$$\boldsymbol{\rho}(t) = \varphi\left(\frac{\sum_{k=1}^{K}\|\tilde{\mathbf{q}}(t+k)\|_1}{\sum_{k=1}^{K}\|\mathbf{q}(t+k)\|_1}\right)\mathbf{v}(t) + \mathbf{u}_{\text{GCN}}(t) \odot [\mathbf{1} - \mathbf{v}(t)], \quad (5)$$

where $\mathbf{v}(t)$ is the indicator vector of schedule $\hat{\boldsymbol{v}}_{\text{GCN}}(t)$, $\tilde{\mathbf{q}}(t+k)$ is the vector of queue lengths under the baseline policy k steps after t, and φ is an activation function that can be either linear $\varphi(x) = x$ or a Heaviside step function $\varphi(x) = H(x - 1)$. Intuitively, whenever the GCN policy beats the baseline in a K-step lookahead scheduling (higher values of $\boldsymbol{\rho}(t)$), the schedule $\hat{\boldsymbol{v}}_{\text{GCN}}(t)$ is encouraged for the network state at t. Consequently, a root-mean-square loss is adopted to train our GCN. For a given experience tuple this is given by

$$\ell(\boldsymbol{\omega}; \mathcal{G}(t), \mathbf{S}(t)) = |\mathcal{V}|^{-\frac{1}{2}}\|\mathbf{u}_{\text{GCN}}(t) - \boldsymbol{\rho}(t)\|_2. \quad (6)$$

With the loss in (6) and the collected experience tuples, we update the parameters $\boldsymbol{\omega}$ of the GCN through batch training, employing the Adam optimizer and exponentially decaying learning rates.
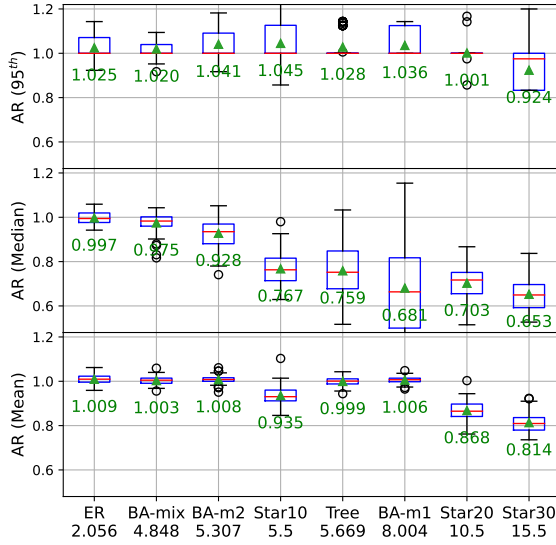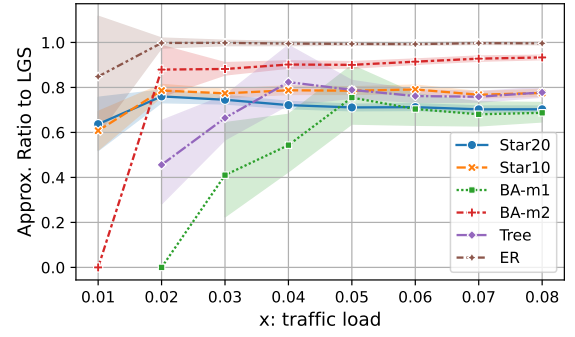
Fig. 4: The approximation ratios of GCN($L = 1$)-based distributed scheduler relative to local greedy scheduler [10] under various conflict graph configurations: (top) 95th percentile, (middle) median, and (bottom) mean backlogs. Smaller values refer to better delay.

4. NUMERICAL EXPERIMENTS

We evaluate our GCN-based distributed scheduler in simulated wireless networks. The simulated conflict graphs include star graph, and random graphs generated following the Erdős–Rényi (ER) [31], Barabási–Albert (BA) [32], and power-law tree models. Intuitively, the ER conflict graphs seek to represent networks of uniformly distributed users with identical transmit power (i.e., unit-disk interference zones). The star and BA conflict graphs arise from networks with several connected components and links of heterogeneous transmit power, e.g., macrocells surrounded by microcells and D2D links. The power-law trees represent wireless backhaul networks where some links have interfering neighbors.

We adopt the following configurations for conflict graphs: *StarX* (star graph with $V = X + 1$), *BA-mX* (BA graph with $V = 70, m = X$), *BA-mix* (BA graphs with $V \in \{100, 150, \ldots, 300\}$ and $m \in \{2, 5, 10, 15, 20\}$), *ER* (ER graphs with $V = 50, p = 0.1$), and *Tree* (power-law tree with $V = 50, \gamma = 3$), where $V = |\mathcal{V}|$ is the number of nodes, m is the number of edges that each new node forms during the preferential attachment process for the BA model, p is the probability of edge-appearance for the ER model, and γ is the exponent for the power-law tree model. Link rates $r_v(t)$, defined as the number of packets that can be transmitted through link v at time t, are independently drawn from a normal distribution $\mathbb{N}(50, 25)$, and clipped to $[0, 100]$, to capture a lognormal fading channel [33]. The exogenous packets at each source user follow a Poisson arrival with a prescribed arrival rate λ. We define network traffic load as $\mu = \lambda / \mathbb{E}(\mathbf{r})$. For each configuration of a graph model, we generate 100 scheduling instances, each contains realizations of a conflict graph and the random processes of arrivals and link rates for $T = 64$ time slots. A 1-hop flow is generated for each link in the network. In a scheduling instance, each tested scheduler is tested under identical realizations of random processes.

A single-layer GCN ($L = 1$) is evaluated. The baseline for training and evaluation is the vanilla LGS with a utility function of $\mathbf{u}(t) = \mathbf{q}(t) \odot \mathbf{r}(t)$ [10]. The feature matrix is set to be the same as the baseline utility $\mathbf{S}(t) = [\mathbf{u}(t)]$. Through trial-and-error, we train the GCN on scheduling instances with mixed conflict graphs of 80% *Star30* and 20% *BA-m2* for effectiveness and generalizability. The

Fig. 5: The approximation ratios of the median backlogs of GCN($L = 1$)-based scheduler w.r.t. the baseline by traffic load.

GCN-DQL settings include $\varphi(x) = H(x - 1)$, $K = 5$, a batch size of 64 for experience replay, and 6000 episodes.[1]

We collect the mean, median, and tail (95th percentile) queue lengths (backlogs) over time and links in the wireless networks, under the GCN-based distributed scheduler and the baseline, in light-to-moderately loaded traffics, $\mu = 0.01, \ldots, 0.08$. The boxplots of approximation ratios (AR) for the 95th percentile, median, and average backlogs by graph model, of the GCN-based scheduler to the baseline (smaller than 1 denotes improvement) with $\mu = 0.07$ are illustrated in Figs. 4 (top to bottom), respectively, where the means are in green text and marked by green triangles. The peak to average degree ratio (written under each graph annotation in Fig. 4) measures the graph centralization. Compared to the baseline, our GCN-based scheduler can reduce the average and median backlogs, especially when the conflct graph is more centralized and the central links are more likely to be congested by the memoryless baseline. On star graphs, the average and median backlogs are respectively reduced by $6.5 \sim 18.6\%$ and $23.3 \sim 34.7\%$. On more complex topologies, the mean backlogs are the same as the baseline, while the median backlogs are reduced proportionally to the graph centralization, e.g. from 2.5% on BA-mix to 32.9% on BA-m1. The tail backlogs of GCN-based scheduler are increased by $0.1 \sim 4.5\%$ on average, while the median values are the same as the baseline, except on Star30 where tail backlogs are improved in both mean and median values. With a different baseline utility of $\mathbf{u}(t) = \min(\mathbf{q}(t), \mathbf{r}(t))$, the average backlogs on BA and Tree graphs can also be improved by the GCN. Considering sojourn time [23] as the baseline utility, the average sojourn time of a packet can be improved by $1.6 \sim 4.6\%$ on star graphs. These results show that our approach can improve the delay of the majority of the links over the baseline, at the cost of slight increase of the tail delay. The improvement of the median backlogs brought by the GCN-based scheduler is consistent under different traffic loads, as illustrated in Fig. 5, which shows its good generalizability to network traffic conditions.

5. CONCLUSIONS

We presented a GCN-based distributed and scalable scheduler to improve the delay performance in wireless networks by combining the efficiency of a local greedy scheduler and the ability of GCNs to encode the network state and topological information. The proposed scheduler markedly improves upon classical baselines, especially in wireless networks with several connected components and heterogeneous transmit power, and shows good generalizability over graph types, graph sizes, and traffic loads.

---

[1]Training typically takes 3-4 hours on a workstation with a specification of 16GB memory, 8 cores, and Geforce GTX 1070 GPU. The source code is published at https://github.com/zhongyuanzhao/gcn-dql

6. REFERENCES

[1] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, 2006.

[2] S. K. Sarkar, T. G. Basavaraju, and C. Puttamadappa, *Ad hoc Mobile Wireless Networks: Principles, Protocols and Applications (2nd Ed)*. CRC Press, 2013.

[3] A. Kott, A. Swami, and B. J. West, "The internet of battle things," *Computer*, vol. 49, no. 12, pp. 70–75, 2016.

[4] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133995–134030, 2020.

[5] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1132–1145, 2009.

[6] A. G. Marques, N. Gatsis, and G. B. Giannakis, "Optimal cross-layer design of wireless fading multi-hop networks," in *Cross Layer Designs in WLAN Systems* (N. Zorba, C. Skianis, and C. Verikoukis, eds.), Leicester, UK: Troubador Pub, 2011.

[7] A. Kabbani, T. Salonidis, and E. W. Knightly, "Distributed low-complexity maximum-throughput scheduling for wireless backhaul networks," in *IEEE Intl. Conf. on Computer Comms. (INFOCOM)*, pp. 2063–2071, 2007.

[8] I. C. Paschalidis, F. Huang, and W. Lai, "A message-passing algorithm for wireless network scheduling," *IEEE/ACM Trans. Netw.*, vol. 23, p. 1528–1541, Oct. 2015.

[9] S. Basagni, "Finding a maximal weighted independent set in wireless networks," *Telecomm. Systems*, vol. 18, no. 1-3, pp. 155–168, 2001.

[10] C. Joo and N. B. Shroff, "Local greedy approximation for scheduling in multihop wireless networks," *IEEE Trans. on Mobile Computing*, vol. 11, no. 3, pp. 414–426, 2012.

[11] C. Joo, X. Lin, J. Ryu, and N. B. Shroff, "Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1476–1488, 2015.

[12] S. Sanghavi, D. Shah, and A. S. Willsky, "Message passing for maximum weight independent set," *IEEE Trans. Info. Theory*, vol. 55, no. 11, pp. 4822–4834, 2009.

[13] P. Du and Y. Zhang, "A new distributed approximation algorithm for the maximum weight independent set problem," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[14] X. Li, P. Tolmachev, M. Pauley, and J. H. Manton, "A distributed transmission scheduling algorithm for wireless networks based on the Ising model," in *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 6–10, 2018.

[15] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, and M. Alouini, "Distributed hybrid scheduling in multi-cloud networks using conflict graphs," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 209–224, 2018.

[16] C. Joo, G. Sharma, N. B. Shroff, and R. R. Mazumdar, "On the complexity of scheduling in wireless networks," *EURASIP Journal on Wireless Commun. and Netw.*, vol. 2010, no. 1, p. 418934, 2010.

[17] W. Cheng, X. Cheng, T. Znati, X. Lu, and Z. Lu, "The complexity of channel scheduling in multi-radio multi-channel wireless networks," in *IEEE Intl. Conf. on Computer Comms. (INFOCOM)*, pp. 1512–1520, 2009.

[18] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Distributed scheduling using graph neural networks," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pp. 4720–4724, 2021.

[19] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Link scheduling using graph neural networks," in *submitted to IEEE J. Sel. Topics Signal Process., arXiv:2109.05536*, 2021.

[20] D. Xue and E. Ekici, "Delay-guaranteed cross-layer scheduling in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1696–1707, 2012.

[21] J. J. Jaramillo, R. Srikant, and L. Ying, "Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 979–987, 2011.

[22] I. H. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. of the 11th ACM intl. symposium on Mobile ad hoc netw. and comp. (MobiHoc)*, pp. 31–40, 2010.

[23] L. Hai, Q. Gao, J. Wang, H. Zhuang, and P. Wang, "Delay-optimal back-pressure routing algorithm for multihop wireless networks," *IEEE Trans. Vehicular Tech.*, vol. 67, no. 3, pp. 2617–2630, 2018.

[24] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *IEEE Intl. Symposium on Info. Theory (ISIT)*, pp. 561–565, 2017.

[25] M. Gupta, A. Rao, E. Visotsky, A. Ghosh, and J. G. Andrews, "Learning link schedules in self-backhauled millimeter wave cellular networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8024–8038, 2020.

[26] M. Lee, G. Yu, and G. Y. Li, "Graph embedding-based wireless link scheduling with few training samples," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2282–2294, 2021.

[27] J. Gao, Y. Shen, M. Ito, and N. Shiratori, "Multi-agent Q-learning aided backpressure routing algorithm for delay reduction," *arXiv preprint arXiv:1708.06926*, 2017.

[28] Z. Wang, M. Eisen, and A. Ribeiro, "Learning decentralized wireless resource allocations with graph neural networks," *arXiv preprint arXiv:2107.01489*, 2021.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Intl. Conf. Learn. Repres. (ICLR)*, 2017.

[30] J. Yang, S. C. Draper, and R. Nowak, "Learning the interference graph of a wireless network," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 631–646, 2016.

[31] P. Erdős and A. Rényi, "On random graphs I," *Publ. Math. Debrecen 6*, pp. 290–297, 1959.

[32] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[33] H. Mousavi, I. S. Amiri, M. Mostafavi, and C. Choon, "LTE physical layer: Performance analysis and evaluation," *Applied Computing and Informatics*, vol. 15, no. 1, pp. 34 – 44, 2019.