A Learning-based Distributed Algorithm for Scheduling in Multi-hop Wireless Networks

Daehyun Park, Sunjung Kang, *Student Member, IEEE*, and Changhee Joo, *Senior Member, IEEE*

***Abstract*—We address the joint problem of learning and scheduling in multi-hop wireless network without a prior knowledge on link rates. Previous scheduling algorithms need the link rate information, and learning algorithms often require a centralized entity and polynomial complexity. These become a major obstacle to develop an efficient learning-based distributed scheme for resource allocation in large-scale multi-hop networks. In this work, by incorporating with learning algorithm, we develop provably efficient scheduling scheme under packet arrival dynamics without a priori link rate information. We extend the results to distributed implementation and evaluation their performance through simulations.**

***Index Terms*—Distributed algorithm, learning, multi-hop networks, provable efficiency, wireless scheduling.**

I. Introduction

As one of the key functions in wireless communication networks, link scheduling determines which links should be activated at what time. The problem is challenging, in particular, in multi-hop[1] networks due to non-linear interference relationship between wireless links. The seminal work of Tassiulas and Ephremides has shown that the maximum weighted matching (MWM) algorithm that maximizes the queue weighted rate sum can achieve the optimal throughput under packet arrival dynamics [8]. Due to high computational complexity of MWM [1], alternative low-complexity scheduling solutions with comparable performance such as greedy maximal matching (GMM) or longest queue first (LQF) have attracted much attention [2], [3]. However, since GMM still has linear complexity that increases with the network size, it can be hardly used in large-size multi-hop networks.

Manuscript received August 18, 2021; approved for publication August 24, 2021. This paper is specially handled by EIC and Division Editor with the help of three anonymous reviewers in a fast manner.

This work was supported in part by the NRF grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C2013065), and in part by the MSIT, Korea, under the ICT Creative Consilience program (IITP-2021-2020-0-01819) supervised by the IITP.

D. Park was with ECE, UNIST, Ulsan, South Korea, email: eoy13@unist.ac.kr.

S. Kang is with ECE, OSU, Columbus, Ohio, email: kang.853@osu.edu.

C. Joo is the corresponding author, and he is with CSE, Korea University, Seoul, South Korea, email: changhee@korea.ac.kr.

Digital Object Identifier: 10.23919/JCN.2021.000030

[1]In this work, we use the term 'multi-hop' for the arbitrary interference relationship between wireless links, and consider only single-hop traffic flows, i.e., a packet departs the network after a single transmission. If a scheduling scheme is suboptimal with these single-hop flows over multi-hop network, then clearly it is suboptimal with multi-hop flows. This approach has been widely adopted to investigate the performance of scheduling schemes without being affected by the functions of other layers such as routing and congestion control [1]–[7].

There has been extensive research on developing efficient scheduling algorithms that have sublinear complexity, yet perform provably well in multi-hop wireless networks. An approximation to GMM with logarithmic complexity has been developed in [4]. Random access technique with explicit neighborhood information exchanges has been explored at some expense of performance [5]–[7], [9]. Several studies have shown that the optimal throughput performance is achievable, either by taking the pick-and-compare approach [10], [11], or by exploiting the carrier-sensing functionality [12], [13]. There have been also attempts to develop provably efficient scheduling algorithms that work with time-varying wireless channels [4], [14] or with complex SINR interference model [15], [16]. The aforementioned scheduling schemes provide performance guarantees under packet arrival dynamics. However, they are limited to deterministic link rates that are *known a priori* or at the time of scheduling. Their extension to the case when the link rates are unknown is not straightforward.

In this work, we consider the scheduling problem, where link rates and statistics are unknown a priori. This occurs when new applications try to operate efficiently under uncertainty caused by wireless fading, interference, limited feedback, measurement error, system dynamics, etc [17]–[19]. We assume that an instance link rate is revealed when it is accessed/scheduled, and it is drawn from an unknown static distribution. Our goal is to find an appropriate sequence of link schedules that maximize throughput under packet arrival dynamics, while quickly learning the link rates and queue states.

Focusing on the learning aspect, the problem can be viewed as a variant of multi-armed bandit (MAB) problems, in which one repetitively plays a set of arms to maximize the reward sum [20]. The performance of a learning algorithm is often evaluated by *regret*, which is the difference in the total expected reward obtained by an optimal policy and that by the learning algorithm. Lai and Robbins have shown that the regret grows at least at logarithmic rate of time [21], and several index-type learning algorithms with the order-optimal regret have been developed [22], [23]. For a large-scale multi-hop wireless network, it is imperative to design algorithms that are amenable to implement in a distributed manner. In [24], the authors have developed a distributed learning algorithm that selects best M out of N arms, where each of M users selects an arm taking into consideration mutual collision. By employing a time-division selection approach, the scheme is shown to achieve logarithmic regret. Chen et al. [20] and Gai et al. [25] have considered more general problems of combi-

1229-2370/21/$10.00 © 2021 KICS

natorial MAB (CMAB) with arbitrary constraint. They have employed an (α, β)-approximation oracle that can achieve α fraction of the optimal value with probability β, and developed learning schemes that can achieve the logarithmic growth for $\alpha\beta$ fraction of the optimal expected regret (denoted by $\alpha\beta$-*regret*). However, an oracle with good $\alpha\beta$ (i.e., close to 1) often has a high-order polynomial complexity, and thus as the network scales, it is not clear whether the scheme is amenable to implement in a distributed manner.

Applying learning algorithms to scheduling in wireless networks, the works of [26], [27] have addressed the regret-minimization problem in cognitive radio network settings, and developed distributed schemes with logarithmic regret through prioritized ranking and adaptive randomization. The authors of [28] have developed fully distributed schemes that can achieve the logarithmic regret without any information exchange. In [29], the authors have successfully lowered the algorithmic complexity to $O(1)$ while achieving the logarithmic regret performance. Although these aforementioned learning algorithms are amenable to distributed implementation, they are limited to single-hop networks, such as wireless access networks, and to saturated traffic scenarios (i.e., links always have packets to send), and thus cannot accordingly respond to packet arrival dynamics. Recently, Stahlbuhk et al. [19] has developed a joint learning and scheduling scheme that provides provable efficiency under packet arrival dynamics, by incorporating GMM scheduling algorithm and UCB-based learning algorithm. Albeit interesting, it achieves only $1/2$ of the capacity region and has linear complexity, which makes it less attractive for large-size networks.

In this work, we consider the joint problem of learning and scheduling in multi-hop wireless networks, and develop *low-complexity schemes that achieves near-full capacity region under packet arrival dynamics*. Our contribution can be summarized as follows:

- We develop a joint learning and scheduling scheme with $O(k)$ computational complexity, by successfully incorporating a graph augmentation algorithm with the UCB index. Parameter k is settable for a certain level of performance, in which case the complexity becomes $O(1)$.
- We show that A^k-UCB is an (α, β)-approximation oracle with $\alpha = \frac{k-1}{k+1}$ and some small non-zero β, but can indeed achieve the logarithmic growth for $\frac{k-1}{k+1}$-regret, regardless of the value of β, which is in contrast to $\alpha\beta$-regret shown in [20], [25].
- By using the frame structure, we show that A^k-UCB achieves $\frac{k-1}{k+1}$ fraction of the capacity region under packet arrival dynamics.
- We extend our result and develop dA^k-UCB scheme that is amenable to implement in a completely distributed manner.

The rest of paper is organized as follows. Section II describes our system model. We propose a joint scheme of learning and scheduling in Section III, and analytically evaluate its performance in Section IV. We further extend our scheme for distributed implementation in Section V. Finally, we numerically evaluate our schemes in Section VI and conclude our work in Section VII.

II. System Model

We consider a multi-hop wireless network denoted by graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ with the set $\mathcal{V}$ of nodes and the set $\mathcal{L}$ of directional links. We assume that the connectivity is reciprocal, i.e., if $(u, v) \in \mathcal{L}$, then $(v, u) \in \mathcal{L}$. A set of links that can be scheduled at the same time is constrained by the primary interference model, under which any node v (either transmitter or receiver) in the network can communicate with at most one of its neighbor nodes $\mathcal{N}(v)$, where $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (v, u) \in \mathcal{L}\}$. Slightly abusing the notation, we also denote the set of links that is connected to v by $\mathcal{N}(v)$. The primary interference model can represent Bluetooth or FH-CDMA networks as well as capture the essential feature of wireless interference [2], [19], and has been adopted in many studies on wireless scheduling, e.g., see [2]–[7] for more detailed description. Time is slotted, which can be achieved by being equipped with high accuracy GPS. At each time slot, a set of links that satisfies the interference constraints can be simultaneously activated. Such a set of links is called a *matching* (or a feasible schedule), and let $\mathcal{S}$ denote the set of all matchings.

At each link $i \in \mathcal{L}$, we assume packets arrive following a Bernoulli process with probability λ_i (e.g., see [11]). Let $\boldsymbol{\lambda}$ denote its vector and $a_i(t) \in \{0, 1\}$ denote the number of arrived packets in time slot t. We have $\mathbb{E}[a_i(t)] = \lambda_i$. We assume that the rate of link i is time-varying due to multi-path fading and unknown interference as in [14], and it is independently drawn from a (possibly different) distribution with mean μ_i. Let $\boldsymbol{\mu}$ denote its vector and $X_i(t) \in [0, 1]$ denote the instance rate of link i when it is activated at time slot t, with $\mathbb{E}[X_i(t)] = \mu_i$. The extension to multiple packet arrivals and departures is straightforward. We assume that $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are unknown.

At time slot t, if a policy activates matching $\mathbf{S}_t$, then each link $i \in \mathbf{S}_t$ accesses the medium and transmits $X_i(t)$ packets[2] during the time slot. Each link i is associated with an unbounded buffer that queues up packets for transmission. Let $q_i(t)$ denote the queue length at link i at the beginning of time slot t, which evolves as

$$q_i(t+1) = \begin{cases} [q_i(t) - X_i(t)]^+ + a_i(t), & \text{if } i \in \mathbf{S}_t, \\ q_i(t) + a_i(t), & \text{if } i \notin \mathbf{S}_t, \end{cases} \quad (1)$$

where $[\cdot]^+ = \max\{\cdot, 0\}$. Let $\boldsymbol{q}(t)$ denote its vector, and let $q^*(t) = \max_{i \in \mathcal{L}} q_i(t)$ denote the maximum queue length in the network at time slot t.

We consider a frame structure where each frame has length of T time slots. Frame n begins at time slot $t_n = (n-1)T+1$. During frame n, i.e., for time slots $t \in [t_n, t_{n+1})$, we define weight $W_i(t)$ and its mean w_i of link i, respectively, as

$$W_i(t) = \frac{q_i(t_n)}{q^*(t_n)} X_i(t), \text{ and } w_i = \frac{q_i(t_n)}{q^*(t_n)} \mu_i. \quad (2)$$

[2]or transmits 1 packet with success probability $X_i(t)$.

For $q^*(t_n) = 0$, we define $W_i(t) = X_i(t)$ and $w_i = \mu_i$. Let $\boldsymbol{w}$ denote its vector $(w_1, w_2, \cdots, w_{|\mathcal{L}|})$, where $|\cdot|$ is the cardinality of the set. We denote the link weight sum of matching S by

$$r_{\boldsymbol{w}}(S) = \textstyle\sum_{i \in S} w_i. \tag{3}$$

For convenience, we let $r^*_{\boldsymbol{w}} = \max_{S \in \mathcal{S}} \; r_{\boldsymbol{w}}(S)$ denote the largest weight sum over all matchings, and we also denote a set of optimal matchings by $\mathcal{S}^*_{\boldsymbol{w}} = \arg\max_{S \in \mathcal{S}} r_{\boldsymbol{w}}(S)$. For $\alpha \in (0, 1]$, we define a set of near-optimal matchings with respect to vector $\boldsymbol{w}$ as

$$\mathcal{S}^{\alpha}_{\boldsymbol{w}} = \{S \in \mathcal{S} \mid r_{\boldsymbol{w}}(S) \geq \alpha \cdot r^*_{\boldsymbol{w}}\}, \tag{4}$$

and define its complement as $\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}} = \mathcal{S} - \mathcal{S}^{\alpha}_{\boldsymbol{w}}$.

In the CMAB framework, a link corresponds to an arm, a matching to a super arm, and the instance link rate to the reward of the link, respectively. We use the terms interchangeably. Note that the regret is defined as the accumulated expected difference between the reward sum associated with an optimal matching and that obtained by the MAB algorithm. Similar to [20], we define α-*regret* as, for some $\alpha \in (0, 1]$,

$$Reg^{\alpha}(t) = t \cdot \alpha \cdot r^*_{\boldsymbol{w}} - \mathbb{E}\left[\textstyle\sum_{\tau=1}^{t} r_{\boldsymbol{w}}(\mathbf{S}_\tau)\right], \tag{5}$$

which evaluates the performance of an CMAB task at time t.

In the viewpoint of resource allocation, achieving a high reward sum is equivalent to achieving a larger queue-weighted link rate sum, which implies that the links with high demands and high service rates are scheduled first, and thus tends to stabilize the network. A network is said to be *stable* if the queues of all links are *rate stable*, i.e., $\lim_{t\to\infty} q_i(t)/t = 0$ with probability 1 for all $i \in \mathcal{L}$. Let Λ denote the *capacity region*, which is the set of arrival rate vectors $\boldsymbol{\lambda}$ such that for any $\boldsymbol{\lambda} \in \Lambda$, there exists a policy that can make the network stable. We say that a scheduling policy has the *stability region* $\gamma\Lambda$ for some $\gamma \in [0, 1]$, if it can stabilize the networks for any arrival $\boldsymbol{\lambda} \in \gamma\Lambda$.

We aim to develop a joint scheme of learning and scheduling that determines $\mathbf{S}_t$ to

$$\begin{aligned} \textbf{maximize} \quad & \gamma \\ \textbf{subject to} \quad & \lim_{t\to\infty} \tfrac{q_i(t)}{t} = 0, \text{ for any } \boldsymbol{\lambda} \in \gamma\Lambda, \\ & X_i(t) \overset{iid}{\sim} \mathcal{D}_i, \text{ and (1)}, \end{aligned}$$

where $\mathcal{D}_i$ denotes the distribution with finite support $[0, 1]$. The arrival vector $\boldsymbol{\lambda}$ and the distributions $\{\mathcal{D}_i\}$ are *unknown* to the controller, and an observation on $X_i(t)$ is available only by scheduling link i at time t.

III. Augmentation with UCB Index (A^k-UCB)

We develop a provably efficient joint scheme of learning and scheduling, by incorporating the augmentation algorithm presented in [11] with UCB index. We first describe the overall algorithm, and then explain the detailed operations.

We consider each frame time as an independent learning period, which allows us to decouple the learning from the scheduling. In the following, we describe the operation of our algorithm during a frame time. For the ease of exposition, we assume that our algorithm runs for time slot $[1, T]$, where T is the frame length.

Algorithm 1 Frame-based joint learning and scheduling

/* Repeat at each frame time */
1: Obtain queue constant q_i and q^*
2: Initialize $\hat{w}_i$ and $\hat{\tau}_i$
3: **for** $t = 1$ to $|\mathcal{L}|$ **do**
4: Schedule arbitrary matching $\mathbf{S}_t$ that has link t
5: Update $\hat{w}_i$ and $\hat{\tau}_i$ for each link $i \in \mathbf{S}_t$
6: **for** $t = |\mathcal{L}| + 1$ to T **do**
7: Compute UCB index $\bar{w}_{i,t} \leftarrow \hat{w}_i + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_i}}$
8: $\langle$ Select matching $\mathbf{S}_t$ using $\bar{\boldsymbol{w}}$ $\rangle$
9: Schedule $\mathbf{S}_t$
10: Update $\hat{w}_i$ and $\hat{\tau}_i$ for each link $i \in \mathbf{S}_t$

We start with some notations. Let $q_i = q_i(1)$ denote the initial (i.e., at the beginning of the frame) queue length of link (arm) i, and let $q^* = \max_i q_i$. Let $\hat{\tau}_i(t)$ denote the number of times that arm i is played up to time slot t, and let $\hat{\tau}_S(t)$ denote the number of times that matching S is played. The UCB index of arm i [22] is defined as

$$\bar{w}_{i,t} = \hat{w}_i(t-1) + \sqrt{\tfrac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_i(t-1)}}, \tag{6}$$

where $\hat{w}_i(t) = \frac{q_i}{q^*} \cdot \frac{1}{\hat{\tau}_i(t)} \sum_{n=1}^{t} X_i(n) \cdot \mathbb{I}\{i \in S_n\}$ denotes average reward of arm i at time slot t weighted by $\frac{q_i}{q^*}$, and $\mathbb{I}\{e\} \in \{0, 1\}$ denotes the indicator function that equals 1 if event e occurs, and 0 otherwise. All the variables of $\bar{w}_{i,t}, \hat{w}_i, \hat{\tau}_i, q_i, q^*$ will be reset at the beginning of each frame. Let $\bar{\boldsymbol{w}}_t = (\bar{w}_{1,t}, \bar{w}_{2,t}, \cdots, \bar{w}_{|\mathcal{L}|,t})$ denotes the UCB index vector. Then $r_{\bar{\boldsymbol{w}}_t}(S)$ and $r^*_{\bar{\boldsymbol{w}}_t}$ are the index sum over links in matching S and its maximum value over all possible matchings, respectively. Also, we denote $\mathcal{S}^*_{\bar{\boldsymbol{w}}_t}$ and $\mathcal{S}^{\alpha}_{\bar{\boldsymbol{w}}_t}$ as the set of matchings that achieve $r^*_{\bar{\boldsymbol{w}}_t}$ and those that achieve at least $\alpha r^*_{\bar{\boldsymbol{w}}_t}$, respectively.

We develop our joint learning and scheduling algorithm based on the generic UCB index, as shown in Algorithm 1, where we omit subscript t of $\hat{w}_i(t)$ and $\hat{\tau}_i(t)$ for brevity. At time slot $t = 1$, it obtains two constant weight parameters q_i and q^* (line 1), and schedules arbitrary matchings for $|\mathcal{L}|$ time slots such that each link can be scheduled at least once (lines 3-5). Afterwards, it computes the index of each arm (line 7), selects matching $\mathbf{S}_t$ using the indices (line 8), and schedules it (lines 9-10).

The key part of the algorithm is about how to select matching $\mathbf{S}_t$ in line 8 such that it achieves high performance at low complexity. It has been shown in [20], [25] that, if we use an (α, β)-approximation oracle in the matching selection, the joint algorithm has $\alpha\beta$-regret performance, which leads to achieving $\alpha\beta$ fraction of the capacity region. The problem is, that the parameter β can be very small in particular when the network size is large. To this end, we introduce a class of augmentation algorithms with parameter k, which is an (α, β)-approximation oracle with $\alpha = (k-1)/(k+1)$ and

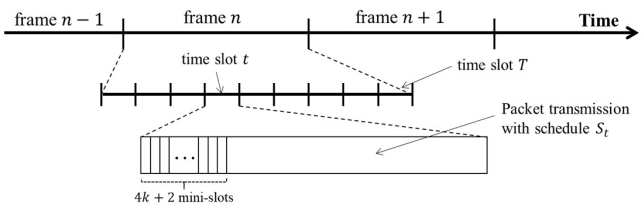


Fig. 1. Time structure.

some small[3] $\beta > 0$, and has $O(k)$ complexity.

We overview the augmentation algorithm. For the detailed description, we refer to [11] or our technical report [30]. We start with some definitions. Given a matching S, an *augmentation* A of matching S is a path or cycle where every alternate link is in S and has the property that if all links in $A \cap S$ are removed from S and all links in $A - S$ are added to that S, then the resulting set of links is another matching in G. The latter process of finding new matching is called *augmenting* S with A, and the resulting matching is denoted by $S \oplus A = (S-A) \cup (A-S)$. A pair of augmentations A_1 and A_2 of matching S is $disjoint$ if no two links in $A_1 - S$ and $A_2 - S$ are adjacent, i.e., if they do not share a common node. Let $\mathcal{A}$ denote a set of disjoint augmentations of matching S where every pair in $\mathcal{A}$ is disjoint. Then $\bigcup_{A \in \mathcal{A}} (S \oplus A)$ is also a matching in G.

The overall procedure of the augmentation algorithm at each time slot t is as follows.

1) At the beginning of the time slot, each link i is associated with some known weight $w_i(t)$.
2) Given a valid matching $\mathbf{S}_{t-1}$ that is the schedule at time $t-1$, it randomly generates a set $\mathcal{A}$ of disjoint augmentations of $\mathbf{S}_{t-1}$.
3) It compares the weight sum of $A - \mathbf{S}_{t-1}$ and $A \cap \mathbf{S}_{t-1}$ for each $A \in \mathcal{A}$. Let $B(A)$ be the one with the larger weight sum among the two.
4) It takes the new schedule $\mathbf{S}_t$ as $\bigcup_{A \in \mathcal{A}} B(A)$.

For the comparison of the weight sum in the 3rd step, we define the gain of augmentation A as

$$G_t(A) = \sum_{i \in A - \mathbf{S}_{t-1}} w_i - \sum_{j \in A \cap \mathbf{S}_{t-1}} w_j, \qquad (7)$$

and obtain new schedule $\mathbf{S}_t$ by augmenting $\mathbf{S}_{t-1}$ with all $A \in \mathcal{A}$ of $G_t(A) > 0$.

The augmentation algorithm can accomplish the above procedure in a distributed fashion. The algorithm has two configuration parameters p and k, and consists of the following four stages in each time slot: initialization, augmenting, checking a cycle, and back-propagating/scheduling. For the ease of exposition, we consider additional time structure of mini-slots, as shown in Fig. 1, and all the four stages end in $4k+2$ mini-slots.

i) **Initialization stage:** At mini-slot $\tau = 1$, each node v selects itself as a seed with probability p. Once selected, it becomes an *active* node. It starts an augmentation A_v and randomly selects $\bar{Z}_v \in [1, k]$, which is the maximum size[4] of A_v. Then, it adds the first link to A_v from its neighboring links $\mathcal{N}(v)$: if there is a link in $\mathbf{S}_{t-1} \cap \mathcal{N}(v)$, then we include the link in A_v, and otherwise, we randomly choose a link from $\mathcal{N}(v)$. Once the first link (v, n) is chosen, the two nodes v and n coordinate with each other by exchanging necessary information through request (REQ) and acknowledgement (ACK) messages. The information (including current A_v, current $G_t(A_v)$ according to (7), $\bar{Z}_v$, etc) allows node n to continue building the augmentation. At the end of the first mini-slot, node v becomes inactive and node n becomes active.
ii) **Augmenting stage:** It extends A_v by adding a link at each mini-slot $\tau \in [2, 2k+1]$. Current active node selects new link that should be either a random neighboring link outside $\mathbf{S}_{t-1}$ (if the previously selected link is in $\mathbf{S}_{t-1}$) or a neighboring link in $\mathbf{S}_{t-1}$ (if the previously selected link is not in $\mathbf{S}_{t-1}$). It updates $G_t(A_v)$ and A_v accordingly, and proceeds a similar coordination through REQ and ACK messages, and at the end of mini-slot, active node changes to the corresponding end node of the new link. The extension continues until one of the following conditions hold: the size of A_v equals $\bar{Z}_v$, A_v cannot be extended any more, or the extension fails due to message collision (see Fig. 2).
iii) **Cycle-checking stage:** When the augmenting stage finishes, the last node is called the *terminus*. The terminus checks whether the final augmentation A_v forms a cycle or not. If it forms a cycle, the gain is updated to include the last link (n, s).
iv) **Back-propagating/scheduling stage:** The last stage is for back-propagating the final gain from the terminus to the seed through A_v, and in the meantime, constructing S^* either by scheduling links outside $\mathbf{S}_{t-1}$ if the gain is positive, or by scheduling links in $\mathbf{S}_{t-1}$ if the gain is negative. This takes additional $2k+1$ mini-slots at most. The final result S^* will be used as the new schedule $\mathbf{S}_t$ during time slot t.

Remarks: In our description, we present $\mathbf{S}_{t-1}$ as if it is a global variable, but each node v indeed requires only the local view of it, i.e., $\mathcal{N}(v) \cap \mathbf{S}_{t-1}$, which can be obtained during the back-propagation in the last stage of the previous time slot. After all the stages, a set of augmentations (one per a seed node) will be generated. Since a size-$\bar{Z}$ augmentation A can have at most $\bar{Z}+1$ links of $\mathbf{S}_{t-1}$ and $\bar{Z}$ new links, the number of total links in A can be up to $2\bar{Z}+1 \leq 2k+1$.

Fig. 2 illustrates an example operation of the augmentation algorithm during time slot t in a 3×4 grid topology. Nodes are dots, and solid lines are links. The previous schedule $\mathbf{S}_{t-1}$ is marked by thick solid lines in Fig. 2(a)-(e). At the beginning ($\tau = 1$), each node selects itself as a seed with probability p. In this example, three nodes are selected and become an active as marked by (white) numbered circles in Fig. 2(a). Active nodes 1 and 2 have to start its augmentation with the previous scheduled link, while active node 3 selects one of three neighboring nodes at random. The solid arrow

[3]It depends on the network topology and the setting of p. In general, we have a smaller β for a larger network.

[4]The size Z of an augmentation A is defined as the number of new links in A, i.e., $Z = |A - \mathbf{S}_{t-1}|$.

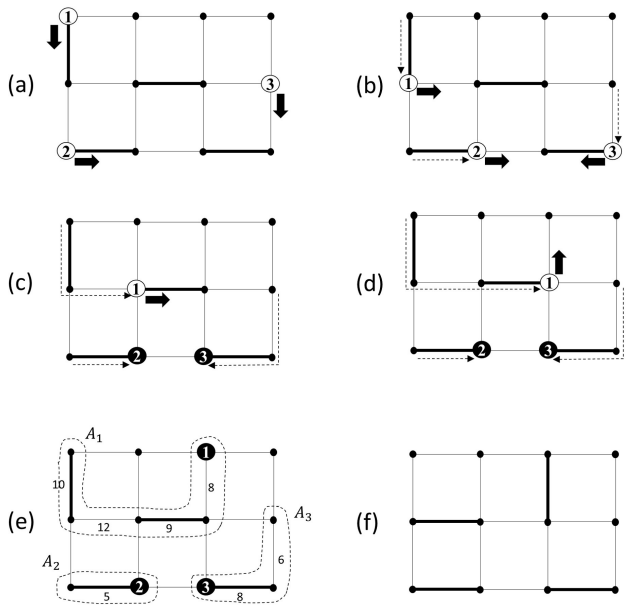


Fig. 2. Example operation of the augmentation algorithm with $k = 2$ in a time slot. After the cycle-checking stage, link weights of final augmentations are shown in (e). After back-propagation, links are accordingly augmented with A_1, and the final schedule is marked by thick solid lines in (f).

denotes the link selected by the active node. Once selected, the nodes exchange the necessary information. In the next mini-slot ($\tau = 2$), active nodes change as shown in Fig. 2(b). Narrow dotted arrows denote the augmentation up to now. The active nodes continue to build the augmentation repeating the procedure, until the augmentation cannot be extended or it reaches the maximum size. In the meantime, if two augmentations collide as shown by active nodes 2 and 3 in Fig. 2(b), both augmentations terminate. The node at the collision point will belong to the augmentation that follows a link in $\mathbf{S}_{t-1}$, as shown in Fig. 2(c). The terminus nodes are marked by solid (black) number circles. The result after $(2k+1)$th mini-slots is shown in Fig. 2(e), where each of three augmentations is marked by a dotted enclosure. The number of links in the augmentations denote their weight. Then after checking a cycle, the back-propagating stage follows. Each terminus makes the final decision by comparing the weight sum as in (7). The decision propagates backward through the augmentation, and leads to new schedule $\mathbf{S}_t$ as shown in Fig. 2(f).

By using the aforementioned augmentation algorithm in matching selection of line 8 in Algorithm 1, we complete A^k-UCB scheme. However, evaluating the performance of A^k-UCB is not straightforward. Since the scheduler do not know the true link rate μ_i and instead use the experience-based UCB index $\bar{w}_{i,t}$, the previous analysis of [11] for scheduling performance is not applicable due to inaccurate link rate information. Also for learning performance, considering A^k-UCB as an (α, β)-oracle is not much helpful due to the fact that probability β can be arbitrarily small as the network scales up.

Our main contribution is to analytically characterize the performance of our joint learning and scheduling scheme A^k-UCB, and to show that it can achieves the rate-optimal logarithmic growth of $\frac{k-1}{k+1}$-regret regardless of the network size in the learning, and further it has the close-to-optimal stability region that equals $\frac{k-1}{k+1}\Lambda$ in the scheduling.

IV. Performance Evaluation

We first consider the regret performance of A^k-UCB in a single frame, and then evaluate its scheduling performance across frames.

A. Regret Performance in a Single Frame

We show that A^k-UCB has distribution-dependent upper bound of $O(\log T)$ on the regret in a frame of length T. We start with the following lemma.

Lemma 1. *Given any $\mathbf{S}_{t-1}$, weight $\bar{\boldsymbol{w}}_t$, and a fixed $k > 0$, there exists $\delta > 0$ such that, with probability at least δ, the augmentation algorithm generates a set $\mathcal{A}^*$ of disjoint augmentations that satisfies $(\mathbf{S}_{t-1} \oplus \mathcal{A}^*) \in \mathcal{S}^{\alpha}_{\bar{\boldsymbol{w}}_t}$, i.e., $\Pr\{(\mathbf{S}_{t-1} \oplus \mathcal{A}^*) \in \mathcal{S}^{\alpha}_{\bar{\boldsymbol{w}}_t}\} \geq \delta$, or equivalently,*

$$\Pr\left\{ r_{\bar{\boldsymbol{w}}_t}(\mathbf{S}_{t-1} \oplus \mathcal{A}^*) \geq \tfrac{k-1}{k+1} \cdot r^*_{\bar{\boldsymbol{w}}_t} \right\} \geq \delta, \tag{8}$$

where $\delta \geq \min\{1, (\frac{p}{1-p})^{|\mathcal{V}|}\} \cdot (\frac{1-p}{k\Sigma})^{|\mathcal{V}|}$, $|\mathcal{V}|$ is the number of nodes, and Σ is the maximum node degree.

Lemma 1 means that the augmentation algorithm is an (α, β)-approximation oracle with $\alpha = \frac{k-1}{k+1}$ and $\beta > 0$, where β can be arbitrarily small according to the network size. The proof follows the same line of analysis of [11] and thus omitted. For the completeness, we provide the proof in [30].

Next, we need a generalized version of the decomposition inequality for α-regret. From (5), we have

$$\begin{aligned} Reg^{\alpha}(t) &= t \cdot \alpha \cdot r^*_{\boldsymbol{w}} - \mathbb{E}\left[\textstyle\sum_{\tau=1}^{t} r_{\boldsymbol{w}}(\mathbf{S}_\tau)\right] \\ &= \textstyle\sum_{\tau=1}^{t} \sum_{S \in \mathcal{S}} \mathbb{E}\left[\mathbb{I}\{\mathbf{S}_\tau = S\} \cdot (\alpha r^*_{\boldsymbol{w}} - r_{\boldsymbol{w}}(S))\right] \\ &\leq \textstyle\sum_{S \in \mathcal{S}} \mathbb{E}[\hat{\tau}_S(t)] \cdot \Delta^{\alpha}_{\max}, \end{aligned} \tag{9}$$

where $\Delta^{\alpha}_{\max} = \alpha \cdot r^*_{\boldsymbol{w}} - \min_{S \in \bar{\mathcal{S}}^{\alpha}} r_{\boldsymbol{w}}(S)$ is the maximum near-optimal gap. Similarly we define the minimum near-optimal gap $\Delta^{\alpha}_{\min} = \alpha \cdot r^*_{\boldsymbol{w}} - \max_{S \in \bar{\mathcal{S}}^{\alpha}} r_{\boldsymbol{w}}(S)$.

The following lemma ensures that, if a non-near-optimal matching in $\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}$ is played many times, then its index sum is smaller than that of any near-optimal matching in $\mathcal{S}^{\alpha}_{\boldsymbol{w}}$.

Lemma 2. *Given a frame of length T, if a non-near-optimal matching $S \in \bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}$ is played more than $l_T = \lceil \frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln T}{\Delta^{\alpha}_{\min}} \rceil$ times by t-th time slot in the frame, then the probability that the total sum of UCB indices over S at time slot t is greater than that over any near-optimal matching $S' \in \mathcal{S}^{\alpha}_{\boldsymbol{w}}$ is bounded by*

$$\Pr\left\{ r_{\bar{\boldsymbol{w}}_t}(S) \geq r_{\bar{\boldsymbol{w}}_t}(S') \right\} \leq 2|\mathcal{L}|t^{-2}, \tag{10}$$

for all $t \leq T$ such that $\hat{\tau}_S(t) \geq l_T$.

We emphasize that $\mathcal{S}^{\alpha}_{\boldsymbol{w}}$ and $\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}$ are defined with true weight $\boldsymbol{w}$, while the matching comparison is based on UCB index $\bar{\boldsymbol{w}}_t$. The lemma shows that the augmentation algorithm may still work well, even when the true weight is replaced with the

UCB index. The proof of the lemma is analogous to Lemma $A.1$ of [29] but has some differences due to 'near-optimality'. It can be found in [30].

One of our main results, the regret bound of A^k-UCB, can be obtained as follows.

Proposition 1. *For a network graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, A^k-UCB achieves the regret performance bound of*

$$Reg^{\alpha}(t) \leq \Delta^{\alpha}_{\max}\left[D_1 \cdot \frac{\log t}{(\Delta^{\alpha}_{\min})^2} + D_2\right],$$

for all $t \in \{1, \cdots, T\}$, where $D_1 = \left(1 + \frac{1}{\delta}\right) \cdot 4|\mathcal{L}|^2(|\mathcal{L}|+1) \cdot (|\mathcal{S}|-1)$, $D_2 = \frac{|\mathcal{S}|-1}{\delta}(1 + \frac{|\mathcal{L}|\delta\pi^2}{3} + \frac{|\mathcal{L}|(|\mathcal{S}|-2)\pi^2}{6}) + \frac{1-\delta}{\delta} + \frac{2|\mathcal{L}|\pi^2}{3\delta}$, $\alpha = \frac{k-1}{k+1}$, and $\delta = \min\{1, (\frac{p}{1-p})^{|\mathcal{V}|}\} \cdot (\frac{1-p}{k\Sigma})^{|\mathcal{V}|}$.

It shows that A^k-UCB achieves the logarithmic growth $O(\log T)$ of α-regret. Note that although the result is somewhat similar to those in [20], [25], their proof techniques are not applicable. Suppose that at time slot t, A^k-UCB randomly generates a set $\mathcal{A}_t$ of augmentations based on the previous schedule $\mathbf{S}_{t-1}$, and $\mathcal{A}_t$ consists of a single augmentation for the ease of exposition. It is possible that both the matchings, previous schedule $\mathbf{S}_{t-1}$ and schedule $\mathbf{S}_{t-1} \oplus \mathcal{A}_t$ generated by the augmentation algorithm, are non-near-optimal. This implies that we cannot ensure that the index sum of the chosen schedule (i.e., either $\mathbf{S}_{t-1}$ or $\mathbf{S}_{t-1} \oplus \mathcal{A}_t$) is greater than $\alpha r^*_{\bar{\boldsymbol{w}}_t}$, because the index-sum comparison is done only between the two non-near-optimal matchings. This, the lack of comparison with the optimal matching (or near-optimal matching in our case) at every time slot, makes the previous regret analysis technique non-applicable. We successfully address the difficulties by grouping the plays of non-near-optimal matchings.

Overall, we show that the number of explorations to non-near-optimal matchings is bounded. To this end, we consider a sequence of time points where a non-near-optimal matching is sufficiently played at each point. They serve as a foothold to count the total number of plays of non-near-optimal matchings.

To begin with, for an arbitrary fixed time $h > 0$, let $l_h = \lceil \frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln h}{(\Delta^{\alpha}_{\min})^2} \rceil$, and let $\hat{t}_h$ denote the first time when all non-near-optimal matchings are sufficiently (i.e., more than l_h times) explored, i.e.,

$$\hat{t}_h = \min\left\{t \;\middle|\; \hat{\tau}_S(t) \geq l_h \text{ for all } S \in \bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}\right\}.$$

(1) When $\hat{t}_h \leq h$: Let $\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}} = \{S^1, S^2, \cdots, S^M\}$ with $M = |\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}|$. Further we define $\overline{\mathcal{S}}(t) = \{S \in \bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}} \mid \hat{\tau}_S(t) \geq l_h\}$, which is the set of non-near-optimal matchings that are scheduled sufficiently many times by time t, and $\underline{\mathcal{S}}(t) = \bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}} - \overline{\mathcal{S}}(t)$ denotes the set of not-yet-sufficiently-scheduled non-near-optimal matchings. Also, let t^n denote the time when matching S^n is sufficiently scheduled, i.e., $\hat{\tau}_{S^n}(t^n) = l_h$. Without loss of generality, we assume $t^1 < t^2 < \cdots < t^M = \hat{t}_h$.

To apply the decomposition inequality (9), we need to estimate the expected value of $\sum_{S\in\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}} \hat{\tau}_S(\hat{t}_h)$, which can be written as

$$\begin{aligned}&\textstyle\sum_{S\in\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}} \hat{\tau}_S(\hat{t}_h) = \sum_{S\in\bar{\mathcal{S}}^{\alpha}_{\boldsymbol{w}}} \sum_{t=1}^{\hat{t}_h} \mathbb{I}\{\mathbf{S}_t = S\}\\ &= l_h M + \textstyle\sum_{n=1}^{M-1} \sum_{t=t^n+1}^{t^{n+1}} \sum_{S\in\overline{\mathcal{S}}(t^n)} \mathbb{I}\{\mathbf{S}_t = S\}.\end{aligned} \tag{11}$$

Hence, we need to estimate $\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S\}$ for $t \in (t^n, t^{n+1}]$, which can be obtained in the following lemma.

Lemma 3. *For each $t \in (t^n, t^{n+1}]$, we have*

$$\begin{aligned}&\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S\}\\ &\leq (1-\delta) \cdot \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_{t-1} = S\}\\ &\quad + \Pr\{\mathbf{S}_{t-1} \in \underline{\mathcal{S}}(t^n)\} + \left(|\overline{\mathcal{S}}(t^n)| + \delta\right) \cdot 2|\mathcal{L}|t^{-2}.\end{aligned} \tag{12}$$

Proof. We first divide the case into three exclusive sub-cases based on the previous schedule $\mathbf{S}_{t-1}$: events $\mathbb{A} = \{\mathbf{S}_{t-1} \in \mathcal{S}^{\alpha}_{\boldsymbol{w}}\}$, $\mathbb{B} = \{\mathbf{S}_{t-1} \in \underline{\mathcal{S}}(t^n)\}$, and $\mathbb{C} = \{\mathbf{S}_{t-1} \in \overline{\mathcal{S}}(t^n)\}$. Then we have

$$\begin{aligned}&\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S\}\\ &= \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\} &(13)\\ &\quad + \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{B}\} \cdot \Pr\{\mathbb{B}\} &(14)\\ &\quad + \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{C}\} \cdot \Pr\{\mathbb{C}\}. &(15)\end{aligned}$$

Let $\mathcal{A}_t$ denote the set of augmentations chosen under our algorithm at time t. We can obtain a bound on (13) as

$$\begin{aligned}&\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\}\\ &\leq \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{r_{\bar{\boldsymbol{w}}_t}(S) \geq r_{\bar{\boldsymbol{w}}_t}(\mathbf{S}_{t-1}) \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\}\\ &\leq |\overline{\mathcal{S}}(t^n)| \cdot 2|\mathcal{L}|t^{-2},\end{aligned} \tag{16}$$

where the last inequality comes from Lemma 2. The result holds for all $t \in (t^n, t^{n+1}]$. For the second term (14), we have

$$\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{B}\} \cdot \Pr\{\mathbb{B}\} \leq \Pr\{\mathbb{B}\}. \tag{17}$$

Finally, the third term (15) denotes the probability to transit from a sufficiently-played non-near-optimal matching to a sufficiently-played non-near-optimal matching, and thus we have

$$\begin{aligned}&\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t = S \mid \mathbb{C}\} \cdot \Pr\{\mathbb{C}\}\\ &= \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_t \in \overline{\mathcal{S}}(t^n) | \mathbf{S}_{t-1} = S\} \cdot \Pr\{\mathbf{S}_{t-1} = S\}.\end{aligned}$$

Letting $S' = S \oplus \mathcal{A}_t$ and using Lemma 1, the conditional probability can be derived as

$$\begin{aligned}&\Pr\{\mathbf{S}_t \in \overline{\mathcal{S}}(t^n) | \mathbf{S}_{t-1} = S\}\\ &\leq \Pr\{\mathbf{S}_t \in \overline{\mathcal{S}}(t^n) | \mathbf{S}_{t-1} = S, S' \in \mathcal{S}^{\alpha}_{\boldsymbol{w}}\} \cdot \delta + (1-\delta)\\ &= \Pr\{r_{\bar{\boldsymbol{w}}_t}(S) \geq r_{\bar{\boldsymbol{w}}_t}(S')\} \cdot \delta + (1-\delta)\\ &\leq 2|\mathcal{L}|t^{-2} \cdot \delta + (1-\delta).\end{aligned}$$

where the equality holds since $\mathbf{S}_t$ should be S (otherwise, $\mathbf{S}_t = (S \oplus \mathcal{A}_t) \notin \overline{\mathcal{S}}(t^n)$) and thus S should have the larger weight sum to be chosen by the augmentation algorithm, and the last inequality comes from Lemma 2. Hence, the third term (15) can be upper bounded by

$$\begin{aligned}&\textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_{t-1} = S\} \cdot \left(2\delta|\mathcal{L}|t^{-2} + 1 - \delta\right)\\ &\leq 2\delta|\mathcal{L}|t^{-2} + (1-\delta) \textstyle\sum_{S\in\overline{\mathcal{S}}(t^n)} \Pr\{\mathbf{S}_{t-1} = S\},\end{aligned} \tag{18}$$

for all $t \in (t^n, t^{n+1}]$.

The result can be obtained by combining (16), (17), and (18). □

In order to apply Lemma 3 to (11), we rewrite it in a recursive form. Let $\eta = 1 - \delta$, $G_n = \left(|\overline{\mathcal{S}}(t^n)| + \delta\right) \cdot 2|\mathcal{L}| =$

$(n+\delta)\cdot 2|\mathcal{L}|$, and $\Theta_n(t)=\sum_{S\in\overline{\mathcal{S}}(t^n)}\Pr\{\mathbf{S}_t=S\}$. We have a recursive form of (12) as

$$\Theta_n(t)\le \Pr\{\mathbf{S}_{t-1}\in\underline{\mathcal{S}}(t^n)\}+G_n\cdot t^{-2}+\eta\Theta_n(t-1),$$

for $t\in(t^n,t^{n+1}]$. Extending the right side further down to t^n, we can obtain that

$$\begin{aligned}\Theta_n(t)&\le \eta^{t-t^n}\Theta_n(t^n) &(19)\\ &\quad+G_n\textstyle\sum_{i=t^n+1}^{t}\eta^{t-i}\cdot i^{-2} &(20)\\ &\quad+\textstyle\sum_{i=t^n+1}^{t}\eta^{t-i}\cdot\Pr\{\mathbf{S}_{i-1}\in\underline{\mathcal{S}}(t^n)\}. &(21)\end{aligned}$$

By summing it over $t\in(t^n,t^{n+1}]$ on the both sides, we obtain the following lemma.

Lemma 4. *The total number of times that sufficiently played non-near-optimal matchings are selected during $(t^n,t^{n+1}]$ is bounded by*

$$\textstyle\sum_{t=t^n+1}^{t^{n+1}}\Theta_n(t)\le\frac{1}{\delta}(1+\frac{\pi^2}{6}G_n+\mathbb{E}[\sum_{S\in\underline{\mathcal{S}}(t^n)}\tau_{S,n+1}]), \quad (22)$$

where $\tau_{S,n+1}$ denote the number of time slots that S is scheduled in $(t^n,t^{n+1}]$.

The proof of Lemma 4 is omitted and can be found in [30].

Now, by taking expectation on (11), we can obtain the expected total number of times that non-near optimal matchings are selected up to time $\hat{t}_h(\le h)$ as

$$\begin{aligned}&\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}\left[\hat{\tau}_S(\hat{t}_h)\right]=l_hM+\sum_{n=1}^{M-1}\sum_{t=t^n+1}^{t^{n+1}}\Theta_n(t)\\ &\le l_hM+\tfrac{1}{\delta}\textstyle\sum_{n=1}^{M-1}\left(1+G_n\frac{\pi^2}{6}+\mathbb{E}\Big[\sum_{S\in\underline{\mathcal{S}}(t^n)}\tau_{S,n+1}\Big]\right),\\ &\le l_hM+\tfrac{M}{\delta}\left(1+\tfrac{|\mathcal{L}|\delta\pi^2}{3}+\tfrac{|\mathcal{L}|(M-1)\pi^2}{6}+l_h\right).\end{aligned}$$

The last inequality holds since (i) $\sum_{n=1}^{M-1}G_n=\sum_{n=1}^{M-1}(n+\delta)\cdot 2|\mathcal{L}|\le M\cdot|\mathcal{L}|\cdot(2\delta+(M-1))$, and (ii) $\sum_{S\in\underline{\mathcal{S}}(t^n)}\tau_{S,n+1}$ is the total number that the matchings that have been chosen less than l_h up to t^n are chosen during $(t^n,t^{n+1}]$ and thus results in $\sum_{n=1}^{M-1}\sum_{S\in\underline{\mathcal{S}}(t^n)}\tau_{S,n+1}\le\sum_{k=2}^{M}l_h\le l_hM$. From $M\le|\mathcal{S}|-1$, we have

$$\begin{aligned}\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}\left[\hat{\tau}_S(\hat{t}_h)\right]&\le D_1\cdot\tfrac{\ln h}{(\Delta^\alpha_{\min})^2}\\ &\quad+\tfrac{|\mathcal{S}|-1}{\delta}(1+\tfrac{|\mathcal{L}|\delta\pi^2}{3}+\tfrac{|\mathcal{L}|(|\mathcal{S}|-2)\pi^2}{6}). &(23)\end{aligned}$$

This provides a bound on the number of times that non-near-optimal matchings are selected up to $\hat{t}_h$. For the rest time $t\in(\hat{t}_h,h]$, we need to compute $\sum_{t=\hat{t}_h+1}^{h}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}$. Let $S'=\mathbf{S}_{t-1}\oplus\mathcal{A}_t$. Since next schedule $\mathbf{S}_t$ is either $\mathbf{S}_{t-1}$ and S' under the algorithm, we divide the event $\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}$ into three sub-cases based on $\mathbf{S}_{t-1}$ and S', and compute the probability as

$$\begin{aligned}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}&=\Pr\{S'\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}},\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\\ &\quad+\Pr\{S'\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}},\ \mathbf{S}_{t-1}\in\mathcal{S}^\alpha_{\boldsymbol{w}},r_{\boldsymbol{w}}(S')\ge r_{\boldsymbol{w}}(\mathbf{S}_{t-1})\}\\ &\quad+\Pr\{S'\in\mathcal{S}^\alpha_{\boldsymbol{w}},\ \mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}},r_{\boldsymbol{w}}(S')\le r_{\boldsymbol{w}}(\mathbf{S}_{t-1})\}.\end{aligned}$$

This leads to

$$\begin{aligned}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}&\le\Pr\{S'\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\mid\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\cdot\Pr\{\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\\ &\quad+\Pr\{r_{\boldsymbol{w}}(S')\ge r_{\boldsymbol{w}}(\mathbf{S}_{t-1})\mid S'\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}},\ \mathbf{S}_{t-1}\in\mathcal{S}^\alpha_{\boldsymbol{w}}\}\\ &\quad+\Pr\{r_{\boldsymbol{w}}(S')\le r_{\boldsymbol{w}}(\mathbf{S}_{t-1})\mid S'\in\mathcal{S}^\alpha_{\boldsymbol{w}},\ \mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}.\end{aligned}$$

From Lemma 1, we have $\Pr\{S'\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\mid\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}=1-\Pr\{S'\in\mathcal{S}^\alpha_{\boldsymbol{w}}\mid\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\le 1-\delta=\eta$. Since $\hat{\tau}_S(t)\ge l_h$ for all S and $t\in(\hat{t}_h,h]$, Lemma 2 provides an upper bound $2|\mathcal{L}|t^{-2}$ on each conditional probability in the second and the third terms. As a result, we can obtain

$$\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\le\eta\cdot\Pr\{\mathbf{S}_{t-1}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}+4|\mathcal{L}|t^{-2}.$$

By extending the inequality in a recursive manner down to $\hat{t}_h$, we obtain that

$$\begin{aligned}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}&\le\eta^{t-\hat{t}_h}\cdot\Pr\{\mathbf{S}_{\hat{t}_h}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\\ &\quad+4|\mathcal{L}|\textstyle\sum_{i=\hat{t}_h+1}^{t}\eta^{t-i}i^{-2}\\ &=\eta^{t-\hat{t}_h}+4|\mathcal{L}|\textstyle\sum_{i=\hat{t}_h+1}^{t}\eta^{t-i}i^{-2},\end{aligned}$$

where the last equality holds since $\Pr\{\mathbf{S}_{\hat{t}_h}\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}=1$ from the definition of $\hat{t}_h$. Summing over $t\in(\hat{t}_h,h]$ on the both sides, and from $\eta=1-\delta$, we have

$$\textstyle\sum_{t=\hat{t}_h+1}^{h}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\}\le\frac{1-\delta}{\delta}+\frac{2|\mathcal{L}|\pi^2}{3\delta}. \quad (24)$$

Combining (23) and (24), we obtain

$$\begin{aligned}&\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}[\hat{\tau}_S(h)]\\ &=\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}\left[\hat{\tau}_S(\hat{t}_h)\right]+\sum_{t=\hat{t}_h+1}^{h}\Pr\{\mathbf{S}_t\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}\} &(25)\\ &\le D_1\cdot\tfrac{\ln h}{(\Delta^\alpha_{\min})^2}+D_2,\end{aligned}$$

where $D_1=\left(1+\frac{1}{\delta}\right)\cdot 4|\mathcal{L}|^2(|\mathcal{L}|+1)\cdot(|\mathcal{S}|-1)$, and $D_2=\frac{|\mathcal{S}|-1}{\delta}(1+\frac{|\mathcal{L}|\delta\pi^2}{3}+\frac{|\mathcal{L}|(|\mathcal{S}|-2)\pi^2}{6})+\frac{1-\delta}{\delta}+\frac{2|\mathcal{L}|\pi^2}{3\delta}$.

(2) When $\hat{t}_h>h$ (i.e,. $\exists S$ such that $\hat{\tau}_S(h)<l_h$) : With the same definitions of l_h, $\overline{\mathcal{S}}(t)$, and $\underline{\mathcal{S}}(t)$, let $|\overline{\mathcal{S}}|=|\overline{\mathcal{S}}(h)|$ and $|\underline{\mathcal{S}}|=|\overline{\mathcal{S}}(h)|$. At this time, we define $\overline{\mathcal{S}}(h)=\{S^1,S^2,\cdots,S^{|\overline{\mathcal{S}}|}\}$ and let t^n denote the time at which matching S^n is sufficiently scheduled, i.e., $\hat{\tau}_{S^n}(t^n)=l_h$. Without loss of generality, we assume $t^1<t^2<\cdots<t^{|\overline{\mathcal{S}}|}$. By time slot h, $\underline{\mathcal{S}}(h)$ is non-empty (since $\hat{t}_h>h$), and it is clear that $\sum_{S\in\underline{\mathcal{S}}(h)}\hat{\tau}_S(h)\le l_h|\underline{\mathcal{S}}|$.

Similar to the case when $\hat{t}_h\le h$, we can obtain

$$\begin{aligned}&\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}\left[\hat{\tau}_S(h)\right]\\ &=\textstyle\sum_{S\in\underline{\mathcal{S}}(h)}\mathbb{E}[\hat{\tau}_S(h)]+\sum_{t=1}^{h}\sum_{S\in\overline{\mathcal{S}}(h)}\Pr\{\mathbf{S}_t=S\}\\ &\le l_h|\underline{\mathcal{S}}|+l_h|\overline{\mathcal{S}}|+\textstyle\sum_{n=1}^{|\overline{\mathcal{S}}|}\sum_{t=t^n+1}^{t^{n+1}}\Theta_n(t)\\ &\le l_hM+\textstyle\sum_{n=1}^{|\overline{\mathcal{S}}|}\frac{1}{\delta}(1+\frac{\pi^2}{6}G_n+\mathbb{E}[\sum_{S\in\underline{\mathcal{S}}(t^n)}\tau_{x,n+1}]),\end{aligned}$$

where the last inequality comes from Lemma 4. As in (23), we can obtain

$$\begin{aligned}&\textstyle\sum_{S\in\bar{\mathcal{S}}^\alpha_{\boldsymbol{w}}}\mathbb{E}\left[\hat{\tau}_S(h)\right]\\ &\le l_hM+\tfrac{|\overline{\mathcal{S}}|}{\delta}\left(1+\tfrac{|\mathcal{L}|\pi^2\delta}{3}+\tfrac{|\mathcal{L}|(|\overline{\mathcal{S}}|+1)\pi^2}{6}+l_h\right) &(26)\\ &\le D_1\tfrac{\ln h}{(\Delta^\alpha_{\min})^2}+D_2,\end{aligned}$$

where the last inequality holds due to $|\overline{\mathcal{S}}|\le M-1$. Proposition 1 can be obtained by applying (25) and (26) to the decomposition inequality (9).

Remarks: Despite the logarithmic bound, the algorithm may suffer from slow convergence due to large values of constant D_1 and D_2. On the other hand, the bound is quite loose because we consider each matching separately. In practice, a link belongs to multiple matchings, and thus it can learn much faster.

B. Scheduling Efficiency

We now consider the throughput performance of A^k-UCB across multiple frames. It can be obtained through the Lyapunov technique with time unit of frame length.

Proposition 2. *For a sufficiently large frame length T, A^k-UCB is rate-stable for any arrival rate strictly inside $\frac{k-1}{k+1}\Lambda$.*

Proof. Given any $\boldsymbol{\lambda}$ strictly inside $\alpha\Lambda$ with $\alpha = \frac{k-1}{k+1}$, we consider the Lyapunov function $L(t_n) = \frac{1}{2}\sum_{i\in\mathcal{L}} (q_i(t_n))^2$ at the start time t_n of the n-th frame. If the Lyapunov function has a negative drift for sufficiently large queue lengths, then all the queues will remain finite.

From the queue evolution (1), we have

$$q_i(t_{n+1}) \le \Big(q_i(t_n) - \textstyle\sum_{t=t_n}^{t_n+T-1} X_i(t)\cdot\mathbb{I}\{i\in\mathbf{S}_t\}\Big)^+ + \textstyle\sum_{t=t_n}^{t_n+T-1} a_i(t),$$

where $\{\mathbf{S}_t\}$ denotes the sequence of matchings chosen by A^k-UCB. Let $D(t_n) = L(t_{n+1}) - L(t_n)$. The drift during a frame time can be written as

$$\begin{aligned}\mathbb{E}\left[D(t_n)\ |\boldsymbol{q}(t_n)\right] &\le \tfrac{1}{2}\textstyle\sum_{i\in\mathcal{L}}\mathbb{E}[(\sum_{t=t_n}^{t_n+T-1} a_i(t))^2|\boldsymbol{q}(t_n)]\\ &+ \tfrac{1}{2}\textstyle\sum_{i\in\mathcal{L}}\mathbb{E}[(\sum_{t=t_n}^{t_n+T-1} X_i(t)\cdot\mathbb{I}\{i\in\mathbf{S}_t\})^2|\boldsymbol{q}(t_n)]\\ &+ \textstyle\sum_{t=t_n}^{t_n+T-1}\mathbb{E}[\sum_{i\in\mathcal{L}} q_i(t_n)a_i(t)\\ &- \sum_{i\in\mathbf{S}_t} q_i(t_n)X_i(t))\ |\boldsymbol{q}(t_n)],\end{aligned}$$

where the first two terms can be bounded by CT for some constant C, because $a_i(t)$, $X_i(t)$, and $|\mathcal{L}|$ are bounded. Suppose that we have weight vector $\boldsymbol{w}$ at time t_n. Let S^* denote an optimal matchings during the corresponding frame time, i.e., $S^* \in \mathcal{S}_w^* = \arg\max_{S\in\mathcal{S}}\sum_{i\in S} w_i$, and let $r_{\boldsymbol{w}}^* = \sum_{i\in S^*} w_i$. Since $\boldsymbol{\lambda}$ strictly inside $\alpha\Lambda$, there exists $\epsilon > 0$ such that $\boldsymbol{\lambda} + \epsilon\mathbf{1} \in \alpha\Lambda$, where $\mathbf{1}$ is the vector of all ones. Then from $w_i = \frac{q_i(t_n)}{q^*(t_n)}\mu_i$, we can obtain

$$\begin{aligned}&\mathbb{E}\left[D(t_n)\ |\boldsymbol{q}(t_n)\right]\\ &\le CT + \textstyle\sum_{t=t_n}^{t_n+T-1}(\mathbb{E}[\sum_{i\in\mathcal{L}} q_i(t_n)a_i(t)\ |\boldsymbol{q}(t_n)]\\ &\quad - \mathbb{E}[\textstyle\sum_{i\in\mathbf{S}_t} q_i(t_n)X_i(t)\ |\boldsymbol{q}(t_n)])\\ &= CT + q^*(t_n)\textstyle\sum_{t=t_n}^{t_n+T-1}\left(\sum_{i\in\mathcal{L}}\frac{q_i(t_n)}{q^*(t_n)}\lambda_i - \alpha r_{\boldsymbol{w}}^*\right)\\ &\quad + q^*(t_n)\textstyle\sum_{t=t_n}^{t_n+T-1}\left(\alpha r_{\boldsymbol{w}}^* - \mathbb{E}\left[r_{\boldsymbol{w}}(\mathbf{S}_t)\ |\boldsymbol{q}(t_n)\right]\right)\\ &\le CT - \epsilon T\textstyle\sum_{i\in\mathcal{L}} q_i(t_n) + q^*(t_n)\cdot Reg^\alpha(T),\end{aligned}$$

where the equality holds due to the independence of link rates, and the last inequality holds since $\boldsymbol{\lambda} + \epsilon\mathbf{1} \in \alpha\Lambda$ and thus $\sum_{i\in\mathcal{L}}\frac{q_i(t_n)}{q^*(t_n)}(\lambda_i+\epsilon) < \alpha r_{\boldsymbol{w}}^*$. Dividing both sides by T, we have

$$\tfrac{1}{T}\mathbb{E}\left[D(t_n)\ |\boldsymbol{q}(t_n)\right] \le C - \epsilon\textstyle\sum_{i\in\mathcal{L}} q_i(t_n) + q^*(t_n)\cdot\frac{Reg^\alpha(T)}{T}.$$

Since Proposition 1 implies that $\frac{Reg^\alpha(T)}{T} < \epsilon$ for sufficiently large T, we have a negative drift for sufficiently large queue lengths. □

Proposition 2 means that A^k-UCB can stabilize the queue lengths under packet arrival dynamics for any $\boldsymbol{\lambda} \in \frac{k-1}{k+1}\Lambda$.

V. Distributed Implementation (dA^k-UCB)

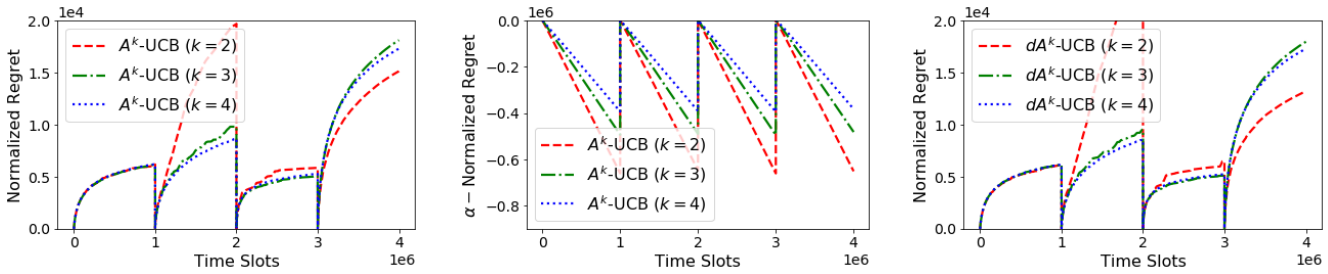
Although the augmentation algorithm has $O(k)$ complexity and amenable to implement in a distributed fashion, the link index $\bar{\boldsymbol{w}}_t$ of A^k-UCB includes *global information* $q^*(t_n)$ – the largest queue length in the network at the start of each frame n. This normalization is due to Hoeffding inequality and essential for the provable regret performance bound.

We pay attention to the fact that A^k-UCB indeed learns the expected value of the queue weighted link rate, i.e., $q_i(t_n)\mu_i$, and the global information $q^*(t_n)$ takes the role of normalizing the weight in the range of $[0,1]$. This implies that it may be able to separate the normalizing parameter from the learning. To this end, we develop a distributed version of A^k-UCB, denoted by dA^k-UCB, and describe it with two key differences. For the ease of exposition, we assume that $\mathcal{A}_t$ consists of a single augmentation.

1) **Local normalizer:** Each node v maintains a local normalizer $\tilde{q}_v$, which is initialized to $\max_{u\in\mathcal{N}(v)} q_{(u,v)}(t_n)$ at the beginning of each frame n. At each time slot t, node v in an augmentation updates its local normalizer twice as follows. 1) In each initialization stage or path augmenting stage, the REQ message from u to v includes additional information of $\tilde{q}_u$. The receiving node v sets $\tilde{q}_v \leftarrow \max\{\tilde{q}_u, \tilde{q}_v\}$. This repeats while building the augmentation. After the cycle-checking stage, the terminus w has $\tilde{q}_w = \tilde{q}^*$ that is the largest local normalizer in the augmentation. 2) In the back-propagating stage, this value $\tilde{q}^*$ is back-propagated together and each node v in the augmentation sets $\tilde{q}_v \leftarrow \tilde{q}^*$. Hence, at the end of the time slot, all the nodes in the augmentation have the same local normalizer $\tilde{q}^*$.
2) **Separate gain computation:** In the meantime, we change the way to compute the gain. To elaborate, let G'_u denote the new gain normalized by $\tilde{q}_u$. At each mini-slot in the path augmenting stage, whenever node u transmits an REQ message to node v, we divide G'_u into $G'_{u,1} + G'_{u,2}$, where $G'_{u,1}$ is for average reward (normalized by factor $\tilde{q}_u$) and $G'_{u,2}$ for confidence interval. They are included in the REQ message, separately. Then, after the receiving node v updates the local normalizer $\tilde{q}_v$, it renormalizes the received reward gain as $G'_{u,1}\cdot\tilde{q}_u/\tilde{q}_v$. Once the next link is decided as $i = (v,n)$, it computes $G'_{v,1}$ accordingly by either adding or subtracting its average reward normalized by $\tilde{q}_v$, i.e., $\hat{w}'_i(t) = \frac{q_i(t_n)}{\tilde{q}_v}\cdot\frac{1}{\hat{\tau}_i(t)}\sum_{j=t_n+1}^{t} X_i(j)\cdot\mathbb{I}\{i\in S_j\}$. $G'_{u,2}$ can be obtained simply by adding the confidence interval. Let A' is the augmentation up to node v, and let $A'_1 = A' - \mathbf{S}_{t-1}$ and let $A'_2 = A'\cap\mathbf{S}_{t-1}$. Then the gains

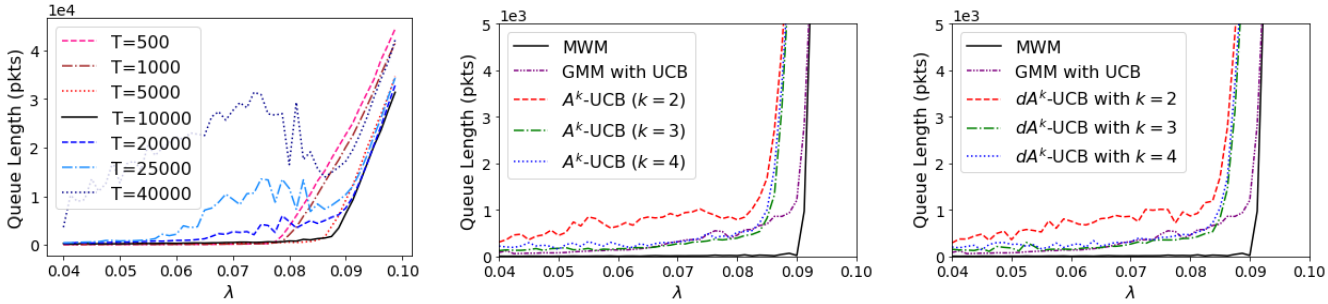
$$\begin{aligned}G'_{v,1} &= \textstyle\sum_{i\in A'_1}\hat{w}'_i - \sum_{j\in A'_2}\hat{w}'_j\\ &= \tfrac{\tilde{q}_u}{\tilde{q}_v}G'_{u,1} + (\mathbb{I}\{v\in A'_1\} - \mathbb{I}\{v\in A'_2\})\cdot\hat{w}'_v,\\ G'_{v,2} &= \textstyle\sum_{i\in A'_1}\sqrt{\frac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_i}} - \sum_{j\in A'_2}\sqrt{\frac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_j}}\\ &= G'_{u,2} + (\mathbb{I}\{v\in A'_1\} - \mathbb{I}\{v\in A'_2\})\cdot\sqrt{\tfrac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_v}},\end{aligned}$$

can be computed given the value of $\tilde{q}_u$ and the gains $G'_{u,1}, G'_{u,2}$. By repeating this during the augmenting



(a) A^k-UCB (b) A^k-UCB (α-regret) (c) $d\mathrm{A}^k$-UCB

Fig. 3. Regret traces for learning performance. For comparison across frames, the regret is reset to 0 at each frame boundary ($T = 10^6$ time slots), and normalized by the maximum expected reward $r^*_{\boldsymbol{w}}$.



(a) A^k-UCB ($k=3$) with different T's (b) A^k-UCB (zoom-in) (c) $d\mathrm{A}^k$-UCB (zoom-in)

Fig. 4. Queue lengths for scheduling efficiency. Given a scheduler, if the arrival rate gets closer to the boundary of its stability region, the queue length soars quickly.

stage, we can obtain the gain normalized by $\tilde{q}^*$ at the terminus.

Remarks: During a frame time, the local normalizer of a node is non-decreasing over time slots. In addition, at the same time slot, two nodes in the network may have a different normalizer value. Hence, our previous analysis results for A^k-UCB cannot be directly applied to $d\mathrm{A}^k$-UCB. However, we highlight that, given a time slot, all the nodes in the same augmentation have the same value of the (local) normalizer, which is of importance, since the gain comparison for making a decision occurs only within an augmentation. On the other hand, as the time slot t increases, the value of the global normalizer $q^*(t_n)$ is disseminated throughout the network and all the local normalizers will converge to this value. Considering that, it is not difficult to show that there exists some T' such that all nodes v have $\tilde{q}_v = q^*(t_n)$ with probability close to 1 for all $t > T'$, we believe that $d\mathrm{A}^k$-UCB also achieves $O(\log T)$ regret performance and $\frac{k-1}{k+1}\Lambda$ capacity, if the frame length T is sufficiently large. Rigorous proof remains as future work.

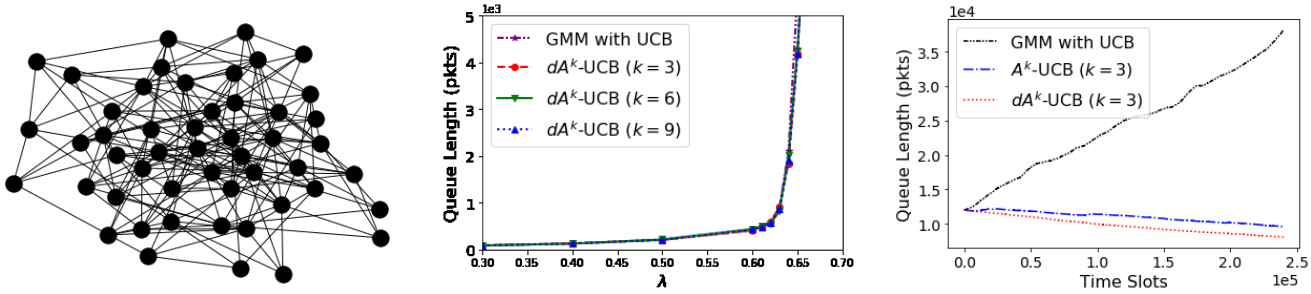
VI. Numerical Results

We evaluate the performance of our proposed schemes through simulations. We first consider a 4x4 grid network topology with the primary interference model, and then conduct extend simulations with a randomly generated network. Time is slotted. At each time slot, a packet arrives at link i with probability λ_i, and for a scheduled link j, a packet successfully departs the link with probability μ_j. Both $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are unknown to the controller. The arrivals and the departures are independent across the links and time slots.

Regret performance: We first investigate the regret performance of A^k-UCB and $d\mathrm{A}^k$-UCB. We set the seed probability $p = 0.2$ for the schemes. We consider a large frame length of $T = 10^6$ time slots to observe their regret growth. An identical arrival rate $\lambda_i = 0.08$ is set for all links i, and the departure rate μ_i is set uniformly at random in range $[0.25, 0.75]$[5]. We simulate the two schemes with different k's and measure their regret performance. For the comparison across frames, the regret value is set to 0 at each frame start, and normalized with respect to the maximum expected reward sum $r^*_{\boldsymbol{w}}$ within the frame.

Fig. 3(a) illustrates the regret traces of A^k-UCB, which is an average of 10 simulation runs. We can observe the logarithmic regret growth in both cases. Recall that our regret analysis in Proposition 1 is for α-regret with $\alpha = \frac{k-1}{k+1}$. Thus empirical performance of the proposed schemes are much better than the analytical bound. The performance in terms of α-regret is shown in Fig. 3(b), where the gap from 0 can be interpreted as the level of practical difficulty in achieving analytic performance bound: as k increases, it harder to achieve $\frac{k-1}{k+1}$-regret. Fig. 3(c) shows the regret values of $d\mathrm{A}^k$-UCB. Comparing with those of A^k-UCB, they achieve similar learning performance in terms of regret, and thus we can conclude that the performance loss due to local normalizer

[5]Due to randomized μ_i, each link has different traffic load despite the identical arrival rate. Setting a different arrival rate for each link leads to similar results.



(a) A randomly generated network (b) Performance in the random network (c) Performance in a ring network

Fig. 5. Stability in different network topologies. dA^k-UCB achieves high performance in a larger randomly-generated network, and GMM with UCB may suffer from low performance in a specific ring network.

in dA^k-UCB is negligible in practice.

Scheduling efficiency: We evaluate scheduling efficiency of the proposed schemes in the grid network. We use the same simulation settings, but change frame length T and arrival rate λ ($\lambda_i = \lambda$ for all i). By increasing λ, the arrival rate gets closer to the boundary of its stability region, and when this occurs, the queue length will soar quickly. We conduct each simulation for 10^6 time slots and measure queue lengths when the simulations end. For each λ, we average the queue length of 10 simulation runs. By observing the arrival rate where the queue length starts increasing quickly, we can indirectly compare the achievable stability region $\gamma\Lambda$ for different scheduling policies [7]. A policy with larger γ is better.

Fig. 4(a) demonstrates how the bound changes according to the frame length. From the results, we can observe that the critical point of λ, around which the queue length starts soaring, is increasing for $T \leq 10^6$ and then decreasing for $T \geq 2{\cdot}10^6$. This is somewhat expected, since too small frame length will lead to incomplete learning, and a larger frame length results in a relatively slower response to the queue dynamics.

We now evaluate the performance of A^k-UCB in comparison with two schemes of MWM and UCB-based GMM. MWM is a well-known optimal scheduler [8], and it is a centralized algorithm that not only requires the knowledge about the weight $q_i(t)\mu_i$ at each time slot t, but also has a high-order computational complexity. In our simulations, we use its performance as a reference value. The UCB-based GMM [19]) finds a matching by including the link with the highest UCB index first. It is known to achieves $\frac{1}{2}\Lambda$ and has the linear computational complexity.

Fig. 4(b) demonstrate the queue lengths of MWM, UCB-based GMM, and A^k-UCB. MWM operates at each time slot, and for UCB-based GMM and A^k-UCB, we use $T = 5000$. Each simulation runs for 10^6 time slots (i.e., 200 frames) and we measure the queue lengths after the simulations. For MWM, the queue length quickly increases at around $\lambda = 0.09$, which can be considered as the boundary of the capacity region Λ. Under A^k-UCB with $k = 2, 3, 4$, the queue lengths increase quickly around $\lambda = 0.084$ for all k, which exceeds their theoretic bound $\frac{k-1}{k+1} \cdot 0.09 = 0.03, 0.045, 0.054$, respectively. Interestingly, the impact of k on throughput is not significant, which seems to be due to the small network size – we can observe that a larger k leads to lower queue lengths in all arrival rates and thus achieves better delay performance. The UCB-based GMM achieves the performance closet to that of MWM, which is also far beyond its theoretic bound $\frac{1}{2}$. Fig. 4(c) shows that dA^k-UCB achieves almost the same performance as A^k-UCB.

Performance in randomly generated networks: We now evaluate the performance of the proposed schemes in a larger, irregular-shaped network. To this end, we randomly generate a network of 50 nodes and 200 links as shown in Fig. 5(a), and run simulations for 1000 frame times with $T = 500$ (i.e., total $5 \cdot 10^5$ time slots). For each link i, we set the successful transmission rate μ_i uniformly at random in range $[0.25, 0.75]$, and set the arrival rate as $\lambda_i = \lambda \cdot \rho_i$, where ρ_i is chosen uniformly at random in range $[0.4, 0.7]$. Due to high computational complexity of MWM, we simulate only UCB-based GMM, A^k-UCB, and dA$^k-$UCB in this experiment, and use the performance of UCB-based GMM as a reference value. It has been observed that GMM algorithm often achieves the optimal scheduling performance in this randomized network environment [7].

Fig. 5(b) demonstrates the queue lengths of UCB-based GMM and dA^k-UCB with $k = 3, 6, 9$. All 4 schemes achieve almost-identical performance, and the setting of k is not sensitive to the performance. The results of A^k-UCB are almost identical to that of dA^k-UCB, and thus omitted.

Low performance of UCB-based GMM: So far, UCB-based GMM achieves close-to-optimal performance despite its low performance guarantee of $\frac{1}{2}\Lambda$. It is an interesting question whether the performance bound is not tight due to technical difficulties and its true performance is close to the optimal. Unfortunately, however, we shows in the next experiment that this is not the case, and UCB-based GMM may suffer from low performance in a certain scenario. We consider a 6-link ring topology, where the links are numbered from 1 to 6 in a clockwise direction. The service rate of each link follows a Bernoulli distribution with mean $\frac{1}{2}$ and the packet arrival on each link is also a Bernoulli process with mean[6] $\frac{1}{6} + \epsilon$ where $\epsilon = 0.08$. We set the frame length $T = 6000$. Other environment settings are the same as before, except that

[6]This implies that $\boldsymbol{\lambda} \in (\frac{2}{3} + 4\epsilon)\Lambda$ since an optimal scheduler can support arrival rate of up to $\frac{1}{4}$ on each link.

the initial queue length is $\{\frac{3T}{6}, \frac{2T}{6}, \frac{T}{6}, \frac{3T}{6}, \frac{2T}{6}, \frac{T}{6}\}$. Fig. 5(c) shows the queue length traces for UCB-based GMM and our proposed schemes. We can observe that while the queue lengths of A^k-UCB and dA^k-UCB are stabilized, those of the UCB-based GMM keep increasing. This is because, the greedy algorithm tends select a matching with the two links of the largest queue at the beginning of each frame. In contrast, A^k-UCB and dA^k-UCB select a matching with three links by considering their weight sum. This result implies that in a certain circumstance, UCB-based GMM may suffer from low scheduling efficiency.

VII. Conclusion

In this work, we addressed the joint problem of learning and scheduling in multi-hop wireless networks. Without a priori knowledge on link rates, we aim to find a sequence of schedules such that all the queue lengths remain finite under packet arrival dynamics. By incorporating the augmentation algorithm into a learning procedure, we develop provably efficient low-complexity schemes that i) achieve logarithmic regret growth in learning, and ii) have the throughput performance that can be arbitrarily close to the optimal. We extend the result to a distributed scheme that is amenable to implement in large-scale networks. We also verify our results through simulations.

References

[1] C. Joo, G. Sharma, N. B. Shroff, and R. R. Mazumdar, "On the complexity of scheduling in wireless networks," *EURASIP J. Wireless Commun. Netw.*, Oct. 2010.

[2] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[3] C. Joo, X. Lin, and N. B. Shroff, "Greedy maximal matching: Performance limits for arbitrary network graphs under the node-exclusive interference model," *IEEE Trans. Autom. Control*, vol. 54, no. 12, pp. 2734–2744, Dec. 2009.

[4] C. Joo and N. B. Shroff, "Local greedy approximation for scheduling in multi-hop wireless networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 3, pp. 414–426, Mar. 2012.

[5] X. Wu, R. Srikant, and J. R. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 595–605, 2007.

[6] X. Lin and S. B. Rasool, "Distributed and provably efficient algorithms for joint channel-assignment, scheduling, and routing in multichannel Ad hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1874–1887, 2009.

[7] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, Oct. 2009.

[8] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximal throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[9] J. Choi, "On improving throughput of multichannel ALOHA using preamble-based exploration," *J. Commun. Netw.*, vol. 22, no. 5, pp. 380–389, 2020.

[10] B. Hajek and G. Sasaki, "Link scheduling in polynominal time," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, Sept. 1988.

[11] L. Bui, S. Sanghavi, and R. Srikant, "Distributed link scheduling with constant overhead," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1467–1480, Oct. 2009.

[12] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 13, pp. 960–972, June 2010.

[13] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, June 2012.

[14] C. Joo, "On random access scheduling for multimedia traffic in multi-hop wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, pp. 647–656, Apr. 2013.

[15] S. A. Borbash and A. Ephremides, "Wireless link scheduling with power control and SINR constraints," *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 5106–5111, Nov. 2006.

[16] J.-G. Choi, C. Joo, J. Zhang, and N. B. Shroff, "Distributed link scheduling under SINR model in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1204–1217, Aug. 2014.

[17] F. Li, D. Yu, H. Yang, J. Yu, H. Karl, and X. Cheng, "Multi-amed-bandit-based spectrum scheduling algorithms in wireless networks: A survey," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 24–30, 2020.

[18] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in Ad hoc networks: A POMDP framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, pp. 589–600, Apr. 2007.

[19] T. Stahlbuhk, B. Shrader, and E. Modiano, "Learning algorithms for scheduling in wireless networks with unknown channel statistics," *Ad Hoc Netw.*, vol. 85, pp. 131–144, 2019.

[20] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. ICML*, 2013.

[21] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002.

[23] V. Anantharam, P. Varaiya, and J. Walrand, "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part I: I.I.D. rewards," *IEEE Trans. Autom. Control*, vol. 32, no. 11, pp. 968–976, Nov. 1987.

[24] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Trans. Signal Processing*, vol. 58, no. 11, pp. 5667–5681, Nov. 2010.

[25] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.

[26] Y. Gai and B. Krishnamachari, "Decentralized online learning algorithms for opportunistic spectrum access," in *Proc. IEEE GLOBECOM*, Dec. 2011.

[27] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 4, pp. 731–745, Apr. 2011.

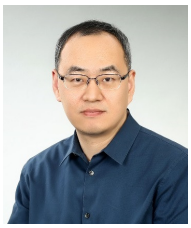
[28] H. Tibrewal, S. Patchala, M. K. Hanawal, and S. J. Darak, "Distributed learning and optimal assignment in multiplayer heterogeneous networks," in *Proc. IEEE INFOCOM*, 2019.

[29] S. Kang and C. Joo, "Low-complexity learning for dynamic spectrum access in multi-user multi-channel networks," *IEEE Trans. Mobile Comput.*, 2021.

[30] D. Park, S. Kang, and C. Joo, "Distributed link scheduling with unknown link rates in multi-hop wireless networks," https://www.dropbox.com/s/uh07dc2xbj55dgm/tech_report.pdf?dl=0, 2021.

Daehyun Park received his M.S. degree from the school of ECE at Ulsan National Institute of Science and Technology (UNIST) in 2020. His research interests include multi-armed bandits.

Sunjung Kang received her M.S. degree from the school of ECE at Ulsan National Institute of Science and Technology (UNIST) in 2018. She is currently a Ph.D. student in the department of ECE at The Ohio State University. Her research interests include the age of information, remote estimation and multi-armed bandits.



Changhee Joo received the Ph.D. degree from Seoul National University in 2005. He was with Purdue University and The Ohio State University, and then worked at Korea University of Technology and Education (KoreaTech), and Ulsan National Institute of Science and Technology (UNIST). Since 2019, he has been with Korea University. His research interests are in the broad areas networking, learning, modeling, and optimization. He was a recipient of the IEEE INFOCOM 2008 Best Paper Award, the KICS Haedong Young Scholar Award (2014), the ICTC 2015 Best Paper Award, and the GAMENETS 2018 Best Paper Award. He was an Associate Editor of the *IEEE/ACM Transactions on Networking*, and currently an Editor of the *IEEE Transactions Vehicular Technology*, a Division Editor of the *Journal of Communications and Networks*, and has served several primary conferences as a technical program committee member, including IEEE INFOCOM, ACM MOBIHOC, IEEE WiOpt, and IEEE GLOBECOM.