Master's Thesis

Distributed Link Scheduling with Unknown Link Rates in Multi-hop Wireless Networks

Daehyun Park

Department of Computer Science and Engineering

Graduate School of UNIST

2020

Distributed Link Scheduling with Unknown Link Rates in Multi-hop Wireless Networks

Daehyun Park

Department of Computer Science and Engineering

Graduate School of UNIST

Distributed Link Scheduling with Unknown Link Rates in Multi-hop Wireless Networks

A thesis/dissertation

submitted to the Graduate School of UNIST

in partial fulfillment of the

requirements for the degree of

Master of Science

Daehyun Park

05.27.2020

Approved by

____

Advisor

Distributed Link Scheduling with Unknown Link Rates in Multi-hop Wireless Networks

Daehyun Park

This certifies that the thesis/dissertation of Daehyun Park is approved.

05.27.2020

signature

___

Advisor: Youngbin Im

signature

___

Changhee Joo

signature

___

Hyoil Kim

Abstract

We address the joint problem of learning and scheduling in multi-hop wireless network without a prior knowledge on link rates. Previous efficient scheduling algorithms need the link rate information and most learning algorithms require a centralized entity with polynomial complexity, which becomes a major obstacle to develop a learning-based scheduling scheme for large-scale multi-hop networks. In this work, we incorporate a low-complexity scheduling algorithm into the learning procedure and develop a new joint scheme for learning and scheduling. We show that it achieves rate optimal performance in learning and also achieves close-to-optimal throughput performance. Further, we extend it to practically distributed algorithm that is amenable to implement in large-scale networks. To our best knowledge, it is the first $O(1)$-complexity distributed scheme for both learning and scheduling. We verify our results through simulations.

Contents

I. INTRODUCTION 2

II. SYSTEM MODEL 4

III. ALGORITHMS 5

3.1 Augmentation algorithm 5

3.2 Learning through augmentation 10

IV. PERFORMANCE EVALUATION 11

4.1 Regret performance in a single frame 12

4.2 Scheduling efficiency 14

V. DISTRIBUTED ALGORITHM 15

VI. NUMERICAL RESULTS 16

VII. CONCLUSION 20

REFERENCES 20

APPENDIX 23

A. PROOF OF LEMMA 1 23

B. PROOF OF PROPOSITION 1 25

C. PROOF OF LEMMA 1 26

D. PROOF OF PROPOSITION 2 27

Figures

FIGURE 1: TIME-SLOT STRUCTURE 6

FIGURE 2: EXAMPLE OPERATION OF AUGMENTATION ALGORITHM
WITH $k = 2$ 9

FIGURE 3: REGRET PERFORMANCE. 17

FIGURE 4: QUEUE LENGTHS 18

FIGURE 5: STABILITY IN A 6-LINK RING TOPOLOGY 19

FIGURE 6 : STABILITY IN A LARGE-SCALE NETWORK TOPOLOGY 20

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 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 [24]. Due to high computational complexity of MWM [14], alternative low-complexity scheduling solutions with comparable performance such as Greedy Maximal Matching (GMM) or Longest Queue First (LQF) have attracted much attention [13][20]. However, since GMM still has linear complexity that increases with the network size, it can be hardly used in large size multi-hop networks.

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 [16]. Random access technique with explicit neighborhood information exchanges has been explored at some expense of performance [15][19][26]. Several studies have shown that the optimal throughput performance is achievable, either by taking the *pick-and-compare* approach [5][10], or by exploiting the carrier-sensing functionality [11][22]. There have been also attempts to develop provably efficient scheduling algorithms that work with time-varying wireless channels [12][16] or with complex SINR interference model [4][7]. The aforementioned scheduling schemes, however, operate with deterministic link rates that are known a priori or at the time of scheduling. The extension of these scheduling schemes to the case when the link rates are unknown is not straightforward.

In this work, we consider the scheduling problem in multi-hop wireless networks, where link rates and statistics are unknown a priori. The uncertainty is often caused by wireless fading, interference, limited feedback, measurement error, system dynamics, etc [23][27]. 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 a sequence of non-interfering link sets (i.e., feasible schedules) to maximize throughput performance while learning link rates.

From 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 [6]. 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 [18], and several index-type learning algorithms with the order-optimal regret have been developed [2][3].

For a large-scale multi-hop wireless network, it is imperative for the algorithms to be amenable to implement in a distributed manner. In [21], 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. The works of [1][8] have addressed the problem in cognitive radio network settings and developed distributed schemes with logarithmic regret through prioritized ranking and adaptive randomization. Although these learning algorithms are amenable to distributed implementation, they are limited to a single-hop network with a fixed number M. Gai et al. [9] and Chen et al. [6] have considered more general problems of combinatorial MAB (CMAB) with arbitrary constraint that is applicable to multi-hop networks. 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 αβ 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.

Recent work of Stahlbuhk et al. [23] is the most related to ours. They have incorporated GMM, which is $\left(\frac{1}{2}, 1\right)$-approximation oracle with linear complexity, for both learning and scheduling in multi-hop wireless networks with unknown link rates, and shown that the GMM-based scheme achieves the logarithmic growth for $\frac{1}{2}$-regret. Albeit interesting, its stability region is limited to $\frac{1}{2}$ of the capacity region and has linear complexity that makes it less attractive for large-size networks. In this work, we consider the joint problem of learning and scheduling, and develop low-complexity scheme that achieves near-full capacity region. Further, we extend it to distributed implementation. Our contribution can be summarized as follows:

- We incorporate the augmentation algorithm of [10] into the CMAB framework.
- We show that the augmentation algorithm, which can be considered as an (α, β)-approximation oracle, achieves the rate-optimal logarithmic growth of $\alpha - regret$, for any $\beta > 0$.
- We show that our scheme achieves the stability region that is arbitrarily close to the capacity region.
- We develop a modified version of the scheme that is amenable to implement in a distributed manner.

The rest of paper is organized as follows. Section II describes our system model. In Section III, we introduce the augmentation algorithm and incorporate it into our joint scheme of learning and scheduling. After analyzing the performance of the proposed scheme in Section IV, we extend our algorithm to distributed implementation in Section V. Finally, we evaluate our schemes through simulations 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 bidirectional links. We assume that the links are 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 in the network can communicate with at most one of its neighbor nodes $\mathcal{N}(v)$, where $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (v, u) \in \mathcal{L}\}$. It can model Bluetooth or FH-CDMA networks as well as captures the essential feature of wireless interference [20][23]. 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 and let $\mathcal{S}$ denote the set of all available matchings in $\mathcal{G}$.

At each link $i \in \mathcal{L}$, we assume packets arrive following a Bernoulli process with probability λ_i. 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, and it is independently drawn from a (possibility 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. We have $\mathbb{E}[X_i(t)] = \mu_i$. We assume that that $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are unknown.

At time slot t, if a policy activates matching S_t, then each link $i \in S_t$ accesses the medium and transmits $X_i(t)$ packets[1] 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

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

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. n-th frame begins at time slot $t_n = (n-1)T + 1$. During the n-th frame, 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

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

[1]or transmits 1 packet with 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

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

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 super arm, and the instance link rate to the reward of the link, respectively. We use the terms interchangeably. Note that the regret its defined as the accumulated expected difference between the reward sum associated with an optimal matching and that obtained by the CMAB algorithm. Similar to [6], we define $\alpha - regret$ as, for some $\alpha \in (0,1]$,

$$Reg^\alpha(t) = t \cdot \alpha \cdot r^\alpha_{\boldsymbol{w}} - \mathbb{E}[\textstyle\sum_{\tau=1}^t r_{\boldsymbol{w}}(S_\tau)], \tag{5}$$

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., $\lim_{t\to\infty} \frac{q_i(t)}{t} = 0$ 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 $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$**,** there exists a policy that can make the network stable. We say that a policy has the stability region $\gamma\Lambda$ for some $\gamma \in (0,1]$, if it can stabilize the networks for any arrival $\boldsymbol{\lambda} \in \gamma\Lambda$

III. ALGORITHMS

Motivated by [5] and [17], we develop an efficient joint scheme of learning and scheduling, by employing augmentation algorithm [5]. We first describe the main characteristics of augmentation algorithm and restate it for the ease of explanation. Then, we incorporate it into our online scheme for learning and scheduling.

3.1 Augmentation algorithm

Firstly, we explain the overall procedure of the original version of algorithm that first appeared in [5], briefly. Based on it, we recompile the algorithm with some changes and restate the description of it in detail, which is necessary for both completeness and accessibility of this work. We start with some definitions. Given a matching S, an *augmentation* A of matching S is a path or cycle where links inside and outside S are connected alternatively 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 $\mathcal{G}$. 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 $S \oplus (\bigcup_{A \in \mathcal{A}} A)$ is also a matching in $\mathcal{G}$. The original algorithm builds such a set of disjoint augmentations in a distributed fashion to make a matching for the next schedule.

The overall procedure of the original augmentation algorithm is as follows. At the beginning of each time slot t, suppose that each link i is associated with some known weight $w_{i,t}$. (i) Given a valid matching S_{t-1} that is the schedule at time $t-1$, it generates a set $\mathcal{A}$ of disjoint augmentations of S_{t-1}, and (ii) compares the weight sum of $A - S_{t-1}$ and $A \cap S_{t-1}$ for each $A \in \mathcal{A}$. Let $B(A)$ be the one with the larger weight sum among the two. (iii) The augmentation algorithm obtains the new schedule S_t as $\bigcup_{A \in \mathcal{A}} B(A) \cup (S_{t-1} - \mathcal{A})$. For the comparison of the weight sum, we define the gain of augmentation $\mathcal{A}$ as

$$G_t^{\boldsymbol{w}}(A) = \sum_{i \in A - S_{t-1}} w_{i,t} - \sum_{j \in A \cap S_{t-1}} w_{j,t} \qquad (6)$$

and obtain new schedule S_t by augmenting S_{t-1} with all $A \in \mathcal{A}$ of $G_t^{\boldsymbol{w}}(A) > 0$. This comparison makes controller select better schedule than prior one.

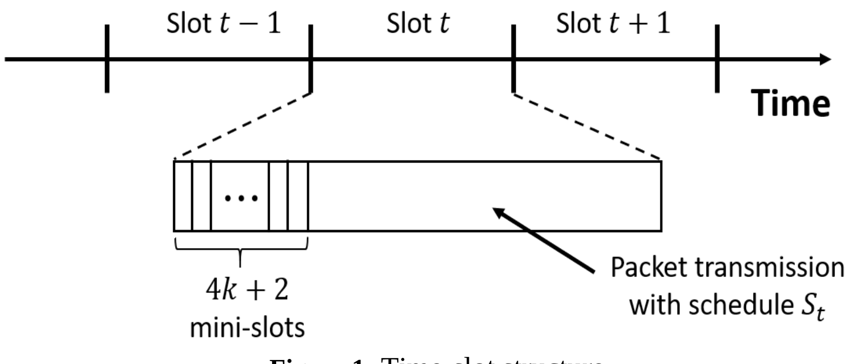


Figure 1: Time-slot structure

Now, we improve the original algorithm to another one for the ease of understanding its implementation and better performance. Also, the revised augmentation algorithm can accomplish the above procedure in a distributed fashion. The algorithm needs two configuration parameters p and k, and consists of the following three stages in each time slot: building augmentations, checking a cycle, and switching(back-propagation)/scheduling. For the ease of exposition, we consider additional time structure of discrete mini-slots, as shown in Fig. 1 and all the three stages end in $4k + 2$ mini-slots. Additionally, let the term "old link" denote a link that belongs to the previous schedule, then S_{t-1} can be seen as a set of old links. The term "new link" is the converse of the concept of old link. In this algorithm, the size of augmentation is defined as the number of new links included in the augmentation. The term "active node" denotes the extending point of augmentation (there is at most one active node for each augmentation).

```
■ Building augmentation
1. Each node selects itself as a seed node with probability $p$.
    (a) Each seed node $s$ initializes its augmentation $A_s$.
    (b) Each seed node $s$ randomly sets $k_s \in [1, k]$.
    (c) Each seed node $s$ becomes the active node of $A_s$
2. At each mini slot $\tau \in [1,2k+1]$,
    (a) Active node $v$ of each augmentation selects node $u$ in $NL(v)$: $u =$ select_next_hop$(v)$.
        i. If $u == NULL$,
            - node $v$ becomes inactive and sets itself as the terminus.
            - Go to 2. (c)    /* $A_s$ with the terminus stops extending */
    (b) Active node $v$ sends REQ to node $u$   /* If node $u$ does not belong to other augmentation
        $A_{s'}$ and receives only one REQ, then node $u$ responds with an RES. */
        i. If node $v$ receives one RES,
            - Append link $(v, u)$ into $A_s$
            - Active node changes from node $v$ to node $u$.
            - If the number of old links in $A_s$ equals $k_s + 1$, node $u$ becomes inactive and sets
              itself as the terminus.
        ii. If node $v$ does not receive any RES,
            - Node $v$ becomes terminus.
            - If link $(v, u) \in S_{t-1}$, append link $(v, u)$ into $A_s$.
    (c) Repeats step 2 for $A_s$ such that
        i. $A_s$ does not have a terminus, and
        ii. the number of new links in $A_s$ is less than or equal to $k_s$.
```

In the original algorithm [5], the order of appending link into augmentation and receiving RES changes when the last included link is old link. We revise it to get appending link after receiving RES regardless of the condition. This revised procedure makes readers understand more clearly and generates the same result of the original one (we add appending condition for the same result in 2.b.ii). Actually, for its implementation, an active node needs to send REQ to a candidate node with some updated information about its augmentation up to that time like its gain $G_t^{\mathbf{w}}(A_s)$. Then, terminus can immediately decide to switch or not when just stop extending. The function *select_next_hop*(v), which requires an active node v, returns a candidate for the next active node to extend augmentation.

Algorithm 1 *Select_next_hop(v)*

```
1:  **if** active node $v$ is the seed node then
2:      **if** $\exists w$ s.t. $(v, w) \in S_{t-1}$ **then**
3:          set $u \leftarrow w$
4:      **else**
5:          set $u$ at random in $\mathcal{N}(v)$
6:      **return** $u$
7: **else** /* node $v$ is not a seed node */
8:      There exists the previous active node $v'$
9:      **if** $(v', v) \notin S_{t-1}$ and $\exists v''$ s.t.$(v, v'') \in S_{t-1}$ **then**
10:         set $u \leftarrow v''$
11:     **else if** $(v', v) \in S_{t-1}$ and $\exists v''\ (\neq v') \in \mathcal{N}(v)$ **then**
12:         set $u$ at random in $\mathcal{N}(v) - \{v'\}$
13:     **else**
14:         set $u \leftarrow NULL$
15:     **return** $u$
```

Remarks: In our description, we present S_{t-1} as if it is a global variable, but each node v indeed requires only the local view of it, i.e., $\{(v, v') \mid v' \in \mathcal{N}(v)\} \cap S_{t-1}$, which can be obtained during the back-propagation in the last stage. After all the stages, a set of augmentations (one per a seed node) will be generated. Since a size of augmentation A_s can have at most $k_s + 1$ links of S_{t-1} and k_s new links, the number of total links in A_s can be up to $2k_s + 1 \leq 2k + 1$.

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 S_{t-1} is marked by thick solid lines in Fig.2 (a). At the beginning of time slot t, each node selects itself as a seed with probability p. Suppose that three nodes are selected in the overall view, which are marked by (white) numbered circles in Fig.2 (a). We use these numbered circles to denote current active nodes. According to the algorithm, active nodes of 1 and 2 will select the previous scheduled link for next active node, while active node 3 will select one of three neighboring nodes at random. These selected next active nodes are pointed by short bold arrows. Then they exchange REQ/RES with necessary information. In the next mini-slot, active nodes change as shown in Fig. $2(b)$. Narrow dotted arrows denote the augmentations up to now. The procedure repeats until at most $2k+1$-th mini-slot, or the augmentation cannot be extended. (the algorithm waits for $2k+1$-th mini slot even if all augmentation cannot be extended before that). In the meantime, if two augmentations collide as shown by active nodes 2 and 3 in Fig. $2(b)$, both the augmentations end.

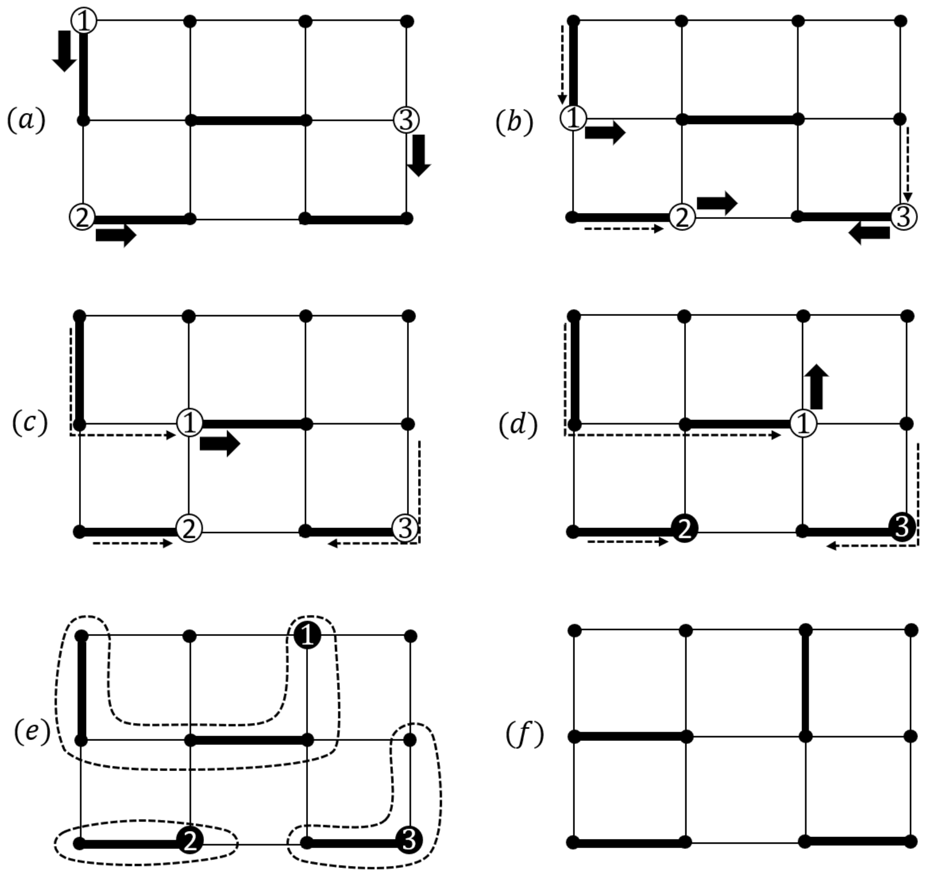


Figure 2: Example operation of Augmentation algorithm with $\boldsymbol{k=2}$

After the procedure, links in augmentation ① are altered.

The node at the collision point belong to the augmentation that follows a link in S_{t-1}, as shown in Fig.2 (c), where the terminus nodes are marked by solid (black) number circles. After the last operation of the $2k+1$-th mini-slot as shown in Fig.2 (d), building the augmentation finishes. Each augmentation is enclosed by a dotted line in Fig.2 (e). Then the back-propagating stage starts. Each terminus makes the final decision from the accumulated gain, and propagates the decision backward through the augmentation, which results in new schedule S_t as shown in Fig.2 (f).

The maximum intended size k of augmentation determines the performance bound. If we can set the weight in (6) to $w_i = q_i(t) \cdot \mu_i$, then the augmentation algorithm achieves $\frac{k-1}{k+1}$ fraction of the capacity region Λ [5]. This approach, however, requires a priori knowledge of μ_i for each link i, which is not available in our scenarios and has to be learned from X_i(t) that is drawn from *unknown* distribution.

On the other hand, the augmentation algorithm can serve as an (α, β)-approximation oracle that takes weight $\boldsymbol{w}$ as input, and outputs matching S such that $\Pr\{S \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}\} \geq \beta$ with $\alpha = \frac{k-1}{k+1}$. By adopting this oracle, one can develop a learning algorithm that achieves the logarithmic growth of $\alpha\beta$-regret with $\alpha = \frac{k-1}{k+1}$ [6]. However, as the network scales, the probability β of the augmentation algorithm can be very small (see Proposition 1), leading to a performance bound that is not much meaningful.

In this work, we develop an analysis framework that characterizes the performance of our scheme based on the augmentation algorithm, and show that it can achieves the rate-optimal logarithmic growth of $\frac{k-1}{k+1}$-regret regardless of the network size, and further it has the stability region that equals $\frac{k-1}{k+1}\Lambda$.

3.2 Learning through augmentation

The problem of link-rate learning in our settings can be considered a CMAB problem with *linear reward*, i.e., the goal is to maximize the total (weighted) reward sum, and of *semi-bandit* type, i.e., individual reward of each played arm is revealed. For this class of learning problems, the well-known UCB index [3] is widely used, and we also adopt it.

We consider a frame structure, where each frame time corresponds to an independent learning period: when a new frame starts, all the learning parameters are reset, and new learning period starts. The frame-based approach allows us to decouple the learning from the scheduling as in [23]. In this section, we present our algorithm during one frame time. For the ease of exposition, we assume that our algorithm runs for time slot $[1, T]$, where T is the frame length.

We start with some definitions. Let $q_i = q_i(1)$ and $\boldsymbol{q}$ denote the initial queue length of link (arm) i and its vector, respectively, and $q^* = q^*(1)$ denote their maximum value in the network. 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 [3] is defined as

$$\bar{w}_{i,t} = \widehat{w}_{i,t-1} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_{i,t-1}}}\ , \tag{7}$$

where $\hat{w}_{i,t} = \frac{q_i}{q^*} \cdot \frac{1}{\hat{\tau}_{i,t}} \sum_{\tau=1}^{t} X_i(\tau) \cdot \mathbb{I}\{i \in S_\tau\}$ denote average reward of arm i at time slot t. Let $\overline{\boldsymbol{w}}_t = (\overline{w}_{1,t}, \overline{w}_{2,t}, \cdots, \overline{w}_{|\mathcal{L}|,t})$ denote the index vector. Then we denote $r_{\overline{\boldsymbol{w}}_t}(S)$ and $r^*_{\overline{\boldsymbol{w}}_t}$ as the index sum over links in matching S and its maximum value over all matchings, respectively. Also, we denote $S^*_{\overline{\boldsymbol{w}}_t}$ and $S^\alpha_{\overline{\boldsymbol{w}}_t}$ as the set of matchings that achieve $r^*_{\overline{\boldsymbol{w}}_t}$ and those that achieve at least $\alpha \cdot r^*_{\overline{\boldsymbol{w}}_t}$, respectively.

Algorithm 2 Learning-based scheduling for a frame

```
1: Obtain queue constant $\boldsymbol{q}$ and $q^*$
2: Initialize $\hat{w}_i$ and $\hat{\tau}_i$ for all $i \in \mathcal{L}$
3: **for** $t = 1$ to $|\mathcal{L}|$ **do**
4:     Schedule arbitrary matching $S_t$ that has link $t$
5:     Update $\hat{w}_i$ and $\hat{\tau}_i$ for each link $i \in S_t$
6: **for** $t = |\mathcal{L}| + 1$ to $T$ **do**
7:     Compute UCB index $\overline{w}_i \leftarrow \hat{w}_i + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{\hat{\tau}_i}}$
8:     < Select matching $S_t$ with gain $G_t^{\overline{\boldsymbol{w}}}(\cdot)$ >
```

Our learning-based scheduling algorithm is shown in Algorithm 2, 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 $\boldsymbol{q}$ 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 S_t with gain using the indices instead of $\boldsymbol{w}$ (line 8), and schedules it (lines 9-10).

We develop a joint scheme of learning and scheduling, by plugging the augmentation algorithm into line 8 of Algorithm 2, and setting the index $\overline{\boldsymbol{w}}_t$ as its input weight. We denote the scheme by A^k-UCB, where parameter k denotes the maximum augmentation size. Note that it greatly reduces the algorithm complexity down to $O(1)$. On the other hand, it no longer provides every-time-slot guarantee on the index sum as the greedy algorithm does in [23]. In the following section, we show that, with appropriate settings, A^k-UCB achieves $\frac{k-1}{k+1}$ fraction of the optimal expected reward, regardless of the network size, and thus achieves $\frac{k-1}{k+1}$ fraction of the capacity region, which can be arbitrarily close to 1 by increasing k.

IV. PERFORMANCE EVALUATION

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

4.1 Regret performance in a single frame

We aim to 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 one lemma and one proposition, which state that the augmentation algorithm is an (α, β)-approximation oracle with $\alpha = \frac{k-1}{k+1}$ and $\beta = \delta$ for some $\delta > 0$. Their proofs follow the same line of analysis of [5], and can be found in Appendices A and B.

LEMMA 1. *Given matching S_{t-1}, $weight$ $\bar{\boldsymbol{w}}_t$, and a fixed $k > 0$, the augmentation algorithm can generate a set $\mathcal{A}^*$ of disjoint augmentation of S_{t-1} such that*

$$r_{\bar{\boldsymbol{w}}_t}(S_{t-1} \oplus \mathcal{A}^*) \geq \frac{k-1}{k+1} \cdot r^*_{\bar{\boldsymbol{w}}_t},$$

and every augmentation A in $\mathcal{A}^*$ has a size no greater than k.

Lemma 1 also holds when using $\boldsymbol{w}$ instead to $\bar{\boldsymbol{w}}_t$.

PROPOSITION 1. *Given any 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 $(S_{t-1} \oplus \mathcal{A}^*) \in \mathcal{S}^{\alpha}_{\bar{\boldsymbol{w}}_t}$, i.e.,* $\Pr\{(S_{t-1} \oplus \mathcal{A}^*) \in \mathcal{S}^{\alpha}_{\bar{\boldsymbol{w}}_t}\} \geq \delta$, *or equivalently,*

$$\Pr\left\{r_{\bar{\boldsymbol{w}}_t}(S_{t-1} \oplus \mathcal{A}^*) \geq \alpha \cdot r^*_{\bar{\boldsymbol{w}}_t}\right\} \geq \delta \tag{8}$$

where $\alpha = \frac{k-1}{k+1}$.

It can be shown that $\delta \geq \min\left\{1, \left(\frac{p}{1-p}\right)^{|\mathcal{V}|}\right\} \cdot \left(\frac{1-p}{k\sum}\right)^{|\mathcal{V}|}$, where p is the seed probability, $|\mathcal{V}|$ is the number of nodes, and $\sum$ is the maximum node degree.

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

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

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

The following lemma, inspired by [17], ensures that if a non-near-optimal matching in $\mathcal{S}_{\boldsymbol{w}}^{\alpha}$ is played many times, then its index sum is smaller than that of any near-optimal matching in $\mathcal{S}_{\boldsymbol{w}}^{\alpha}$. Let $\Delta_{\min}^{\alpha} = \alpha r_{\boldsymbol{w}}^{*} - \max_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} r_{\boldsymbol{w}}(S)$ denotes the minimum near-optimal gap.

Lemma 2. *Given time slot $t > 0$, if a non-near-optimal matching $S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}$ is played more than $l_t = \left\lceil \frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln t}{\Delta_{\min}^{\alpha}} \right\rceil$ 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 $S' \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}$ is bounded by*

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

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

We highlight that $\mathcal{S}_{\boldsymbol{w}}^{\alpha}$ and $\bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}$ 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 by the UCB index. The proof of the lemma is analogous to Lemma A.1 of [17], and included in Appendix C. Using Lemma 2, we can obtain the following regret bound of A^k-UCB, which is one of our main results.

Proposition 2. *For a network graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, A^k-UCB achieves the regret performance bound of*

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

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

Proposition 2 shows that A^k-UCB achieves the logarithmic growth $O(\log T)$ of α-regret. Note that although the result is somewhat similar to the previous works, the proof technique is quite different. At time slot t, A^k-UCB randomly generates a set $\mathcal{A}_t$ of augmentations based on the previous schedule S_{t-1} and $S_{t-1} \oplus \mathcal{A}_t$ are non-near-optimal. This implies that we cannot ensure that the index sum of the chosen schedule (i.e., either S_{t-1} or $S_{t-1} \oplus \mathcal{A}_t$) is greater than $\alpha \cdot r_{\bar{\boldsymbol{w}}_t}^{*}$, because the comparison is done only between the two non-near-optimal matchings. Thus, the traditional technique for the regret analysis cannot be directly applied. We successfully address the technical difficulties by considering the plays of non-near-optimal matchings as a group. The detailed proof can be found in Appendix D.

4.2 Scheduling efficiency

We now consider the throughput performance of A^K-UCB across multiple frames.

PROPOSITION 3. *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. We use the standard Lyapunov technique with time unit of frame length. Given any $\boldsymbol{\lambda}$ strictly inside $\alpha\Lambda$ with $\alpha = \frac{k-1}{k+1}$, we consider the Lyapunov function $L(t_n) = \frac{1}{2}\sum_{i\in\mathcal{L}}\big(q_i(t_n)\big)^2$ 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

$$q_i(t_{n+1}) \le \left(q_i(t_n) - \sum_{t=t_n}^{t_n+T-1} X_i(t)\cdot \mathbb{I}\{i\in S_t\}\right)^+ + \sum_{t=t_n}^{t_n+T-1} a_i(t),$$

where $\{S_t\}$ 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

$$\begin{aligned}\mathbb{E}[D(t_n)\mid \boldsymbol{q}(t_n)] \le 1/2 &\sum_{i\in\mathcal{L}} \mathbb{E}\left[\left(\sum_{t=t_n}^{t_n+T-1} a_i(t)\right)^2 \mid \boldsymbol{q}(t_n)\right] \\ &+ 1/2\sum_{i\in\mathcal{L}} \mathbb{E}\left[\left(\sum_{t=t_n}^{t_n+T-1} X_i(t)\cdot\mathbb{I}\{i\in S_t\}\right)^2 \mid \boldsymbol{q}(t_n)\right] \\ &+ \sum_{t=t_n}^{t_n+T-1}\mathbb{E}\left[\sum_{i\in\mathcal{L}} q_i(t_n)a_i(t) - \sum_{i\in S_t} q_i(t_n)X_i(t)\mid \boldsymbol{q}(t_n)\right]\end{aligned}$$

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 matching during the corresponding frame time, i.e., $S^* \in \mathcal{S}^*_{\boldsymbol{w}} = \arg\max_{S\in\mathcal{S}} \sum_{i\in S} w_i$, 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

$$\begin{aligned}\mathbb{E}[D(t_n)\mid \boldsymbol{q}(t_n)] &\le CT + \sum_{t=t_n}^{t_n+T-1}\mathbb{E}\left[\sum_{i\in\mathcal{L}} q_i(t_n)a_i(t)\mid \boldsymbol{q}(t_n)\right] \\ &\quad - \sum_{t=t_n}^{t_n+T-1}\mathbb{E}\left[\sum_{i\in S_t} q_i(t_n)X_i(t)\mid \boldsymbol{q}(t_n)\right] \\ &= CT + q^*(t_n)\sum_{t=t_n}^{t_n+T-1}\left(\sum_{i\in\mathcal{L}}\frac{q_i(t_n)}{q^*(t_n)}\lambda_i - \alpha r^*_{\boldsymbol{w}}\right) \\ &\quad + q^*(t_n)\sum_{t=t_n}^{t_n+T-1}\left(\alpha r^*_{\boldsymbol{w}} - \mathbb{E}\left[\sum_{i\in S_t} r_{\boldsymbol{w}}(S_t)\mid \boldsymbol{q}(t_n)\right]\right)\end{aligned}$$

$$\leq CT - \epsilon T \sum\nolimits_{i \in S_t} q_i(t_n) + q^*(t_n) \cdot Reg^{\alpha}(T)$$

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

$$\frac{1}{T}\mathbb{E}[D(t_n) | \, \boldsymbol{q}(t_n)] \ \leq C - \epsilon \sum\nolimits_{i \in S_t} q_i(t_n) + q^*(t_n) \cdot \frac{Reg^{\alpha}(T)}{T}.$$

Since Proposition 2 implies that $\frac{Reg^{\alpha}(T)}{T} < \epsilon$ for sufficiently large T, we have a negative drift for sufficiently large queue lengths. □

V. DISTRIBUTED ALGORITHM

We note that the augmentation algorithm is amenable to implement in a distributed fashion with $O(1)$ complexity [5]. However, in A^k-UCB, we need to compute the index $\overline{\boldsymbol{w}}_t$, which requires global information $q^*(t_n)$ – the largest queue length in the network at the start of frame n. The same problem can be observed in the previous greedy algorithm [23].

We pay attention to the fact that A^k-UCB indeed learns the expected value of the queue weighted link rate, i.e., $q_i(t_n) \cdot \mathbb{E}[X_i(t)]$, within a frame time. The global information $q^*(t_n)$ takes the role of normalizing the weight in the range of [0,1].

This implies that we may separate the normalizing parameter from the learning and develop a practically distributed version of A^k-UCB, denoted by dA^k-UCB. In the following, we describe the differences of A^k-UCB and dA^k-UCB. 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 time. At each time slot t of frame n, node v in an augmentation updates its local normalizer twice as follows. 1) In the initialization stage and the path augmenting stage, each 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 repeat while building the augmentation. When the stage of checking a cycle finishes, the terminus w has $\tilde{q}_w = \tilde{q}^*$ that is the largest local normalizer in the augmentation resets $\tilde{q}_v \leftarrow \tilde{q}^*$. Hence, at the end of the time slot, all the nodes in the augmentation have the same local normalizer value $\tilde{q}^*$.

(2) **Gain separation:** In the meantime, we change the way to compute the gain in order to use the local normalizer $\tilde{q}^*$ in the place of the global normalizer $\tilde{q}^*(t_n)$. To elaborate, let G'_u denote the new gain normalized by $\tilde{q}_u$. At each mini-slot in the path augmenting stage,

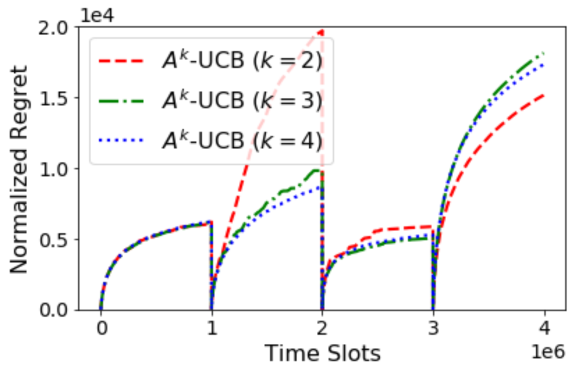
whenever node u transmits an REQ message to node v, we compute the gain $G'_{u,1}$ for average reward (normalized by factor $\tilde{q}_u$) and the gain $G'_{u,2}$ for confidence interval separately (i.e., $G'_u = G'_{u,1} + G'_{u,2}$), and include both in the REQ message. Then, the receiving node v, after updates its local normalizer $\tilde{q}_v$, re-normalizes 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., $\widehat{w}'_i(t) = \frac{q_i(t_n)}{\tilde{q}_v} \cdot \frac{1}{\hat{\tau}_{i,t}} \sum_{j=t_n+1}^{t} X_i(j) \cdot \mathbb{I}\{i \in S_j\} \cdot G'_{u,2}$ can be obtained simply by adding the confidence interval. As this repeat during the augmenting stage, at the terminus, we can obtain the gain from the indices normalized by $\tilde{q}^*$.

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 dA^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 dA^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, and we verify our claim through numerical results.

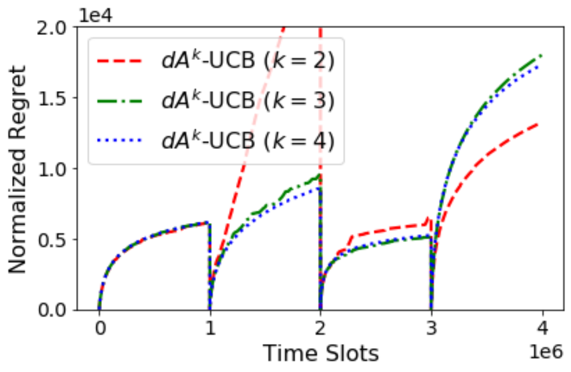
VI. NUMERICAL RESULTS

We evaluate the performance of our proposed schemes through simulations. We consider a 4×4 grid network topology with the primary interference model. Time is slotted, and at each time slot, a packet arrives at link i with probability $\lambda_i = 0.08$ for all i. The instance link rate $X_i(t)$ is drawn from an i.i.d. Bernoulli distribution with mean μ_i, where μ_i is set uniformly at random in range $[0.25, 0.75]$.

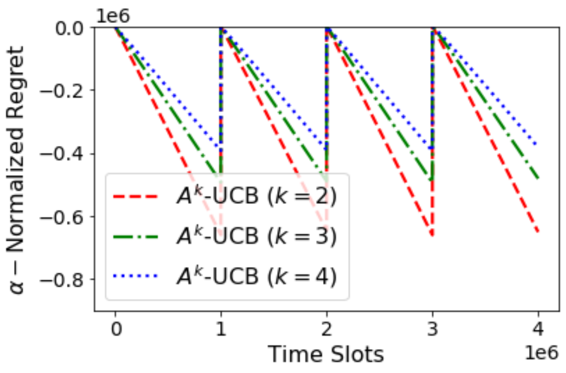
We first investigate the regret performance of A^k-UCB and dA^k-UCB. We set the seed probability $p = 0.2$, and set a large frame length of $T = 10^6$ time slots to observe the regret growth. Fig. 3 show the regret performance of the two schemes with different k. Each value is an average of 10 simulation runs. For 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_w^* within the frame.



(a) Regret of A^k-UCB



(b) Regret of dA^k-UCB



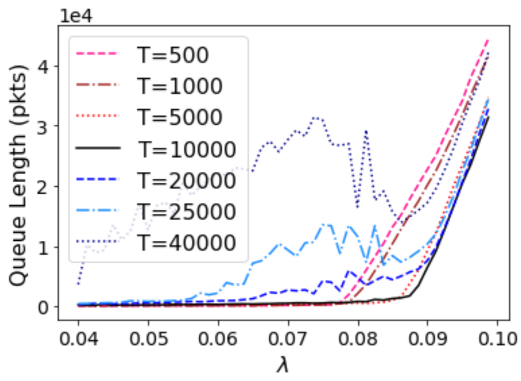
$(\boldsymbol{c})$ $\boldsymbol{\alpha}$-regret of $\boldsymbol{A^k}$-UCB

Figure 3: Regret performance.

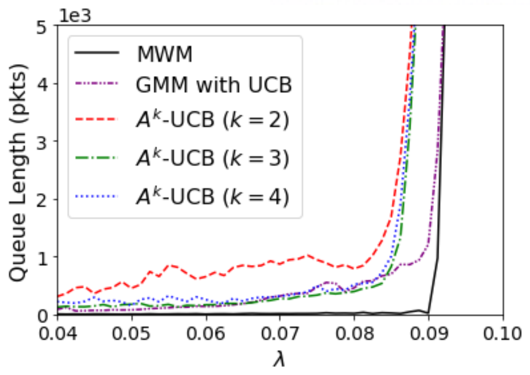
For the performance comparison across frames, the regret is reset to $\mathbf{0}$ *at each frame boundary (*$\boldsymbol{T} = \mathbf{10^6}$ *time slots). We also normalize the regret by the maximum expected reward* $\mathbf{r}_w^*$*.*

Fig.3 (a) and Fig.3 (b) illustrate the logarithmic regret growth for both A^k-UCB and dA^k-UCB, respectively, during a frame time, respectively. Note that the difference between the two schemes is not significant, which is consistently observed throughout our numerical results. Henceforward, we omit some results of dA^k-UCB due to the limited space. We note that the regret performance of Fig.3(a) is much better than our analysis, which provides the logarithmic growth of only α-regret, with $\alpha = \frac{k-1}{k+1}$. The performance in terms of α-regret is shown in Fig. 3(c) for A^k-UCB. The gap from 0 can be interpreted as the level of practical difficulty in achieving analytic performance bound: it gets relatively harder to achieve $\frac{k-1}{k+1}$ -regret as k increases.

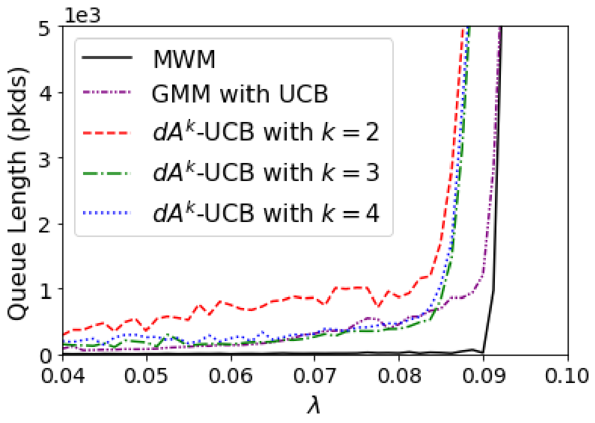
Next, we evaluate throughput performance of our schemes. We set the arrival rate $\lambda_i = \lambda$ for all i and increase λ. We consider the same simulation settings with different T's and λ's. Each simulation continues for 10^6 time slots. We run 10 simulations for each λ, and measure average queue length when the simulations end. Note that, under any scheduling scheme, when the arrival



(a) A^k-UCB with $k = 3$ and different frame length T



(b) Scheduling performance of A^k-UCB



(c) Scheduling performance of dA^k-UCB

Figure 4: Queue lengths

As the arrival rate gets closer to the stability region of a scheme, queue lengths soar quickly.

rate gets closer to the boundary of its stability region, the queue length soars quickly. Thus, we can estimate the performance bound of the scheme from the queue lengths. 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, initially increases for $T \leq 10^6$ and then decreases for $T \geq 2 \cdot 10^6$. It can be expected since too small frame length leads to incomplete learning while too large frame length leads to slow response to the queue dynamics.

We now compare the performance of A^k-UCB with two comparable schemes: One is the optimal MWM that operates with the knowledge of $\boldsymbol{\mu}$. It finds the optimal matching with weight of $q_i(t)\mu_i$ at each time slot t as described in [24] and schedules the optimal matching. The other is the index-based GMM (or online GMM [23]) that uses the UCB index and finds a feasible schedule in the largest-index-first manner. Note that both are a centralized algorithm with polynomial time complexity. For A^k-UCB, we set $T = 5000$ and $p = 0.02$.

Fig. $4(b)$ and Fig. $4(c)$ demonstrate the queue lengths of A^k-UCB and dA^k-UCB, respectively, after 10^6 time slots that corresponds to 200 frames. The queue length of MWM soars at round $\lambda =$

0.09, which can be considered as the boundary of the capacity region Λ since MWM achieves the optimal performance. Under of A^k-UCB and dA^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. Note that a value $k > 4$ is not helpful due to the small network size. Interestingly, the impact of k on empirical throughput seems to be not significant (which may not be true when the network size is larger), but larger k leads to lower queue lengths in all arrival rates, and thus, better delay performance. The index-based GMM achieves better throughput performance close to MWM, which is also far beyond its theoretic bound $\frac{1}{2}$. Similar results have also been reported [15].

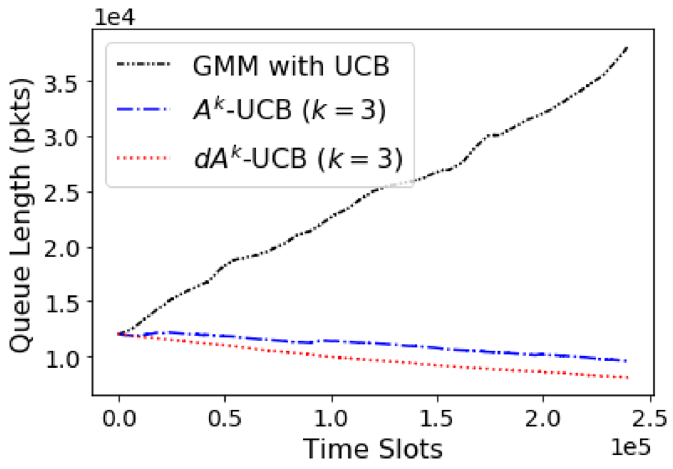
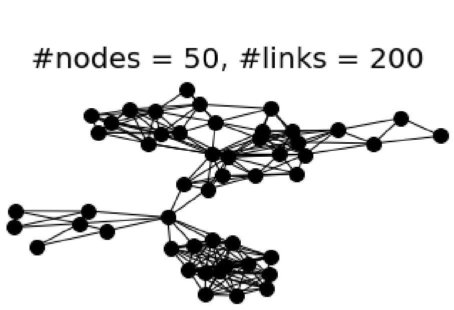
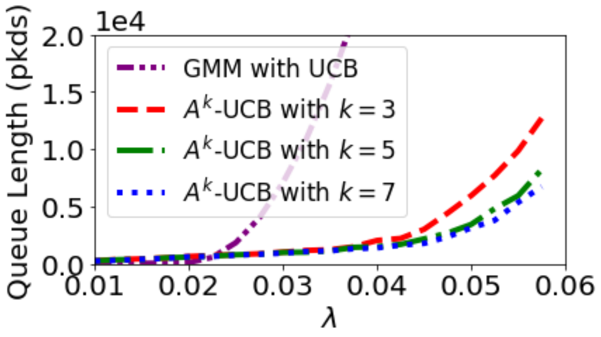


Figure 5: Stability in a 6-link ring topology

Fig. 5 shows that in a certain scenario, the index-based GMM has poor performance. Motivated by [13], 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 $\frac{1}{6} + \epsilon$ where $\epsilon = 0.08$. We set the frame length $T = 6000$. Other environment settings are the same as before, except that the initial queue length is $\left\{\frac{3T}{6}, \frac{2T}{6}, \frac{T}{6}, \frac{3T}{6}, \frac{2T}{6}, \frac{T}{6}\right\}$. The results show that while the queue lengths of A^k-UCB and dA^k-UCB are stabilized, those of the index-based GMM keep increasing. This is because, the greedy algorithm tends select a matching while 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.



(a) Randomly generated network topology



(b) Scheduling performance of A^k-UCB

Figure 6 : Stability in a large-scale network topology

We also experiment in a larger, randomly generated network (of 50 nodes and 200 links). Due to the high order computational complexity, it is hard to simulate MWM in larger network graphs, which takes very long time. So, we compared our algorithm with GMM instead MWM and Fig. 6 shows the index-based GMM may lead to instability under moderate traffic load. In contrast, our algorithm A^k-UCB is *analytically proven* to not lead to instability for any traffic $\boldsymbol{\lambda} \in \frac{k-1}{k+1}\Lambda$ and has better performance than GMM's, indeed.

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. By incorporating the augmentation algorithm into a learning procedure, we develop provably efficient low-complexity schemes that (i) achieve logarithmic regret growth in learning, (ii) have the throughput performance that can be arbitrarily close to the optimal. (iii) 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] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami. 2011. Distributed Algorithms for Learning and Cognitive Medium Access with Logarithmic Regret. IEEE J. Sel. Areas Commun. 29, 4 (April 2011), 731–745.

[2] V. Anantharam, P. Varaiya, and J. Walrand. 1987. Asymptotically Efficient Allocation Rules for the Multiarmed Bandit Problem with Multiple Plays-Part I: I.I.D. Rewards. IEEE Trans. Autom. Control 32, 11 (November 1987), 968–976.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning 47, 2 (May 2002), 235–256.

[4] S. A. Borbash and A. Ephremides. 2006. Wireless Link Scheduling With Power Control and SINR Constraints. IEEE Trans. Inf. Theory 52, 11 (November 2006), 5106–5111. https://doi.org/10.1109/TIT.2006.883617

[5] L. Bui, S. Sanghavi, and R. Srikant. 2009. Distributed Link Scheduling with Constant Overhead. IEEE/ACM Trans. Netw. 17, 5 (October 2009), 1467–1480.

[6] Wei Chen, YajunWang, and Yang Yuan. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In International Conference on Machine Learning.

[7] Jin-Ghoo Choi, Changhee Joo, Junshan Zhang, and Ness B. Shroff. 2014. Distributed Link Scheduling Under SINR Model in Multihop Wireless Networks. IEEE/ACM Trans. Netw. 22, 4 (Aug 2014), 1204–1217.

[8] Y. Gai and B. Krishnamachari. 2011. Decentralized Online Learning Algorithms for Opportunistic Spectrum Access. In IEEE GLOBECOM.

[9] Y. Gai, B. Krishnamachari, and R. Jain. 2012. Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations. IEEE/ACM Trans. Netw. 20, 5 (Oct 2012), 1466–1478.

[10] B. Hajek and G. Sasaki. 1988. Link Scheduling in Polynominal Time. IEEE Trans. Inf. Theory 34, 5 (September 1988).

[11] Linbin Jiang and JeanWalrand. 2010. ADistributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks. IEEE/ACM Trans. Netw. 18, 13 (June 2010), 960–972.

[12] Changhee Joo. 2013. On Random Access Scheduling for Multimedia Traffic in Multi-hop Wireless Networks. IEEE Trans. Mobile Comput. 12, 4 (April 2013), 647–656.

[13] Changhee Joo, Xiaojun Lin, and Ness B. Shroff. 2009. Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-Exclusive Interference Model. IEEE Trans. Autom. Control 54, 12 (Dec 2009), 2734–2744.

[14] Changhee Joo, Gaurav Sharma, Ness B. Shroff, and Ravi R. Mazumdar. 2010. On the Complexity of Scheduling in Wireless Networks. EURASIP Journal of Wireless Communications and Networking (October 2010).

[15] Changhee Joo and Ness B. Shroff. 2009. Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks. IEEE/ACM Trans. Netw. 17, 5 (October 2009).

[16] Changhee Joo and Ness B. Shroff. 2012. Local Greedy Approximation for Scheduling in Multi-hop Wireless Networks. IEEE Trans. Mobile Comput. 11, 3 (March 2012), 414–426.

[17] Sunjung Kang and Changhee Joo. 2018. Low-Complexity Learning for Dynamic Spectrum Access in Multi-User Multi-Channel Networks. In IEEE INFOCOM.

[18] T.L Lai and Herbert Robbins. 1985. Asymptotically Efficient Adaptive Allocation Rules. Adv. Appl. Math. 6, 1 (March 1985), 4–22.

[19] Xiaojun Lin and Shahzada B. Rasool. 2009. Distributed and Provably Efficient Algorithms for Joint Channel-assignment, Scheduling, and Routing in Multichannel Ad Hoc Wireless Networks. IEEE/ACM Trans. Netw. 17, 6 (Dec. 2009), 1874–1887.

[20] Xiaojun Lin and Ness B. Shroff. 2006. The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks. IEEE/ACM Trans. Netw. 14, 2 (April 2006), 302–315.

[21] K. Liu and Q. Zhao. 2010. Distributed Learning in Multi-Armed Bandit With Multiple Players. IEEE Trans. Signal Processing 58, 11 (Nov 2010), 5667–5681.

[22] Jian Ni, Bo Tan, and R. Srikant. 2012. Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks. IEEE/ACM Trans. Netw. 20, 3 (June 2012).

[23] Thomas Stahlbuhk, Brooke Shrader, and Eytan Modiano. 2018. Learning Algorithms for Scheduling in Wireless Networks with Unknown Channel Statistics. In ACM MobiHoc.

[24] L. Tassiulas and A. Ephremides. 1992. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximal Throughput in Multihop Radio Networks. IEEE Trans. Autom. Control 37, 12 (December 1992), 1936–1948.

[25] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang. 2019. Resource Allocation for Ultra-Dense Networks: A Survey, Some Research Issues and Challenges. IEEE Communications Surveys Tutorials 21, 3 (Aug 2019), 2134–2168.

[26] Xinzhou Wu, R. Srikant, and James R. Perkins. 2007. Scheduling Efficiency of Distributed Greedy Scheduling Algorithms in Wireless Networks. IEEE Trans. Mobile Comput. 6, 6 (2007), 595–605.

[27] Q. Zhao, L. Tong, A. Swami, and Y. Chen. 2007. Decentralized Cognitive MAC for Opportunistic Spectrum Access in Ad Hoc Networks: A POMDP Framework. IEEE J. Sel. Areas Commun. 25, 3 (April 2007), 589–600.

APPENDIX

A. PROOF OF LEMMA 1

The lemma shows the possibility for our algorithm to yield a disjoint augmentation of S_{t-1} satisfying the inequality. To this end, we consider the best augmentation set that the algorithm can generate. It can be obtained by combining the previous schedule S_{t-1} and an optimal matching $S_{\boldsymbol{w}}^*$ with the largest weight sum.

We consider two specific matchings (i.e., super arms) of S_{t-1}, the schedule at the previous time slot, and $S_{\boldsymbol{w}}^* \in \mathcal{S}_{\boldsymbol{w}}^*$, and optimal matching with the largest weight sum under weight $\boldsymbol{w}$. Let us define symmetric difference $S_d = S_{t-1} \Delta S_{\boldsymbol{w}}^* = (S_{t-1} - S_{\boldsymbol{w}}^*) \cup (S_{\boldsymbol{w}}^* - S_{t-1})$, and consider graph $\mathcal{G}' = (\mathcal{V}, S_d)$ that contains only the links in S_d. Note that for any vertex in $\mathcal{G}'$, its degree is at most 2 because S_{t-1} and $S_{\boldsymbol{w}}^*$ are a matching. Further, we can define *component* as a set of connected links in $\mathcal{G}'$, which is either a path or an even-length cycle such that links in S_{t-1} and links in $S_{\boldsymbol{w}}^*$ are alternating, i.e., an augmentation. There can be multiple components in $\mathcal{G}'$. For a component C, let $C_{opt} = C \cap S_{\boldsymbol{w}}^*$ and $C_{t-1} = C \cap S_{t-1}$. We note that $size(C) = |C_{opt}|$ and both C_{opt} and C_{t-1} are a matching in $\mathcal{G}$.

Given weight $\boldsymbol{w}$, let $G_t^{\boldsymbol{w}}(\mathcal{A})$ denote the gain of augmentation defined as in (6). We also extend the definition to a set $\mathcal{A}$ of disjoint augmentation by adding up the gains, i.e., $G_t^{\boldsymbol{w}}(\boldsymbol{\mathcal{A}}) = \sum_{A\in\mathcal{A}} G_t^{\boldsymbol{w}}(A)$**.** The following two lemmas show that each component C can be decomposed into small-size augmentations with a set of large-weight sum links.

LEMMA 3. *For component C that is a path, there exists a set of disjoint augmentations $\mathcal{A}(C) \subset C$ such that*

(1) $size(A) \leq k$ for all augmentations $A \in \mathcal{A}(C)$,

(2) $G_t^{\boldsymbol{w}}(\mathcal{A}(C)) \geq \frac{k}{k+1} \cdot r_{\boldsymbol{w}}(C_{opt}) - r_{\boldsymbol{w}}(C_{t-1})$.

Proof. When $size(C) \leq k$, we set $\mathcal{A}(C) = C$. The gain, by definition, equals $G_t^{\boldsymbol{w}}(\mathcal{A}(C)) = r_{\boldsymbol{w}}(C_{opt}) - r_{\boldsymbol{w}}(C_{t-1})$. Thus, the two conditions are satisfied.

When $size(C) > k$, we construct a family of augmentation sets $\{\mathcal{A}_i\}$ and show that at least one of the sets satisfies the two conditions. For path C, we select an endpoint. From the endpoint, we denote the first $(k+1)$ links in C_{opt} by $e_1, e_2, \ldots, e_{k+1}$. For each e_i, we can construct an augmentation set as follows. From C, we remove e_i and every $2(k+1)$-th link thereafter (until we reach the other endpoint). Since C is an augmentation and $e_i \in C_{opt}$, any subsequently removed links also belong to C_{opt}. After the removals, augmentation C is divided into a set of disjoint augmentations, each of which includes at most k links of C_{opt} . Let $\mathcal{A}_i$ denote this set of disjoint augmentations. By repeating the procedure for each $1 \leq i \leq k+1$, we can construct a family of augmentation sets $\{\mathcal{A}_i\}_{i=1}^{k+1}$ such that any augmentation $A \in \cup_{i=1}^{k+1} \mathcal{A}_i$ satisfies $size(A) \leq k$.

Note that, in the construction procedure of $\{\mathcal{A}_i\}_{i=1}^{k+1}$, each link of C_{opt} is removed exactly once, and no link of $C_{t-1}$is removed. This implies that the gain sum over $\{\mathcal{A}_i\}$ satisfies $\sum_{i=1}^{k+1} G_t^{\boldsymbol{w}}(\mathcal{A}_i) = k \cdot r_{\boldsymbol{w}}(C_{opt}) - (k+1) \cdot r_{\boldsymbol{w}}(C_{t-1})$. Hence, there exists at least one j such that

$$G_t^{\boldsymbol{w}}(\mathcal{A}_j) \geq \frac{k}{k+1} \cdot r_{\boldsymbol{w}}(C_{opt}) - r_{\boldsymbol{w}}(C_{t-1}).$$

$\square$

Lemma 3 shows the existence of a good set of disjoint augmentations in every path component in graph $\mathcal{G}'$. We use Lemma 3 to prove a slightly weaker result for a cycle component in $\mathcal{G}'$.

LEMMA 4. *F*or component C that is a cycle, there exists a set of disjoint augmentations $\mathcal{A}(C)$ *s.t.*

(1) $size(A) \leq k$ for all augmentations $A \in \mathcal{A}(C)$,

(2) $G_t^{\boldsymbol{w}}(\mathcal{A}(C)) \geq \frac{k-1}{k+1} \cdot r_{\boldsymbol{w}}(C_{opt}) - r_{\boldsymbol{w}}(C_{t-1})$.

Proof. As in the proof of Lemma 3, when $size(C) \leq k$, we can set $\mathcal{A}(C) = C$, and the two conditions are satisfied. Thus, we assume that size(C) > k.

Let $e \in C_{\text{opt}}$ denote the link with the smallest weight in C_{opt}. Consider path $\hat{C} = C - e$ and define $\hat{C}_{\text{opt}} = \hat{C} \cap S_w^*$ and $\hat{C}_{t-1} = \hat{C} \cap S_{t-1}$. Since $\hat{C}_{\text{opt}}$ is obtained by removing the link of the smallest weight from C_{opt}, it has a larger average weight than C_{opt}, i.e., $\frac{r_w(\hat{C}_{opt})}{|C_{opt}|-1} \geq \frac{r_w(C_{opt})}{|C_{opt}|}$, which implies that, from $|C_{opt}| = size(C) > k$,

$$r_w(\hat{C}_{\text{opt}}) \geq \frac{k}{k+1} \cdot r_w(C_{opt}).$$

Lemma 3 tell us that there exists a set $\mathcal{A}(\hat{C})$ of disjoint augmentations, where (i) the size of each augmentation is at most k, and (ii) the following inequality holds:

$$G_t^w(\mathcal{A}(C)) \geq \frac{k}{k+1} \cdot r_w(\hat{C}_{\text{opt}}) - r_w(\hat{C}_{t-1}) \geq \frac{k-1}{k+1} \cdot r_w(C_{opt}) - r_w(C_{t-1})$$

□

We now build $\mathcal{A}^*$ for the proof of Lemma 1. For each component C of S_d, the corresponding $\mathcal{A}(C)$ can be specified by Lemma 3 or Lemma 4. We set $\mathcal{A}^* = \cup_C \mathcal{A}(C)$. Then all augmentations $A \in \mathcal{A}^*$have $size(A) \leq k$. Further, we have

$$G_t^w(\mathcal{A}^*) \geq \frac{k}{k+1} \cdot \sum_C r_w(C_{opt}) - \sum_C r_w(C_{t-1}) \geq \frac{k}{k+1} \cdot \sum_{i \in S_w^*} w_i - \sum_{i \in S_{t-1}} w_i.$$

Hence, we have $r_w(S_{t-1} \oplus \mathcal{A}^*) = G_t^w(\mathcal{A}^*) + r_w(S_{t-1}) \geq \frac{k-1}{k+1} r_w^*$.

B. PROOF OF PROPOSITION 1

Suppose that, given S_{t-1}, our algorithm has generated the set $\mathcal{A}$ of disjoint augmentations. Let $\mathcal{A}^*$ denote a near optimal set of augmentations satisfying $r_w(S_{t-1} \oplus \mathcal{A}^*) \geq \frac{k-1}{k+1} r_w^*$, which consists of d disjoint augmentations $\{A_1, A_2, .., A_d\}$. For each A_i, let n_i denote an (arbitrary) endpoint of A_i if A_i is a path, or an arbitrary node in A_i if A_i is a cycle. Consider an event $\boldsymbol{\mathcal{E}}$ under our algorithm such that all the following are true:

- the set of self-selected seeds equals $\{n_1, n_2 \ldots, n_d\}$,
- all seed nodes n_i select its intended size as $size(A_i)$, and
- the generated set of augmentation $\mathcal{A}$ equals $\mathcal{A}^*$.

Since it is clear that $\Pr\{\mathcal{A} = \mathcal{A}^* \mid S_{t-1}\} \geq \Pr\{\boldsymbol{\mathcal{E}} \mid S_{t-1}\}$, we focus on a lower bound of $\Pr\{\boldsymbol{\mathcal{E}} \mid S_{t-1}\}$.

Note that i) the seed selection of $\{n_i\}$ occurs with probability $p^d\ (1-p)^{V-d}$, where $V = |\mathcal{V}|$, and ii) the probability that all seeds n_i independently select its intended size of $size(A_i)$ is k^{-d}. iii) Finally, we compute the probability that the generated set of augmentation $\mathcal{A}$ equals $\mathcal{A}^*$, which occurs when, at each iteration, each active node selects its corresponding neighbor that belongs to $\mathcal{A}^*$. Each selection has probability Σ^{-1} , where Σ is the maximum node degree, and total V selections will be made under the algorithm. Thus, combining the three probabilities, we have that

$$\Pr\{\boldsymbol{\mathcal{E}}\ |S_{t-1}\} \geq p^d\ (1\ -p)^{V-d}\ \cdot k^{-d} \cdot \Sigma^{-\mathrm{V}} \geq \min\left\{1, \left(\frac{p}{1-p}\right)^V\right\}\left(\frac{1-p}{k\Sigma}\right)^V. \tag{11}$$

setting $\delta = \min\left\{1, \left(\frac{p}{1-p}\right)^V\right\}\left(\frac{1-p}{k\Sigma}\right)^V$ completes the proof. □

C. PROOF OF LEMMA 2

At time $t > 0$, define $l_t = \left\lceil\frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln t}{\Delta_{\min}^{\alpha}}\right\rceil$. Suppose that a non-near-optimal matching $S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}$ such that $\hat{\tau}_S(t) \geq l_t$ is scheduled. It also implies $\hat{\tau}_i\ (t) \geq l_t$ for all links $i\ \in\ S$. Also, consider an arbitrary near-optimal matching $S' \in\ \mathcal{S}_{\boldsymbol{w}}^{\alpha}$, and let $S = \{e_1, e_2, \ldots, e_A\}$ and $S' = \{e_1\ , e_2\ , \ldots, e_B\ \}$, where $A = |S|$ and $B = |S'|$, respectively. We consider the event that the total index sum over links in S is greater than that over links in S'.

$$\begin{aligned}\mathbb{I}\{r_{\bar{\boldsymbol{w}}_t}(S) \geq r_{\bar{\boldsymbol{w}}_t}(S')\} &= \mathbb{I}\left\{\sum_{i=1}^{A} \bar{w}_{e_i,t} \geq \sum_{j=1}^{B} \bar{w}_{e'_j,t}\right\} \\ &\leq \mathbb{I}\left\{\max_{l_t \leq c_1,\ldots,c_A < t} \sum_{i=1}^{A}\left(\widehat{w}_{e_i,c_i} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}}\right)\right. \\ &\qquad \left.\geq \max_{0<c'_1,\ldots,c'_B<t} \sum_{j=1}^{B}\left(\widehat{w}_{e'_j,c'_j} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c'_j}}\right)\right\}\end{aligned} \tag{12}$$

which is bounded by $\sum_{c_1=1}^{t} \cdots \sum_{c_A=1}^{t} \sum_{c'_1=1}^{t} \cdots \sum_{c'_B=1}^{t} \mathbb{I}\{\mathbb{Z}\}$, where event $\mathbb{Z}$ is defined as

$$\sum_{i=1}^{A}\left(\widehat{w}_{e_i,c_i} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}}\right) \geq \sum_{j=1}^{B}\left(\widehat{w}_{e'_j,c'_j} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c'_j}}\right).$$

For the ease of explanation, we set $\widehat{w}_{e_i,c_i}$ to denote the sample mean of link e_i scheduled c_i times. We further decompose $\mathbb{Z}$ into $\{\mathbb{A}_i\}$, $\{\mathbb{B}_j\}$ and $\mathbb{C}$ as

$$\mathbb{A}_i\colon \widehat{w}_{e_i,c_i} - \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}} \geq w_{e_i} \tag{13}$$

$$\mathbb{B}_j: \widehat{w}_{e'_j c'_j} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c'_j}} \le w_{e'_j}$$

$$\mathbb{C}: \sum_{j=1}^{B} w_{e'_j} - \sum_{i=1}^{A} w_{e_i} < 2\sum_{i=1}^{A}\left(\sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}}\right)$$

Note that if $\mathbb{I}\{\mathbb{Z}\} = 1$, we should have $\sum_{i=1}^{A}\mathbb{I}\{\mathbb{A}_i\} + \sum_{j=1}^{B}\mathbb{I}\{\mathbb{B}_j\} + \mathbb{I}\{\mathbb{C}\} \ge 1$, which can be easily shown by contradiction. The probability of each event of $\mathbb{A}_i$ and $\mathbb{B}_j$ can be bounded by the Chernoff-Hoeffding bound as

$$\Pr\left\{\widehat{w}_{e_i,c_i} - \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}} \ge w_{e_i}\right\} \le t^{-2(|\mathcal{L}|+1)},$$
$$\Pr\left\{\widehat{w}_{e'_j c'_j} + \sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c'_j}} \le w_{e'_j}\right\} \le t^{-2(|\mathcal{L}|+1)}, \qquad (14)$$

Further, we can show $\Pr\{\mathbb{C}\} = 0$ if $c_i \ge \left\lceil \frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln t}{\Delta_{\min}^{\alpha}} \right\rceil$ for all links $e_i \in S$ as follows. Suppose that $\mathbb{C}$ occurs. Then we should $0 > \sum_{j=1}^{B} w_{e'_j} - \sum_{i=1}^{A} w_{e_i} - 2\sum_{i=1}^{A}\left(\sqrt{\frac{(|\mathcal{L}|+1)\ln t}{c_i}}\right)$. The right term is no smaller than $\alpha \cdot r_{\boldsymbol{w}}^{*} - \max_{S\in\bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} r_{\boldsymbol{w}}(S) - \Delta_{\min}^{\alpha}$, which equals 0 by definition of $\Delta_{\min}^{\alpha}$, resulting in a contradiction.

Using these and taking expectation on (12), we have

$$\Pr\{r_{\overline{\boldsymbol{w}}_t}(S) \ge r_{\overline{\boldsymbol{w}}_t}(S')\} \le \sum_{c_1=1}^{t}\cdots\sum_{c_A=1}^{t}\sum_{c'_1=1}^{t}\cdots\sum_{c'_B=1}^{t}\Pr\{\mathbb{Z}\}$$
$$\le \sum_{c_1=1}^{t}\cdots\sum_{c_A=1}^{t}\sum_{c'_1=1}^{t}\cdots\sum_{c'_B=1}^{t}\left(\sum_{i=1}^{A}\Pr\{\mathbb{A}_i\} + \sum_{j=1}^{B}\Pr\{\mathbb{B}_j\}\right)$$
$$\le t^{A+B}\cdot 2|\mathcal{L}|t^{-2(|\mathcal{L}|+1)} \le 2|\mathcal{L}|t^{-2}.$$

□

D. PROOF OF PROPOSITION 2

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 the horizon time T > 0, let $l_T = \left\lceil \frac{4|\mathcal{L}|^2(|\mathcal{L}|+1)\ln t}{\Delta_{\min}^{\alpha}} \right\rceil$, and let T' denote the first time when all non-near-optimal matchings are sufficiently (i.e., more than l_T times) explored, i.e.,

$$T' = \min\{t \mid \hat{\tau}_{S,t} \geq l_T \text{ for all } S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}\}.$$

(1) When $T' \leq T$: Let $\bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha} = \{S^1, S^2, \dots, S^M\}$ with $M = |\bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}|$. Further we define $\bar{\mathcal{S}}(t) = \{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha} \mid \hat{\tau}_{S,t} \geq l_T\}$, which is the set of non-near-optimal matchings that are scheduled sufficiently many times by time t , and $\underline{\mathcal{S}}(t) = \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha} - \bar{\mathcal{S}}(t)$ 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., $\hat{\tau}_{S^n}(T^n) = l_T$. Without loss of generality, we assume $T^1 < T^2 < \cdots < T^M = T'$.

To apply the decomposition inequality (9), we need to estimate the expected value of $\sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \hat{\tau}_{S,T'}$, which can be written as

$$\begin{aligned} \sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \hat{\tau}_{S,T'} &= \sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \sum_{t=1}^{T'} \mathbb{I}\{S_t = S\} \\ &= l_T M + \sum_{n=1}^{M-1} \sum_{t=T^n}^{T^{n+1}} \sum_{S \in \bar{\mathcal{S}}(T^n)} \mathbb{I}\{S_t = S\} \end{aligned} \tag{15}$$

Hence, we need to estimate $\sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \Pr\{S_t = S\}$ for $t \in (T^n, T^{n+1}]$, which can be obtained as in the following lemma.

LEMMA 5. *For each* $t \in (T^n, T^{n+1}]$, *we have*

$$\begin{aligned} \sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \Pr\{S_t = S\} &\leq (1-\delta) \cdot \sum_{S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}} \Pr\{S_{t-1} = S\} \\ &\quad + \Pr\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\} + (|\bar{\mathcal{S}}(T^n)| + \delta) \cdot 2|\mathcal{L}| t^{-2} \end{aligned} \tag{16}$$

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

$$\sum_{S \in \bar{\mathcal{S}}(T^n)} \Pr\{S_t = S\}$$

$$= \sum_{S \in \bar{\mathcal{S}}(T^n)} \Pr\{S_t = S \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\} \tag{17}$$

$$+ \sum_{S \in \bar{\mathcal{S}}(T^n)} \Pr\{S_t = S \mid \mathbb{B}\} \cdot \Pr\{\mathbb{B}\} \tag{18}$$

$$+ \sum_{S \in \bar{\mathcal{S}}(T^n)} \Pr\{S_t = S \mid \mathbb{C}\} \cdot \Pr\{\mathbb{C}\} \tag{19}$$

Let $\mathcal{A}_t$ denote the set of augmentations chosen under our algorithm at time t. We can obtain a bound on (17) as

$$\sum_{S \in \bar{\mathcal{S}}(T^n)} \Pr\{S_t = S \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\}.$$

$$\le \sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{r_{\overline{\boldsymbol{w}}_t}(S) \ge r_{\overline{\boldsymbol{w}}_t}(S_{t-1}) \mid \mathbb{A}\} \cdot \Pr\{\mathbb{A}\}$$

$$\le |\bar{S}(T^n)| \cdot 2|\mathcal{L}|t^{-2} \tag{20}$$

where the last inequality comes from Lemma 2. The result holds for all $t \in (T^n, T^{n+1}]$. For the second term (18), we have

$$\sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{S_t = S \mid \mathbb{B}\} \cdot \Pr\{\mathbb{B}\} \le \Pr\{\mathbb{B}\} \tag{21}$$

Finally, the third term (19) 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

$$\sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{S_t = S \mid \mathbb{C}\} \cdot \Pr\{\mathbb{C}\}$$

$$\le \sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{S_t \in \bar{S}(T^n) \mid S_{t-1} = S\} \cdot \Pr\{S_{t-1} = S\}.$$

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

$\Pr\{S_t \in \bar{S}(T^n) \mid S_{t-1} = S\}$

$\le \Pr\{S_t \in \bar{S}(T^n) \mid S_{t-1} = S, S' \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}\} \cdot \Pr\{S' \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}\} + \Pr\{S' \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}\}$

$\le \Pr\{S_t \in \bar{S}(T^n) \mid S_{t-1} = S, S' \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}\} \cdot \delta + 1 - \delta$

$\le \Pr\{r_{\overline{\boldsymbol{w}}_t}(S) \ge r_{\overline{\boldsymbol{w}}_t}(S_{t-1}) \mid S \in \bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}, S' \in \mathcal{S}_{\boldsymbol{w}}^{\alpha}\} \cdot \delta + 1 - \delta$

$\le 2|\mathcal{L}|t^{-2} \cdot \delta + 1 - \delta.$

where the second inequality comes from Proposition 1, the equality holds since S_t should be S (otherwise, $S_t = (S \oplus \mathcal{A}_t) \notin \bar{S}(T^n)$) 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 (19) can be upper bounded by

$$\sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{S_{t-1} = S\} \cdot (2\delta|\mathcal{L}|t^{-2} + 1 - \delta)$$

$$\le 2\delta|\mathcal{L}|t^{-2} + (1-\delta)\sum\nolimits_{S\in\bar{S}(T^n)} \Pr\{S_{t-1} = S\}, \tag{22}$$

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

The result can be obtained by combining (20), (21), and (22). □

In order to apply Lemma 5 to (15), we rewrite it in a recursive form. Let $\eta = 1 - \delta$, $H_n = (|\bar{S}(T^n)| + \delta) \cdot 2|\mathcal{L}| = (n + \delta) \cdot 2|\mathcal{L}|$, and $\Theta_n(t) = \sum_{S\in\bar{S}(T^n)} \Pr\{S_t = S\}$. We have a recursive form of (16) as

$$\Theta_n(t) \le \eta^{t-T^n}\Theta_n(T^n) \tag{23}$$

$$+H_n \sum\nolimits_{i=T^n+1}^{t} \eta^{t-i} i^{-2} \tag{24}$$

$$+\sum_{i=T^n+1}^{t} \eta^{t-i} \cdot \Pr\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\}. \tag{25}$$

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

LEMMA 6. *The total number of times that sufficiently played non-near-optimal matchings are selected during* $(T^n, T^{n+1}]$ is bounded by

$$\sum_{i=T^n+1}^{T^n+1} \Theta_n(t) \leq \frac{1}{\delta}\left(1 + \frac{\pi^2}{6} H_n + \mathbb{E}\left[\sum_{S \in \underline{\mathcal{S}}(T^n)} \tau_{S,n+1}\right]\right), \tag{26}$$

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 6 follows the same line of analysis as [17] and included for the completion.

Proof. By summing up (25) over $t \in (T^n, T^{n+1}]$, we have

$$\begin{aligned}
&\sum_{t=T^n}^{T^{n+1}} \sum_{i=T^n+1}^{t} \eta^{t-i} \cdot \Pr\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\} \\
&= \sum_{t=T^n}^{T^{n+1}} \Pr\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\} \cdot \left(\sum_{i=0}^{T^{n+1}-t} \eta^i\right) \\
&\leq \frac{1}{1-\eta} \mathbb{E}\left[\sum_{t=T^n}^{T^{n+1}} \mathbb{I}\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\}\right] \\
&= \frac{1}{\delta} \mathbb{E}\left[\sum_{S \in \underline{\mathcal{S}}(T^n)} \tau_{S,n+1}\right],
\end{aligned} \tag{27}$$

where the inequality holds because $\sum_{i=0}^{T^{n+1}-t} \eta^i \leq \frac{1}{1-\eta}$, $\Pr\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\} = \mathbb{E}[\mathbb{I}\{S_{t-1} \in \underline{\mathcal{S}}(T^n)\}]$, and $\mathbb{I}\{S_{T^n} \in \underline{\mathcal{S}}(T^n)\} = 0$ since the matching scheduled at T^n does not belong to $\underline{\mathcal{S}}(T^n)$ by definition.

Also, by summing up (24) over $(T^n, T^{n+1}]$, we have

$$\sum_{t=T^n}^{T^{n+1}} H_n \sum_{i=T^n+1}^{t} \eta^{t-i} \cdot i^{-2} \leq H_n \cdot \frac{1}{\delta} \cdot \frac{\pi^2}{6}. \tag{28}$$

Similarly, we take the sum of (23) as

$$\sum_{t=T^n}^{T^{n+1}} \eta^{t-T^n} \Theta_n(T^n) \leq \sum_{t=T^n}^{T^{n+1}} \eta^{t-T^n} \leq \frac{1}{\delta} \tag{29}$$

Combining (27), (28), and (29), we obtain the result. □

Now, by taking expectation on (15), we can obtain the expected total number of times that non-near optimal matchings are selected up to time $T'(\leq T)$ as

$$\sum_{S \in \bar{\mathcal{S}}_w^\alpha} \mathbb{E}[\hat{\tau}_S(T')] = l_T M + \sum_{n=1}^{M-1} \sum_{t=T^n}^{T^{n+1}} \Theta_n(t)$$

$$\leq l_T M + \frac{1}{\delta}\sum_{n=1}^{M-1}\left(1 + \frac{H_n\pi^2}{6} + \mathbb{E}\left[\sum_{S\in\underline{\mathcal{S}}(T^n)} \tau_{S,n+1}\right]\right)$$
$$\leq l_T M + \frac{M}{\delta}\left(1 + \frac{|\mathcal{L}|\delta\pi^2}{6} + \frac{|\mathcal{L}|(M-1)\pi^2}{6} + l_T\right).$$

The last inequality holds since (i) $\sum_{n=1}^{M-1} H_n = \sum_{n=1}^{M-1}(n+\delta)\cdot 2|\mathcal{L}| \leq M\cdot|\mathcal{L}|\cdot\big(2\delta + (M-1)\big)$, 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_T up to T^n are chosen during $(T^n, T^{n+1}]$ and thus results in $\sum_{n=1}^{M-1}\sum_{S\in\underline{\mathcal{S}}(T^n)} \tau_{S,n+1} \leq \sum_{k=2}^{M} l_T \leq l_T M$. From $M \leq |\mathcal{S}| - 1$, we have

$$\sum_{S\in\bar{\mathcal{S}}_w^\alpha} \mathbb{E}[\hat{\tau}_S(T')] \leq D_1 \cdot \frac{\ln T}{(\Delta_{\min}^\alpha)^2} + \frac{|\mathcal{S}|-1}{\delta}\left(1 + \frac{|\mathcal{L}|\delta\pi^2}{6} + \frac{|\mathcal{L}|(|\mathcal{S}|-2)\pi^2}{6}\right) \tag{30}$$

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

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

$$\begin{aligned}\Pr\{S_t \in \bar{\mathcal{S}}_w^\alpha\} = {} & \Pr\{S' \in \bar{\mathcal{S}}_w^\alpha, S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\} \\ & + \Pr\{S' \in \bar{\mathcal{S}}_w^\alpha, S_{t-1} \in \mathcal{S}_w^\alpha, r_w(S') \geq r_w(S_{t-1})\} \\ & + \Pr\{S' \in \mathcal{S}_w^\alpha, S_{t-1} \in \bar{\mathcal{S}}_w^\alpha, r_w(S') \leq r_w(S_{t-1})\}\end{aligned}$$

This leads to

$$\begin{aligned}\Pr\{S_t \in \bar{\mathcal{S}}_w^\alpha\} \leq {} & \Pr\{S' \in \bar{\mathcal{S}}_w^\alpha \mid S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\}\cdot\Pr\{S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\} \\ & + \Pr\{r_w(S') \geq r_w(S_{t-1}) \mid S' \in \bar{\mathcal{S}}_w^\alpha, S_{t-1} \in \mathcal{S}_w^\alpha\} \\ & + \Pr\{r_w(S') \leq r_w(S_{t-1}) \mid S' \in \mathcal{S}_w^\alpha, S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\}.\end{aligned}$$

From Proposition 1, we have $\Pr\{S' \in \bar{\mathcal{S}}_w^\alpha \mid S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\} = 1 - \Pr\{S' \in \mathcal{S}_w^\alpha \mid S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\} \leq 1 - \delta = \eta$. Since $\hat{\tau}_S(t) \geq l_T$ for all S and $t \in (T', T]$, 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

$$\Pr\{S_t \in \bar{\mathcal{S}}_w^\alpha\} \leq \eta\Pr\{S_{t-1} \in \bar{\mathcal{S}}_w^\alpha\} + 4|\mathcal{L}|t^{-2}.$$

By extending the inequality in a recursive manner down toT', we obtain that

$$\Pr\{S_t \in \bar{\mathcal{S}}_w^\alpha\} \leq \eta^{t-T'}\Pr\{S_{T'} \in \bar{\mathcal{S}}_w^\alpha\} + 4|\mathcal{L}|\sum_{t=T'+1}^{t}\eta^{t-i}i^{-2}$$
$$= \eta^{t-T'} + 4|\mathcal{L}|\sum_{t=T'+1}^{t}\eta^{t-i}i^{-2}$$

where the last equality holds since $\Pr\{S_{T'} \in \bar{\mathcal{S}}_w^\alpha\} = 1$ from the definition of T'. Summing over $t \in (T', T]$ on the both sides, and from $\eta = 1 - \delta$, we have

$$\sum_{t=T'+1}^{T}\Pr\{S_t \in \bar{\mathcal{S}}_w^\alpha\} \leq \frac{1-\delta}{\delta} + \frac{2|\mathcal{L}|\pi^2}{3\delta}. \tag{31}$$

Combining (30) and (31), we obtain

$$\sum\nolimits_{S\in\bar{\mathcal{S}}_{w}^{\alpha}}\mathbb{E}[\hat{\tau}_S(T)] = \sum\nolimits_{S\in\bar{\mathcal{S}}_{w}^{\alpha}}\mathbb{E}[\hat{\tau}_S(T')] + \sum\nolimits_{t=T'+1}^{T}\Pr\{S_t\in\bar{\mathcal{S}}_{\boldsymbol{w}}^{\alpha}\}$$

$$\leq D_1\cdot\frac{\ln T}{(\Delta_{\min}^{\alpha})^2}+D_2 \tag{32}$$

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

(2) When $T'>T$ (i.e., $\exists S$ such that $\hat{\tau}_S<l_T$) : With the same definitions of l_T, $\bar{\mathcal{S}}(t)$, and $\underline{\mathcal{S}}(t)$, let $|\bar{\mathcal{S}}| = |\bar{\mathcal{S}}(T)|$ and $|\underline{\mathcal{S}}| = |\underline{\mathcal{S}}(T)|$. At this time, we define $\bar{\mathcal{S}}(T)=\{S^1,S^2,\dots,S^{|\bar{\mathcal{S}}|}\}$ and let T^n denote the time at which matching S^n is sufficiently scheduled, i.e., $\hat{\tau}_{S^n}(T^n)=l_T$. Without loss of generality, we assume $T^1<T^2<\cdots<T^{|\bar{\mathcal{S}}|}$. By time slot T, $\underline{\mathcal{S}}(T)$ is non-empty (since $T'>T$), and it is clear that $\sum_{S\in\underline{\mathcal{S}}(T)}\hat{\tau}_{S,T}\leq l_T|\underline{\mathcal{S}}|$.

Similar to the case when $T'\leq T$, we can obtain

$$\sum\nolimits_{S\in\bar{\mathcal{S}}_{w}^{\alpha}}\mathbb{E}[\hat{\tau}_S(T)] = \sum\nolimits_{S\in\underline{\mathcal{S}}(T)}\mathbb{E}\left[\hat{\tau}_{S,T}\right] + \sum\nolimits_{t=1}^{T}\sum\nolimits_{S\in\bar{\mathcal{S}}(T)}\Pr\{S_t=S\}$$

$$\leq l_T|\underline{\mathcal{S}}|+l_T|\bar{\mathcal{S}}|+\sum\nolimits_{n=1}^{|\bar{\mathcal{S}}|}\sum\nolimits_{t=T^n+1}^{T^{n+1}}\sum\nolimits_{S\in\bar{\mathcal{S}}(T)}\Theta_n(t)$$

$$\leq l_T M+\sum\nolimits_{n=1}^{|\bar{\mathcal{S}}|}\frac{1}{\delta}\left(1+\frac{H_n\pi^2}{6}+\mathbb{E}\left[\sum\nolimits_{S\in\underline{\mathcal{S}}(T^n)}\tau_{S,n+1}\right]\right),$$

where the last inequality comes from Lemma 6. As in (30), we can obtain

$$\leq l_T M+\frac{|\bar{\mathcal{S}}|}{\delta}\left(1+\frac{|\mathcal{L}|\delta\pi^2}{6}+\frac{|\mathcal{L}|(|\bar{\mathcal{S}}|+1)\pi^2}{6}+l_T\right)$$

$$\leq D_1\cdot\frac{\ln T}{(\Delta_{\min}^{\alpha})^2}+D_2 \tag{33}$$

where the last inequality holds due to $|\bar{\mathcal{S}}|\leq M-1$.

Proposition 2 can be obtained by applying (32) and (33) to the decomposition inequality (9).

□