Ad Hoc Networks 85 (2019) 131–144

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Learning algorithms for scheduling in wireless networks with unknown channel statistics☆,☆☆

Thomas Stahlbuhk[a,*], Brooke Shrader[a], Eytan Modiano[b]

[a] MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02421, United States
[b] Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139, United States

ARTICLE INFO

Article history:
Received 29 May 2018
Revised 31 August 2018
Accepted 16 October 2018
Available online 17 October 2018

Keywords:
Wireless networks
Network control
Transmission scheduling

ABSTRACT

We study the problem of learning channel statistics to efficiently schedule transmissions in wireless networks subject to interference constraints. We propose an algorithm that uses greedily-constructed schedules in order to learn the channels' transmission rates, while simultaneously exploiting previous observations to obtain high throughput. Comparison to the offline solution shows our algorithm to have good performance that scales well with the number of links in the network. We then turn our attention to the stochastic setting where packets randomly arrive to the network and await transmission in queues at the nodes. We develop a queue-length-based scheduling policy that uses the channel learning algorithm as a component. We analyze our method in time-varying environments and show that it achieves the same stability region as that of a greedy policy with full channel knowledge.

1. Introduction

A major challenge in the design of wireless networks is the need to schedule transmissions so that the nodes in the network can efficiently share the common spectrum. Nodes that are nearby can interfere with one another, and, therefore, network controllers must choose simultaneous link activations that do not violate interference constraints. Additionally, scheduling decisions must take channel state information into account. In many settings, it is not possible to know the channel in advance. The nodes can only learn the channel statistics by transmitting on the channels and using receiver feedback to observe the results. Due to multi-path fading and environmental interference, the channel state is often time-varying, and multiple observations are needed in order to obtain an accurate estimate of the channel statistics. This introduces a tradeoff between exploration and exploitation. The nodes may have to forgo scheduling those channels that, so far, have given the highest throughput in order to improve their understanding of under-observed channels.

With perfect channel knowledge, it is well known that the max-weight policy, first explored in [2], provides a throughput optimal scheduling scheme that activates edges based on queue length and channel conditions, subject to interference constraints. The max-weight policy weights each edge by the product of its queue backlog and channel rate and chooses a link activation that maximizes the summation of weights in the set [3]. Depending on the interference constraints imposed on the network, this requires that the controller solves a complex combinatorial optimization problem at each time step of the algorithm.

A drawback of the max-weight policy is that it is heavily centralized and assumes that the network controller knows the queue backlog at every edge in the network. For this reason, the use of greedy scheduling methods have been explored in the literature [4–8]. Under these methods, at each time step, the activation set is chosen according to a greedy combinatorial algorithm. In many settings, this construction can be performed in a distributed manner by the nodes of the network. The cost of using greedy algorithms is that, in general, they are not able to guarantee throughput optimality. However, under certain interference constraints, they can obtain network stability over a guaranteed fraction of the network's throughput region.

The aforementioned work assumes perfect knowledge of the channel conditions, which enables the combinatorial scheduling problem to be solved to optimality or approximation at each time step. However, in practice the channel conditions are not known a

☆ Funding: This work was sponsored by NSF Grants AST-1547331, CNS-1701964 and CNS-1524317. This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.
☆☆ A subset of the material presented in this paper appeared in [1].
* Corresponding author.
E-mail addresses: Thomas.Stahlbuhk@ll.mit.edu (T. Stahlbuhk), brooke.shrader@ll.mit.edu (B. Shrader), modiano@mit.edu (E. Modiano).

https://doi.org/10.1016/j.adhoc.2018.10.006
1570-8705/

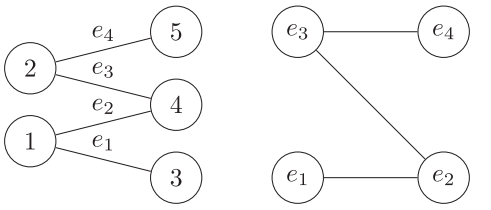
priori by the network and instead must be learned through interaction with the channel. In this work, we explore scheduling algorithms for learning channel statistics while simultaneously adapting to stochastic traffic demand. We begin by exploring how combinatorial multi-armed bandit algorithms can be used to tradeoff the exploration/exploitation requirements of channel learning and scheduling. In particular, we explore the use of the *Upper Confidence Bound* (UCB) method (see [9–11]). Bandit algorithms are historically evaluated using regret, which is defined to be the expected difference between the performance of an algorithm that must learn the best action versus a method that knows the best action a priori. It is well known that UCB obtains a regret that grows sub-linearly over time [12]. In this work, we provide a summary of these results and provide novel contributions for several combinatorial problems induced by scheduling constraints that have been previously considered in the wireless networking literature.

We then turn our attention to the problem of integrating learning algorithms into network controllers that must schedule activations to support stochastic traffic demand. To this end, we consider the use of frame-based max-weight scheduling that uses UCB methods to solve the exploration/exploitation tradeoff over the duration of a frame. Frame-based scheduling has been previously used to solve scheduling problems that are subject to Markov decision processes [13–15]. The inspiration for this method is to allow a complex decision process to be embedded within the max-weight framework. Over the length of a frame, the decision process optimizes transmissions to meet a target, while the algorithm changes the target at the frame boundaries to adapt to the arrival processes. The existence of efficient methods for solving the decision process is therefore critical to the success of the algorithm. Our proof of stability uses the sub-linear growth of regret over time to prove negative drift over the frame boundaries.

The use of learning algorithms in wireless systems has been previously considered in the literature. In particular, learning algorithms for channel access and contention resolution have been extensively analyzed (see [16–21]). In this setup, the wireless nodes can transmit on a set of shared channels in order to communicate to a receiver. The nodes must then access the different channels to both assess their capacity and effectively communicate, while resolving contention with one another. This line of work was extended in [22–24], where learning algorithms for channel access with spectrum reuse were considered.

In contrast, in the following, we consider ad-hoc networks that are geographically dispersed. Ad-hoc networks are decentralized wireless networks, that do not rely on pre-existing infrastructure to establish communications, and have application to sensor networks and autonomous systems. In this setting, nodes must schedule their transmissions; transmitters and receivers must be paired with one another and interfering pairs must not be simultaneously scheduled. The challenge in achieving this coordination is to determine which links to activate at any given time, as only a subset of the links can be activated simultaneously. Additionally, this decision often must be made by a distributed method. This application motivates our following analysis.

The rest of this paper is organized as follows. Section 2 introduces the wireless network model and learning problem. In Section 3, we provide a summary of previous results for the closely related combinatorial multi-armed bandit problem. In Section 4, we provide a distributed, greedy algorithm for learning and scheduling. The algorithm's performance relative to an offline scheduler is analyzed in Section 5. We then show how to use learning algorithms for scheduling networks with stochastic arrivals in Section 6. Our results are then extended to a general class of interference constraints in Section 7. Simulation results are presented in Section 8.



(a) Graph G (b) Conflict Graph G^c

Fig. 1. Example graph G and its conflict graph G^c. Edges that share a common node conflict and cannot be in the same activation set.

2. Network model and problem formulation

We consider a wireless network modeled as a graph $G = (N, E)$ where N is a set of wireless nodes and E is a set of undirected edges over which the nodes can communicate. The network operates over discrete time slots $t = 1, 2, \ldots, n$ for a finite time horizon n. At each time t, the nodes can choose an activation set, $M_t \subseteq E$, of edges to communicate. The set of edges that can be simultaneously activated is limited by transmitter interference. In this work, we consider activation sets that are limited by an underlying conflict graph.

A conflict graph is denoted $G^c = (N^c, E^c)$ and consists of a set of nodes N^c and undirected edges E^c. For each edge in G (i.e., each $e \in E$) there is a corresponding node in N^c. Then, for every two edges in E that conflict, and thus cannot be simultaneously activated, there exists an edge in E^c between the nodes in N^c that correspond to those two edges. The structure of the conflict graph depends on the wireless technologies used at the nodes.

In Fig. 1, we show an example of a network and its conflict graph. In this example, there are 5 nodes with 4 edges and no two edges that share a common node may be activated at the same time.[1] On the left-hand side we show the graph and on the right the corresponding conflict graph. Note that a valid activation set is an independent set in the conflict graph.[2]

Given a conflict graph, each edge has a set of adjacent edges that it interferes with and therefore cannot be simultaneously activated with. We assume that the network knows the conflict graph at the start of time and therefore does not need to learn which pairs of edges conflict. Let $\mathcal{M}$ denote the set of all activation sets that are permissible under the given conflict graph.

Our objective is to choose at each time slot an activation set that maximizes the network's throughput. To this end, we introduce a variable $w_t(e)$ that indicates the *capacity* of edge e at time slot t (i.e., the achievable data rate over the edge). At the start of the time slot, the value of $w_t(e)$ is not immediately known to the network. If edge e is included in the activation set M_t, then the nodes adjacent to e can access the link, observe the channel's instantaneous state, and communicate at rate $w_t(e)$. Thus, activating set M_t yields a *reward* equal to $\sum_{e \in M_t} w_t(e)$. Due to fading and external interference, $w_t(e)$ is time-varying. In this work, we assume $w_t(e)$ fluctuates over time as an *i.i.d.* random process with mean $\overline{w}(e)$. This model encompasses Bernoulli random failures as a special case. Note that edge weights in the same time slot can be correlated (e.g., if the same external interference signal impacts multiple edges).

Now, if the means of the variables $w_t(e)$ were known by the network a priori, then an optimal policy would choose to activate

[1] This constraint is called primary interference and is discussed in the following section.

[2] An independent set in a graph is a subset of the nodes such that no two nodes share a common edge.

at every time slot the activation set

$$M^* \triangleq \arg\max_{M \in \mathcal{M}} \sum_{e \in M} \overline{w}(e). \quad (1)$$

However, the statistics of $w_t(e)$ are usually not known at the start of time and must instead be learned through observation. This introduces an exploration/exploitation tradeoff, where the network must simultaneously learn the rates of its edges and use this knowledge to transmit data.[3] Moreover, the value of $w_t(e)$ is only observed by the nodes that are adjacent to edge e and must be disseminated across the network in order to compute M^*. In many applications, this network-wide coordination is impractical. To overcome this, the activation set can be sub-optimally constructed via a distributed, greedy method. We further discuss this below.

2.1. Notation and additional assumptions

Before continuing we provide some additional notation that will be used in the following sections. Without loss of generality, each edge reward is normalized so that $w_t(e) \in [0, 1]$. For each edge e, we use $\mathcal{N}(e)$ to denote the set of edges adjacent to e in the network's conflict graph (i.e., e's *neighborhood*). For now, we assume any two edges that are adjacent in the conflict graph do not have the same expected edge weight. This assumption is mild. Since the expected edge weights are real numbers, we would in general expect that no two weights have exactly the same mean. Then, $\mathcal{N}(e)$ can be partitioned into two sets: a superior neighborhood

$$\mathcal{N}^s(e) \triangleq \{e_0 \in \mathcal{N}(e) : \overline{w}(e_0) > \overline{w}(e)\}$$

and an inferior neighborhood

$$\mathcal{N}^i(e) \triangleq \{e_0 \in \mathcal{N}(e) : \overline{w}(e_0) < \overline{w}(e)\}.$$

Note that $e_0 \in \mathcal{N}^s(e)$ if and only if $e \in \mathcal{N}^i(e_0)$.

For any two edges e, $e_0 \in E$ define the expected difference between the edges' weights (referred to as *gap*) to be $\Delta_{e,e_0} \triangleq \overline{w}(e_0) - \overline{w}(e)$. Furthermore, define the minimum gap between two *adjacent* edge weights in the conflict graph to be

$$\Delta_{\min} \triangleq \min_{e \in E, e_0 \in \mathcal{N}^s(e)} \Delta_{e,e_0}.$$

Note that by its definition, $\Delta_{\min} \in (0, 1]$.

3. Scheduling with uncertain channel statistics

The above problem of Section 2 can be seen as a subset of the combinatorial multi-armed bandit problem first considered in [25]. In the combinatorial bandit problem, a decision maker has access to a set of elements that each offer an amount of reward that varies over time according to an *i.i.d.* random process with unknown mean. At each time step, a subset of the elements can be chosen subject to a known combinatorial constraint.[4] The individual rewards of each element in the chosen subset is then revealed to the decision maker and the total aggregate reward of the subset is collected. The rewards that would have been obtained from all elements not in the chosen subset remain hidden.

The combinatorial bandit problem could be solved inefficiently by simply using the classical multi-armed bandit framework [12]. This approach would declare each subset that is admissible under the combinatorial constraint to be an arm in the multi-armed bandit problem. Then the selection of combinatorial subsets could be decided using classical multi-armed bandit policies (e.g., [9–11]), where the aggregate reward of the chosen subset would serve as an observation for the associated arm. The problem with this approach is that it does not exploit the fact that subsets that share elements reveal information about one another [25]. For example, in our problem of the previous section, for an edge that is a member of two different activation sets, activating that edge in either set reveals partial information about the performance of the other.

Therefore, in [25], policies that assign weights to each individual element and then select subsets based on these individual weights were proposed. In [25], it was shown that such policies can have a regret that is logarithmic in time and polynomial in the combinatorial problem's size. In contrast, inefficient policies that assign weights to combinatorial subsets can have a regret that, while still logarithmic in time, is *exponential* in the combinatorial problem's size. Subsequent work in [26–28] has yielded improved bounds that, while still having a logarithmic dependence on time, have a decreased polynomial order on the problem's size. Furthermore, [27] provides a gap independent bound on the regret as well.

The above works provide a fairly complete solution for the combinatorial multi-armed bandit problem. However, one direction that is less well explored in the literature is the behavior of greedy, sub-optimal algorithms when applied to combinatorial multi-armed bandits. Greedy algorithms are of special interest in wireless networking as they can often be easily implemented by a distributed process.

Greedy algorithms for solving combinatorial multi-armed bandits where the combinatorial constraint had a matroid structure were explored in [29,30]. However, most interference constraints considered in the wireless networking literature do not induce matroid constraints on the activation sets. In [31], the performance of greedy methods applied to general combinatorial bandit problems was analyzed with respect to the offline greedy solution. In that work, it was shown that regret scales with, at most, the number of incorrect decision branches that may be followed before arriving to the greedy solution multiplied by the cost that is incurred by following the wrong decision branch. This result holds for general combinatorial structures. In this work, however, our focus will be on combinatorial constraints that have a pairwise-exclusive relationship through a conflict graph, which is a standard model for wireless networks [32]. By focusing on these constraints, we will be able to derive improved bounds.

In the following sections, we will derive regret bounds on the performance of greedy algorithms for learning activation sets, where each set is constrained to be an independent set in a conflict graph. We begin our analysis by specifically focusing on the case of primary interference. By doing so, we will derive insights that facilitate analyzing more complex interference models, which will be done towards the end of the paper.

Under primary interference, each wireless node in the network is constrained to transmit to or receive from at most one other node in the network (i.e., no node can engage in communication with multiple nodes at a given time). This constraint captures the behavior of ultra-wideband and spread spectrum communications and has been extensively used to model wireless networks in the literature [33–35]. Under the primary interference constraint, each activation set must be a matching in the network's graph.[5] Then, using the notation of the previous section, M_t would correspond to the matching chosen at time t by the controller and $\mathcal{M}$ would denote the set of all matchings.[6]

Given the objective of (1), at each time step, the controller seeks to activate the Maximum Weighted Matching (MWM) in the graph (i.e., the matching whose expected sum rewards is maxi-

[3] We concretely describe how to apply our results to networks with stochastic arrivals, where queueing dynamics must be considered, in Section 6.

[4] Note that the interference graph in our problem specifies a pairwise-exclusive combinatorial relationship.

[5] A matching is defined to be a set of edges such that no two edges share a common node.

[6] Note that under primary interference, $\mathcal{N}(e)$ is the set of all edges that share a node with edge $e \in E$.

mum over all other matchings). However, in this work we will focus on algorithms that, rather than finding a MWM, will use greedy methods to construct a Greedy Maximal Matching (GMM). A GMM is constructed, starting from an empty set, by adding the next best edge to the matching until no further edge may be added without violating the matching constraint. It is well known that any GMM is a $\frac{1}{2}$-approximation of the MWM [36] and that a GMM can be found via distributed algorithms. In Section 4, we present an algorithm for learning GMMs. The regret performance of the algorithm is analyzed in Section 5. In Section 6, we show how the distributed algorithm can be combined with frame-based scheduling to control networks with stochastic traffic demands.

In Section 7, we consider more general conflict graph constraints. Under these constraints, the activation sets correspond to the independent sets of the conflict graph. We will specifically consider networks that are parameterized by an interference degree, χ. The interference degree of the graph is defined as follows. For each $e \in E$, define

$$\chi(e) \triangleq \max_{M \in \mathcal{M}} |\mathcal{N}(e) \cap M|.$$

Then the interference degree of the entire graph is given by

$$\chi \triangleq \max_{e \in E} \chi(e).$$

Throughout the rest of this paper, we assume $\chi > 0$ since otherwise the problem is trivial. It is well known that greedy methods achieve a $\frac{1}{\chi}$-approximation of the maximum weighted set when applied to a conflict graph with interference degree χ.[7] Wireless networks parameterized by different interference degrees have been considered in the literature. (See [32], which examines several interference models and gives the corresponding interference degree.) In Section 7, we derive regret bounds for greedy learning algorithms applied to networks with interference degree constraints.

4. Learning greedy maximal matchings

In this section, we present an algorithm for learning the edge weights and scheduling activation sets under the primary interference model. Recall that under primary interference, each chosen activation set M_t must be a matching in the graph. The learning algorithm will be analyzed in the following sections. Our presentation will proceed in two parts. In Section 4.1, we present a high-level view of a method that can construct a GMM in the graph and discuss how it can be implemented in a distributed manner across the nodes of the network. We then subsequently present in Section 4.2 a learning algorithm that constructs greedy maximal matchings as a subroutine.

4.1. Greedy maximal matching

In this subsection we discuss greedy maximal matchings and briefly review the distributed method of [37] for constructing them.

For a given graph G and edge weight function $W : E \mapsto \mathbb{R}$, the GMM is the matching constructed by the myopic policy that, starting with a base empty set, iteratively adds *the next best* edge to the matching until no more edges can be added without violating the matching constraint [36]. See Fig. 2. The algorithm maintains a set of candidate edges E' that can be added to M. At each step, the algorithm selects a locally heaviest edge in E' to add to M. An edge is termed locally heaviest in E' if it does not have a neighbor $e_0 \in \mathcal{N}(e) \cap E'$ such that e_0 has a greater weight than e. After an edge is added to the matching, the algorithm removes the neighbors of that edge from E' since they may no longer be added to M without violating the matching constraint. This process continues until E' is empty. In [37] it is shown that this algorithm can be easily implemented in a distributed manner. This is achieved by having each node iteratively choose its best unmatched neighbor to match with. An edge then enters the matching when two neighbors choose one another. In [37] it is shown that this method converges with each node sending at most one message over each incident edge and thus has low control overhead.

[7] Note that under primary interference $\chi = 2$.

$E' \leftarrow E$
$M \leftarrow \emptyset$
while $E' \neq \emptyset$ **do**
  Choose an edge e that is locally heaviest in E'
  $M \leftarrow M \cup e$
  Remove e and all adjacent edges $\mathcal{N}(e)$ from E'
end while
Return matching M

Fig. 2. Greedy Maximal Matching (GMM) Algorithm [36].

$T_1(e) \leftarrow 0$, for all $e \in E$
$\widehat{w}_1(e) \leftarrow 0$, for all $e \in E$
for all $t = 1, 2, \ldots, n$ **do**
  $U_t(e) \leftarrow \widehat{w}_t(e) + \sqrt{\frac{2 \log t}{T_t(e)}}$, for all $e \in E$
  Construct a GMM M_t using $U_t(e)$ as edge weights
  for all $e \in M_t$ **do**
    Observe $w_t(e)$
    $\widehat{w}_{t+1}(e) \leftarrow \frac{T_t(e)\widehat{w}_t(e) + w_t(e)}{T_t(e)+1}$
    $T_{t+1}(e) \leftarrow T_t(e) + 1$
  end for
  for all $e \notin M_t$ **do**
    $\widehat{w}_{t+1}(e) \leftarrow \widehat{w}_t(e)$
    $T_{t+1}(e) \leftarrow T_t(e)$
  end for
end for

Fig. 3. Online Learning Algorithm.

We define the *offline* solution to be the GMM, M^o, that is found by a method that knows the average edge rewards $\overline{w}(e)$ and uses these values as the edge weights W in the GMM algorithm. An important result from [36] is that

$$\sum_{e^o \in M^o} \overline{w}(e^o) \geq \frac{1}{2} \sum_{e^* \in M^*} \overline{w}(e^*). \tag{2}$$

That is, the GMM is guaranteed to achieve at least $\frac{1}{2}$ the reward of the optimal matching. Since we assume no two adjacent edges have the same average reward, it can be shown that the GMM is unique (i.e., the algorithm will only return M^o for the offline solution). However, as noted in Section 2, in general a network will not know the average rewards $\overline{w}(e)$ a priori and must instead estimate these values from observations. In the following subsection, we present an *online* learning algorithm for choosing matching M_t at time t that uses the GMM algorithm as a subroutine.

4.2. Online learning algorithm

We now present the online learning algorithm that uses previous observations of edge rewards in order to choose a matching M_t at each time t (see Fig. 3). The algorithm maintains at time t two sets of variables: $T_t(e)$ and $\widehat{w}_t(e)$, where $T_t(e)$ is the number

of times edge e has been chosen to be in the graph matching up to time t (i.e., the number of times the edge's weight has been observed), and $\widehat{w}_t(e)$ is the sample mean of the observations.

At each time t, the algorithm calculates the Upper Confidence Bound (UCB) weight [11] for each edge:

$$U_t(e) \triangleq \widehat{w}_t(e) + \sqrt{\frac{2\log t}{T_t(e)}}. \tag{3}$$

These weights are then used to construct a GMM M_t as shown in Fig. 2. If an edge e is chosen to be in M_t, the edge is activated by its adjacent nodes and its weight $w_t(e)$ is observed by these nodes. Then, $\widehat{w}_{t+1}(e)$ and $T_{t+1}(e)$ are updated to reflect this new observation. All other edges not in M_t have their corresponding variables unchanged.

We assume $U_t(e)$ is defined on the extended real number line and takes the value infinity when $T_t(e) = 0$. Then in the construction of matching M_t, edges that have not yet been observed are given preference over edges that have been observed. It is clear that by time $t = |E| + 1$ every edge will have been observed at least once.

The above online learning method is easily made distributed. In the distributed implementation, each node maintains local variables $\widehat{w}_t(e)$ and $T_t(e)$ for each of its adjacent edges, and uses the UCB weights to distributively obtain a GMM. Then the nodes that are adjacent to an edge in the matching will activate the edge, observe the resulting reward, and update their internal variables accordingly.

5. Performance analysis under primary interference

We evaluate the performance of the online learning algorithm of Fig. 3 by comparing it to the offline solution. In general, we expect that as time advances, the sample means $\widehat{w}_t(e)$ used by the algorithm will converge to $\overline{w}(e)$, and, thus, the sequence of matchings M_t will approach M^o. In this section, we analyze the difference in the rewards obtained by the online and offline solution during this process.

We define the regret between the offline and online algorithms as

$$R^o(n) \triangleq E\left[\sum_{t=1}^{n}\left(\sum_{e^o\in M^o} w_t(e^o) - \sum_{e\in M_t} w_t(e)\right)\right]. \tag{4}$$

This metric evaluates the expected cost incurred by the online algorithm's need to learn the edge weights, and is a commonly considered metric in the stochastic multi-armed bandit literature (see [12]). We proceed to analyze this regret by first upper bounding its value. We then provide a lower bound on the regret that is asymptotically tight in the problem's parameters. Our analysis will show that the online learning algorithm efficiently learns the offline GMM.

5.1. Regret bounds

We begin our discussion by first establishing several relationships between the offline solution M^o and online solution M_t. An important observation, noted in [35], is that given two maximal matchings M_t and M^o on graph G, the union of the matchings' edges $(M_t \cup M^o)$ form a set of connected components (see Fig. 4). Each connected component is either: (1) a single edge between two nodes, (2) an even-length ring of alternating edges in M_t and M^o, or (3) a path of alternating edges [35]. From this we see that if an edge is in M_t but not M^o (i.e., set $M_t - M^o$), then it must be adjacent to at least one but no more than two edges in M^o. The analogous statement applies to those edges in $M^o - M_t$. Using the above, we give, in Lemmas 1 and 2, two important relationships between M_t and M^o. Note that these lemmas follow from the analysis of greedy algorithms for graph matchings and are provided here for completeness.

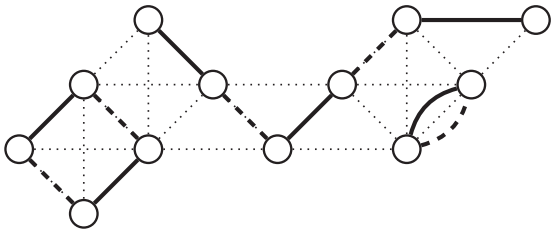


Fig. 4. Example graph with edge set shown in light dots. Offline solution M^o is shown in solid lines, and online solution M_t is shown in dashed lines.

Lemma 1. *For all $e \in M_t$, either*

1) $e \in M^o$, *or*
2) $\exists e^o \in M^o \cap \mathcal{N}^s(e)$.

Proof. If $e \in M^o$ then by the definition of a matching, no adjacent edge to e may be in M^o. Thus, if (1) is true (2) cannot be.

Now, assume $e \notin M^o$. Consider the offline solution M^o constructed by running the algorithm in Fig. 2 with edge weights $\overline{w}(e)$. By the algorithm's definition, in the iterative construction of M^o, e was removed from E' when one of its neighbors e^o became locally heaviest (implying $\overline{w}(e^o) > \overline{w}(e)$) and entered the matching M^o. Thus, (2) is true. □

An immediate consequence of Lemma 1 is that an edge $e \in M_t$ can be adjacent to at most one edge in M^o with an expected edge weight that is less than e's expected weight. Given the above, we proceed with the following analogous lemma.

Lemma 2. *At each time t and edge $e^o \in M^o$, either*

1) $e^o \in M_t$, *or*
2) $\exists e \in M_t \cap \mathcal{N}(e^o)$ *such that* $U_t(e) \geq U_t(e^o)$.

Proof. The online algorithm constructs the matching M_t via the same method as the offline algorithm, except the online algorithm uses the UCB variables $U_t(e)$ as edge weights instead of the expected edge weights $\overline{w}(e)$ as in the offline algorithm. Thus, the result follows from a simple modification of the proof of Lemma 1. □

Now, consider the edges sorted in descending order based on mean rewards $\overline{w}(e)$. If the sorting based on UCB weights $U_t(e)$ preserves this order, then the online and offline matchings will be equal. However, errors emerge when the UCB weights swap the order of two adjacent edges in the graph. Complicating matters, whether an edge is or is not added to M_t cannot be concluded simply by determining whether its UCB weight places it out-of-order with respect to its neighbors.

For example, consider an edge $e \notin M^o$. This edge may have entered M_t because its UCB weight falsely indicated that it had a higher mean reward, $\overline{w}(e)$, than a neighboring edge $e^o \in M^o$ that should have been added instead. This can be thought of as a local inaccuracy in the UCB weights. However, e may also have been placed in M_t because its superior neighbor e^o was prevented from entering the matching by a local inaccuracy e^o had with one of its other neighbors. Then, since e^o was excluded from M_t, e was promoted into the matching.

From the above, one can see that a local error can propagate through the graph. Indeed, the difference between the two matchings in Fig. 4 could be caused by only two out-of-order UCB weights. For greedy algorithms in arbitrary combinatorial problems, this behavior could be very problematic; a single mistake

could set the greedy policy down a path where it accrues very large regret that grows with the number of elements in the combination. However, the above lemmas can be used to show that although a single, unrepresentative UCB weight can cause M_t to look very different from M^o, the amount of regret that is accrued is constrained to the cost of the original mistake. This allows us to bound the regret using the following theorem. The proof is provided in Appendix A.

Theorem 1.

$$R^o(n) \leq \sum_{e\in E}\left(\sum_{e^o\in M^o\cap\mathcal{N}^s(e)} \overline{w}(e^o)\left(\frac{8\log n}{\Delta^2_{e,e^o}}+\frac{\pi^2}{3}\right)\right)+\frac{|N||E|}{2} \tag{5}$$

which implies the following asymptotic bound

$$\limsup_{n\to\infty}\frac{R^o(n)}{\log n}\leq 16\frac{|E|}{\Delta^2_{\min}}. \tag{6}$$

Note that (6) follows by bounding $\overline{w}(e)$ by 1 and Δ_{e,e^o} with $\Delta_{\min}$ in (5). To give intuition, note that the summations in (5) include every edge e and its (no more than two) superior neighbors in M^o. These are the boundaries where regretful decisions can be made. However, given the mistake, the resulting expected regret accounted for in the summation is a value that depends only on the time horizon n and local edge weights. As a result, (6) shows that, asymptotically in n, the expected regret grows at most linearly with the number of edges in $|E|$.

Furthermore, it can be shown that the number of times that M_t does not equal M^o is bounded asymptotically by the following theorem.

Theorem 2. *The expected number of time slots during which the online learning algorithm does not select M_t to be equal to M^o is $O(\log n)$.*[8]

The proof of Theorem 2 can be found in Appendix B, and we will use the result to construct a lower bound on the regret of the online algorithm.

The right-hand side of (6) scales linearly with the number of edges in the graph and $\frac{1}{\Delta^2_{\min}}$. For UCB methods in stochastic multi-armed bandits, it is well-known (cf., [12, Theorem 2.1]) that there exists an upper bound on regret that scales inversely proportional to the minimum difference between two arms. Thus, a reasonable question is whether we could improve the bound of (6) with a bound that scales linearly with $\frac{1}{\Delta_{\min}}$. In the following, we will show that there exists a graph G and distribution on edge weights $w_t(e)$ where the regret of the online learning algorithm has the same scaling behavior as in (6). Thus, the above bound is tight in its parameters.

Next, we provide an instantiation of graph G and edge weights $w_t(e)$ such that there exists a lower bound on $R^o(n)$ that has the same dependence on $\Delta^2_{\min}$ and $|E|$ as (6). We begin with the theorem statement.

Theorem 3. *There exists a graph G and specified edge weights such that for the online learning algorithm*

$$\liminf_{n\to\infty}\frac{R^o(n)}{\log n}\geq\frac{1}{24}\frac{|E|}{\Delta^2_{\min}}. \tag{7}$$

To this end, consider a graph G composed of a set of $\frac{|E|}{3}$ connected components as shown in Fig. 5. (Each component is disconnected from the others.) The weights of every edge in the graph are drawn from a Bernoulli distribution. For each connected component, the exterior edges, e_{ext1} and e_{ext2}, are perfectly correlated (i.e., at each time t have the same realization) and take the value 1 with probability 0.5. Likewise, for each component, the weight of the interior edge, e_{int}, is independent of the exterior edge weights and takes value 1 with probability $0.5-\Delta$ for some constant $\Delta\in(0,\,0.5)$. Then, using Theorem 2 which guarantees that, for any graph instantiation G and edge weight distributions, all matchings not equal to M^o are chosen $O(\log n)$ times in expectation and applying a well known result for multi-armed bandits (see [12, Theorem 2.2]) gives the theorem. The complete proof can be found in Appendix C.

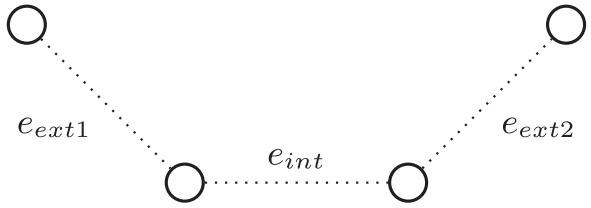


Fig. 5. A single connected component in the graph used for Theorem 3.

It is important to note that Theorem 3 applies only to the algorithm in Fig. 3 and is not meant to bound the regret for every possible policy. The theorem results from the fact that the online learning algorithm aggressively targets finding M^o, choosing all other matchings $O(\log n)$ times. This comes at a cost. It drives the algorithm to include in the matching an edge that has slightly better expectation than two of its potential neighbors, even if choosing those two neighbors in combination is clearly the better option in terms of reward. This leads to the $\Delta^2_{\min}$ term in the denominator of (7).

5.2. Simulation of the learning algorithm's convergence

To illustrate the above results, in Fig. 6 we show the performance of the online learning algorithm on a 4×4 grid with Bernoulli rewards on each edge. The graph is pictured in Fig. 6(a), where each edge is labeled with its mean reward $\overline{w}(e)$. The offline matching M^o is shown in solid lines. Note that M^o is not equal to the MWM for this problem. In Fig. 6(b) the average regret $\frac{R^o(n)}{n}$ is shown to be decaying to zero with the time horizon n as the sequence of matchings M_t settle on M^o. Note that in this example the average regret approaches zero from above. However, since in general $M^o\neq M^*$, this is not always the case, and it is not hard to construct problem instantiations where the average regret becomes negative and approaches zero from below. This can occur when there is a matching *near to* M^o that outperforms the GMM.

In Fig. 6(c), we show the rate at which four different edges, labeled 1 through 4 in Fig. 6(a), are selected to be in matching M_t over time horizon n (i.e., $\frac{T_n(e)}{n}$). As would be expected, those edges in M^o approach 1 with increasing n, and those outside of M^o approach 0. We note that the online algorithm has an easier time discerning that edge 2 should not be in the GMM than it does edge 3. This is because edge 2 has a large gap between its average reward and the average reward of its neighbors in M^o.

6. Stochastic network control

We now turn to the problem of learning channel conditions in order to control networks with stochastic traffic demands. In this section, we will continue to focus on the primary interference model and the learning algorithm of Fig. 3. At the end of this section, we address other interference models and learning algorithms.

[8] For two functions $f(n)$ and $g(n)$, we denote $f(n)=O(g(n))$ if $\exists n_0$ and $\beta>0$ such that $|f(n)|\leq\beta|g(n)|$ for all $n\geq n_0$.

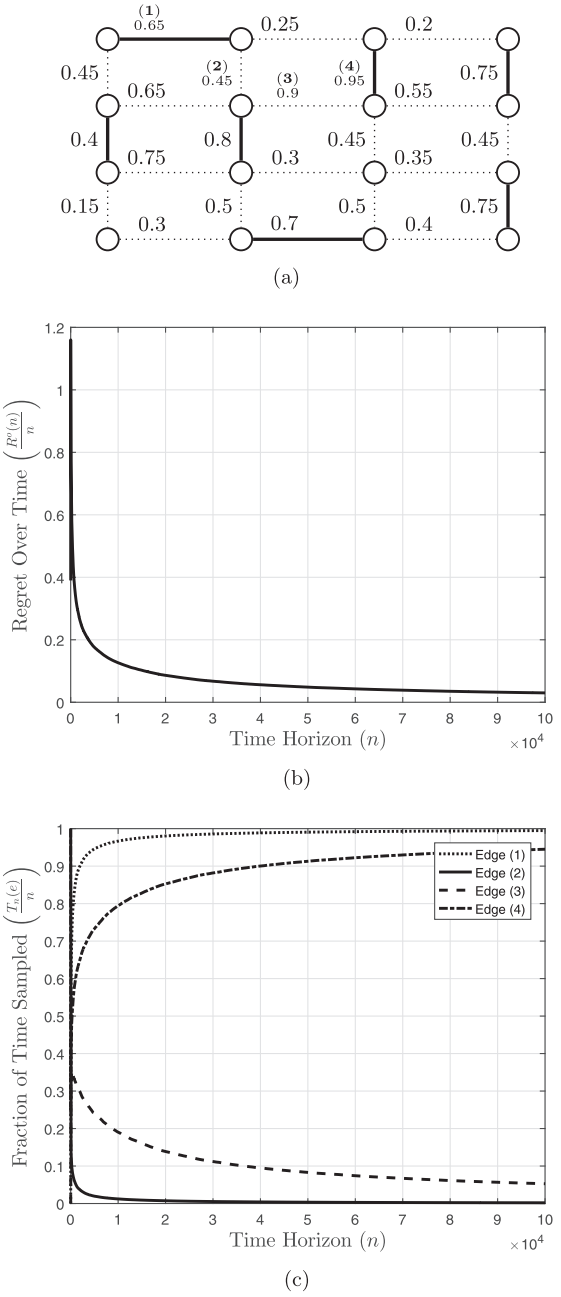


Fig. 6. (a) Graph with edge set in light dots; edges labeled with their mean rewards $\overline{w}(e)$. Matching M^o is shown in solid lines. (b) Average regret accrued over the time horizon n. (c) Fraction of time slots that labeled edges are chosen to be in matching M_t. Plots are averaged over 100 pseudo-random runs.

Time is slotted and indexed as $t = 1, 2, \ldots$. Let $A_t(e)$ denote the number of packets that arrive to the network at time t to be transmitted over edge e. All traffic is single-hop and, for simplicity, we assume traverses the edge in a single direction. Extensions to accommodate bidirectional and multi-hop traffic can be derived from our results. We consider $A_t(e)$ to be a Bernoulli process with rate $\lambda(e)$. We denote the vector of arrivals $\vec{\lambda} \in \mathbb{R}_+^{|E|}$. Upon arrival, packets wait in a queue until successfully transmitted over the edge. Denote the queue backlog at time t as $Q_t(e)$.

We let $c_t(e) \in [0, 1]$ be the number of packets that can traverse edge e at time t and assume this process is *i.i.d.* with mean $\bar{c}(e)$. When $c_t(e) \in (0, 1)$, we allow for a fraction of a packet to be transmitted. Note that in the previous sections we used $w_t(e)$ to denote an edge's throughput and it was this edge weight that our online algorithm was focused on learning. In this section, $w_t(e)$ will be used to denote a normalized edge capacity as defined below. Importantly, as above, $w_t(e)$ will be the edge weight that is learned by our algorithm. However, as opposed to the previous sections, the results of this section do not require that adjacent edges have different average weights.

At time t, the network can activate an activation set M_t of edges. Then, the queue backlogs evolve according to the following update equation:

$$Q_{t+1}(e) = \begin{cases} \max\{Q_t(e) - c_t(e), 0\} + A_t(e) & \text{if } e \in M_t \\ Q_t(e) + A_t(e) & \text{if } e \notin M_t \end{cases}$$

A queue is defined to be *rate stable* if [3]

$$\lim_{t\to\infty} \frac{Q_t(e)}{t} = 0, \text{ with prob. 1.}$$

Then, our objective is to design a scheduling policy that stabilizes all queues in the network. For a given graph G and average capacities $\bar{c}(e)$, we let Λ denote the stability region of the network. Recall that the stability region is defined to be the set of all arrival rate vectors, $\vec{\lambda}$, that permit a scheduling policy that can rate stabilize the queues.

The max-weight policy that at each time t weights each edge by $Q_t(e)\bar{c}(e)$ and then schedules the MWM in the graph, is known to stabilize any arrival rate vector interior to Λ [3]. If instead of scheduling a MWM, a GMM is scheduled at each time step, then from [4,5], any arrival rate vector interior to $\frac{\Lambda}{2}$ can be stabilized. (i.e., The set $\vec{\lambda}$ such that $2\vec{\lambda}$ is interior to Λ.) In the proceeding, we refer to this latter policy as the *offline GMM scheduler*. To directly implement the offline GMM scheduler, the nodes must know mean capacities $\bar{c}(e)$. Since these quantities are not known a priori, we proceed to describe a learning algorithm that can stabilize the same $\frac{\Lambda}{2}$ throughput region. Our method will use the online learning algorithm as a component.

We propose a frame-based scheduling policy to stabilize the network. Our method partitions time into frames of length n time slots, where frame $k \in \mathbb{N}$ begins at time slot $t_k = (k-1) \times n + 1$. At the start of each frame, a snapshot of the queue backlogs $Q_{t_k}(e)$ is taken. Then, over the duration of each frame, the policy schedules a sequence of matchings M_t using the algorithm in Fig. 3.[9] However, rather than observing the edge capacities directly as in Section 4, the online learning algorithm uses the following normalized observations:

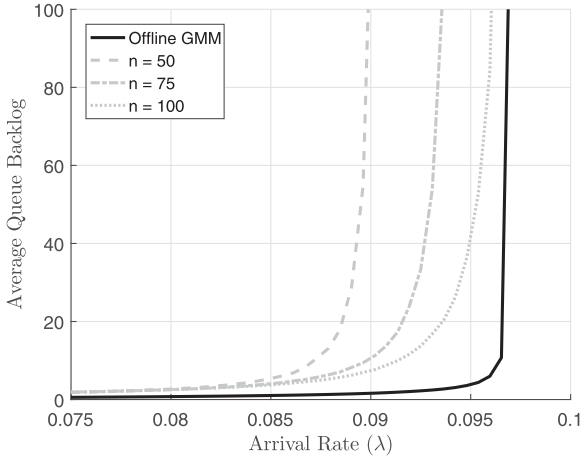
$$w_t(e) \triangleq \frac{Q_{t_k}(e)c_t(e)}{\max_{e\in E} Q_{t_k}(e)}. \tag{8}$$

Note that (8) is the (normalized) product of the queue backlog at the start of the frame with the edge capacity that is observed by activating edge e at time t. If $\max_{e\in E} Q_{t_k}(e) = 0$, we set $w_t(e) \triangleq 0$. Since, $c_t(e) \in [0, 1]$, $w_t(e) \in [0, 1]$ as well. Furthermore, it is easy to see that, when $\max_{e\in E} Q_{t_k}(e) \neq 0$, the average edge weight conditioned on Q_{t_k} is given by

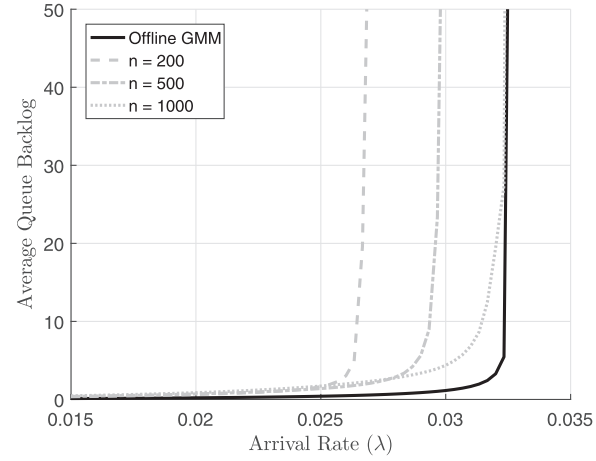
$$\overline{w}(e) = \frac{Q_{t_k}(e)\bar{c}(e)}{\max_{e\in E} Q_{t_k}(e)}, \tag{9}$$

and it is this value that the online learning algorithm is estimating with its UCB weights. In this section, two adjacent edges are allowed to have the same mean weight $\overline{w}(e)$. Over each frame, the objective of the learning algorithm is to converge to the GMM that would be constructed using edge weights $\overline{w}(e)$ (i.e., the offline solution).

[9] The UCB weights consider time to be "restarted" at the beginning of each frame. Thus, for frame k, $U_t(e) \triangleq \widehat{w}_t(e) + \sqrt{(2\log(t_k - t + 1))/T_t(e)}$.



(a) Bernoulli channel capacities



(b) Rayleigh channel capacities

Fig. 7. Average queue backlog for the online learning algorithm with varying frame size n and the offline GMM scheduler operating on a 4×4 grid with uniform arrival rate λ to each edge.

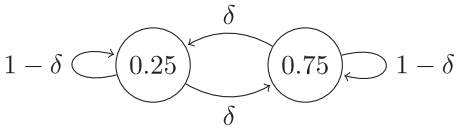


Fig. 8. Markov chain for time-varying channel state. When in a given state, the edge has Bernoulli rates with the labeled mean.

We now analyze the stability of the frame-based policy. Assume the arrival vector is interior to the GMM throughput region (i.e., $\frac{\Lambda}{2}$). This implies there exists a scalar $\epsilon > 0$ such that $\vec{\lambda} + \epsilon\vec{1} \in \frac{\Lambda}{2}$ where $\vec{1}$ is the vector of all ones. For each frame k, define the Lyapunov function $L(t_k) \triangleq \frac{1}{2}\sum_{e\in E}\left(Q_{t_k}(e)\right)^2$ and the Lyapunov drift $D(t_k) \triangleq L(t_{k+1}) - L(t_k)$. Then, using standard techniques (see [3]) it can be shown that

$$E\left[D(t_k)|\vec{Q}_{t_k}\right] \le V + n\sum_{e\in E}\lambda(e)Q_{t_k}(e) - E\left[\sum_{t=t_k}^{t_k+n-1}\sum_{e\in M_t} c_t(e)Q_{t_k}(e)\middle|\vec{Q}_{t_k}\right] \quad (10)$$

where $\vec{Q}_{t_k}$ is the vector of queue backlogs at the start of frame k and V is a finite constant for a fixed value of n. Now, if the right-hand side of (10) becomes negative as $L(t_k)$ becomes large, it follows that all queues in the network are rate stable [3]. Define M^* to be a (possibly nonunique) MWM in graph G given mean edge weights $\overline{w}(e)$ defined in (9) (i.e., the offline maximum weighted matching). Adding and subtracting $\frac{1}{2}E\left[\sum_{t=t_k}^{t_k+n-1}\sum_{e\in M^*} c_t(e)Q_{t_k}(e)\middle|\vec{Q}_{t_k}\right]$ from the right-hand side of (10) and using $\vec{\lambda} + \epsilon\vec{1} \in \frac{\Lambda}{2}$ gives

$$E\left[D(t_k)|\vec{Q}_{t_k}\right] \le V - \epsilon n\sum_{e\in E}Q_{t_k}(e) + E\left[\sum_{t=t_k}^{t_k+n-1}\left(\frac{1}{2}\sum_{e\in M^*}c_t(e)Q_{t_k}(e) - \sum_{e\in M_t}c_t(e)Q_{t_k}(e)\right)\middle|\vec{Q}_{t_k}\right]. \quad (11)$$

Since, by the definition of $w_t(e)$ in (8),

$$E\left[\sum_{t=t_k}^{t_k+n-1}\left(\frac{1}{2}\sum_{e\in M^*}c_t(e)Q_{t_k}(e) - \sum_{e\in M_t}c_t(e)Q_{t_k}(e)\right)\middle|\vec{Q}_{t_k}\right] = E\left[\sum_{t=t_k}^{t_k+n-1}\left(\frac{1}{2}\sum_{e\in M^*}w_t(e) - \sum_{e\in M_t}w_t(e)\right)\middle|\vec{Q}_{t_k}\right]\left(\max_{e\in E}Q_{t_k}(e)\right)$$

we see from (11) that if

$$\frac{1}{n}E\left[\sum_{t=t_k}^{t_k+n-1}\left(\frac{1}{2}\sum_{e\in M^*}w_t(e) - \sum_{e\in M_t}w_t(e)\right)\middle|\vec{Q}_{t_k}\right] < \epsilon \quad (12)$$

the Lyapunov drift will be negative for sufficiently large values of $L(t_k)$ and thus all queues will be rate stable [3].

We proceed to analyze the rate at which the left-hand side of (12) goes to zero when the online learning algorithm is used to schedule the activations M_t over the frame. For each frame, the online learning algorithm is run independently of the previous frames and does not use the observations obtained in those frames. We define the regret *with respect to the approximation ratio* to be

$$R^a(n) \triangleq E\left[\sum_{t=1}^{n}\left(\frac{1}{2}\sum_{e^*\in M^*}w_t(e^*) - \sum_{e\in M_t}w_t(e)\right)\right].$$

Note that $\frac{R^a(n)}{n}$ is the left-hand side of (12) where, for simplicity, we have dropped the notation for conditioning on $\vec{Q}_{t_k}$ and indicate the start of the frame with time 1. As opposed to the definition of regret $R^o(n)$ in (4) which compared the performance of the online learning algorithm to the offline solution M^o, $R^a(n)$ compares the performance relative to the approximation-ratio of the offline MWM, M^*. We then have the following lemma.

Lemma 3.

$$R^a(n) \le \frac{1}{2}\sum_{e\in E}\left(\sum_{e^*\in M^*\cap\mathcal{N}^s(e)}\left(\frac{8\log n}{\Delta_{e,e^*}} + \frac{\pi^2}{3}\Delta_{e,e^*}\right)\right) + \frac{1}{2}|E||M^*|$$

which implies the following asymptotic bound[10]

$$\limsup_{n\to\infty}\frac{R^a(n)}{\log n} \le 8\frac{|E|}{\Delta_{\min}}.$$

The complete proof is given in Appendix D.

Note that Δ_{e,e^*} is a function of $\vec{Q}_{t_k}$ and can be arbitrarily small. Therefore, Lemma 3 does not imply that there exists a value of n such that (12) holds for all $L(t_k)$ sufficiently large. However, we now apply a well known technique from the multi-armed bandit

[10] This asymptotic bound holds when there exists a pair of adjacent edges in the graph such that the edges do not have the same expected weight. Then, $\Delta_{\min}$ is defined to be the minimum of the gaps not equal to zero. Otherwise, we can replace the asymptotic bound with 0.

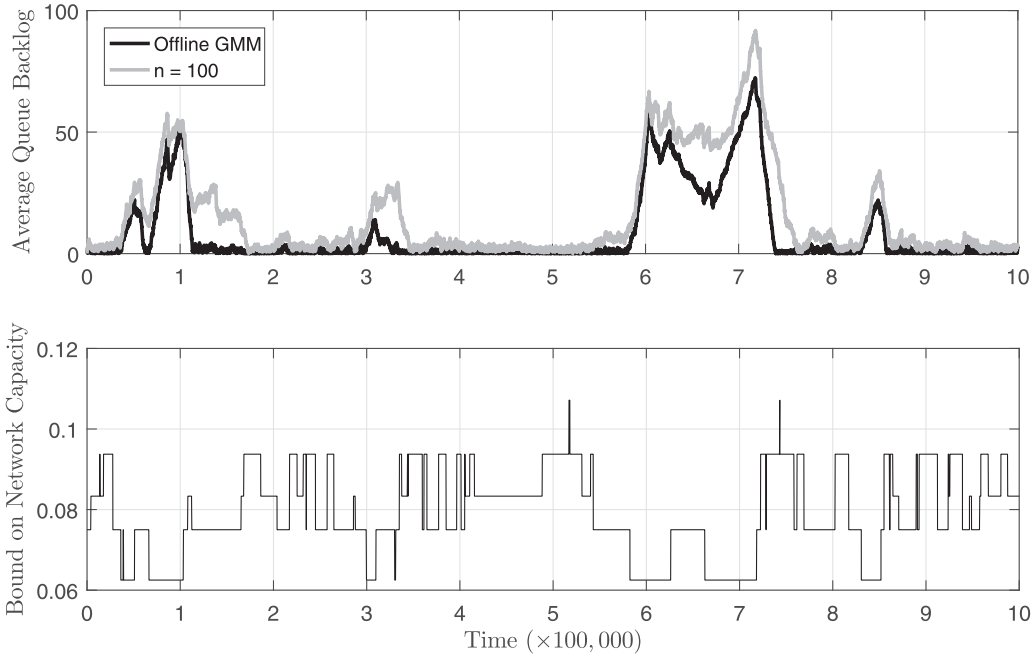


Fig. 9. (Top) The average queue backlog across the network for the frame-based learning algorithm with frame size $n = 100$ and the offline GMM scheduler. The values of $\bar{c}(e)$ are determined by the Markov Chain of Fig. 8. (Bottom) Instantaneous upper bound on the network's capacity.

literature in order to obtain a bound on $R^a(n)$ that is independent of the gaps (see [12, Section 2.4.3]). The technique designates each pair of adjacent edges as belonging to one of two sets: those that have an expected difference greater than positive value Δ_{th} and those that have an expected difference less than Δ_{th}. Since the contribution to $R^a(n)$ of the latter group must grow slowly with time, strategically choosing Δ_{th} and using Lemma 3 gives

Corollary 1.

$$R^a(n) \leq 4\sqrt{2|E||M^*|n\log n} + |E|\left(\frac{\pi^2}{3} + \frac{|M^*|}{2}\right).$$

The complete proof is given in Appendix E.

Note that $|M^*| \leq \frac{|N|}{2}$. Therefore, we can obtain a bound that is only a function of the number of edges and nodes in the graph and the time horizon n. Moreover, as $n \to \infty$, $\frac{R^a(n)}{n} \to 0$. Comparing to (12), we see that there exists an n large enough such that the Lyapunov drift is negative for large queue backlogs. Thus, by choosing a value of n such that (12) is true, the network can be rate stabilized. We therefore have the following theorem.

Theorem 4. *Under the primary interference constraint, the frame-based learning policy can achieve rate stability for any set of arrival rates interior to the* $\frac{\Lambda}{2}$ *region.*

We now briefly address some practical aspects of the frame-based method. The method relearns the channel statistics at every frame boundary. When $w_t(e)$ is strictly *i.i.d.* for all time t, this is unnecessary. However, in many practical wireless networks, although $w_t(e)$ is *i.i.d.* over short time horizons (e.g., because of multi-path fading and interference), over larger horizons the underlying statistics of $w_t(e)$ vary as the network and its environment evolve over time. Then, the frame-based method learns the channel statistics in order to favor edges with temporary, high throughputs. The frame corresponds to the window over which the statistics are averaged. In general, the length of this window, n, will be application dependent and may even need to adapt to changing network dynamics. The online learning algorithm is suited for this scenario as its computations do not depend on n. In Section 8, we illustrate the algorithm's performance in a network with changing statistics.

Finally, we discuss the use of the frame-based algorithm under other interference constraints and learning algorithms. Note that the key to proving network stability, above, was to show that there exists a gap independent bound for the regret relative to the approximation ratio. In Corollary 1, we showed that the distributed, learning algorithm of Fig. 3 has such a bound. Previously, in [27], it was shown that learning algorithms that solve the combinatorial multi-armed bandit algorithm using a centralized method can likewise achieve a similar bound. Thus, we see that when centralized control can be implemented, the frame-based algorithm can use these methods over its frame boundaries to achieve the entire network stability region. Note that these bounds hold for general combinatorial constraints and, therefore, scheduling constraints that are subject to conflict graphs. Importantly, the duration of the frame length depends on the speed at which learning can take place. Longer frame boundaries, in general, lead to larger packet delays. Thus, it is critical to establish methods that are quick to learn the network's edge weights. This requirement motivated our previous derivation of regret bounds for networks that are constrained by primary interference and reliant upon distributed, learning algorithms. In the next section, we extend these results to networks that have conflict graphs with a fixed interference degree.

7. Extensions to other scheduling constraints

In this section, we extend our analysis to conflict graphs that have a fixed interference degree χ. We proceed to derive bounds that are analogous to Theorem 1, Lemma 3, and Corollary 1.

Note that the algorithm in Fig. 3 can be used to learn activation sets in networks with general interference constraints. The only difference is that the chosen activation set M_t will now be an independent set in the conflict graph and not necessarily a graph matching. Now, when the algorithm of Fig. 2 is called to construct the activation set, it iteratively adds elements to the set until no more edges can be added without violating the conflict graph's

constraints. As before, an edge is added when it becomes locally heaviest (i.e., it is in E' with a weight that is no less than any of its conflicting neighbors, $\mathcal{N}(e)$, that are still in E'). Note that this algorithm can be made distributed by having the end nodes of each edge inform the end nodes of the conflicting edges of changes in status. Since conflicts tend to be localized in the network, this prevents the nodes from having to know the entire network's state.

Let M^o be the greedy offline activation set that is found by the greedy method when the average weights $\overline{w}(e)$ are known. Note that Lemmas 1 and 2 continue to be true for general conflict graph constraints (i.e., there was nothing specific to the primary interference model that made these claims true). Defining the learning algorithm's regret with respect to the greedy offline activation set M^o as $R^o(n)$ (see (4)), we then have the following bounds.

Corollary 2. *For a generalized interference graph with interference degree* χ,

$$R^o(n) \leq |M^o| \sum_{e\in E}\left(\sum_{e^o\in M^o\cap\mathcal{N}^s(e)}\left(\frac{8\log n}{\Delta^2_{e,e^o}}+\frac{\pi^2}{3}\right)\right)+|M^o||E|$$

which implies the following asymptotic bound

$$\limsup_{n\to\infty}\frac{R^o(n)}{\log n}\leq 8\chi\frac{|E||M^o|}{\Delta^2_{\min}}.$$

The proof of Corollary 2 follows from the proof of Theorem 2 and noting that the expected regret at time t given that $M_t\neq M^o$ is at most $|M^o|$. Note that this bound is, in general, worse than the bound of Theorem 1. This is because an estimation error, that causes an erroneous edge to be added to the greedy schedule, can set the greedy algorithm down a path where it chooses a much worse activation set than M^o. As can be seen in the proof of Theorem 1, this effect does not occur in GMMs.

Now, for a conflict graph with interference degree, χ, define the regret with respect to the $\frac{1}{\chi}$-approximation ratio to be

$$R^a_\chi(n)\triangleq E\left[\sum_{t=1}^{n}\left(\frac{1}{\chi}\sum_{e^*\in M^*}w_t(e^*)-\sum_{e\in M_t}w_t(e)\right)\right].$$

Then, in a proof similar to Lemma 3 and Corollary 1, we can show the following.

Corollary 3. *For a generalized interference graph with interference degree* χ,

$$R^a_\chi(n)\leq \frac{1}{\chi}\sum_{e\in E}\left(\sum_{e^*\in M^*\cap\mathcal{N}^s(e)}\left(\frac{8\log n}{\Delta_{e,e^*}}+\frac{\pi^2}{3}\Delta_{e,e^*}\right)\right)+\frac{1}{\chi}|E||M^*|$$

which implies the following asymptotic bound

$$\limsup_{n\to\infty}\frac{R^a_\chi(n)}{\log n}\leq 8\frac{|E|}{\Delta_{\min}}$$

and the following gap independent bound

$$R^a_\chi(n)\leq 4\sqrt{2|E||M^*|n\log n}+|E|\left(\frac{\pi^2}{3}+\frac{|M^*|}{\chi}\right).$$

Note that, as with Lemma 3 and Corollary 1, this corollary does note require that conflicting edges have different average weights and that it produces a gap independent bound. Thus, if the greedy method is used in frame-based scheduling, in an analysis similar to the previous section, we can show that it achieves network stability for the $\frac{\Lambda}{\chi}$–throughput region.

8. Simulation results

In this section, we illustrate the performance of the online learning algorithm for stochastic network control. Our simulations are for a 4×4 grid topology (as in Fig. 6(a)) where each edge receives Bernoulli arrivals with the same fixed rate, λ, for all edges. We consider the primary interference constraint where each activation set must be a matching in the graph.

We first consider an environment where the mean edge capacities $\overline{c}(e)$ do not vary with time. In Fig. 7 we show the performance of the offline GMM scheduler (that knows the means $\overline{c}(e)$ a priori) and the frame-based learning algorithm for varying frame sizes, n. The plots show the nodes' average queue backlog over a simulation of 10 million time slots for increasing arrival rate λ. The point where the backlogs grow rapidly indicates the boundary of that method's stability region. We show results for two different distributions on the channel capacities $c_t(e)$. In Fig. 7(a) the capacities are Bernoulli (i.e., the channel is on or off) with a mean that was chosen, at the start of time, uniformly between 0.25 and 0.75. In Fig. 7(b) the distributions are Rayleigh with a scale parameter chosen uniformly between 0.05 and 0.25.

An upper bound on the maximum arrival rate λ that can be supported by the network is given by

$$\lambda\leq\min_{v\in N}\frac{1}{\sum_{e\in\mathcal{A}(v)}\left(\overline{c}(e)\right)^{-1}} \tag{13}$$

where $\mathcal{A}(v)$ denotes the set of edges adjacent to node v. Applying this bound to the values of $\overline{c}(e)$ used in Fig. 7(a) and (b) gives 0.0969 and 0.0325, respectively. Therefore, we see that in both cases, the offline GMM scheduler nearly achieves the network's capacity. As n becomes large, the online learning algorithm's stability region approaches the stability region of the offline GMM scheduler.

In Fig. 9 we illustrate the performance of the offline GMM scheduler and the frame-based learning algorithm when each edge's average capacity, $\overline{c}(e)$, is time-varying. In this scenario, for each edge $e\in E$, $\overline{c}(e)$ evolves over the time slots according to the Markov chain in Fig. 8. We choose $\delta = 2\times 10^{-5}$ in order to ensure the channels' statistics are stationary for long periods of time. Given an edge's mean value $\overline{c}(e)$ at time t, the edge capacities $c_t(e)$ are Bernoulli with the given rate. The arrival rate to each edge is $\lambda = 0.07$. In the bottom of Fig. 9, we plot the bound on the network's capacity given by (13) for the mean edge capacities at time t. Note that when the bound falls below 0.07 the queue backlogs of both the offline GMM scheduler and the frame-based learning method grow since the arrival rate exceeds the network's instantaneous capacity. Overall, the frame-based learning algorithm's performance mirrors the performance of the offline GMM scheduler with the frame-based learning algorithm maintaining a slightly larger backlog.

9. Conclusion

Scheduling policies based upon greedily constructed schedules have been well studied in the literature and have produced distributed algorithms with good performance. However, previous work required channel state information be known in advance. In this work, we considered the problem of jointly learning the channel statistics while simultaneously scheduling transmissions to support high throughput. We presented learning algorithms that choose sequences of greedily constructed schedules in order to solve this exploration/exploitation tradeoff. By analyzing how estimation errors impacted the greedy construction, we were able to derive novel bounds on the algorithms' regrets. Subsequently, for networks with stochastic traffic arrivals, we illustrated how our methods could be used in frame-based scheduling.

Appendix A. Proof of Theorem 1

We use Lemmas 1 and 2 to establish a bound on the regret of any sample path of $w_t(e)$. Note that the weights $w_t(e)$ determine $U_t(e)$ and M_t at each time step. We begin by noting that

$$\sum_{e^o\in M^o} w_t(e^o) - \sum_{e\in M_t} w_t(e) = \sum_{e^o\in M^o - M_t} w_t(e^o) - \sum_{e\in M_t - M^o} w_t(e). \quad \text{(A.1)}$$

From Lemma 2, each edge $e^o \in M^o - M_t$ must be adjacent to an edge $e \in M_t$ with $U_t(e) \geq U_t(e^o)$. Since this edge is in $\mathcal{N}(e^o)$ it cannot be in M^o. Now, this adjacent edge can be in either $\mathcal{N}^s(e^o)$ or $\mathcal{N}^i(e^o)$. Thus, denoting the indicator function as $\mathbf{1}_{\{.\}}$, we can upper bound (A.1) with

$$\begin{aligned}&\sum_{e^o\in M^o} w_t(e^o) - \sum_{e\in M_t} w_t(e) \leq \\ &\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^s(e^o)} w_t(e^o)\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}} \\ &+ \sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} w_t(e^o)\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}} - \sum_{e\in M_t - M^o} w_t(e). \quad \text{(A.2)}\end{aligned}$$

Note that the inequality in (A.2) results from the possibility that there can be two edges in M_t that are adjacent to edge e^o that both have greater UCB weights than e^o. In this case, the right-hand side would over count the contribution of $w_t(e^o)$ to the regret.

Now, from Lemma 1, each edge $e \in M_t - M^o$ must be adjacent to at least one edge $e^o \in M^o$ with $\overline{w}(e^o) > \overline{w}(e)$. As noted above, this implies that e can be adjacent to at most one edge $e^o \in M^o$ with weight $\overline{w}(e^o) < \overline{w}(e)$. In other words, there exists at most one edge $e^o \in M^o$ such that $e \in \mathcal{N}^s(e^o)$. From this we can upper bound (A.2) as

$$\begin{aligned}&\sum_{e^o\in M^o} w_t(e^o) - \sum_{e\in M_t} w_t(e) \leq \\ &\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^s(e^o)} (w_t(e^o) - w_t(e))\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}} \\ &+ \sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} w_t(e^o)\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}}. \quad \text{(A.3)}\end{aligned}$$

Note that the above inequality results from the fact that we may undercount the (negative) contribution of the edges $e \in M_t - M^o$ on the right-hand side.

We now bound the regret (cf., (4)) over times $|E|+1, |E|+2, \ldots, n$. Starting with (A.3), summing over time, and taking expectation

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\sum_{e\in M^o} w_t(e) - \sum_{e\in M_t} w_t(e)\right)\right] \leq E\Bigg[\sum_{t=|E|+1}^{n}\Bigg(\ldots \\ &\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^s(e^o)} (w_t(e^o) - w_t(e))\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}} \\ &+ \sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} w_t(e^o)\mathbf{1}_{\{e\in M_t\cap U_t(e)\geq U_t(e^o)\}}\Bigg)\Bigg]. \quad \text{(A.4)}\end{aligned}$$

Now, at time t, random variables $w_t(e)$ and events

$$\{e \in M_t \cap U_t(e) \geq U_t(e^o)\}$$

are independent since the UCB variables are only dependent on observations prior to time t. Thus, we can bound (A.4) as

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\sum_{e\in M^o} w_t(e) - \sum_{e\in M_t} w_t(e)\right)\right] \leq \sum_{t=|E|+1}^{n}\Bigg(\ldots \\ &\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^s(e^o)} (\overline{w}(e^o) - \overline{w}(e))P(e \in M_t \cap U_t(e) \geq U_t(e^o)) \\ &+ \sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} \overline{w}(e^o)P(e \in M_t \cap U_t(e) \geq U_t(e^o))\Bigg). \quad \text{(A.5)}\end{aligned}$$

For edge pair $e^o \in M^o$ and $e \in \mathcal{N}^s(e^o)$, $\overline{w}(e^o) - \overline{w}(e) < 0$. Therefore, we can upper bound the first term on the right-hand side of (A.5) with 0. Doing so, we obtain from (A.5)

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\sum_{e\in M^o} w_t(e) - \sum_{e\in M_t} w_t(e)\right)\right] \\ &\leq \sum_{t=|E|+1}^{n}\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} \overline{w}(e^o)P(e \in M_t \cap U_t(e) \geq U_t(e^o)) \\ &= \sum_{e^o\in M^o}\overline{w}(e^o)\sum_{e\in\mathcal{N}^i(e^o)}\sum_{t=|E|+1}^{n} P(e \in M_t \cap U_t(e) \geq U_t(e^o)) \quad \text{(A.6)}\end{aligned}$$

where the equality follows by changing the order of summation.

Using the proof of [11, Theorem 1], we have for $e^o \in M^o$ and $e \in \mathcal{N}^i(e^o)$

$$\sum_{t=|E|+1}^{n} P(e \in M_t \cap U_t(e) \geq U_t(e^o)) \leq \frac{8\log n}{\Delta^2_{e,e^o}} + \frac{\pi^2}{3}.$$

Plugging this into (A.6) gives

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\sum_{e\in M^o} w_t(e) - \sum_{e\in M_t} w_t(e)\right)\right] \\ &\leq \sum_{e^o\in M^o}\overline{w}(e^o)\sum_{e\in\mathcal{N}^i(e^o)}\left(\frac{8\log n}{\Delta^2_{e,e^o}} + \frac{\pi^2}{3}\right) \\ &= \sum_{e\in E}\left(\sum_{e^o\in M^o\cap\mathcal{N}^s(e)}\overline{w}(e^o)\left(\frac{8\log n}{\Delta^2_{e,e^o}} + \frac{\pi^2}{3}\right)\right).\end{aligned}$$

where the equality follows from changing the order of summation. Bounding the regret over the first $|E|$ time steps with $\frac{|N|}{2}|E|$ gives the result.

Appendix B. Proof of Theorem 2

Assume at time t, $M_t \neq M^o$. Then there must exist at least one edge $e \notin M^o$ that is in M_t. Consider the first such edge $e \notin M^o$ that enters M_t in the greedy construction of M_t by Algorithm 2. By Lemma 1, there is a neighbor of e that is in M^o and $\mathcal{N}^s(e)$. Let e^o denote this neighbor. Since e was added to M_t instead of e^o, $U_t(e) \geq U_t(e^o)$. This implies

$$\begin{aligned}&\{M_t \neq M^o\} \subseteq \\ &\{\exists e \in M_t \text{ and } e^o \in M^o \cap \mathcal{N}^s(e) : U_t(e) \geq U_t(e^o)\}.\end{aligned}$$

Taking a union bound we see that

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\mathbf{1}_{\{M_t\neq M^o\}}\right] \\ &\leq \sum_{t=|E|+1}^{n}\sum_{e^o\in M^o}\sum_{e\in\mathcal{N}^i(e^o)} P(e \in M_t \cap U_t(e) \geq U_t(e^o)).\end{aligned}$$

Comparing to equation (A.6), we see that following a similar analysis as above that

$$E\left[\sum_{t=|E|+1}^{n}\mathbf{1}_{\{M_t\neq M^o\}}\right]\leq\sum_{e\in E}\left(\sum_{e^o\in M^o\cap\mathcal{N}^s(e)}\left(\frac{8\log n}{\Delta_{e,e^o}^2}+\frac{\pi^2}{3}\right)\right)$$

which gives the result.

Appendix C. Proof of Theorem 3

Consider the graph G and the distributions on the edge weights $w_t(e)$ explained in Section 5 after the theorem. Given that the exterior edges are perfectly correlated, the problem facing the algorithm is to determine whether the weight of the first exterior edge is greater than the weight of the interior edge; since the greater of these two edges should be included in the GMM. Thus, each connected component can be viewed as a two-armed stochastic bandit problem with independent Bernoulli rewards.

From Theorem 2, for any given graph G and defined edge weight distributions, the expected number of times a matching not equal to M^o is chosen to be activated is $O(\log n)$. This implies that any edge not in M^o is activated in expectation $o(n^\alpha)$ times for all $\alpha>0$.

Denote the edges of the jth connected component as e^j_{int}, e^j_{ext1}, and e^j_{ext2}. Define $T_n^j\left(e^j_{int}\right)$ the number of times the interior edge of the jth connected component is pulled over n time slots. From the above, each connected component is equivalent to solving a two-armed bandit problem with Bernoulli rewards, and we are using a strategy that selects the inferior arm $o(n^\alpha)$ times in expectation for all $\alpha>0$. Then from [12, Theorem 2.2] which bounds, asymptotically, the number of expected times the inferior edge is pulled in the two-armed bandit by any strategy meeting the $o(n^\alpha)$ constraint, we see that

$$\liminf_{n\to\infty}\frac{E\left[T_n^j\left(e^j_{int}\right)\right]}{\log n}\geq\frac{1}{kl(0.5-\Delta,0.5)}\tag{C.1}$$

where $kl(p,q)$ is the Kullback–Leibler divergence between two Bernoulli random variables with parameters p and q. Therefore, the regret across all connected components can be bounded as

$$\begin{aligned}\liminf_{n\to\infty}&\frac{R^o(n)}{\log n}\\&=\liminf_{n\to\infty}\sum_{j=1}^{\frac{|E|}{3}}\left(2\overline{w}(e^j_{ext1})-\overline{w}(e^j_{int})\right)\frac{E\left[T_n^j\left(e^j_{int}\right)\right]}{\log n}\\&\geq\frac{|E|}{3}(0.5+\Delta)\frac{1}{kl(0.5-\Delta,0.5)}.\end{aligned}\tag{C.2}$$

where the inequality follows from (C.1) and plugging in the expected edge weights.

Now, by a well known simple inequality

$$kl(p,q)\leq\frac{(p-q)^2}{q(1-q)}.$$

Using this in (C.2) we see that

$$\liminf_{n\to\infty}\frac{R^o(n)}{\log n}\geq\frac{|E|}{3}(0.5+\Delta)\frac{1}{4\Delta^2}\geq\frac{|E|}{24}\frac{1}{\Delta^2}$$

where the second inequality follows from dropping the Δ term from the numerator. Since Δ is by the definition of the graph equivalent to $\Delta_{\min}$, the proof is established.

Appendix D. Proof of Lemma 3

Consider any sample path for $w_t(e)$ and the corresponding values $U_t(e)$ and chosen matchings M_t. Then for time $t>|E|$,

$$\begin{aligned}&\frac{1}{2}\sum_{e^*\in M^*}w_t(e^*)-\sum_{e\in M_t}w_t(e)\\&\leq\frac{1}{2}\left(\sum_{e^*\in M^*-M_t}w_t(e^*)-2\sum_{e\in M_t-M^*}w_t(e)\right)\end{aligned}\tag{D.1}$$

where the inequality follows from simple algebra.

Now, by an argument similar to Lemma 2, if $e^*\in M^*-M_t$ then it must be adjacent to an edge $e\in M_t$ such that $U_t(e)\geq U_t(e^*)$. Furthermore, we note that each edge $e\in M_t-M^*$ can be adjacent to at most two edges in M^* (see Fig. 4 for example). Thus, denoting the indicator function as $\mathbf{1}_{\{.\}}$, we can bound (D.1) as

$$\begin{aligned}&\frac{1}{2}\sum_{e^*\in M^*}w_t(e^*)-\sum_{e\in M_t}w_t(e)\leq\\&\frac{1}{2}\sum_{e^*\in M^*}\sum_{e\in\mathcal{N}(e^*)}(w_t(e^*)-w_t(e))\mathbf{1}_{\{U_t(e)\geq U_t(e^*)\cap e\in M_t\}}\end{aligned}\tag{D.2}$$

Noting that the event $\{U_t(e)\geq U_t(e^*)\cap e\in M_t\}$ is independent of the values of $w_t(e)$ and $w_t(e^*)$ we see that

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\frac{1}{2}\sum_{e^*\in M^*}w_t(e^*)-\sum_{e\in M_t}w_t(e)\right)\right]\\&\leq\frac{1}{2}\sum_{t=|E|+1}^{n}\sum_{e^*\in M^*}\sum_{e\in\mathcal{N}(e^*)}\Delta_{e,e^*}P(U_t(e)\geq U_t(e^*)\cap e\in M_t)\\&\leq\frac{1}{2}\sum_{e^*\in M^*}\sum_{e\in\mathcal{N}^i(e^*)}\Delta_{e,e^*}\sum_{t=|E|+1}^{n}P(U_t(e)\geq U_t(e^*)\cap e\in M_t)\end{aligned}\tag{D.3}$$

where the second inequality follows from changing the order of summation and noting that for all pairs $e^*\in M^*$ and $e\in\mathcal{N}(e^*)-\mathcal{N}^i(e^*)$, $\Delta_{e,e^*}\leq 0$. Now, using the proof of [11, Theorem 1], for $e^*\in M^*$ and $e\in\mathcal{N}^i(e^*)$,

$$\sum_{t=|E|+1}^{n}P(e\in M_t\cap U_t(e)\geq U_t(e^*))\leq\frac{8\log n}{\Delta_{e,e^*}^2}+\frac{\pi^2}{3}.$$

Plugging into (D.3), we obtain

$$\begin{aligned}&E\left[\sum_{t=|E|+1}^{n}\left(\frac{1}{2}\sum_{e^*\in M^*}\overline{w}(e^*)-\sum_{e\in M_t}\overline{w}(e)\right)\right]\\&\leq\frac{1}{2}\sum_{e\in E}\left(\sum_{e^*\in M^*\cap\mathcal{N}^s(e)}\left(\frac{8\log n}{\Delta_{e,e^*}}+\frac{\pi^2}{3}\Delta_{e,e^*}\right)\right)\end{aligned}$$

where on the right-hand side we have changed the order of summation. Bounding the total regret over the first $|E|$ time steps with $\frac{1}{2}|E||M^*|$ gives the result.

Appendix E. Proof of Corollary 1

We start with Eq. (D.3). We will now apply a well known trick in the bandit literature (cf., [12, Section 2.4.3]) by partitioning the pairs $e^*\in M^*$ and $e\in\mathcal{N}^i(e^*)$ into two disjoint sets using a threshold Δ_{th}. Our two sets are those pairs that have a gap $\Delta_{e,e^*}<\Delta_{th}$ and those that have a gap $\Delta_{e,e^*}\geq\Delta_{th}$.

For the set where $\Delta_{e,e^*} < \Delta_{th}$ we begin by noting that each $e^* \in M^*$ may be adjacent to at most two edges in M_t. Therefore,

$$\frac{1}{2}\sum_{e^*\in M^*}\sum_{\substack{e\in\mathcal{N}^1(e^*):\\ \Delta_{e,e^*}<\Delta_{th}}}\Delta_{e,e^*}\sum_{t=|E|+1}^{n}P(U_t(e)\geq U_t(e^*)\cap e\in M_t)$$

$$< \frac{1}{2}\sum_{e^*\in M^*}\Delta_{th}E\left[\sum_{t=|E|+1}^{n}\sum_{\substack{e\in\mathcal{N}^1(e^*):\\ \Delta_{e,e^*}<\Delta_{th}}}\mathbf{1}_{\{e\in M_t\}}\right]$$

$$\leq \frac{1}{2}|M^*|\Delta_{th}2n \tag{E.1}$$

where the first inequality follows from bounding all Δ_{e,e^*} by Δ_{th} and using $\mathbf{1}_{\{U_t(e)\geq U_t(e^*)\cap e\in M_t\}} \leq \mathbf{1}_{\{e\in M_t\}}$. The second inequality follows from noting that over any sample path at most two neighbors of $e^* \in M^*$ may be in M_t at any given time.

Likewise, for the set $\Delta_{e,e^*} \geq \Delta_{th}$ we can bound, in an analysis similar to the proof of Lemma 3

$$\frac{1}{2}\sum_{e^*\in M^*}\sum_{\substack{e\in\mathcal{N}^1(e^*):\\ \Delta_{e,e^*}\geq\Delta_{th}}}\Delta_{e,e^*}\sum_{t=|E|+1}^{n}P(U_t(e)\geq U_t(e^*)\cap e\in M_t)$$

$$\leq \frac{1}{2}\sum_{e\in E}2\left(\frac{8\log n}{\Delta_{th}}+\frac{\pi^2}{3}\right) \tag{E.2}$$

Combining (E.1) and (E.2) into (D.3) and using $\frac{1}{2}|E||M^*|$ to bound the regret of the first $|E|$ steps we obtain

$$R^a(n) \leq |E|\left(\frac{8\log n}{\Delta_{th}}+\frac{\pi^2}{3}\right)+(|M^*|\Delta_{th})n+\frac{1}{2}|E||M^*|.$$

Now, choosing the threshold

$$\Delta_{th} = \sqrt{\frac{8|E|\log n}{|M^*|n}}$$

gives the result.

References

[1] T. Stahlbuhk, B. Shrader, E. Modiano, Learning algorithms for scheduling in wireless networks with unknown channel statistics, in: Proceedings of the ACM MobiHoc (2018).

[2] L. Tassiulas, A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, IEEE Trans. on Autom. Control 37 (12) (1992) 1936–1948.

[3] M.J. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems, Morgan & Claypool, 2010.

[4] X. Lin, N. Shroff, The impact of imperfect scheduling on cross-layer rate control in wireless networks, in: Proceedings of the IEEE INFOCOM, 2005, pp. 1804–1814.

[5] L. Chen, et al., Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks, in: Proceedings of the IEEE INFOCOM, 2006.

[6] G. Sharma, C. Joo, N.B. Shroff, Distributed scheduling schemes for throughput guarantees in wireless networks, in: Proceedings of the Allerton Conference, 2006.

[7] C. Joo, X. Lin, N.B. Shroff, Greedy maximal matching: performance limits for arbitrary network graphs under the node-exclusive interference model, IEEE Trans. on Autom. Control 54 (12) (2009) 2734–2744.

[8] B. Birand, et al., Analyzing the performance of greedy maximal scheduling via local pooling and graph theory, IEEE/ACM Trans. on Netw. 20 (1) (2012) 163–176.

[9] T.L. Lai, H. Robbins, Asymptotically efficient adaptive allocation rules, Adv. Appl. Math. 6 (1985) 4–22.

[10] R. Agrawal, Sample mean based index policies with o(log n) regret for the multi-armed bandit problem, Adv. Appl. Probab. 27 (4) (1995) 1054–1078.

[11] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2–3) (2002) 235–256.

[12] S. Bubeck, N. Cesa-Bianchi, in: Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit problems, 5, Foundations and Trends in Machine Learning, 2012, pp. 1–122.

[13] G. Celik, L.B. Le, E. Modiano, Dynamic server allocation over time varying channels with switchover delay, IEEE Trans. on Inf. Theory 58 (9) (2012) 5856–5877.

[14] G. Celik, E. Modiano, Scheduling in networks with time-varying channels and reconfiguration delay, IEEE/ACM Trans. Netw. 23 (1) (2015) 99–113.

[15] K. Jagannathan, et al., A state action frequency approach to throughput maximization over uncertain wireless channels, Internet Math. 9 (2–3) (2013) 136–160.

[16] Y. Gai, B. Krishnamachari, R. Jain, Learning Multiuser Channel Allocations in Cognitive Radio Networks: A Combinatorial Multi-Armed Bandit formulation, IEEE New Frontiers in Dynamic Spectrum, 2010.

[17] K. Liu, Q. Zhao, Distributed learning in multi-armed bandit with multiple players, IEEE Trans. Signal Process. 58 (11) (2010) 5667–5681.

[18] A. Anandkumar, et al., Distributed algorithms for learning and cognitive medium access with logarithmic regret, IEEE J. Sel. Areas Commun. 29 (4) (2011) 731–745.

[19] D. Kalathil, N. Nayyar, R. Jain, Decentralized learning for multiplayer multiarmed bandits, IEEE Trans. Inf. Theory 60 (4) (2014) 2331–2345.

[20] N. Nayyar, D. Kalathil, R. Jain, On regret-optimal learning in decentralized multiplayer multiarmed bandits, IEEE Trans. Control of Netw. Syst. 5 (1) (2016) 597–606.

[21] O. Avner, S. Mannor, Multi-user lax communications: a multi-armed bandit approach., in: Proceedings of the IEEE INFOCOM, 2016.

[22] M. Lelarge, A. Proutiere, M.S. Talebi, Spectrum bandit optimization, in: Proceedings of the Information Theory Workshop, 2013.

[23] Y. Zhou, et al., Almost optimal channel access in multi-hop networks with unknown channel variables, in: Proceedings of the IEEE Distributed Computing Systems, 2014, pp. 461–470.

[24] Y. Zhang, et al., Learning temporal-spatial spectrum reuse, IEEE Trans. Commun. 64 (7) (2016) 3092–3103.

[25] Y. Gai, B. Krishnamachari, R. Jain, Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observation, IEEE/ACM Trans. Netw. 20 (5) (2012) 1466–1478.

[26] W. Chen, Y. Wang, Y. Yuan, Combinatorial multi-armed bandit: general framework and applications, in: Proceedings of the ICML, 2013, pp. 151–159.

[27] B. Kveton, et al., Tight regret bounds for stochastic combinatorial semi-bandits, in: Proceedings of the AISTATS, 2015, pp. 535–543.

[28] R. Combes, et al., Combinatorial bandits revisited, in: Proceedings of the Neural Information Processing Systems, 2015, pp. 2116–2124.

[29] B. Kveton, et al., Matroid bandits: fast combinatorial optimization with learning, Proceedings of Uncertainty in Artificial Intelligence (UAI), 2014.

[30] M.S. Talebi, A. Proutiere, An optimal algorithm for stochastic matroid bandit optimization, in: Proceedings of the ACM AAMAS, 2016, pp. 548–556.

[31] T. Lin, J. Li, W. Chen, Stochastic online greedy learning with semi-bandit feedbacks, in: Proceedings of the Neural Information Processing Systems, 2015.

[32] P. Chaporkar, et al., Throughput and fairness guarantees through maximal scheduling in wireless networks, IEEE Trans. Inf. Theory 54 (2) (2008) 572–594.

[33] B. Hajek, G. Sasaki, Link scheduling in polynomial time, IEEE Trans. Inf. Theory 34 (5) (1988) 910–917.

[34] M. Kodialam, T. Nandagopal, Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem, in: Proceedings of the ACM MobiCom, 2003, pp. 42–54.

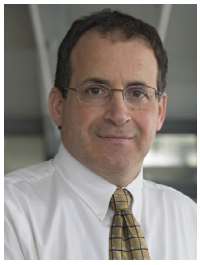
[35] E. Modiano, D. Shah, G. Zussman, Maximizing throughput in wireless networks via gossiping, in: Proceedings of the ACM Sigmetrics, 2006, pp. 27–38.

[36] R. Preis, Linear time 1/2–approximation algorithm for maximum weighted matching in general graphs, in: Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science, 1999, pp. 259–269.

[37] J.-H. Hoepman, Simple distributed weighted matchings, arXiv preprint cs/0410047 (2004).

Thomas Stahlbuhk received his B.S. and M.S. degrees in Electrical Engineering from the University of California San Diego in 2008 and 2009, respectively, and received his Ph.D. degree in Communications and Networks from the Massachusetts Institute of Technology (MIT) in 2018. He is currently a member of the Technical Staff at MIT Lincoln Laboratory, working in communication networks research. His research interests are in wireless networks, dynamic programming, and applied probability.

Brooke Shrader received the B.S. degree from Rice University, the M.S. degree from the Swedish Royal Institute of Technology (KTH), and the Ph.D. degree from the University of Maryland, College Park, all in electrical engineering. She is a Senior Member of Technical Staff with the Massachusetts Institute of Technology Lincoln Laboratory, where she has been since 2008. Her research interests lie in communication systems, wireless networks, and related disciplines, including information theory, control, and queueing models. She currently serves as Associate Editor for the IEEE/ACM Transactions on Networking.



Eytan Modiano received his B.S. degree in Electrical Engineering and Computer Science from the University of Connecticut at Storrs in 1986 and his M.S. and Ph.D. degrees, both in Electrical Engineering, from the University of Maryland, College Park, MD, in 1989 and 1992, respectively. He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow during 1992–1993. Between 1993 and 1999 he was with MIT Lincoln Laboratory. Since 1999 he has been on the faculty at MIT, where he is a Professor and Associate Department Head in the Department of Aeronautics and Astronautics, and Associate Director of the Laboratory for Information and Decision Systems (LIDS). His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks. He is the co-recipient of the MobiHoc 2016 best paper award, the WiOpt 2013 best paper award, and the Sigmetrics 2006 Best paper award. He is the Editor-in-Chief for IEEE/ACM Transactions on Networking, and served as Associate Editor for IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking. He was the Technical Program co-chair for IEEE WiOpt 2006, IEEE Infocom 2007, ACM MobiHoc 2007, and DRCN 2015. He is a Fellow of the IEEE and an Associate Fellow of the AIAA, and served on the IEEE Fellows committee.