Local Greedy Approximation for Scheduling in Multi-hop Wireless Networks

Changhee Joo, *Member, IEEE,* and Ness B. Shroff, *Fellow, IEEE*

Abstract—In recent years, there has been a significant amount of work done in developing low-complexity scheduling schemes to achieve high performance in multi-hop wireless networks. A centralized sub-optimal scheduling policy, called *Greedy Maximal Scheduling* (*GMS*) is a good candidate because its empirically observed performance is close to optimal in a variety of network settings. However, its distributed realization requires high complexity, which becomes a major obstacle for practical implementation. In this paper, we develop simple distributed greedy algorithms for scheduling in multi-hop wireless networks. We reduce the complexity by relaxing the global ordering requirement of *GMS*, up to near-zero. Simulation results show that the new algorithms approximate the performance of *GMS*, and outperform the state-of-the-art distributed scheduling policies.

Index Terms—Wireless scheduling, distributed system, greedy algorithm.

1 Introduction

In the past few years, there have been significant advances made in our understanding of the wireless scheduling problem [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. Scheduling is a process that determines which links transmit, at what times, and at what power levels. Throughput optimal scheduling is in general a non-linear, non-convex optimization problem mainly due to interference constraints between links, and thus requires high computational complexity. In addition, the nature of multi-hop wireless networks demands a distributed solution based on local information, which often causes additional complexity.

The scheduling problem is especially important because it has been shown that the scheduling component results in the highest complexity among various network functionalities (e.g., see recent studies on cross-layer optimization [3], [13], [14]). Although the optimal scheduling solution has been known for a long time [2], it requires a high order polynomial complexity even under the simplest 1-hop interference model[1] [15], and is in general NP-Hard [16]. Hence, it is difficult to implement the optimal solution. To this end, more practical scheduling solutions, i.e., simpler sub-optimal solutions, have been developed [4], [5], [6], [7], [8], [9], [10], [11], [12] in order to reduce the complexity and at the same time, with the aim of approximating the optimal performance.

One of the most well-known sub-optimal scheduling policy is the *Greedy Maximal Scheduling* (*GMS*) policy or *Longest Queue First* (*LQF*) policy. *GMS* schedules links in decreasing order of the queue length conforming to interference constraints. It has been known to achieve an efficiency ratio of $\frac{1}{2}$ under the 1-hop interference model [17], where the *efficiency ratio* is defined as the largest fraction of the optimal capacity region that the scheduling policy can support. *GMS* is an important scheduling policy because it has a good provable performance bound superior to many distributed scheduling policies and it empirically achieves the same performance as throughput-optimal scheduling in a variety of network settings [12]. For practical implementation in multi-hop wireless networks, *GMS* has been realized as a distributed algorithm [4], [5], [18]. However, these algorithms are quite complex to ensure the precise queue-length ordering of links.

Towards the goal of developing simple and efficient scheduling algorithms, many approaches have been taken in the literature. We classify them into four categories: (1) *Maximal Scheduling* policies [19], [20], [21]: these select a schedule that is *maximal* in the sense that it is not possible to add a link to the schedule without violating the interference constraints. *GMS* is an example of such policy. (2) *Pick and Compare Approach* [6], [7], [8], [9]: a scheduling policy in this class picks a schedule at random, and decides on the current or the previous schedule by comparing their performance. By choosing a better (or no worse schedule), it eventually results in optimal scheduling. (3) *Carrier-Sensing-Multiple-Access/Collision-Avoidance (CSMA/CA) based Approach* [22], [23], [24] finds the optimal scheduling without directly comparing the schedules, under the assumption that wireless nodes can detect transmissions of their interfering neighbors by using carrier sensing technology. *Q-CSMA* in [24] falls in this category. Under *Q-CSMA*, a link schedules itself in a probabilistic manner unless its transmission does not interfere with any on-

- *C. Joo is with the Dept. of EECE, Korea University of Technology and Education, Cheoan, Chungnam, Korea.*
  E-mail: cjoo@kut.ac.kr
- *N. B. Shroff is with the Depts. of ECE and CSE, the Ohio State University, Columbus, OH, USA.*

A preliminary version of this work was presented at ACM MobiHoc'08 [1].

1. Under the 1-hop interference model, any two links sharing a node cannot transmit simultaneously. It is also known as the *node-exclusive* interference model or the *primary* interference model.

going transmissions. The convergence of stationary distribution of link activities to an optimal solution has been shown. While policies in class (2) and class (3) achieve optimal throughput performance, they suffer from high complexity (i.e., high-order message exchanges), long convergence time, low delay performance, and/or requirement for carrier sensing functionality. (4) Scheduling policies in the class of *Constant-time Random Access Approach* provide an explicit tradeoff between complexity and performance [10], [11], [12]. They need local message exchanges to resolve contention, and the performance bound is expressed in terms of the complexity (i.e., overhead for message exchanges); the higher complexity they have, the better performance they guarantee. While their empirical performance approaches the optimal, it only happens at the cost of high complexity. Queue-length based random access scheduling (*QL-RAS*) proposed in [12] belongs to this class. *QL-RAS* schedules a link with a probability that is proportional to the ratio of its queue length to the sum of queue lengths over its neighboring links.

In this paper, we propose local greedy algorithms that achieve good throughput performance with lower complexity and delay. The proposed algorithms are not only amenable to distributed implementations with local message exchanges but also approximate the performance of *GMS* with a much lower complexity. We successfully reduce the complexity based on the observation that links with the largest queue length within their interference range are crucial in characterizing the capacity region of *GMS*.

The rest of the paper is organized as follows. The system model is described in Section 2. In Section 3, the local greedy scheduling schemes are described and their performance analyzed. The idea is extended in Section 4 to the development of a simpler scheme with near-zero complexity. The complexity of the proposed local greedy algorithms is analyzed in Section 5. Performance evaluation comparing with *GMS* and other random access scheduling policies is shown in Section 6. Finally, we conclude our paper in Section 7.

2 Network Model and Greedy Maximal Scheduling

We consider a network graph $G(V, E, I)$ having a set V of nodes, a set E of links, and an interference constraint matrix I. We assume that the network is a time-slotted system with global synchronization. Each time slot is divided into a contention period and a transmission period. A schedule is determined during the contention period, which consists of a number of mini-slots, and served during the transmission period. We model the interference constraints between the links by the K-hop interference model, under which any two links within a K-hop distance cannot be scheduled in the same time slot. Note that the different interference models approximate different network systems; e.g., the 1-hop interference model is appropriate for Bluetooth or FH-CDMA networks [25], [26] and the 2-hop interference model is often used for IEEE 802.11 Distributed Coordination Function (DCF) wireless networks [19], [20]. Let N_l denote the set of links interfering with link l (including itself). We say that link j is a (K-hop) *neighbor* of link l if $j \in N_l$ and $j \neq l$.

Let λ_l denote the offered load at link l, and let $\vec{\lambda} := [\lambda_1, \lambda_2, \ldots, \lambda_{|E|}]$. The capacity region of a scheduling policy can be defined as the set of offered loads under which the scheduling policy can stabilize the network[2]. Let $\Omega_{\mathcal{P}}$ denote the capacity region of scheduling policy $\mathcal{P}$. We define the optimal capacity region Ω^* as the union of the capacity regions of all feasible policies, i.e.,

$$\Omega^* := \cup_{\mathcal{P}} \ \Omega_{\mathcal{P}}.$$

The throughput-optimal scheduling policy can be defined as a scheduling policy achieving Ω^* or as a combination of policies $\mathcal{P}$ resulting in the throughput region Ω^*.

Assume that each link l has a fixed link capacity r_l. It has been known that the throughput-optimal scheduling policy can be realized by maximizing the queue weighted rate sum at each time slot t [2], i.e.,

$$\vec{S}^*(t) \leftarrow \underset{\vec{S} \in \mathcal{F}}{\operatorname{argmax}} \sum_{l \in E} Q_l(t) \cdot r_l \mathbb{1}_{\{l \in \vec{S}\}}, \tag{1}$$

where $\vec{S}^*(t)$ is the vector of rate assignment of the throughput-optimal scheduling policy, $Q_l(t)$ is the queue length of link l, $\vec{S}$ is a *feasible schedule*, which is defined as the set of links that can make simultaneous transmissions without violating the interference constraints, $\mathcal{F}$ is the set of all feasible schedules, and $\mathbb{1}_{\{l \in \vec{S}\}}$ is an indicator function that equals to 1 if $l \in \vec{S}$ and 0 if $l \notin \vec{S}$. As mentioned earlier, it has been shown that the optimal solution has enormous computational complexity; $O(|V|^3)$ under the 1-hop interference model [15], where $|V|$ is the number of nodes in the network, and NP-Hard under the K-hop interference model for $K \geq 2$ [16].

A simpler sub-optimal scheduling policy called *GMS* was originally proposed in [27]. *GMS* makes a scheduling decision by choosing the link with the longest queue first. Specifically, at time slot t, it determines its schedule as follows. Starting with an empty set $\vec{S}(t)$, it first includes in $\vec{S}(t)$ the link with the largest queue length, and removes from the network graph this link and all its K-hop neighbors. Then from the new network graph, it finds the link with the largest queue length and adds it to $\vec{S}(t)$. Again it removes the added link and all its K-hop neighbors from the network graph. The procedure repeats until there is no remaining link. The resulting schedule $\vec{S}(t)$ is the final schedule. It is well-known that under the 1-hop interference model, *GMS* achieves at least $\frac{1}{2}$ of the optimal performance [17], and under the K-hop interference model, it guarantees

2. Under a given offered load, a scheduling policy stabilizes the network if it keeps all average queue lengths in the network finite.

only a much smaller fraction, i.e., $\frac{1}{(\Delta_l)^{K-1}}$, where Δ_l is the maximum link-degree. Recent studies have shown that the performance limits of *GMS* depend on some topological properties [28], [29], [30], [31] as well as the underlying interference model. For example, under the K-hop interference model, *GMS* achieves the full capacity region in tree network topologies [30] and at least $\frac{1}{6}$ in geometric unit-disk network graphs [31].

While *GMS* exhibits good provable performance and achieves empirical performance indistinguishable from the optimal throughput in a variety of settings [12], it still requires centralized information to achieve global link ordering. However, practical demands in multi-hop wireless networks require distributed implementation. In this direction, some distributed versions of *GMS* have been proposed [4], [5], [18]. The state-of-the-art distributed *GMS* [5] obtains its schedule by regulating each node to do a series of operations based on local information. Specifically, it has the following procedure. Each node i requests a matching to node j, where j is selected such that link (i, j) has the largest weight among eligible (i.e., not matched yet) neighbors of node i. It waits for a response from node j; if node j accepts the request, link (i, j) gets matched and both nodes i, j broadcast the new matching information to their neighbors and refuse other requests; if node j refuses the request, which means that node j gets matched with some other node and no longer eligible, node i finds another neighbor k, where link (i, k) has the largest weight among remaining eligible neighbors, and sends new matching request to node k. This procedure repeats until node i gets matched or there is no eligible neighbor. The resulting set of matched links is the schedule of *GMS*. Note that this decentralization incurs additional complexity because a link cannot make a scheduling decision until all its neighboring links with a larger weight make a decision. Although the process can be accelerated using parallel executions, it still requires $O(|V|)$ complexity in the worst case [18].

In the following section we propose a distributed scheduling policy that approximates the performance of *GMS* and has a significantly lower computational complexity than distributed *GMS*. The main idea is to schedule links with the largest queue lengths among their local neighbors.

3 Local Greedy Scheduling

We start with an algorithmic description with an example, and show how it approximates the performance of *GMS*. We also extend the basic algorithm for improved performance.

3.1 Basic algorithm

We consider a network graph $G(V, E, I)$ with a set V of nodes, a set E of links, and an interference constraint matrix I. Let $\vec{S}$ denote a feasible schedule and let $\mathcal{F}$ denote the set of all feasible schedules on $G(V, E, I)$. Let $\mathcal{D}$ denote a subset of $\mathcal{F}$, i.e., $\mathcal{D} = \{\vec{S}_1, \vec{S}_2, \ldots, \vec{S}_{|\mathcal{D}|}\}$, satisfying the following two constraints:

$$\begin{aligned} &\bullet\ \vec{S}_i \cap \vec{S}_j = \emptyset \text{ for all } i \neq j, \\ &\bullet\ \cup_{i=1}^{|\mathcal{D}|} \vec{S}_i = E. \end{aligned} \tag{2}$$

Each link in E belongs to one and only one schedule in $\mathcal{D}$, and any two interfering links cannot belong to the same schedule. Let $i(l)$ denote the index of the schedule in $\mathcal{D}$ that contains link l, i.e., $l \in \vec{S}_{i(l)}$.

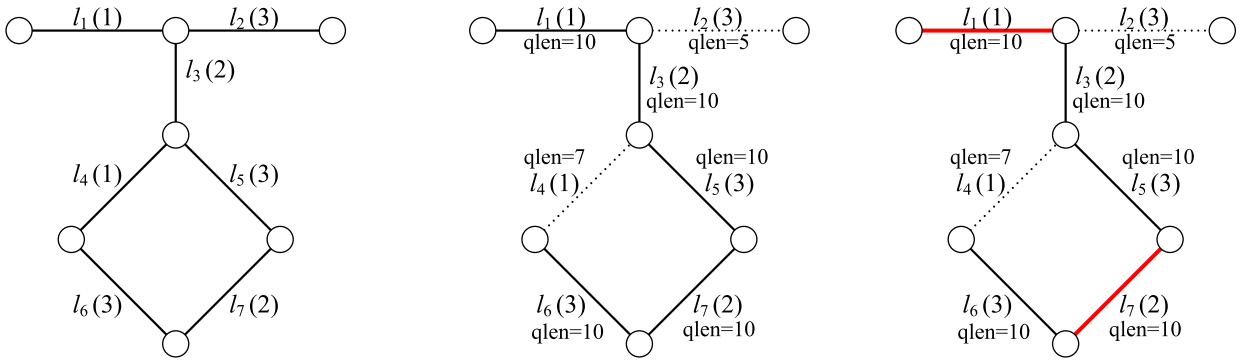
We assume that the set $\mathcal{D}$ is predetermined and that each link knows the index of the schedule that it belongs to. A distributed algorithm that assigns the indexes satisfying (2) will be provided in Section 5.2. We also assume that each link has backlog information of its interfering neighbors. Later we will introduce an efficient algorithm to find $\mathcal{D}$ in Section 5.2, and also relax the latter assumption in Section 5.3 when we discuss complexity issues.

We first design a scheduling policy, called *Local Greedy Scheduling* (*LGS*), which schedules only links with the locally longest queue. Other links that have a smaller queue length than their neighbors are simply not scheduled under *LGS*. Clearly, this restriction will reduce complexity at the cost of some performance. However, we show later that the cost is not significant and the new policy provides a good approximation to the performance of *GMS*.

Note that the worst-case complexity of *LGS* still remains high, which occurs when all the links have the same queue length. In order to determine which link to be scheduled among links with the same queue length, we make use of the predetermined set of feasible schedules $\mathcal{D}$. Specifically, if two interfering links l_1 and l_2 have the same largest queue length, the link with the smaller index will be scheduled, i.e., without loss of generality, if $i(l_1) < i(l_2)$, link l_1 is allowed to transmit. The detailed algorithm is provided in Algorithm 1, where $f(x)$ on line 5 denotes some function such that $f(x) \to 0$ slowly as $x \to \infty$. The function is for the purpose of analysis only and will be replaced by 0 in practical implementation. In this section, we also choose it to be 0 to simplify our description. In a network-wide point of view, *LGS* has a two-tier decision procedure:

1) At each time slot, links with the locally longest queue have the right to transmit.
2) If more than two interfering links have the same largest queue length, they are added to the schedule in an increasing order of index unless they interfere with some links that were added earlier.

To make the decision procedure clear, we provide a scheduling example for *LGS*. We consider the network graph shown in Fig. 1(a) under the 1-hop interference model. We assume that all links have identical link capacity. The index of each link is pre-assigned and presented in parentheses. Note that the index assignment satisfies the interference constraints, i.e., two links sharing a node have a different index.



(a) Network topology

(b) Links with longest queue at a time slot

(c) Resultant schedule $\{l_1, l_7\}$

Fig. 1. An example schedule of *LGS* under the 1-hop interference model. Fig. (a) shows the network topology where all links have the same link capacity. The index of each link is pre-assigned and presented in the parenthesis. Since the index assignment satisfies the interference constraints, any two links sharing a node have a different index. Fig. (b) illustrates the queue length of links at a time slot t. The set of links with the longest queue, i.e., $L(t) = \{l_1, l_3, l_5, l_6, l_7\}$, are marked with solid lines. Fig. (c) presents the final schedule $\vec{S}(t) = \{l_1, l_7\}$. It is deterministically obtained by choosing first the link of the smallest index among links in $L(t)$.

Algorithm 1 Local Greedy Scheduling (*LGS*).

At each time slot, each link l does

1: $sched \leftarrow 0$
2: **if** $Q_l(t) > 0$ **then**
3: $\quad eligible \leftarrow 0$
4: **end if**
5: **if** $\frac{Q_l}{r_l} \geq \left(\max_{k \in N_l} \frac{Q_k}{r_k}\right)^{1-f(\max_{k \in N_l} Q_k)}$ **then**
6: $\quad eligible \leftarrow 1$
7: **end if**
8: **for** each contention mini-slot m $(1 \leq m \leq |\mathcal{D}|)$, **do**
9: $\quad$ **if** $eligible = 1$ and $i(l) = m$ **then**
10: $\quad\quad sched \leftarrow 1$
11: $\quad\quad$ Broadcast a control message to its K-hop neighbors
12: $\quad\quad eligible \leftarrow (-1)$
13: $\quad$ **else**
14: $\quad\quad$ **if** Receive a control message **then**
15: $\quad\quad\quad eligible \leftarrow (-1)$
16: $\quad\quad$ **end if**
17: $\quad$ **end if**
18: **end for**
19: **for** transmission period **do**
20: $\quad$ **if** $sched = 1$ **then**
21: $\quad\quad$ Transmit data packets
22: $\quad$ **end if**
23: **end for**

Suppose that at a time slot t, each link has queue length as shown in Fig. 1(b). Let $L(t)$ denote the set of links with the largest queue length, i.e., $L(t) = \{l_1, l_3, l_5, l_6, l_7\}$, are marked as solid lines in Fig. 1(b). Since links with the locally longest queue are eligible (lines 5-7 in Algorithm 1 above), all links in $L(t)$ are eligible and participate in the contention. All other links $\{l_2, l_4\}$ remain inactive. Each eligible link is given a chance to attempt at a mini-slot following the index order, i.e., link l_1 attempts in the 1-st mini-slot (lines 8-9). Since the index assignment satisfies the interference constraints (from $\mathcal{D} \subset \mathcal{F}$), the links attempting transmissions in the same mini-slot do not interfere with each other, and hence, there is no collision. Once link l_1 makes an attempt, it is included in the schedule (line 10) and broadcasts a control message (line 11). Links l_2, l_3 that receive the message disable themselves, and they are no longer eligible (lines 14-16). The procedure repeats with the reduced set of eligible links. At the 2-nd mini-slot, link l_7 attempts and it is added to the schedule. The broadcast message from link l_7 makes links l_5, l_6 ineligible. At the 3-rd mini-slot, no one can attempt and the scheduling ends. The resulting schedule is $\{l_1, l_7\}$ as shown in Fig. 1(c). Scheduled links transmit data packets in the transmission period (lines 19-23).

The key idea is to schedule links with the *locally* longest queue and to resolve the contention between them *without collision* using a predetermined link ordering. Achieving the collision-free resolution is important since it ensures that the resultant schedule is *maximal on the set of links with the locally longest queue*. We show in the next section that if an appropriate function $f(\cdot)$ can be chosen, this property actually allows *LGS* to achieve a capacity region which is equivalent to *GMS*. Other links with a smaller queue length than their neighbors do not play a significant role in characterizing the capacity region.

3.2 Throughput performance

We take an analytical view of *LGS* in terms of achievable throughput performance as compared to *GMS*. We characterize throughput performance of a scheduling policy by its *efficiency ratio*, and show that the efficiency ratio of *LGS* is no smaller than that of *GMS* under a certain

assumption. To this end, we make use of a recently developed topological characterization of the network capacity called *local-pooling factor* [31].

We begin with the following definitions:

Definition 1: A scheduling policy $\mathcal{P}$ is said to achieve an *efficiency ratio* of $\gamma^*_{\mathcal{P}}$ if it can support $\gamma^*_{\mathcal{P}}\vec{\lambda}$ for all $\vec{\lambda} \in \Omega^*$, i.e.,

$$\gamma^*_{\mathcal{P}} := \sup\{\gamma \mid \gamma\vec{\lambda} \in \Omega_{\mathcal{P}} \text{ for all } \vec{\lambda} \in \Omega^*\},$$

where $\Omega_{\mathcal{P}}$ is the capacity region of the policy and Ω^* is the optimal capacity region.

Let $\mathcal{M}_L$ denote the set of all maximal schedules on a subset L of links, where a maximal schedule is a feasible schedule such that no link can be added to the schedule without violating the interference constraints. Let $Co(\mathcal{M}_L)$ denote the set of convex combinations of maximal schedules, i.e.,

$$Co(\mathcal{M}_L) := \{\vec{\phi} \mid \vec{\phi} = \textstyle\sum_i w_i\vec{m}_i, \\ \text{where } w_i \geq 0,\ \textstyle\sum_i w_i = 1, \text{ and } \vec{m}_i \in \mathcal{M}_L\}.$$

Let $\vec{x} \succeq \vec{y}$ denote that $\vec{x}$ is component-wise greater than or equal to $\vec{y}$. It is known that for each $\vec{\lambda} \in \Omega^*$, there exist a subset L and a vector $\vec{\mu}$ such that $\vec{\mu} \in Co(\mathcal{M}_L)$ and $\vec{\mu} \succeq \vec{\lambda}$.

Definition 2: A set of links L satisfies *σ-local pooling*, if $\sigma\vec{\mu} \not\succeq \vec{\nu}$ for all $\vec{\mu}, \vec{\nu} \in Co(\mathcal{M}_L)$. In other words, for each $\vec{\mu}, \vec{\nu} \in Co(\mathcal{M}_L)$, there must exist some $k \in L$ such that $\sigma\mu_k < \nu_k$.

Definition 3: The *local-pooling factor* of a graph $G(V, E, I)$ is the supremum of all σ such that every subset $L \subset E$ satisfies σ-local pooling. In other words,

$$\sigma^* := \sup\{\sigma \mid \sigma\vec{\mu} \not\succeq \vec{\nu} \text{ for all } L \text{ and all } \vec{\mu}, \vec{\nu} \in Co(\mathcal{M}_L)\} \\ = \inf\{\sigma \mid \sigma\vec{\mu} \succeq \vec{\nu} \text{ for some } L \text{ and } \vec{\mu}, \vec{\nu} \in Co(\mathcal{M}_L)\}.$$

It has been shown that this topological notion characterizes the performance limits of *GMS* as follows.

Proposition 1 ([31], [32]): Given a network graph $G(V, E, I)$, the efficiency ratio γ^*_{GMS} of *GMS* is equivalent to the local-pooling factor σ^* of the graph, i.e., $\gamma^*_{GMS} = \sigma^*$.

Let $q_l(t)$ denote the fluid limit of $Q_l(t)$, which can be defined as

$$q_l(t) = \lim_{n\to\infty} \frac{Q_l(nt)}{n}.$$

The fluid limit exists and is differentiable almost everywhere under any scheduling policy if the arrival process satisfies the strong law of large number [33]. We further define the following subsets of links: let $\mathcal{L}$ denote the set of links with the *locally* longest queue in fluid limits, and let $\mathcal{G}$ denote the set of links with the *globally* longest queue, respectively, i.e.,

$$\mathcal{L} := \left\{l \mid l = \text{argmax}_{k\in N_l} \tfrac{q_k}{r_k}\right\},$$
$$\mathcal{G} := \left\{l \mid l = \text{argmax}_{k\in E} \tfrac{q_k}{r_k}\right\}.$$

Note that $\mathcal{G} \subseteq \mathcal{L}$. The following Lemma is a by-product of Proposition 1, and of importance to characterize the throughput performance of *LGS*.

Lemma 1: Given a network graph $G(V, E, I)$ that has the local-pooling factor σ^*, policy $\mathcal{P}$ has an efficiency ratio $\gamma^*_{\mathcal{P}}$ no smaller than σ^* if its schedule is maximal on the set of links with the globally longest queue, i.e., on $\mathcal{G}$.

Note that even if a schedule is maximal on $\mathcal{G}$, it does not mean that the schedule is maximal on E, and thus it is different from the schedule of *GMS*. The proof of the lemma can be obtained by following the line of analysis for Proposition 1. We refer interested readers to [32]. The result of Lemma 1 leads to the following proposition.

Proposition 2: If $f(\cdot)$ is chosen appropriately in Algorithm 1 such that all the links in $\mathcal{L}$ are either scheduled or interfered by some other links in $\mathcal{L}$, then *LGS* achieves an efficiency ratio γ^*_{LGS} that is no smaller than γ^*_{GMS}.

Proof: We assume a network graph $G(V, E, I)$ that has the local-pooling factor σ^*. Suppose that the function $f(\cdot)$ is chosen such that $n^{-f(n)} \to 1$ slowly as $n \to \infty$.

For a link $l \notin \mathcal{L}$, if $\max_{k\in N_l} q_k(t) > 0$, then there exist a small $\epsilon > 0$ and a large n_0 such that $\frac{Q_l(nt)}{r_l} \leq \max_{k\in N_l} \frac{Q_k(nt)}{r_k}(1-\epsilon)$ for all $n > n_0$, because $\frac{q_l(t)}{r_l} < \max_{k\in N_l} \frac{q_k(t)}{r_k}$ from the definition of $\mathcal{L}$. Note that from $\max_{k\in N_l} q_k(t) > 0$, we have $\max_{k\in N_l} Q_k(nt) = \Theta(n)$. Then we can obtain that $\frac{Q_l}{r_l} < (\max_{k\in N_l} \frac{Q_k}{r_k})^{1-f(\max_{k\in N_l} Q_k)}$ from the property of $f(\cdot)$, which results in that links $l \notin \mathcal{L}$ become ineligible from line 5 in Algorithm 1.

Similarly, if $l \in \mathcal{L}$ and $q_l(t) > 0$, from $\frac{q_l(t)}{r_l} = \max_{k\in N_l} \frac{q_k(t)}{r_k}$, there exists some function $g(n) \uparrow 1$ such that $\frac{Q_l(nt)}{r_l} = \max_{k\in N_l} \frac{Q_k(nt)}{r_k} \cdot g(n)$. Then we can obtain that $g(n) \geq (\max_{k\in N_l} \frac{Q_k(nt)}{r_k})^{-f(\max_{k\in N_l} Q_k(nt))}$ for large enough n from the slow convergence of $n^{-f(n)}$. Hence, there exists a large constant n_1 such that $\frac{Q_l}{r_l} \geq (\max_{k\in N_l} \frac{Q_k}{r_k})^{1-f(\max_{k\in N_l} Q_k)}$ for all $n \geq n_1$, which results in that links $l \in \mathcal{L}$ become eligible.

Therefore, after an initial period, Algorithm 1 will schedule only links in $\mathcal{L}$, and at each time slot, it ensures for each eligible link either to be scheduled or to be interfered by some scheduled links. Then, the resulting schedule is maximal on $\mathcal{L}$, i.e., there is no link on $\mathcal{L}$ that can be added to the schedule without violating the interference constraints.

Note that $\mathcal{G} \subseteq \mathcal{L}$ and that no link in $\mathcal{G}$ interferes with links in $\mathcal{L}\backslash\mathcal{G}$. The latter can be easily proved by contradiction using the definitions of $\mathcal{L}$ and $\mathcal{G}$. Then, it is clear that for every maximal schedule $\vec{S} \in \mathcal{M}_{\mathcal{L}}$, the projection of the schedule to $\mathcal{G}$ is maximal on $\mathcal{G}$, i.e., $[\vec{S}]_{\mathcal{G}} \in \mathcal{M}_{\mathcal{G}}$, where $[\cdot]_{\mathcal{G}}$ denotes the projection operation on to $\mathcal{G}$. Hence, Lemma 1 holds for *LGS*. Combining it with Proposition 1, we obtain

$$\gamma^*_{LGS} \geq \sigma^* = \gamma^*_{GMS}. \tag{3}$$

□

Remark: The result of Proposition 2 depends on the finding of an appropriate function $f(\cdot)$. Unfortunately,

in practice, it is hard to find such a function, since it should approach 0 more slowly than the rate at which the queue length of links in $\mathcal{L}$ converge to their fluid limits. Hence, if we can obtain a lower bound on the rates at which the fluid limits converge, then *LGS* can achieve the performance bound (3). Similarly, the proof shown in [1] is incomplete since it implicitly assumes that such a function can be obtained. In our evaluation in Section 6, we set the function to be zero, with which *LGS* cannot guarantee the performance of Proposition 2. However, empirical results shown in Section 6 suggest that *LGS* is a good approximation of *GMS*.

Note that *LGS* intends to decrease the locally largest queue length and increases other queue lengths smaller than their local maximum, which results in the situation where all queues eventually remain near the global maximum of queue lengths. Hence, after some initial time, *LGS* will serve links with the same set of maximal schedules as *GMS*, which explains the reason why the performance of *LGS* is close to that of *GMS*.

3.3 Enhancement

Since *LGS* schedules only links with the locally longest queue, it is possible to enhance its performance by scheduling additional links that remain inactive as long as they can transmit without violating the interference constraints. For example, let us recall the scheduling example shown in Fig. 1. At the final schedule in Fig. 1(c), link l_4 can be scheduled without violating the interference constraints, but it remains inactive because its queue length is smaller than its neighbors. This can be improved by scheduling extra links with a lower priority. We embed the detailed algorithm shown in Algorithm 2 after the contention period and before the transmission period of Algorithm 1.

Algorithm 2 Addition for Enhancement of Local Greedy Scheduling (*LGS-E*).

Embed the following between lines 18 and 19 of Algorithm 1.

```
for each contention mini-slot m (|D|+1 ≤ m ≤ 2|D|), do
    if eligible = 0 and i(l) = m − |D| then
        sched ← 1
        Broadcast a control message to its K-hop neighbors
        eligible ← (−1)
    else
        if Receive a control message from its neighbor then
            eligible ← (−1)
        end if
    end if
end for
```

Algorithm 2 is almost the same as *LGS* except that it considers all links as eligible. It adds links to the schedule in increasing order of index unless they interfere with some links added earlier to the schedule. This enhanced version of *LGS*, or simply *LGS-E*, first schedules links with the locally longest queue and then schedules other eligible links. Hence, by given a priority to links in $\mathcal{L}$ with an appropriately chosen $f(\cdot)$, its resulting schedule is still maximal on $\mathcal{G}$ (and clearly it is also maximal on E). This implies that Lemma 1 and Proposition 2 also hold, and hence, *LGS-E* will approximate the performance of *GMS* as *LGS*.

4 Local Greedy Approximation with Two Contention Mini-slots

Although *LGS* and *LGS-E* approximate the performance of *GMS*, their time complexity depends on the underlying network topology and the interference model, since the size of $\mathcal{D}$ is related to the topological structure. (See Section 5.) We propose another local greedy algorithm, whose complexity does not depend on the network graph, and is close to zero. The solution will be attractive when the network graph has rich connectivity or when the complexity overhead costs a significant amount of the network resources.

Our algorithm is motivated by a recently developed scheduling policy called *Q-CSMA*. It has been shown in [24] that, if each link can detect transmissions of interfering neighbors, the distributed CSMA/CA scheduling algorithm (*Q-CSMA*) can achieve the optimal capacity region in a time-slotted network system. We modify our local greedy algorithm capturing important features of *Q-CSMA*, but without the carrier sensing functionality. The resultant algorithm requires only a couple of contention mini-slots.

We begin by describing the basic operations of *Q-CSMA*. At each time slot t, it first randomly chooses a feasible schedule $m(t)$, which is denoted by a decision schedule, using a randomized matching [20], [24]. Let $\mathcal{M}_0$ denote the set of all decision schedules. It should satisfy that

$$\begin{aligned} \cup_{\vec{m}\in\mathcal{M}_0} &= E, \\ \textit{Prob}\{m(t)=\vec{m}\} &> 0, \\ \sum_{\vec{m}\in\mathcal{M}_0} \textit{Prob}\{m(t)=\vec{m}\} &= 1. \end{aligned} \tag{4}$$

Let $N_l^\circ := N_l \backslash \{l\}$. Given $m(t)$, *Q-CSMA* decides its schedule $S(t)$ using $m(t)$ and $S(t-1)$ in a probabilistic manner:

1) For each link $l \notin m(t)$, link l is included in $S(t)$, if and only if $l \in S(t-1)$.
2) For each link $l \in m(t)$, if $N_l^\circ \cap S(t-1) \neq \emptyset$, link l is NOT included in $S(t)$.
3) For each link $l \in m(t)$, if $N_l^\circ \cap S(t-1) = \emptyset$, link l is included in $S(t)$ with probability p_l, and $l \notin S(t)$ with probability $1-p_l$.

Link l needs the carrier sensing functionality for the scheduling information in its neighborhood at the previous time slot, i.e., for $N_l^\circ \cap S(t-1)$. Repeating the

procedure at each time slot, it has been shown that a stationary distribution can be obtained in a product form for all feasible schedules. Throughput optimality can be achieved, with the time-scale separation assumption, if $p_l = \frac{\exp(w_l(t))}{\exp(w_l(t))+1}$ where $w_l(t)$ is a non-decreasing and continuous function of $Q_l(t)$, e.g., $w_l(t) = \log\log Q_l(t)$. We refer the interested readers to [22], [24] for details.

Algorithm 3 Local Greedy Scheduling with Two contention mini-slots (LGS-Two)

At each time slot, each link l does

1: $pre_sched \leftarrow sched$
2: $sched \leftarrow 0$
3: **if** $Q_l(t) > 0$ and $i(l) = t \mod |\mathcal{D}| + 1$ (i.e., if link $l \in m(t)$) **then**
4: **if** $\frac{Q_l}{r_l} \geq \frac{Q_k}{r_k}$ for all $k \in N_l \cap S(t-1)$ **then**
5: $sched \leftarrow 1$
6: Broadcast a control message to its K-hop neighbors
7: **end if**
8: **end if**
9: **if** $Q_l(t) > 0$ and $i(l) \neq t \mod |\mathcal{D}| + 1$ (i.e., if link $l \notin m(t)$) **then**
10: **if** Not receive a control message and $pre_sched = 1$ **then**
11: $sched \leftarrow 1$
12: Broadcast a control message to its K-hop neighbors
13: **end if**
14: **end if**
15: **for** transmission period **do**
16: **if** $sched = 1$ **then**
17: Transmit data packets
18: **end if**
19: **end for**

We now modify our local greedy algorithm as shown in Algorithm 3. First, instead of choosing the decision schedule $m(t)$ from the set of all feasible schedules, we use the pre-determined set $\mathcal{D}$, i.e., $\mathcal{M}_0 = \mathcal{D}$, and select $m(t) = \vec{S}_i \in \mathcal{D}$ in a round robin manner with $i = t \mod |\mathcal{D}|+1$ (lines 3 and 9). Then from the construction of $\mathcal{D}$ satisfying (2), it is clear that all the conditions of (4) are satisfied. This modification is intended to improve the delay performance by giving each link a chance within a short time period, and to reduce the complexity by removing the contention period.

Second, we change the probabilistic transmission attempt of *Q-CSMA* to a deterministic one as follows. We consider links as eligible for transmissions at time slot t, if they belong to either $m(t)$ or $S(t-1)$. To begin with, each link $l \in m(t)$ includes itself in $S(t)$ if its queue is the longest among those neighbors eligible for transmissions (i.e., if $\frac{Q_l(t)}{r_l} \geq \frac{Q_k(t)}{r_k}$ for all $k \in N_l \cap S(t-1)$). These scheduled links broadcast a control message to inform their neighbors of the scheduling decision (line 6). Then each link $l \in S(t-1) \backslash m(t)$ makes its scheduling decision by including itself in $S(t)$ if none of its neighbors broadcasts a control message (lines 10-11). The newly scheduled links also broadcast a control message to its neighbors (line 12). The control messages are intended for the neighboring links to update the scheduling information $N_l^\circ \cap S(t)$, which will be used for scheduling decision at the next time slot. It can easily be checked that the resultant schedule is feasible since both $m(t)$ and $S(t-1)$ are a feasible schedule. Note that the removal of the probabilistic attempts will violate the assumption for the stationary distribution, and hence throughput optimality can no longer be proved using the techniques in [24]. However, our greedy approximation still captures the key feature of *Q-CSMA* that a link with a longer queue transmits with a higher probability.

Although we explained our algorithm using the framework of *Q-CSMA*, it can be also considered as a time expanded version of *LGS*. Instead of applying all schedules of $\mathcal{D}$ in order, we apply one of them at each time slot. We denote the new scheduling policy by Local Greedy Scheduling with Two contention mini-slots (*LGS-Two*) since it uses only two contention mini-slots. Interestingly, even the two contention mini-slots can be removed, if links have the carrier sensing functionality. We will further discuss this in the next section.

The precise characterization of the performance of local greedy approximations remains an open problem. In this paper, we analyze their complexity, discuss important issues relevant for practical implementation, and evaluate its performance through simulations comparing with other scheduling policies such as centralized *GMS* and *Q-CSMA*.

5 Complexity Analysis

5.1 Contention overhead

By contention overhead, we mean the time complexity required for a scheduling policy to determine its schedule in terms of the number of contention mini-slots. We first analyze the worst-case complexity of *LGS* under the 1-hop interference model, and extend it to the K-hop interference model. The results can be directly applied to *LGS-E*. We also discuss the complexity of *LGS-Two*.

The scheduling of *LGS* illustrated in Algorithm 1 requires at least $|\mathcal{D}|$ mini-slots to determine which link would be included in the schedule. Hence, the complexity of *LGS* depends on the cardinality of $\mathcal{D}$, which is the number of feasible schedules that enclose all links. Under the 1-hop interference model, the bound on $|\mathcal{D}|$ can be found from the following result on the link coloring problem [34].

Theorem 1 (Vizing): The links of a network can be colored so that no two links sharing a node have the same color using at most $\Delta + 1$, where Δ is the maximum node-degree of the network graph.

Theorem 1 immediately implies that $|\mathcal{D}| \leq \Delta+1$ because each set of links with the same color can be used as a feasible schedule in $\mathcal{D}$ satisfying (2). Assuming nodes are

randomly located in a wireless network, the complexity of *LGS* would be $O(\log |V|)$ under the 1-hop interference model[3].

We can extend the analysis to the K-hop interference model. Note that under the 1-hop interference model, the link coloring problem in $G(V, E, I)$ is equivalent to the *vertex*[4] coloring problem in its *conflict* (or interference) graph $G'(V', E', I')$, where the conflict graph G' can be obtained from the original graph G by changing links into vertices and connecting two vertices if they are interfering with each other. A theorem similar to Theorem 1 holds for the vertex coloring problem, i.e., all vertices can be colored with $\Delta_v(G')+1$ colors [34], where $\Delta_v(G')$ is the vertex-degree in the conflict graph G'. From the construction of the conflict graph, it is bounded by $\Delta_v(G') \leq 2(\Delta - 1)$ under the 1-hop interference model.

Let us consider the same set of nodes V and links E under the K-hop interference model. Let $G_K(V, E, I_K)$ denote the graph with the K-hop interference constraints. We can obtain its corresponding conflict graph $G''_K(V'', E'', I''_K)$ from the conflict graph under 1-hop interference model $G'(V', E', I')$ by adding links between vertices within K-hop distance. The resulting graph has the maximum vertex-degree $\Delta_v(G''_K) \leq 2((\Delta - 1) + (\Delta - 1)^2 + \cdots + (\Delta - 1)^K) < 4(\Delta - 1)^K$ for $\Delta \geq 3$. Now the solution of the vertex coloring problem in G''_K would provide the complexity bound of *LGS* under the K-hop interference model. Using a greedy coloring algorithm presented in Section 5.2, we can find a set $\mathcal{D}$ with $|\mathcal{D}| \leq 4(\Delta - 1)^K$. Therefore, in a network with randomly located nodes, *LGS* has $O(\log^K |V|)$ scheduling complexity[5]. Since *LGS-E* requires at most twice many mini-slots as *LGS*, the order result for the complexity does not change.

Before we discuss the complexity of *LGS-Two*, we briefly address the complexity of *Q-CSMA*. While *Q-CSMA* can get information of the previous schedule using the carrier sensing functionality, it still needs to resolve contention for the decision schedule $m(t)$. Hence, the complexity of *Q-CSMA* can be estimated by the number of mini-slots used for calculation of the decision schedule $m(t)$. A randomized matching is used: at each mini-slot, an eligible link can attempt to include itself in $m(t)$ by broadcasting a control message. If link l attempts at a mini-slot that is chosen uniformly at random, its neighbors that receive the message set themselves ineligible and they do not attempt during the rest of the contention period. Then link l includes itself in $m(t)$ if none of its neighbors attempts at the same mini-slot. Otherwise, there is a collision, and link l will receive a control message and sets itself ineligible. The procedure repeats at each mini-slot during the contention period. When the contention period ends, the chosen links will serve as the decision schedule $m(t)$, and the rest of *Q-CSMA* algorithm in the previous section proceeds. Note that the performance of *Q-CSMA* is related to the number of mini-slots. In general, the more mini-slots it has, the more links will be included in the decision schedule, which improves delay performance. Since the randomized maximal matching needs $O(\log^K |V|)$ complexity [21], *Q-CSMA* will also need the same complexity to achieve good delay performance.

LGS-Two, as its name indicates, requires only two contention mini-slots (from lines 6 and 12 of Algorithm 3). Given queue length information, it allows each link to determine its schedule from the schedule of the previous time slot and the current decision schedule chosen in a round robin manner. If each node has the carrier sensing functionality that allows overhearing, it does not even need the two contention mini-slots. In this case, it suffices for the links in $S(t) \cap m(t)$ to transmit data packets a little earlier, which can replace the first broadcast (line 6). Then the other links (i.e., links in $S(t-1) \backslash m(t)$) can determine their schedule based on the overheard transmissions and $S(t-1)$. The second broadcast (line 12) is also unnecessary since the neighboring links will notice the transmission by overhearing. Therefore, *LGS-Two* will be an attractive solution when the time for a mini-slot is precious, or when the network is heavily connected so that a large number of mini-slots are required for *LGS* and *LGS-E*. Note that *Q-CSMA* can also be implemented in a distributed manner with a small contention period, e.g., with one or two mini-slots. However, it will lead to a significant increase of delay since each link has a smaller chance to be included in $m(t)$ (e.g., see Fig. 2(b)). We evaluate the algorithms through simulations in Section 6, and observe that *LGS-Two* achieves high throughput with moderate delays under light traffics.

It is worth noting that the distributed *GMS* algorithms in [4], [5] require a global link ordering, which results in $O(|V|)$ complexity. Although the process can be accelerated by parallel executions to find a local maximum as in [18], the worst-case complexity still scales with the network size. For example, in a linear network where all nodes are placed in a line, it can also have $O(|V|)$ complexity.

5.2 Distributed greedy coloring algorithm

In this section, we provide a distributed greedy (vertex) coloring algorithm in the conflict graph, which can be used for all three local greedy algorithms to determine the set $\mathcal{D}$. Its centralized version can be found in [34]. It

3. It has been shown in [35] that considering a network on a unit area, the minimum transmission range of a node for connectivity is $O(\sqrt{\frac{\log |V|}{|V|}})$. Then the number of nodes within the transmission area is $O(\log |V|)$ since the node density is $O(|V|)$. Clearly, $|\mathcal{D}| = O(\log |V|)$ when the nodes are randomly located. In the worst case topology, a node can have $O(|V|)$ neighbors resulting in $|\mathcal{D}| = O(|V|)$. However, for typical network topologies $|\mathcal{D}|$ is much less than $O(|V|)$

4. We use the term 'vertex' to indicate node in the conflict graph to easily distinguish it from node in the original graph.

5. Note that the time complexity is measured in the number of contention mini-slots. Under the K-hop interference model, each mini-slot requires additional K broadcasts unless nodes have the carrier-sensing functionality.

colors links such that any two interfering links do not share a color. After finishing the coloring, we obtain the sets of links with different colors, where each set of links with the same color equals to an element of $\mathcal{D}$.

Given a network graph $G(V, E, I)$ with the maximum node-degree $\Delta \geq 3$, let $G'(V', E', I')$ denote its corresponding conflict graph under the 1-hop interference model with the maximum vertex-degree $\Delta_v(G') \leq 2(\Delta - 1)$. Under the K-hop interference model, the conflict graph can be extended to take into account the interference constraints by adding links between vertices within K-hop distance in G'. Let $G''_K(V'', E'', I''_K)$ denote the resulting conflict graph, which has the maximum vertex-degree $\Delta_v(G''_K) < 4(\Delta - 1)^K$ for $\Delta \geq 3$ as shown in Section 5.1. Assume that all vertices in V'' (i.e. links in E) are numbered as $v_1, v_2, \ldots, v_{|E|}$. We color vertices such that any two interfering vertices do not have the same color. Since we color one vertex in a round, the whole coloring takes $|E|$ rounds.

Suppose that we have a set $\mathcal{C}$ of colors $c_1, c_2, \ldots, c_{|\mathcal{C}|}$. Each vertex v_i maintains a set $\mathcal{C}_i$ of available colors, which is initially set to $\mathcal{C}$. At each round r, we color a vertex v_r with the color of the lowest index in $\mathcal{C}_r$, i.e., c_x where $x = \min\{y \mid c_y \in \mathcal{C}_r\}$. For each vertex v_j interfering with the vertex v_r, we remove c_x from its available color set, i.e., $\mathcal{C}_j \leftarrow \mathcal{C}_j \backslash \{c_x\}$ for all j such that $(v_r, v_j) \in E''$. Note that each vertex has at most $\Delta_v(G''_K)$ neighboring vertices. Hence, when we color a vertex, we can always find a color index no greater than $\Delta_v(G''_K) + 1$. This means that $\Delta_v(G''_K) + 1$ colors suffice for the coloring, i.e., $|\mathcal{C}| \leq \Delta_v(G''_K) + 1$. The distributed version of this greedy coloring algorithm is shown in Algorithm 4.

Algorithm 4 Distributed Greedy Coloring.

Each vertex (link) v_i does
1: Initializes its available color set $\mathcal{C}_i \leftarrow \{c_1, \ldots, c_{|\mathcal{C}|}\}$
2: **for** each round r $(1 \leq r \leq |E|)$ **do**
3: **if** $i = r$ **then**
4: $x \leftarrow \min\{y \mid c_y \in \mathcal{C}_r\}$
5: color the vertex v_r with c_x
6: Broadcast a control message with c_x to its K-hop neighbors
7: **else**
8: **if** Receive a control message with c_z from its neighbor **then**
9: $\mathcal{C}_i \leftarrow \mathcal{C}_i \backslash \{c_z\}$
10: **end if**
11: **end if**
12: **end for**

The distributed algorithm takes $|E|$ rounds and requires to transmit a control message to the K-hop neighbors in each round (line 6). Since a message has to be forwarded in $G_K(V, E, I_K)$, it requires at least K broadcasts in each round. Hence, the time complexity of the coloring is $O(K|E|)$. Note that we do not need to do the coloring at each scheduling decision. Once colors are assigned and $\mathcal{D}$ is obtained, we reuse the same $\mathcal{D}$ at every scheduling decision unless the network topology changes. In static settings like high-speed access networks, the overhead for coloring is negligible.

5.3 Exchange of queue information

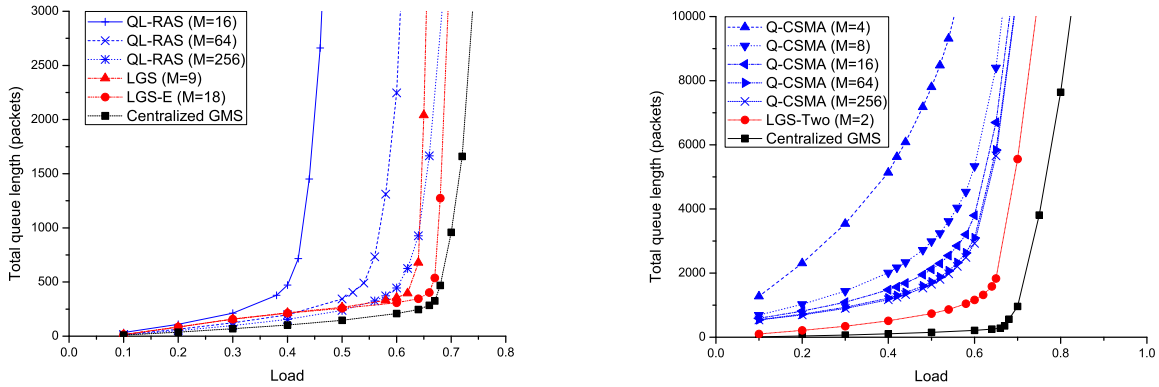
In practical implementation of the local greedy algorithms, the queue information of each link should be distributed to its neighbors. Under the 1-hop interference model, the information exchange can be done relatively quickly, i.e., within $|\mathcal{D}|$ rounds; at the i-th round, all links in $\vec{S}_i \in \mathcal{D}$ can be activated without interference and allow their two end nodes to exchange the queue information. Hence, the overall time complexity including the time required for the information exchange remains $O(\log |V|)$ for *LGS* and *LGS-E*. Under the K-hop interference model, the queue information of a link has to be forwarded to its K-hop neighbors. This will requires K rounds of message exchanges, where each round has a time complexity $O(\log^K |V|)$, increasing the overall complexity to $O(K \log^K |V|)$.

For *LGS-Two*, it is more likely that the network system requires a low complexity. In this case, it would be better to utilize piggy-backing of queue information on the data and acknowledgment packets. An appropriate time-averaging method and/or an auxiliary periodic broadcast of queue information can be useful to maintain correct information of neighboring links.

6 Simulation Results

We evaluate the performance of several scheduling polices under the 1-hop and 2-hop interference models. We compare our local greedy scheduling with several other policies, including centralized *GMS*, *Q-CSMA*, and a queue-length based random access scheduling (*QL-RAS*) [11], [12], which will be explained shortly. We measure the total queue lengths in the network changing the traffic loads. For each scheduling policy, the results show that the queue lengths rapidly increase when the traffic loads approach the boundary of the capacity region. They also show a typical level of queue lengths when the network is stabilized, i.e., delay performance. Before we explain our simulation settings, we brief the *QL-RAS* algorithm.

QL-RAS algorithm is another approximation of centralized *GMS* and is a state-of-the-art distributed scheduling scheme with $O(1)$ complexity in the *Constant-time Random Access Approach* category. Basically, it operates as the randomized matching that is used to find the decision schedule of *Q-CSMA*. The difference is that under *QL-RAS*, links attempt at each mini-slot with a probability, and each link has a different attempt probability (instead of choosing one mini-slot uniformly at random). The probability is given as a function $p(\vec{Q}(N_l), M)$, where $\vec{Q}(N_l)$ is the queue length information of l's neighbors and M is the number of mini-slots. It has been



(a) *QL-RAS*, *LGS*, *LGS-E*, and *GMS*

(b) *Q-CSMA*, *LGS-Two*, and *GMS*

Fig. 2. Performance of scheduling policies with different contention periods under the 1-hop interference model. The centralized *GMS*, *LGS* and *LGS-E* achieve similar throughput performance. *QL-RAS* requires a massive amount of contention mini-slots to achieve a comparable performance to *LGS-E*. Also, *LGS-Two* outperforms *Q-CSMA* achieving moderate delays under light traffic loads. An interesting observation for *Q-CSMA* is that its performance improves with a small increase of contention mini-slots (i.e., from $M = 4$ to 8), which, however, soon stops improving beyond some point (i.e., after $M = 64$).

TABLE 1
Efficiency Ratios under the 1-hop Interference Model.

Scheduling	Efficiency ratio
GMS	σ^*
LGS/LGS-E	$\geq (\sigma^*)^\dagger$
Q-CSMA	1
QL-RAS	$\geq 1/2 - 1/\sqrt{M}$

† Under assumption that we can find an appropriate function $f(\cdot)$ as specified in Proposition 2.

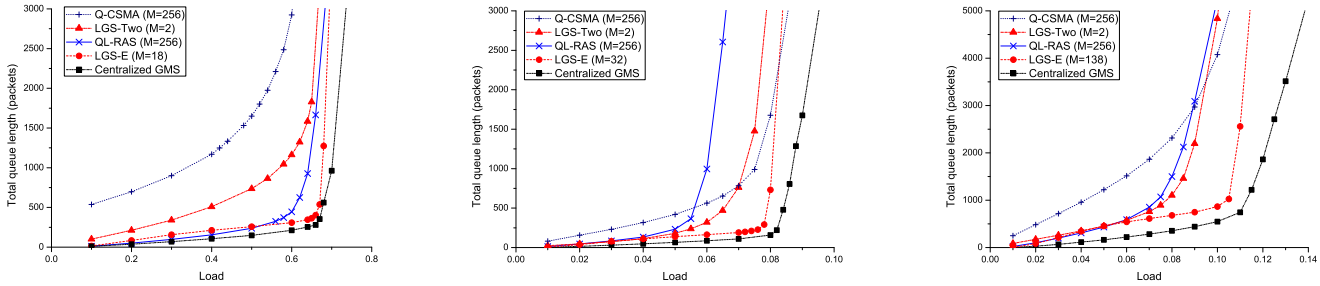
shown that *QL-RAS* achieves at least $(\frac{1}{2} - \frac{1}{\sqrt{M}})$ of the optimal performance under the 1-hop interference model with $p(\vec{Q}(N_l), M) = \frac{\sqrt{M}-1}{2M} \cdot \frac{Q_l/c_l}{\max_{\mathcal{N}\in\{N_1,N_2\}} \sum_{k\in\mathcal{N}} Q_k/c_k}$, where N_1 and N_2 are the set of links connected to each end node of link l, respectively. It is observed in [12] that the empirical performance of *QL-RAS* approaches that of *GMS* as M increases. (We refer to [12] for details.) The performance of different scheduling schemes under the 1-hop interference model is summarized in Table 1.

Our simulation settings are as follows: we generate a network graph on a 1x1 square area by randomly placing 50 nodes. Two nodes are connected by a link if they are within a distance of 0.2 (over 120 links are generated). We first consider the 1-hop interference model, under which the network graph has a maximum node-degree of 8. For the predetermined set of schedules $\mathcal{D}$, we randomly choose a set of matchings satisfying $|\mathcal{D}| \leq 9$. (The greedy coloring algorithm shown in Section 5.2 can be used, but it yields a looser bound of $|\mathcal{D}| \leq 2(\Delta - 1) = 14$.) Each link has a capacity between $[5, 10]$ (uniformly distributed). For each link, we consider single-hop traffic with mean arrival rate either 0 (with probability 0.2), 1 (with probability 0.6), or 2 (with probability 0.2). Packets arrive at links following a Poisson distribution.

Each scheduling policy has M contention mini-slots during the contention period. For example, *LGS* has $M = |\mathcal{D}| = 9$ and *LGS-E* has $M = 2|\mathcal{D}| = 18$ under the 1-hop interference model. For *QL-RAS* and *Q-CSMA*, we change the number of mini-slots, where *Q-CSMA* uses the mini-slots to find the decision schedule $m(t)$. We assume that each link has the queue information of its interfering neighbors, and do not take into account the overhead of the contention mini-slots unless explicitly stated. For *Q-CSMA*, we set $w_l(t) = \log\log Q_l(t)$ and assume that each link has knowledge of the previous schedule in its neighborhood from the carrier sensing.

Fig. 2(a) shows the total queue lengths under *QL-RAS*, *LGS*, *LGS-E*, and centralized *GMS*. For the traffic load vector randomly chosen as in the above, we scale the load vector by multiplying a factor. The x-axis represents the scaling factor. We observe that the total queue length rapidly increases over a certain threshold for each scheduling policy. The load at the threshold can be considered as the boundary of the capacity region of the policy. Fig. 2(a) shows that the capacity region of *GMS* is the largest. However, the performance gap between *GMS*, *LGS*, and *LGS-E* is relatively small. In particular, the capacity boundaries of *LGS-E* and *GMS* are almost the same. Fig. 2(a) also shows that *LGS-E* ($M = 18$) outperforms *QL-RAS* with $M = 16$ and $M = 64$. There is a significant performance difference until *QL-RAS* uses a large number of mini-slots ($M = 256$), which however may result in excessive overhead.

We also compare the performance of *LGS-Two*, centralized *GMS*, and *Q-CSMA*. Fig. 2(b) shows the results, where the performance differences between the



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 3. Performance comparison of scheduling policies under different network scenarios. (a) Scenario 1 with 50 nodes, asymmetric traffic, and the 1-hop interference model. (b) Scenario 2 with 30 nodes, highly asymmetric traffic, and the 2-hop interference model. (c) Scenario 3 with 100 nodes, symmetric traffic, and the 3-hop interference model.

three algorithms are very clear showing that *LGS-Two* outperforms *Q-CSMA*. An interesting point is that the performance of *Q-CSMA* improves with a small increase of the number of mini-slots (i.e., from $M = 4$ to 8), but unlike *QL-RAS*, it does not improve beyond a certain point (i.e., after $M = 64$). Moreover, *Q-CSMA* has large queue lengths even at light loads, which indicates that the delay performance of *Q-CSMA* is poor. The results also suggest that the delay performance can improve if local neighborhood queue information is used.

In Fig. 3, we directly compare all the scheduling policies. Fig. 3(a) shows that under the same network settings (denoted by Scenario 1), the performance (from best to worst) is in the following order: centralized *GMS*, *LGS-E*, *QL-RAS*, *LGS-Two*, and *Q-CSMA*. We highlight that *LGS-E* achieves good performance with small complexity $M = 18$, substantially better than the other distributed policies. We also simulate the scheduling policies in a couple of different network scenarios of different network topologies, traffic loads, and interference models. Scenario 2 has a network topology of 30 nodes, whose locations are chosen at random over 1x1 area. A link has been placed if any two nodes are within distance 0.26 (total 61 links; $\Delta = 7$), and has a capacity between $[5, 10]$. We generate highly asymmetric traffic by injecting single-hop traffic packets (following a Poisson distribution) to each link with a mean arrival rate chosen at uniformly random between $[0, 10]$. We scale the traffic loads by multiplying a varying scaling factor as before. We simulate the scheduling policies under the 2-hop interference model, and have $|\mathcal{D}| = 32$. For Scenario 3, we simulate a network topology of 100 nodes with random locations. A link has been established between two nodes of distance ≤ 0.14 (total 255 links; $\Delta = 11$), and has a capacity between $[5, 10]$. We generate symmetric traffic such that all links have packet arrivals following a Poisson distribution with a unit mean rate (also scaled by a scaling factor). We consider the 3-hop interference model for Scenario 3 ($|\mathcal{D}| = 138$).

Figs. 3(b) and 3(c) show the simulation results. Overall, they are similar to those of Scenario 1 except that the performance of *QL-RAS* substantially decreases. This is in part because the same number of mini-slots $M = 256$ is used in all three scenarios despite the fact that *QL-RAS* usually requires a larger number of mini-slots M under the 2 (or more)-hop interference model [12]. However, although we can improve the performance of *QL-RAS* by increasing M, we will see later that a larger number of mini-slot also increases the overhead complexity. Another interesting observation is that *Q-CSMA* and *LGS-Two* achieve better performance (i.e., close to the performance of *GMS*) as the traffic load becomes asymmetric.

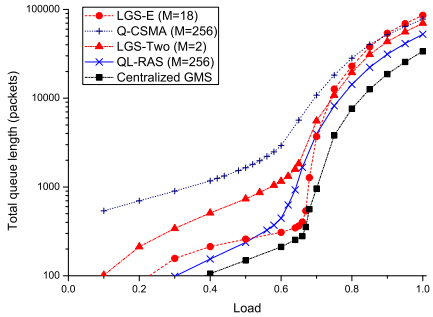


Fig. 4. Performance comparison under heavy traffic load (Scenario 1; log scale).

Under heavy traffic loads, whose results for Scenario 1 are shown in Fig. 4, the performance of the scheduling schemes is now ordered differently (from best to worst): centralized *GMS*, *QL-RAS*, and *Q-CSMA*/*LGS-Two*/*LGS-E*. The differences between *Q-CSMA*, *LGS-Two*, and *LGS-E* are not significant. Although *QL-RAS* performs well both in light and heavy loads, it requires the highest complexity. Later we will see that the performance of *QL-RAS* is severely degraded when the overhead is taken into account. While *LGS-E* shows the worst performance under heavy loads, the performance cross-over occurs when the queue lengths are very large. Hence, *LGS-E* is indeed very attractive providing good delay perfor-

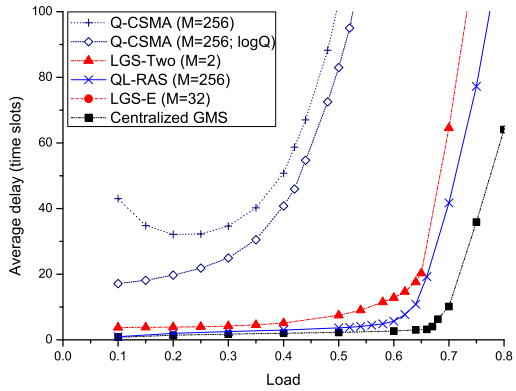


Fig. 5. Delay performance of scheduling policies (Scenario 1).

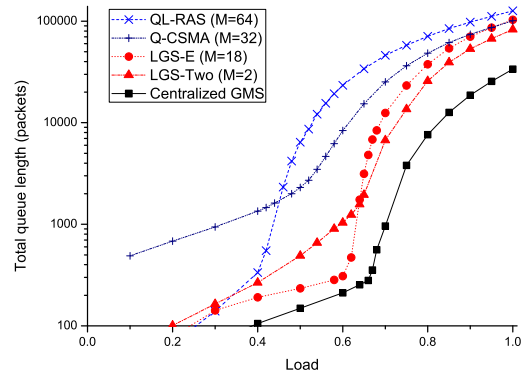


Fig. 6. Performance of scheduling policies under the 1-hop interference model taking into account the overhead of contention mini-slots (log scale). A single mini-slot has a 2^{-8} length of a time slot, and hence, each scheduling policy has the overhead of $2^{-8}M$.

mance when the traffic load is moderate and the network is not heavily connected (for low complexity from small $|\mathcal{D}|$). The results for Scenarios 2 and 3 are very similar and we omit them.

It is worth noting the limitation on the comparison of simulation results with analytical results obtained in the previous sections. Since the efficiency ratio is the metric of *worst-case* performance, it is possible for two scheduling policies, say $\mathcal{P}$ and $\mathcal{Q}$, with $\gamma^*_{\mathcal{P}} \leq \gamma^*_{\mathcal{Q}}$, to have policy $\mathcal{P}$ outperforming policy $\mathcal{Q}$ for a particular network scenario. Also, the analytical results do not provide the precise queue length at which the network stability threshold is reached. A scheduling policy may have a large queue length while stabilizing the network. For example, Fig. 4 shows that though *Q-CSMA* has larger queue lengths than *LGS-E* in light loads, as the traffic load increases, it performs relatively better and outperforms *LGS-E* in heavy loads.

Fig. 5 shows the delay performance of the scheduling schemes. We measure average packet delay under Scenario 1. The results are similar to the queue length results except for *Q-CSMA* in light traffic load. Under *Q-CSMA*, it may take long for a packet to be transmitted even in light traffic load, since the attempt probability of a link depends only on its own queue length and will be small when the queue length is small. Hence, the results show that as the traffic load increases up to 0.2, the delay performance of *Q-CSMA* gets better owing to high attempt probability, and then the performance decreases for traffic loads greater than 0.25 because of high queueing delay. *Q-CSMA* with $w_l = \log Q_l(t)$ (instead of $\log\log Q_l(t)$) improves the delay performance in light traffic loads, which, however, is still substantially worse than the performance of the other scheduling schemes. *GMS* and the class of *LGS* schemes pick links based on *relative* queue length, and thus do not experience large packet delay under light traffic loads.

Now we take into account the complexity overhead. The contention period does in fact cause an overhead, resulting in throughput performance of scheduling policies. If a contention mini-slot takes a fixed time length, the number of mini-slots can be directly translated into the amount of performance degradation. Fig. 6 shows the performance of scheduling policies taking into account the overhead assuming that each mini-slot takes 2^{-8} length of a single time slot. For *QL-RAS* and *Q-CSMA*, we simulate them changing the number of mini-slots and show the best results. The results show that the performance of *QL-RAS* is significantly degraded and lags behind the others under heavy traffic loads, while it still outperforms the others under very light traffic loads. Also the performance gap between *Q-CSMA* and *LGS-Two* is enlarged since *LGS-Two* requires only two contention mini-slots. Note that we take into account only the overhead from the contention period, which may result in overestimating the performance of the scheduling schemes. In a real implementation, the scheduling schemes require queue length information of neighboring links, which can result in additional complexity due to direct message exchanges, or performance degradation due to inaccurate estimates of queue length information, when the information is piggy-backed. In this sense, *Q-CSMA*, which does not require the queue information from its neighbors, has a relatively lower complexity. *Q-CSMA* can reduce the message passing overhead by requiring a carrier sensing mechanism to sense simultaneous transmission of other links.

Finally, we extend our simulations to the 2-hop interference model with a higher overhead. Each contention mini-slot takes 2^{-6} length of a time slot. We omit the results of *QL-RAS* due to its relatively poor performance. The greedy coloring algorithm shown in Section 5.2 is used to find $\mathcal{D}$. In this particular network scenario, the greedy coloring algorithm finds a set of schedules with $|\mathcal{D}| = 29$. Hence, for *LGS-E* with $M = 58$, most of a time slot (about 90%) is used for finding a schedule and only

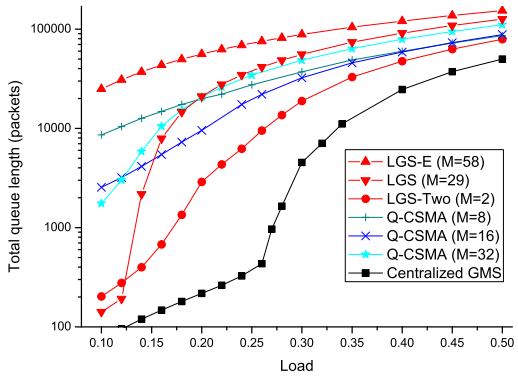


Fig. 7. Performance of scheduling policies under the 2-hop interference model (log scale) with a high overhead per contention mini-slot. (A single mini-slot has a 2^{-6} length of a time slot.) We focus on *LGS-Two* and *Q-CSMA* since they achieve high performance in heavy load with a small number of contention mini-slot. Note that *GMS* is unlikely to be implemented in such settings. We compare the performance of *GMS* for benchmark purpose.

10% for data transmission. Fig. 7 show that the overhead highly affects the performance of scheduling policies. For example, *LGS* outperforms *LGS-E* since it has a smaller complexity, and for *Q-CSMA*, the parameter M needs to be chosen appropriately for good performance. As expected, *LGS-Two* significantly outperforms the other scheduling policies. Its low complexity is very attractive in particular when the network resources are scarce and/or the network is heavily connected.

7 Conclusion

Greedy Maximal Scheduling (*GMS*) is a promising scheduling solution in multi-hop wireless networks that provably outperforms many distributed scheduling policies appears to empirically achieve optimal performance ove a variety of different network topologies and traffic distributions. However, its distributed implementation requires high computational complexity of $O(|V|)$. Recently, a throughput-optimal scheduling scheme that is amenable to distributed implementation, called *Q-CSMA*, has been developed. However, it requires the carrier sensing functionality and suffers from a large delay even under a light traffic load.

In this paper, we propose local greedy algorithms for scheduling, which approximate the performance of *GMS* with lower complexity. The proposed algorithms reduce the complexity of *GMS* by excluding from the schedule links with a smaller queue length than their local neighbors. Although the global link ordering of *GMS* is replaced with a local ordering, we show that its empirical performance is close to *GMS*. It comes from the intuition that the links with locally longest queues are important to characterize the capacity region.

The proposed algorithms acquire this property in a distributed and collision-free fashion with minimal complexity by pre-assigning an index to each link conforming to the interference constraints. It turns out that for *LGS* and *LGS-E*, the minimum cardinality of the assigned index set determines the complexity, which has been shown to be $O(\log|\Delta|)$, where Δ denotes the maximum node degree. For *LGS-Two*, we reduce the complexity to two contention mini-slots, which can be further reduced to near-zero if we employ carrier sensing. We also address the issues of the distributed link coloring for the index assignment and the queue information exchange.

We evaluate our local greedy scheduling through simulations under the K-hop interference models for $K = 1, 2, 3$. In all cases, they achieve the throughput performance close to *GMS* outperforming the state-of-the-art scheduling policies. In particular, *LGS-E* shows good delay performance in moderate traffic loads, and *LGS-Two* outperforms the other scheduling policies when a very low complexity is required due to scarce network resources. Our results also suggest that the delay performance of a throughput-optimal scheduler such as *Q-CSMA* can improve with additional queue information of links' interfering neighbors.

There remain several open problems. We are interested in characterizing the exact throughput performance of the local greedy algorithms and in analyzing their delay performance. Understanding the fundamental tradeoff between these performance metrics and complexity is also of interests. In addition, our development is based on the K-hop interference model, which capture the essential features of the signal-to-interference-and-noise ratio (SINR) based interference model. It is interesting to extend the local greedy schemes to more general interference models like the SINR based interference model.

Acknowledgements

This work was supported in part by NSF project CNS-0721236 and ARO MURI project W911NF-08-1-0238, and in part by the Education and Research Promotion Program of KUT.

References

[1] C. Joo, "A Local Greedy Scheduling Scheme with Provable Performance Guarantee," in *ACM MOBIHOC*, May 2008.

[2] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximal Throughput in Multihop Radio Networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[3] X. Lin and N. B. Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, April 2006.

[4] R. Preis, "Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs," in *Symposium on Theoretical Aspects of Computer Science*, 1999.

[5] J.-H. Hoepman, "Simple Distributed Weighted Matchings," eprint, October 2004. [Online]. Available: http://arxiv.org/abs/cs/0410047v1

[6] E. Modiano, D. Shah, and G. Zussman, "Maximizing Throughput in Wireless Networks via Gossiping," *Sigmetrics Performance Evaluation Review*, vol. 34, no. 1, pp. 27–38, 2006.

[7] Y. Yi and S. Shakkottai, "Learning Contention Patterns and Adapting to Load/Topology Changes in a MAC Scheduling Algorithm," in *IEEE WiMesh*, September 2006.

[8] S. Sanghavi, L. Bui, and R. Srikant, "Distributed Link Scheduling with Constant Overhead," in *ACM Sigmetrics*, June 2007, pp. 313–324.

[9] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial Complexity Algorithms for Full Utilization of Multi-hop Wireless Networks," in *IEEE INFOCOM*, May 2007.

[10] X. Lin and S. Rasool, "Constant-Time Distributed Scheduling Policies for Ad Hoc Wireless Networks," in *IEEE CDC*, December 2006.

[11] A. Gupta, X. Lin, and R. Srikant, "Low-Complexity Distributed Scheduling Algorithms for Wireless Networks," in *IEEE INFOCOM*, May 2007, pp. 1631–1639.

[12] C. Joo and N. B. Shroff, "Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, October 2009.

[13] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu, "Joint Asynchronous Congestion Control and Distributed Scheduling for Multi-Hop Wireless Networks," in *IEEE INFOCOM*, April 2006, pp. 1–12.

[14] A. Eryilmaz and R. Srikant, "Fair Resource Allocation in Wireless Networks Using Queue-length Based Scheduling and Congestion Control," in *IEEE INFOCOM*, March 2005.

[15] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[16] C. Joo, G. Sharma, N. B. Shroff, and R. R. Mazumdar, "On the Complexity of Scheduling in Wireless Networks," *EURASIP Journal of Wireless Communications and Networking*, October 2010.

[17] D. Avis, "A Survey of Heuristics for the Weighted Matching Problem," *Networks*, vol. 13, no. 4, pp. 475–493, 1983.

[18] M. Leconte, J. Ni, and R. Srikant, "Improved Bounds on the Throughput Efficiency of Greedy Maximal Scheduling in Wireless Networks," in *ACM MOBIHOC*, 2009, pp. 165–174.

[19] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput Guarantees in Maximal Scheduling in Wireless Networks," in *the 43rd Annual Allerton Conference on Communication, Control and Computing*, September 2005.

[20] X. Wu and R. Srikant, "Scheduling Efficiency of Distributed Greedy Scheduling Algorithms in Wireless Networks," in *IEEE INFOCOM*, April 2006.

[21] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "Joint Congestion Control and Distributed Scheduling for Throughput Guarantees in Wireless Networks," in *IEEE INFOCOM*, May 2007, pp. 2072–2080.

[22] L. Jiang and J. Walrand, "A Distributed Algorithm for Optimal Throughput and Fairness in Wireless Networks with a General Interference Model," in *the 46th Annual Allerton Conference on Communications, Control, and Computing*, 2008.

[23] P. Marbach and A. Eryilmaz, "A Backlog-Based CSMA-Mechanism to Achieve Fairness and Throughput-Optimality in Multihop Wireless Networks," in *the 46th Annual Allerton Conference on Communications, Control, and Computing*, 2008.

[24] J. Ni and R. Srikant, "Distributed CSMA/CA Algorithms for Achieving Maximum Throughput in Wireless Networks," in *Information Theory and Applications Workshop*, 2009.

[25] B. Hajek and G. Sasaki, "Link Scheduling in Polynominal Time," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, September 1988.

[26] S. Sarkar and L. Tassiulas, "End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Ad-hoc Networks," in *IEEE CDC*, December 2003, pp. 564–569.

[27] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "On the Stability of Input-Queued Switches with Speed-Up," *IEEE/ACM Trans. Netw.*, vol. 9, no. 1, Feburary 2001.

[28] A. Dimakis and J. Walrand, "Sufficient Conditions for Stability of Longest-Queue-First Scheduling: Second-order Properties using Fluid Limits," *Advances in Applied Probability*, vol. 38, no. 2, pp. 505–521, 2006.

[29] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling Distributed Throughput Maximization in Wireless Mesh Networks: A Partitioning Approach," in *ACM MOBICOM*, 2006, pp. 26–37.

[30] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop Local Pooling for Distributed Throughput Maximization in Wireless Networks," in *IEEE INFOCOM*, April 2008.

[31] C. Joo, X. Lin, and N. B. Shroff, "Understanding the Capacity Region of the Greedy Maximal Scheduling Algorithm in Multihop Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1132–1145, August 2009.

[32] ——, "Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-exclusive Interference Model," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 1132–1145, August 2009.

[33] J. G. Dai, "On Positive Harris Recurrence of Multiclass Queueing Networks: A Unified Approach via Fluid Limit Models," *Annals of Applied Probability*, vol. 5, no. 1, pp. 49–77, 1995.

[34] R. Diestel, *Graph Theory*, 3rd ed. Springer, 2005.

[35] P. Gupta and P. R. Kumar, "Critical Power for Asymptotic Connectivity in Wireless Networks," *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, pp. 547–566, 1998.

Changhee Joo (S'98-M'05) received his Ph.D degree from the school of Electrical and Computer Engineering, Seoul National University, Korea, 2005. He joined the Center of Wireless Systems and Applications, Purdue University, and also work at the Ohio State University as a research scientist. In March 2010, he joined Korea University of Technology and Education. His research interests include communication systems, cross-layer network optimization, wireless sensor networks, and network controls. He is a member of IEEE, and a recipient of the IEEE INFOCOM 2008 best paper award.

Ness B. Shroff (S'91-M'93-SM'01-F'07) received his Ph.D. degree from Columbia University, NY in 1994 and joined Purdue university immediately thereafter as an Assistant Professor. At Purdue, he became Professor of the school of Electrical and Computer Engineering in 2003 and director of CWSA in 2004, a university-wide center on wireless systems and applications. In July 2007, he joined Ohio State University as the Ohio Eminent Scholar of Networking and Communications. University.

His research interests span the areas of wireless and wireline communication networks. He is especially interested in fundamental problems in the design, performance, control, and security of these networks. Dr. Shroff is a past editor for IEEE/ACM Trans. on Networking and the IEEE Communications Letters. He currently serves on the editorial board of the Computer Networks Journal. He has served on the technical and executive committees of several major conferences and workshops. He was the TPC co-chair of IEEE INFOCOM'03, ACM Mobihoc'08, and general chair of IEEE CCW'99 and WICON'08. Dr. Shroff is a fellow of the IEEE.

He has received numerous awards for his work, including two best paper awards at IEEE INFOCOM (in 2006 and 2008), the flagship conference of the field. He has also received the IEEE IWQoS 2006 best student paper award, the 2005 best paper of the year award for the Journal of Communications and Networking, the 2003 best paper of the year award for Computer Networks, and the NSF CAREER award in 1996 (his IEEE INFOCOM 2005 paper was selected as one of two runner-up papers).