Hierarchical Scheduling Algorithms with Throughput Guarantees and Low Delay

Peruru Subrahmanya Swamy*, Aravind Srinivasan†, Radha Krishna Ganti*, Krishna Jagannathan*
*Department of Electrical Engineering, IIT Madras, Chennai, India 600036
†Department of Computer Science, University of Maryland, College Park, MD, USA 20742
{p.swamy, rganti, krishnaj}@ee.iitm.ac.in, srin@cs.umd.edu

***Abstract*—We propose distributed scheduling algorithms that guarantee a constant fraction of the maximum throughput for typical wireless topologies, and have $O(1)$ delay and complexity in the network size. Our algorithms resolve collisions among pairs of conflicting nodes by assigning a master-slave hierarchy. When the master-slave hierarchy is chosen randomly, our algorithm matches the throughput performance of the maximal scheduling policies, with a complexity and delay that do not scale with network size. When the master-slave hierarchy is chosen based on the network topology, the throughput performance of our algorithm is characterized by a parameter of the conflict graph called the *master-interference* degree. For commonly used conflict graph topologies, our results lead to the best known throughput guarantees among the algorithms that have $O(1)$ delay and complexity. Numerical results indicate that our algorithms outperform the existing $O(1)$ complexity algorithms like Q-CSMA.**

I. Introduction

The design of efficient distributed scheduling algorithms is a fundamental problem in wireless networks. The goal of a scheduling algorithm is to determine which subset of the non-interfering nodes should transmit at each time instant so that the service requirements of all the nodes are met. Based on the network topology and the interference constraints, there exists a maximal set of rates that can be achieved, known as the achievable rate region. The performance of a scheduling algorithm is measured based on the fraction of rate region that the algorithm can support, the complexity of determining the schedules, and the delay incurred under that algorithm. If an algorithm can support any rate vector in the interior of the rate region, it is said to be a throughput maximizing policy. Max-Weight scheduling algorithm [1] is a centralized scheduling algorithm which is known to be throughput maximizing. However, due to its high complexity, and the fact that it requires global information to determine the schedule, it is not suitable for implementation in large wireless networks.

A. Related work

There have been several efforts [2], [3], [4] in the literature to design distributed approximations to the Max-weight algorithms with lower complexity. Greedy algorithms known as maximal scheduling policies [2] is one such class of approximations. For network topologies that are modelled using a conflict graph, these maximal scheduling policies can provably support $\frac{1}{\chi}$ fraction of the rate region, and they incur a complexity of $O(\log N)$, where χ denotes the maximum *interference degree* of the conflict graph (See Definition 1), and N is the network size. In other words, maximal scheduling policies achieve a reasonable fraction of the throughput, but their complexity scales logarithmically with the network size.

Subsequent efforts [5], [6] have focussed on developing scheduling algorithms whose complexity does not scale with the network size. In particular, [5] supports $\frac{1}{\Delta}$ fraction of the rate region with $O(1)$ complexity, where Δ is the maximum degree of the conflict graph, which can be potentially much higher than the maximum interference degree χ. On the other hand, an Aloha based protocol [6] supports $\frac{1}{e\chi}$ fraction of the rate region, and is hence off by a factor of $\frac{1}{e}$ compared to the maximal scheduling algorithms. In other words, [5], [6] improve the complexity at the cost of reduced throughput.

In another line of work, Gibbs sampling [7] based Q-CSMA algorithms [8], [9] are shown to be throughput maximizing, and have $O(1)$ complexity in finding the schedules. While these algorithms are good in terms of throughput, they suffer from poor delay performance, which often scales exponentially with the network size [10]. This is due to the slow mixing time of the underlying Markov chain that governs the CSMA schedules [10]. In particular, one cannot utilize the full throughput of Q-CSMA based algorithms, without suffering large delay [10]. The authors in [11] proposed a CSMA based algorithm called the fast CSMA, which overcomes the problem of large delay. However, their results are restricted to fully-connected conflict graph topologies.

B. Our contributions

For a scheduling algorithm to be implementable in a large wireless network, it is desirable that its complexity, delay and throughput performances do not scale with the network size. In this work, we intend to understand the fraction of the maximum throughput that can be supported, while insisting that the complexity and delay of the scheduling algorithm be $O(1)$ in the network size.

To that end, we propose distributed scheduling algorithms which are guaranteed (i) to have $O(1)$ delay in the size of the network, (ii) to have $O(1)$ complexity, whenever the maximum neighbourhood size does not scale with network size, and (iii) to support a constant fraction of the rate region for typical wireless network topologies. The central idea is to assign a master-slave hierarchy for each pair of conflicting nodes, and whenever there is a possible conflict of transmission between them, the slave has to back-off transmitting from that slot. The

Supported in part by NSF awards CCF-1422569 and CCF-1749864, and by research awards from Adobe, Inc.

maser-slave hierarchy is either chosen randomly, or based on the conflict graph topology, leading to different performance guarantees.

Some of our key findings are summarized below:

- When the master-slave hierarchy is chosen randomly, our algorithm guarantees $1/\chi$ fraction of the rate region with $O(1)$ delay and complexity. To the best of our knowledge, this is the only distributed algorithm that matches the throughput guarantee of maximal scheduling, with a complexity and delay that do not scale with the network size.
- When the master-slave hierarchy is chosen based on the conflict graph topology, our algorithm guarantees $1/(e\chi_m)$ fraction of the rate region with $O(1)$ delay and complexity. Here, χ_m refers to a term called the *master-interference degree*, which can be drastically smaller than the interference degree χ for many relevant conflict graph topologies.
- When the conflict graph is a disk graph, which is often used to model the interference in wireless networks, the above result guarantees a throughput fraction of $\frac{1}{5e}$ for any arbitrary disk radii. We note that this is the best known throughput guarantee for a disk graph with *heterogeneous radii* among algorithms with $O(1)$ delay and complexity. Furthermore, existing distributed algorithms [2], [6] fail to provide any constant fraction throughput guarantees for heterogeneous disk graphs – this is because the interference degree of an arbitrary disk graph can be unbounded, while the master-interference degree can be upper bounded by a constant 5.
- When the conflict graph is a *chordal* graph (which includes trees), our algorithm guarantees $\frac{1}{e}$ fraction of the throughput.

The rest of the paper is organised as follows. In Section II, we introduce the network model. In Section III, we propose our hierarchical scheduling algorithms, and derive throughput and delay guarantees. In Section IV, we characterize the performance of our algorithms for specific network topologies like disk graphs and chordal graphs. In Section V, we provide numerical results, and conclude the paper in Section VI.

II. Network Model

Interference model: We consider a single-hop wireless network, and model the interference constraints using the widely used conflict graph model [8], [10]. A conflict graph is an undirected graph $G(V, E)$, in which each vertex corresponds to a wireless node, and two nodes share an edge if simultaneous transmissions from the nodes are not allowed. Any two nodes that share an edge are said to be conflicting. Let $\mathcal{N}_i$ denote the set of neighbours of a node i. Let N denote the total number of nodes in the network.

We consider a time slotted model. We use the binary vector $\mathbf{x} = [x_i]_{i=1}^N \in \{0,1\}^N$ to denote the transmission status (schedule) of the nodes in a given slot. Specifically, if node i is transmitting (active) in a given schedule $\mathbf{x}$, then $x_i = 1$. If node i is not transmitting in the given schedule $\mathbf{x}$, then $x_i = 0$. Further, we assume that a node can successfully transfer unit data per slot, if that node is scheduled to transmit, and none of its conflicting nodes are transmitting in that slot. Next, we define the notion of interference degree of a node.

Definition 1. *(Interference degree) Let $G_i(V_i, E_i)$ denote the graph restricted to the neighbourhood of i,* i.e., *$V_i = \{i\} \cup \mathcal{N}_i$, $E_i = \{(i,j) \in E \mid i,j \in V_i\}$. The size of the maximum independent set of the local graph $G_i(V_i, E_i)$ is called the interference degree of node i. It is denoted by t_i.*

Rate region: A schedule $\mathbf{x}$ is said to be *feasible* if no two conflicting nodes are active in that schedule. Hence, the set of feasible schedules is given by

$$\mathcal{I} := \{\mathbf{x} \in \{0,1\}^N \ : \ x_i + x_j \leq 1, \ \forall (i,j) \in E\}.$$

Further, we define the service rate of a node as the probability that a node is active in a given slot. Then the feasible rate region Λ, which is the set of all the service rates that can be supported is equal to the convex hull of $\mathcal{I}$, *i.e.*,

$$\Lambda := \Big\{ \sum_{\mathbf{x} \in \mathcal{I}} \alpha_{\mathbf{x}} \mathbf{x} : \sum_{\mathbf{x} \in \mathcal{I}} \alpha_{\mathbf{x}} = 1, \alpha_{\mathbf{x}} \geq 0, \forall \mathbf{x} \in \mathcal{I} \Big\}.$$

Queuing model: We assume that each node has a separate arrival process, and $\lambda = [\lambda_i]_{i=1}^N$ denote the arrival rates of the nodes. Each node maintains its own buffer, and $q(t) = [q_i(t)]_{i=1}^N$ denote the queue length at time slot t.

We assume that each node can estimate its own arrival rate, and accordingly set its target service rate, *i.e.*, the average rate at which it wants to transmit. Let $\{s_i\}_{i \in V}$ denote the target service rates of the nodes in the network.

Definition 2. *(Load) Given a feasible rate vector $\{s_i\}_{i=1}^N$, we define load ρ as the smallest real number in $[0,1]$ such that $\{s_i\}_{i=1}^N \in \rho\Lambda$.*

Remark: In this paper, we use e to denote $\exp(1)$.

III. Hierarchical Scheduling algorithms

In this section, we propose distributed scheduling algorithms based on hierarchical collision resolution strategies. The central idea is to assign a master-slave hierarchy for each pair of conflicting nodes, and whenever there is a possible conflict of transmission between them, the slave has to back-off transmitting from that slot. The maser-slave hierarchy is either chosen randomly, or based on the conflict graph topology, leading to different performance guarantees. If an algorithm chooses the master-slave hierarchy randomly in each time-slot, we refer it as *random ordering* based algorithm. If an algorithm fixes the hierarchy based on the graph topology, and uses the same hierarchy in each time-slot, we refer it as *fixed ordering* based algorithm.

A. Random ordering based algorithm

We now propose a random ordering based algorithm called Exp-IndSet. We assume that the time is slotted, and each time slot is divided into a control slot and a data slot. All the nodes contest for the channel in the control slot, and resolve any possible collisions using a master-slave hierarchy defined as follows: Each node $i \in V$ independently generates an exponential random variable T_i. Then for any pair $(i,j) \in E$, we define i to be a master-neighbour of j, if $T_i < T_j$.

Once the collisions are resolved[1], the nodes transmit in the data slot. The procedure is repeated to generate independent schedules in each time slot. The algorithm is described below.

Algorithm 1: *Exp-IndSet*

```
1) In the control slot, each node $i \in V$ independently generates an exponential random variable $T_i \sim \exp(s_i)$, and broadcasts to its neighbours.
2) Node $i$ will transmit in the data slot if it does not have any master-neighbours, i.e., $T_i < T_j, \forall j \in \mathcal{N}_i$.
```

Remarks on complexity: In each time slot, for computing the schedule, a node should exchange information only with its neighbours, and hence (i) the algorithm can be implemented in a distributed manner, and (ii) the complexity of generating a schedule depends only on the size of the neighbourhood. In other words, for graphs whose neighbourhood size does not scale with the network size, the complexity of Exp-IndSet algorithm is $O(1)$ with respect to the network size.

The throughput performance of Exp-IndSet is characterized in the following theorem.

Theorem 1. *Let $\{s_i\}_{i=1}^N$ be a feasible target rate vector, and let $\rho \in [0,1]$ be the load associated with it. Let $\{t_i\}_{i=1}^N$ denote the interference degrees of the nodes. Then the* Exp-IndSet *algorithm supports atleast $\frac{1}{\rho t_i}$ fraction of the target rate,* i.e.*,*

$$\mathbb{P}(\textit{node } i \textit{ is active}) \geq \frac{s_i}{\rho t_i}, \ \forall i \in V.$$

Proof. Proof is provided in Appendix A. □

From Theorem 1, it can be observed that, when the load is less than $\frac{1}{t_i}$, a node achieves its full target service rate s_i. In other words, the Exp-IndSet algorithm can support any arrival rate λ that is in $\frac{1}{\chi}$ fraction of the rate region, where $\chi := \max_i t_i$ is the maximum interference degree. This is achieved by setting the target service rate $s_i = \lambda_i$.

This result implies that our Exp-IndSet algorithm has the same throughput guarantee as that of the maximal scheduling algorithm [2], while incurring a much lower complexity. In particular, the complexity of the maximal scheduling algorithm is $O(\log N)$, while the complexity of our Exp-IndSet algorithm is $O(1)$.

Next, we present our algorithm which uses a fixed ordering, *i.e.*, the ordering of the nodes is fixed initially, and the same ordering is used in every time slot.

B. Fixed ordering based algorithm

In this section, we propose an algorithm called *Fixed-IndSet* that can be used when there is a natural ordering associated with the nodes in the network. For example, if we consider a rooted tree graph, there exists a natural ordering of the nodes based on their distance from the root node. Similar orderings can be defined for various topologies of interest such as disk-graphs and chordal graphs (Refer Section IV for more details.).

[1]CSMA algorithms [8] also employ exponential random variable based back-off counters to resolve collisions. However, the schedules of CSMA are correlated correlated across time slots, which often leads to a phenomenon called 'locking' and hence incur large delays.

Given a graph, and an ordering of the nodes in the graph, we define the terms master-neighbour and master-interference degree as follows. We refer any neighbouring node $j \in \mathcal{N}_i$, that occur before the node i in the order, as a master-neighbour of i.

Definition 3. *(Master-interference degree) Given a conflict graph $G(V,E)$, and an ordering associated with the nodes V, let $\mathcal{L}_i$ denote the set of master-neighbours of i. Let $G_i(V_i, E_i)$ denote the graph restricted to the master-neighbourhood of i,* i.e., *$V_i = \{i\} \cup \mathcal{L}_i$, $E_i = \{(i,j) \in E \mid i,j \in V_i\}$. The size of the maximum independent set of the graph $G_i(V_i,E_i)$ is called the master-interference degree of node i, and is denoted by l_i.*

We now propose an algorithm called Fixed-IndSet that assumes that the nodes are ordered, and each node knows its master-neighbours. In this algorithm, each time slot is divided into a control slot and a data slot. In the control slot, each node will contest for the channel with a probability that depends on its target service rate and master-interference degree. A node that has contested for the channel will transmit in the corresponding data slot, if none of its master-neighbours have contested for that slot. The procedure is repeated in each slot to generate independent schedules across time-slots. The algorithm is described below.

Algorithm 2: *Fixed-IndSet*

```
1) Each node will contest for the slot with probability $p_i = 1 - \exp(\frac{-s_i}{L_i})$, where $L_i := \max_{j \in \{i\} \cup \mathcal{L}_i} l_j$.
2) Node $i$ will transmit its packets in the data slot, if
    a) It has contested for that slot, and
    b) None of its master-neighbours have contested for that slot.
```

Since a node has to just communicate with its neighbours to compute their schedule, the complexity of Fixed-IndSet is $O(1)$ for bounded degree graphs. We now characterize the throughput performance of the Fixed-IndSet algorithm.

Theorem 2. *Let $\{s_i\}_{i=1}^N$ be a feasible target rate vector, and let ρ denote the load associated with it. Then the* Fixed-IndSet *guarantees at least $\frac{e^{-\rho}}{L_i}$ fraction of the target rate,* i.e.*,*

$$\mathbb{P}(\textit{node } i \textit{ is active}) \geq \frac{s_i e^{-\rho}}{L_i} \geq \frac{s_i}{eL_i}. \ \forall i \in V.$$

Proof. Proof is provided in Appendix B. □

This result implies that the Fixed-IndSet algorithm can support any arrival rate λ in $\frac{1}{e\chi_m}$ fraction of the rate region, where $\chi_m := \max_{i \in V} l_i$ is the maximum master-interference degree of the conflict graph. This is achieved by setting the target service rate $s_i = eL_i\lambda_i$, or equivalently setting the contention probability in the Fixed-IndSet algorithm as $p_i = 1 - \exp(-\lambda_i e)$.

Remarks on Implementation:

- It is noteworthy to observe that the probability $p_i = 1 - \exp(-\lambda_i e)$, with which a node has to contest for the channel, depends only on its arrival rate. In particular, the nodes do not require the knowledge of the parameter L_i or the master-interference degrees of its neighbours.

- To execute the Fixed-IndSet algorithm, each node should have the knowledge of its master-neighbours. This can be implemented as follows: We assume that each node in the network has a unique ID. Since the ordering is fixed, and remains the same throughout the time, there can be an initialization phase, where the nodes communicate, and identify their master-neighbours, and store their unique ID's. Later in each slot, when a node is requesting for channel, it should broadcast its unique ID in the control slot, which will help the neighbours to understand if it is their master. The specific control signals that are to be exchanged depend on the topology, and the ordering that is considered. This will be discussed in Section IV.

C. Delay performance

The delay of our algorithms does not scale with the network size. Specifically, we consider the queue at a node i, and prove that the expected HOL (head-of-line) delay of a packet is bounded above by a constant that is independent of the network size. Here, HOL delay is defined as the number of time-slots that a packet has to wait at the head of the queue before it starts receiving the service. The following proposition derives an upper bound on the expected HOL delay of a node under the Exp-IndSet algorithm.

Proposition 1. *For any given arrival rate $\lambda = [\lambda_i]_{i=1}^N$ that is in the interior of $\frac{1}{\chi}\Lambda$, the expected HOL delay at node i is upper bounded by $\frac{1}{\lambda_i}$.*

Proof. Let μ_i denote the probability that node i is active in a given time slot. Since Exp-IndSet can support any arrival rate λ in the interior of $\frac{1}{\chi}\Lambda$, we have $\mu_i \geq \lambda_i$. Since the Exp-IndSet algorithm (i) generates independent schedules across time slots, and (ii) the probability of a node i being active is identical in each time slot, the service received by the node i is governed by a Bernoulli process with mean μ_i.

Further, the expected HOL delay at a node i, can be upper bounded by the expected inter service time of the Bernoulli server, which is equal to $\frac{1}{\mu_i}$. Since $\mu_i \geq \lambda_i$, the expected HOL delay is upper bounded by $\frac{1}{\lambda_i}$, which is independent of the network size. $\square$

Similarly, for any arrival rate $\lambda \in \frac{1}{e\chi_m}\Lambda$, it can be proved that the HOL delay of a node under the Fixed-IndSet is upper bounded by $\frac{1}{\lambda_i}$.

Till now, we considered a general topology, and derived the throughput guarantees in terms of the interference degree and the master-interference degree of the graph. In the next section, we consider specific topologies that are of interest to wireless networks, and derive constant factor throughput guarantees.

IV. Throughput guarantees for specific network topologies

In this section, we consider specific network topologies and derive constant factor throughput guarantees. We also compare the performance of Exp-IndSet and Fixed-IndSet algorithms for these topologies. The Exp-IndSet algorithm supports $\frac{1}{\chi}$ fraction, while the Fixed-IndSet supports $\frac{1}{e\chi_m}$ fraction of the rate region. Since the master-neighbourhood is a subset of the actual neighbourhood, it is clear that $\chi_m \leq \chi$. However, it is not clear whether $e\chi_m \leq \chi$. In other words, depending upon the topology of the network, and the ordering used to define the master-neighbours, one algorithm might perform better than the other.

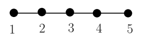

Fig. 1: A line graph

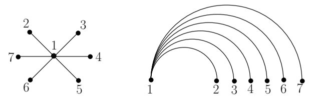

Fig. 2: A star graph, and its ordering

A. Line, Star, and Tree topologies

Line topology: Consider the line topology shown in Figure 1. For each node, we define the node that is on the left of that node as its master-neighbour. Then it is easy to see that $\chi = 2$, and $\chi_m = 1$. Hence, the Exp-IndSet algorithm supports $\frac{1}{2}$ fraction of the rate region while the Fixed-IndSet supports only a fraction of $\frac{1}{e}$.

Star topology: Consider a star topology shown in Figure 2. We define an ordering in which, the center node node is placed ahead of all the leaves (See Figure 2). Then each node has at most one master node, *i.e.*, $\chi_m = 1$. However, the maximum interference degree $\chi = n$, where n is the number of leaves. In other words, Fixed-IndSet can always guarantee $\frac{1}{e}$ fraction of the rate region, while the fraction of rate region supported by the Exp-IndSet can be arbitrarily low, as the number of leaves increases.

Tree topology: Consider a tree topology shown in Figure 3. We define an ordering as follows: One of the node is selected as a root, and the nodes are ordered based on their distance from the root node. Since the root node is the first in that

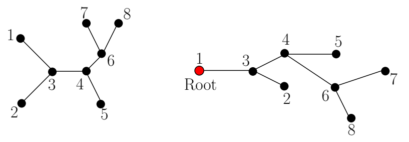

Fig. 3: A tree, and its ordering with respect to a root

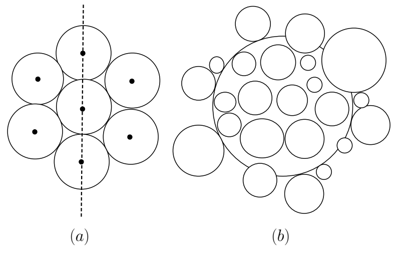

Fig. 4: (a) Homogeneous disk graphs (b) Heterogeneous disk graphs

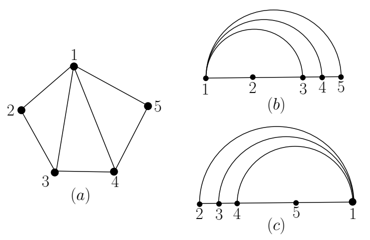

Fig. 5: A chordal graph, and discussion about perfect elimination ordering

order, it will not have any master-neighbours, and its master-interference degree is 0. Further, any other node in the network will have only one node, namely its parent node as its master-neighbour, and hence a master-interference degree of 1. In other words, the maximum master-interference degree $\chi_m = 1$, and the Fixed-IndSet can guarantee $\frac{1}{e}$ fraction of the rate region. On the other hand, it can be seen that the maximum interference degree $\chi = \Delta$, where Δ is the maximum degree of a node in the tree. Hence, the Exp-IndSet supports only $\frac{1}{\Delta}$ fraction of the rate region.

Remark on Implementation: To implement the Fixed-IndSet algorithm, the nodes in the network should elect one of the nodes as a leader (root node), and then each node should know which of its neighbours is its parent. This can be done with $O(\log N)$ time complexity using the existing distributed algorithms for leader selection [12]. It may be noted that this is a one time initialization that has to be done, and is not required to be repeated in each slot.

We now consider the disk graphs that are typically used to model the interference in wireless networks.

B. Disk graphs

Homogeneous disk graphs: In this model, the conflict graph is generated as follows: Each node in the network is associated with a location on the two-dimensional euclidean plane, and has a disk of a fixed radius R around it. Two nodes conflict with each other, if their disks overlap. Under this model, the maximum interference degree χ will be equal to the maximum number of non-overlapping circles that can overlap with a given circle. Using this observation, χ can be upper bounded by a constant 5. This is because, not more than five non-intersecting disks can overlap (or intersect) a given disk of the same radius. This fact can be inferred from Figure 4(a), which shows that, if there are six circles that intersect with a given circle, then they are bound to intersect with each other.

In the Fixed-IndSet algorithm, we define any neighbouring disk, whose center is to the left of a given disk, as its master. Then, it is easy to observe from Figure 4(a), that the maximum master-interference degree χ_m is bounded by 3. Hence, the Fixed-IndSet algorithm supports $\frac{1}{3e}$ fraction of the rate region, while the Exp-IndSet supports a slightly better fraction of $\frac{1}{5}$.

Heterogeneous disk model: This is a generalization of the homogeneous disk model, where each node $v \in V$ is associated with a disk of radius r_v around it. Two nodes $u, v \in V$ are said to conflict with each other if their corresponding disks overlap. It can be seen from Figure 4(b), that the interference degree of a node in this model, can be arbitrarily high, depending on the radii of the disks. Hence, the Exp-IndSet algorithm cannot guarantee any constant fraction of the rate region.

In the Fixed-IndSet algorithm, we define the master-slave hierarchy as follows: In any given pair of conflicting nodes, the node which has the larger disk radius is the master node. Under this hierarchy, the master-interference degree is bounded by a constant 5. This follows from the definition of the master-interference degree, and the fact that not more than 5 non-intersecting circles of radius *larger than* R, can be neighbours of a given disk of radius R. This fact can be inferred from Figure 4(a). Hence, the Fixed-IndSet algorithm can support $\frac{1}{5e}$ fraction of the rate region.

We note that this is the best known throughput guarantee for a disk graph with heterogeneous radii*, among algorithms with $O(1)$ delay and complexity.*

Remark: To implement the Fixed-IndSet algorithm in wireless networks, the nodes need to estimate the transmission power of their neighbours (which is an indication of their disk radii that is required to assign hierarchy). This can be done in the initialization phase using a neighbour discovery algorithm [13] that estimates the transmission power.

C. Chordal graphs

Recent studies on CSMA (Carrier sense multiple access) algorithms [14], [15] observed that the performance guarantees derived on chordal graphs closely match with the performance of random geometric graphs. Hence, it would be interesting to study the performance of our algorithms for chordal graphs.

A graph is said to be chordal if all of its cycles consisting of more than 3 nodes, have a chord. Here, a chord of a cycle refers to an edge joining two non-consecutive nodes of a cycle. See Figure 5(a) for an example of a chordal graph. It is known that the maximum interference degree of a chordal graph can be equal to the maximum degree of the graph[2]. Hence, the Exp-IndSet cannot guarantee any constant fraction of the rate region, if the maximum degree is unbounded. However, with an appropriate choice of master-slave hierarchy, the Fixed-IndSet algorithm can guarantee a constant fraction of the rate region, even when the degree is unbounded. We now define a property called the *perfect elimination ordering* [14], which will be used to define the ordering for Fixed-IndSet algorithm.

Definition 4. *(Perfect elimination order) An ordering of the nodes of the graph $G(V, E)$ such that, for each $v \in V$, the neighbours of v that occur before v in the order, form a clique.*

It is known that for every chordal graph, there exists a perfect elimination order of its vertices [14]. Figure 5(b) shows a perfect elimination ordering of the chordal graph shown in Figure 5(a), *i.e.*, the set of all the neighbours that occur before any given node in the ordering of Figure 5(b) form a clique. Figure 5(c) shows an ordering which is not a perfect elimination order. This is because, all the neighbours of node 1 occur before it in the order, but they do not form a clique.

In the Fixed-IndSet algorithm, if we consider the ordering given by the perfect elimination ordering, all the master-neighbours of any node form a clique. Hence, the size of the

[2]For example, in a star graph, which is a special case of a chordal graph, the maximum degree is same as the maximum interference degree.

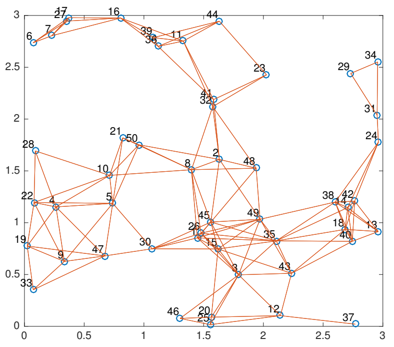

Fig. 6: A 50-node random topology based on disk-graphs with homogeneous radii

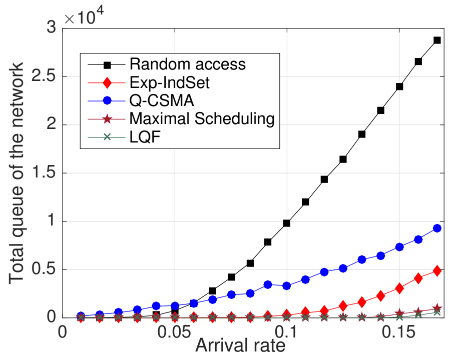

Fig. 7: Disk graph: Comparison of our Exp-IndSet algorithm with the random access algorithm [6], Q-CSMA [9], Maximal Scheduling [2], and Longest queue first algorithm [16].

maximum independent set among the master-neighbours of any node is 1, *i.e.*, $\chi_m = 1$. Hence, Fixed-IndSet can support $\frac{1}{e}$ fraction of the rate region.

Remarks on Implementation: The complexity of the initialization phase for Fixed-IndSet depends on the complexity of computing the perfect elimination ordering which is given by $O(|V| + |E|)$ complexity. A distributed algorithm for computing the perfect elimination order in a wireless network is discussed in [14].

Remark on the choice of ordering: The comparisons in this section suggest that, if there is enough symmetry in the topology like line graphs or homogeneous disk graphs, where every node has structurally similar properties, random ordering seems to be a good choice. However, when there is asymmetry in the topology like a star graph or heterogeneous disk graphs, the fixed-ordering seems to be a good choice.

V. Numerical results

In this section, we consider different network topologies and numerically evaluate the performance of our algorithms. Specifically, the topologies that we consider are: (i) disk graphs (ii) star graph (iii) trees and chordal graphs. For each topology, we verify the bounds on the achievable throughput derived in Theorem 1 and Theorem 2. Then, we numerically compare the performance of the proposed algorithms with various algorithms in the literature. Since the complexity of our Exp-IndSet and Fixed-IndSet algorithms is $O(1)$, we first compare our algorithms with (i) Q-CSMA [9] and (ii) random access protocol in [6], which also incur $O(1)$ complexity in the network size. Further, we compare with two other well-known algorithms, namely the maximal scheduling algorithm [2] that incurs $O(\log N)$ complexity, and the longest queue first (LQF) algorithm [16] that incurs a complexity of $O(N)$.

A. Disk graphs

Homogeneous disk radii: We simulate a random geometric graph to obtain a disk graph with homogeneous radii. Specifically, we consider a square of side length 3, and place 50 nodes uniformly at random. Any two nodes within unit distance are connected by an edge. A realization of this random graph is shown in Figure 6. We consider a Bernoulli arrival process and assume homogeneous arrival rates at each node. A homogeneous arrival rate of 0.2 corresponds to an upper bound on the rate that can be supported under this topology, ie, no algorithm can support an arrival rate larger than 0.2 under this topology.[3] We increase the arrival rate of the nodes till 0.2, and plot the time-average of the total queue length of all the nodes in the network after 10^6 time-slots. The queue length performance of all the considered algorithms is shown in Figure 7.

Firstly, from Figure 7, the throughput guarantee of Exp-IndSet in Theorem 1 can be verified. Specifically, the maximum interference degree for the topology in Figure 6 is 4. Hence, the Exp-IndSet should stabilize any arrival rate less than $\frac{1}{4}\times(0.2) = 0.05$. As seen from Figure 7, the queue length under the Exp-IndSet algorithm is close to zero till an arrival rate of 0.1, and hence the Exp-IndSet meets the throughput guarantee of 0.05.

Further, among all the algorithms with $O(1)$ complexity, our Exp-IndSet algorithm has a better queue length performance. In particular, our Exp-IndSet algorithm results in smaller queue lengths than the Q-CSMA algorithm[4], [9], and the random access algorithm [6]. The maximal scheduling policy, and the LQF policy which incur a larger complexity have similar queue length performance as that of our Exp-IndSet till an arrival rate of 0.1. In the high load regime, for arrival rates larger than 0.1, which are outside the guaranteed fraction of the throughput, the high complexity algorithms perform better than our Exp-IndSet algorithm.

Heterogeneous disk radii: We repeat this experiment for disk graphs with heterogeneous radii and verify the theoretical guarantees of our Exp-IndSet and Fixed-IndSet algorithms. The results can be found in our technical report [17].

B. Star graph

We now consider an example of a star topology which illustrates that for asymmetric topologies, the Fixed-IndSet algorithm might be a good choice. Specifically, in a star

[3]This is an upper bound on the maximum achievable rate vector, which can be computed from the degrees and the interference degrees of the nodes using (2).

[4]Although Q-CSMA algorithm is technically a throughput maximizing policy, the algorithm does not converge for practical time scales and results in larger queue lengths in the shown plots.

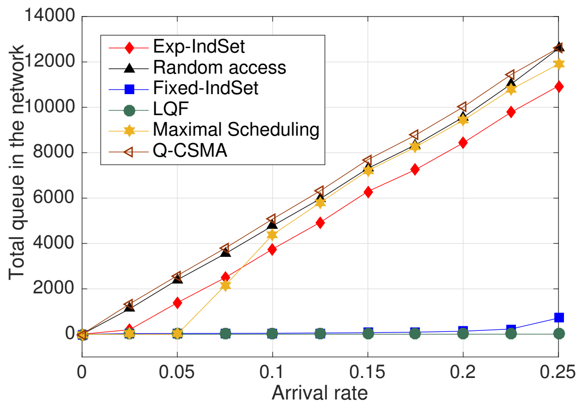

Fig. 8: Performance of various algorithms for a star topology with 50 nodes

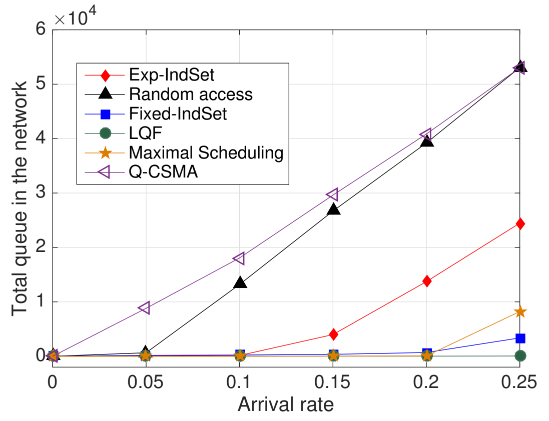

Fig. 9: Performance of various algorithms for a 6-ary tree with 259 nodes

graph, we demonstrate that the Fixed-IndSet algorithm not only performs better than the existing algorithms of $O(1)$ complexity, but it also outperforms the maximal scheduling algorithm that has a complexity of $O(\log N)$. The specific details of the simulations are described below.

We now consider a star network with 50 nodes, and first verify the throughput guarantees that we derived, *i.e.*, the Fixed-IndSet algorithm supports $\frac{1}{e}$ fraction of the full throughput, and the Exp-IndSet algorithm supports $\frac{1}{N}$ fraction of the full throughput. Under this topology, a homogeneous arrival rate of 0.5 corresponds to the maximum throughput, and hence arrival rates of 0.01 and 0.18 correspond to the throughput guarantees of Exp-IndSet and Fixed-IndSet respectively. As seen from Figure 8, under the Exp-IndSet algorithm the queue length is close to zero till an arrival rate of 0.02. Under the Fixed-IndSet algorithm, the queue length is close to zero till an arrival rate of 0.2. Hence both our algorithms comfortably meet their respective throughput guarantees.

Further, it can be observed from Figure 8 that for all the algorithms except our Fixed-IndSet and the LQF algorithm, there is a steep increase in the queue lengths for arrival rates larger than 0.05. In particular, the Fixed-IndSet algorithm results in queue lengths close to zero till an arrival rate of 0.2, and thereby outperforms the maximal scheduling algorithm that has $O(\log N)$ complexity. Further, it can be observed that the performance of our Fixed-IndSet algorithm closely matches the performance of LQF algorithm which has a larger complexity of $O(N)$.

C. *Trees and chordal graphs*

In Figure 9, we illustrate the results obtained for a large 259 node tree topology (6-ary tree of height 4). Similar to the star topology, our Fixed-IndSet algorithm outperforms all the $O(1)$ complexity algorithms and the $O(\log N)$ complexity maximal scheduling algorithm.

The results for chordal graphs are qualitatively similar, and can be found in our technical report [17].

VI. Concluding remarks and Future work

We proposed distributed scheduling algorithms that have $O(1)$ delay and complexity in the network size, and guarantee a constant fraction of the maximum throughput for typical wireless topologies such as disk graphs. Our algorithms resolve collisions among pairs of conflicting nodes by assigning a master-slave hierarchy. When the master-slave hierarchy is chosen randomly, our algorithm has the same throughput guarantee as that of the maximal scheduling policies, with a complexity and delay that do not scale with network size. When the master-slave hierarchy is chosen based on the network topology, our algorithm guarantees a fraction of the throughput that is at most off by a factor of *master-interference* degree. For commonly used conflict graph topologies, our results lead to the best known throughput guarantees among the algorithms that have $O(1)$ delay and complexity. Numerical results indicate that our algorithms outperform the existing $O(1)$ complexity algorithms like Q-CSMA. Further for topologies like tree graphs, which have an inherent hierarchy, our Fixed-IndSet algorithm outperforms the maximal scheduling algorithm which has a higher complexity of $O(\log N)$.

Appendix

A. *Proof of Theorem 1*

The probability that a node i is active in a slot is given by

$$\begin{aligned}\mathbb{P}(\text{node } i \text{ is active}) &= \mathbb{P}(T_i < T_j, \forall j \in \mathcal{N}_i),\\ &\overset{(a)}{=} \frac{s_i}{s_i + \sum_{j\in\mathcal{N}_i} s_j},\end{aligned} \tag{1}$$

where (a) follows from the properties of the exponential distribution.

Recall that the interference degree is defined as the size of the maximum independent set in the neighbourhood graph $G_i(V_i, E_i)$. Since any schedule generated by Exp-IndSet algorithm is an independent set, no more than t_i nodes in that neighbourhood can transmit in a given slot. Hence, for any feasible rate vector $\{\mu_i\}_{i=1}^N \in \Lambda$, the sum of the rates over the local neighbourhood cannot be more than t_i, *i.e.*,

$$\mu_i + \sum_{j\in\mathcal{N}_i} \mu_j \le t_i, \quad \forall i. \tag{2}$$

Since ρ is the load associated with the target rate vector, we have $\frac{1}{\rho}\{s_i\}_{i=1}^{N} \in \Lambda$. Then from (2), we obtain

$$\frac{1}{\rho}(s_i + \sum_{j \in \mathcal{N}_i} s_j) \leq t_i. \tag{3}$$

Substituting (3) in (1) completes the proof of this theorem.

B. Proof of Theorem 2

Recall that the master-interference degree is defined as the size of the maximum independent set in the master-neighbourhood graph. Since any schedule generated by the *Fixed-IndSet* algorithm is an independent set, no more than l_i nodes in the master-neighbourhood can transmit in a given slot. Hence, for any feasible rate vector $\{\mu_i\}_{i=1}^{N} \in \Lambda$, we have

$$\mu_i + \sum_{j \in \mathcal{L}_i} \mu_j \leq l_i. \tag{4}$$

Since ρ is the load associated with the target rate vector, we have $\frac{1}{\rho}\{s_i\}_{i=1}^{N} \in \Lambda$. Using this fact along with (4), we obtain the constraints

$$\begin{aligned} &0 \leq s_i \leq \rho, \;\; i = 1, \ldots, N, \\ &s_i + \sum_{j \in \mathcal{L}_i} s_j \leq \rho l_i, \;\; i = i, \ldots, N. \end{aligned} \tag{5}$$

The probability that a node is active is given by

$$\begin{aligned} &\mathbb{P}(\text{node } i \text{ is active}) \\ &= \left(1 - \exp\left(-\frac{s_i}{L_i}\right)\right) \prod_{j \in \mathcal{L}_i} \exp\left(-\frac{s_j}{L_j}\right), \\ &\overset{(a)}{\geq} \left(1 - \exp\left(-\frac{s_i}{L_i}\right)\right) \prod_{j \in \mathcal{L}_i} \exp\left(-\frac{s_j}{l_i}\right), \\ &= \left(1 - \exp\left(-\frac{s_i}{L_i}\right)\right) \exp\left(-\sum_{j \in \mathcal{L}_i} \frac{s_j}{l_i}\right), \end{aligned} \tag{6}$$

where the inequality (a) follows from the definition of L_j which implies $L_j \geq l_i$. Now using the observation (5) in (6), we have

$$\begin{aligned} &\mathbb{P}(\text{node } i \text{ is active}) \\ &\overset{(b)}{\geq} \left(1 - \exp\left(-\frac{s_i}{L_i}\right)\right) \exp\left(\frac{s_i}{l_i} - \rho\right), \\ &\overset{(c)}{\geq} \left(1 - \exp\left(-\frac{s_i}{L_i}\right)\right) \exp\left(\frac{s_i}{L_i} - \rho\right), \\ &= \left(\exp\left(\frac{s_i}{L_i}\right) - 1\right) e^{-\rho}, \\ &\overset{(d)}{\geq} \frac{s_i}{L_i} e^{-\rho}. \end{aligned}$$

The inequality (b) follows from the observation in (5). The inequality (c) follows from the fact that $L_i \geq l_i$. The inequality (d) follows since $e^x \geq 1 + x$.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 2, p. 572, 2008.

[3] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 4, pp. 1132–1145, 2009.

[4] X. Wu, R. Srikant, and J. R. Perkins, "Queue-length stability of maximal greedy schedules in wireless networks," in *Proceedings of Information Theory and Applications Inaugural Workshop*, 2006, pp. 6–10.

[5] X. Lin and S. B. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 231–242, 2009.

[6] D. Chafekar, D. Levin, V. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Capacity of asynchronous random-access scheduling in wireless networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 2008, pp. 1148–1156.

[7] P. Bremaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues.* springer, 1999, vol. 31.

[8] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 3, pp. 960–972, 2010.

[9] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 3, pp. 825–836, 2012.

[10] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand, "Fast mixing of parallel Glauber dynamics and low-delay CSMA scheduling," *Information Theory, IEEE Transactions on*, vol. 58, no. 10, pp. 6541–6555, 2012.

[11] B. Li and A. Eryilmaz, "Optimal distributed scheduling under time-varying conditions: A fast-CSMA algorithm with applications," *IEEE Transactions on Wireless Communications*, vol. 12, no. 7, pp. 3278–3288, July 2013.

[12] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics.* John Wiley & Sons, 2004, vol. 19.

[13] J. Luo and D. Guo, "Compressed neighbor discovery for wireless ad hoc networks: the Rayleigh fading case," in *Communication, Control, and Computing (Allerton), 2009 47th Annual Allerton Conference on.* IEEE, 2009, pp. 308–313.

[14] B. Van Houdt, "Explicit back-off rates for achieving target throughputs in CSMA/CA networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 765–778, April 2017.

[15] P. S. Swamy, B. V. Pavan Kumar, R. K. Ganti, and K. Jagannathan, "Efficient CSMA based on Kikuchi approximation," in *IEEE International Conference on Signal Processing and Communications (SPCOM)*, June 2016, pp. 1–5.

[16] B. Birand, M. Chudnovsky, B. Ries, P. Seymour, G. Zussman, and Y. Zwols, "Analyzing the performance of greedy maximal scheduling via local pooling and graph theory," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 163–176, 2012.

[17] "Hierarchical scheduling algorithms with throughput guarantees and low delay," Tech. Rep., 2017. [Online]. Available: https://www.dropbox.com/s/1s5r8ifkj7p4pgj/techreport.pdf?dl=0