Analyzing the Performance of Greedy Maximal Scheduling via Local Pooling and Graph Theory

Berk Birand*, Maria Chudnovsky†, Bernard Ries†, Paul Seymour‡, Gil Zussman* and Yori Zwols†
*Department of Electrical Engineering
†Department of Industrial Engineering and Operations Research
Columbia University, New York, NY 10027
‡Department of Mathematics
Princeton University, Princeton, NJ

***Abstract*—Efficient operation of wireless networks and switches requires using simple (and in some cases distributed) scheduling algorithms. In general, simple greedy algorithms (known as Greedy Maximal Scheduling - GMS) are guaranteed to achieve only a fraction of the maximum possible throughput (e.g., 50% throughput in switches). However, it was recently shown that in networks in which the *Local Pooling* conditions are satisfied, GMS achieves 100% throughput. Moreover, in networks in which the σ-Local Pooling conditions hold, GMS achieves $\sigma\%$ throughput. In this paper, we focus on identifying the specific network topologies that satisfy these conditions. In particular, we provide the first characterization of *all the network graphs in which Local Pooling holds under primary interference constraints* (in these networks GMS achieves 100% throughput). This leads to a polynomial time algorithm for identifying Local Pooling-satisfying graphs. Moreover, by using similar graph theoretical methods, we show that *in all bipartite graphs (i.e., input-queued switches) of size up to $7 \times n$, GMS is guaranteed to achieve 66% throughput*, thereby improving upon the previously known 50% lower bound. Finally, we study the performance of GMS in interference graphs and show that in certain specific topologies its performance is very bad. Overall, the paper demonstrates that using graph theoretical techniques can significantly contribute to our understanding of greedy scheduling algorithms.**

***Index Terms*—Local pooling, scheduling, throughput maximization, graph theory, wireless networks, switches.**

I. Introduction

The effective operation of wireless and wireline networks relies on the proper solution of the packet scheduling problem. In wireless networks, the main challenge stems from the need for a decentralized solution to a centralized problem. Even when centralized processing is possible, as is the case in input-queued switches, designing low complexity algorithms that will enable efficient operation is a major challenge.

A centralized joint routing and scheduling policy that achieves the maximum attainable throughput region was presented by Tassiulas and Ephremides [27]. That policy applies to a multihop network with a stochastic packet arrival process and is guaranteed to stabilize the network whenever the arrival rates are within the stability region (i.e., it provides 100% throughput). The results of [27] have been extended to various settings of wireless networks and input-queued switches (e.g., [1], [10], [23]). However, throughput-optimal algorithms based on [27] require the repeated solution of a *global optimization problem*, taking into account the queue backlog information for every link. For example, even under simple primary interference constraints[1], a maximum weight matching problem has to be solved in every slot, requiring an $O(n^3)$ algorithm.

Hence, there has been an increasing interest in simple (potentially distributed) algorithms. One such algorithm is the *Greedy Maximal Scheduling* (GMS) algorithm (also termed Maximal Weight Scheduling or Longest Queue First - LQF). This algorithm selects the set of served links greedily according to the queue lengths [13], [21]. Namely, at each step, the algorithm selects the heaviest link (i.e., with longest queue size), and removes it and the links with which it interferes from the list of candidate links. The algorithm terminates when there are no more candidate links. Such an algorithm can be implemented in a distributed manner [13], [18], [19].

It was shown that the GMS algorithm is guaranteed to achieve 50% throughput in switches [8] and in general graphs under primary interference constraints [21]. Moreover, it was proved in [6] that under secondary interference constraints[2] the throughput obtained by GMS may be significantly smaller than the throughput under a centralized (optimal) scheduler (e.g., $1/8$ of the possible throughput).

Although in *arbitrary topologies* the worst case performance of GMS can be very low, there are some topologies in which 100% *throughput is achieved*. Particularly, Dimakis and Walrand [9] presented sufficient conditions for GMS to provide 100% throughput. These conditions are referred to as *Local Pooling* (LoP) and are related to the structure of the graph. Based on these conditions, it was shown that GMS achieves maximum throughput in tree network graphs under k-hop interference (for any k) [17], [30], in $2 \times n$ switches [5], and in a number of interference graph classes [30].

The LoP conditions were recently generalized to provide

This work was partially supported by NSF grants DMS-0758364 and CNS-0916263, FNR grant TR-PDR BFR08-17, and ONR grant N00014-01-1-0608.

[1]Primary interference constraints imply that each pair of simultaneously active links must be separated by at least one hop (i.e., the set of active links at any point of time constitutes a matching).

[2]Secondary interference constraints imply that each pair of simultaneously active links must be separated by at least two hops (links). These constraints are usually used to model IEEE 802.11 networks [6].

the σ-Local Pooling (σ-LoP) conditions under which GMS achieves $\sigma\%$ throughput [16], [17] (the conditions were reformulated and refined in [20]). Using these conditions, lower bounds on the guaranteed throughput in geometric graphs [17] and in graphs under secondary interference constraints were obtained [19]. Moreover, it was shown in [19] that GMS achieves 100% in all graphs with up to 8 nodes under the secondary interference constraints.

From a practical point of view, identifying graphs that satisfy LoP and σ-LoP can provide important building blocks for partitioning a network (e.g., via channel allocation) into subnetworks in which GMS performs well [5]. Another possible application is to add artificial interference constraints to a graph that does not satisfy the LoP conditions in order to turn it into a LoP-satisfying graph. Adding such constraints may decrease the stability region but would enable GMS to achieve a large portion of the new stability region.

While it is known that under primary interference some graph families (mainly trees and $2 \times n$ bipartite graphs) satisfy LoP, the exact structure of networks that satisfy LoP was not characterized. In this paper, we use graph theoretic methods to *obtain the structure of* all *the network graphs that satisfy LoP under primary interference constraints* (in these networks GMS achieves 100% throughput). This allows us to develop an algorithm that checks if a network graph satisfies LoP in $O(n)$, significantly improving over any other known method (the best known method required solving an NP-Complete problem). We note that although primary interference constraints may not hold in many wireless networking technologies, the characterization provides an important theoretical understanding regarding the performance of simple greedy algorithms. It also shows that the $2 \times n$ switch is the largest switch for which 100% throughput is guaranteed.

We then focus on graphs in which GMS does not achieve 100% throughout. We first consider bipartite network graphs (i.e., input-queued switches) and, in particular, show that *for bipartite graphs of size $k \times n$, where $k \leq 7$ and n is not bounded, GMS achieves at least 66% throughput*. Namely, for switches with up to 7 inputs or 7 outputs, the throughput under GMS is lower bounded by 66%. This significantly improves upon the well known 50% lower bound [8] and confirms many simulation studies (e.g., [11]) in which it was shown that greedy algorithms perform relatively well in switches. To show that this result does not extend to all bipartite graphs, we show that there exists a 10×10 bipartite graph for which $\sigma = 0.6$.

Finally, we consider interference graphs[3] and categorize different graph families according to their σ values. In particular, we show that all co-strongly perfect graphs satisfy LoP. This class encapsulates all the classes of perfect LoP-satisfying interference graphs that were identified before (i.e., interference graphs of trees, chordal graphs, etc.). The observation increases the number of graphs known to satisfy LoP by an order of magnitude. Regarding σ-LoP we show that there are graphs with arbitrarily low σ. Since the worst case specific graph identified up to now had $\sigma = 0.6$ [16] and the lowest lower bound known for a graph family was 1/6 [17], [19], this provides an important insight regarding graphs in which GMS may have bad performance. We conclude with describing a simulation study in which we compared the performance of GMS to the optimal algorithm in graphs with low σ.

To conclude, the main contributions of this paper are two-fold: (i) a characterization of *all* network graphs in which Local Pooling holds under primary interference constraints (in these network graphs Greedy Maximal Scheduling is guaranteed to achieve 100% throughput) and (ii) improved lower bounds on the throughput performance of Greedy Maximal Scheduling in small switches. Overall, the paper demonstrates that using graph theoretical techniques can significantly contribute to our understanding of greedy scheduling algorithms.

This paper is organized as follows. In Section II we present the model. We characterize all graphs that satisfy LoP under primary interference constraints in Section III. In Section IV we show that GMS achieves 66% throughput in switches with up to 7 inputs. We study the performance of GMS in interference graphs in Section V and we conclude and discuss open problems in Section VI. Due to space constraints, some of the proofs are omitted and can be found in [2].

II. Model and Definitions

In this section we first present definitions of our network model under primary interference, and then extend them for general interference graphs.

A. Network Graphs

Consider a *network graph* $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes, and $E \subseteq \{ij : i, j \in V, i \neq j\}$ is a set of links indicating pairs of nodes between which data flow can occur.

Following the model of [5], [9], [16], [27], assume that time is slotted and that packets are of equal size, each packet requiring one time slot of service across a link. The model considers only single-hop traffic. A queue is associated with each edge in the network. We assume that the stochastic arrivals to edge ij have long term rates λ_{ij} and are independent of each other. We denote by $\vec{\boldsymbol{\lambda}}$ the vector of the arrival rates λ_{ij} for every edge ij. For more details regarding the queue evolution process under this model, see [5], [9], [16].

For a graph G, let $\mathbf{M}(G)$ be a 0-1 matrix with $|E(G)|$ rows, whose columns represent the maximal matchings of G. A *scheduling algorithm* selects a set of edges to activate at each time slot, and transmits packets on those edges. Since they must not interfere under primary interference constraints, the selected edges form a matching. In other words, the scheduling algorithm picks a column $\boldsymbol{\pi}(t)$ from the maximal matching matrix $\mathbf{M}(G)$ at every time slot t. If $\boldsymbol{\pi}_k(t) = 1$, one of the two nodes along edge e_k can transmit, and the associated queue is

[3]Although it has been recently shown that in some cases the interference graph does not fully capture the wireless interference characteristics [24], it still provides a reasonable abstraction. Extending the results to general SINR-based constraints is a subject for further research.

decreased by one. We define the *stability region* (or *capacity region*) of a network as follows.

Definition 2.1 (Stability region [27]): The stability region of a network G_N is defined by

$$\Lambda^* = \left\{ \vec{\boldsymbol{\lambda}} \mid \vec{\boldsymbol{\lambda}} < \vec{u} \ \text{ for some } \vec{u} \in Co(\mathbf{M}(G)), \right\},$$

where $Co(\mathbf{M}(G))$ is the convex hull of the columns of $\mathbf{M}(G)$ (inequality operators are taken element-wise when their operands are vectors).

A *stable* scheduling algorithm (which we also refer to as a *throughput-optimal* algorithm or an algorithm that *achieves* 100% *throughput*) is defined as an algorithm for which the Markov chain that represents the evolution of the queues is positive recurrent for all arrivals $\vec{\boldsymbol{\lambda}} \in \Lambda^*$. It was shown in [27] that the Maximum Weight Matching algorithm that selects the matching with the largest total queue sizes at each slot is stable. The *efficiency ratio* γ^* of an algorithm indicates the fraction of the stability region for which the algorithm is stable (in simple words, the queues are bounded for all arrival rates $\vec{\boldsymbol{\lambda}} \in \gamma^* \Lambda^*$).

We briefly reproduce the definitions of Local Pooling (LoP) presented in [5], [9].[4] In the following, $\mathbf{e}$ denotes the vector having each entry equal to one.

Definition 2.2 (Subgraph Local Pooling - SLoP): A network graph G satisfies SLoP, if there exists $\boldsymbol{\alpha} \in [0,1]^{|E|}$ such that $\boldsymbol{\alpha}^T \mathbf{M}(G) = \mathbf{e}^T$.

This definition corresponds also to associating a weight, denoted $\boldsymbol{\alpha}(e)$, to all edges $e \in E$, such that

$$\sum_{e \in Z} \boldsymbol{\alpha}(e) = 1 \text{ for every maximal matching } Z \text{ in } G.$$

If a vector $\boldsymbol{\alpha}$ satisfies the above condition, we will say that it is a *good edge weighting*.

Definition 2.3 (Overall Local Pooling - OLoP): A network graph G satisfies OLoP, if every subgraph S of G satisfies SLoP.

In [9], Dimakis and Walrand proved that if a graph satisfies OLoP, GMS achieves 100% throughput. In networks in which OLoP is not satisfied, σ-Local Pooling [16], [17] provides a way of estimating the efficiency ratio γ^* of GMS. Below, we provide a different definition called σ-SLoP that is equivalent to the original one from [16], [17].

Definition 2.4 (σ-SLoP - Xi et. al. [20]): A network graph G satisfies σ-SLoP, if and only if there exists a vector $\boldsymbol{\alpha} \in [0,1]^{|E|}$ such that

$$\sigma \mathbf{e}^T \leq \boldsymbol{\alpha}^T \mathbf{M}(G) \leq \mathbf{e}^T.$$

This definition can also be written in the following form:

$$\sigma \leq \sum_{e \in Z} \boldsymbol{\alpha}(e) \leq 1 \text{ for every maximal matching } Z \text{ in } G.$$

Clearly, if a graph satisfies σ-SLoP, it also satisfies σ'-SLoP for every $\sigma' < \sigma$. Therefore, it is sufficient to focus on the largest value of σ such that G satisfies σ-SLoP. This value is denoted by $\sigma(G)$:

$$\sigma(G) := \max \{ \sigma \mid G \text{ satisfies } \sigma\text{-SLoP} \}. \tag{1}$$

This definition can also be written as a Linear Program whose solution yields the $\sigma(G)$ for a given graph G [20].

$$\begin{aligned} \sigma(G) = \quad & \max \ \sigma \\ & \text{subject to } \sigma \mathbf{e}^T \leq \boldsymbol{\alpha}^T \mathbf{M}(G) \leq \mathbf{e}^T. \end{aligned} \tag{2}$$

We say that a graph satisfies σ-OLoP if all of its subgraphs satisfy σ-SLoP. We can then define the local pooling factor σ^* as follows:

Definition 2.5 (Joo et. al. [16]): The local pooling factor $\sigma^*(G)$ of a network graph G is the smallest value of σ for which σ-SLoP is satisfied for all subgraphs.

This definition can also be written in terms of $\sigma(G)$:

$$\sigma^*(G) := \min \{ \sigma(S) \mid \text{ for all subgraphs } S \text{ of } G \}. \tag{3}$$

It was proved in [16] that the local pooling factor σ^* of a graph is equal to the efficiency ratio γ^* of GMS in that graph. For instance, if a graph has a local-pooling factor of $2/3$, GMS is stable for all arrival rates $\vec{\boldsymbol{\lambda}} \in \frac{2}{3}\Lambda^*$ and therefore achieves 66% throughput. Note that $\sigma^*(G) = 1$, if and only if G satisfies the OLoP condition.

In the following, we use the above definitions to present a non-trivial lower-bound on $\sigma(G)$ [19]. Define

$$\nu(G) = \max\{|Z| : Z \text{ is a maximal matching in } G\},$$
$$\mu(G) = \min\{|Z| : Z \text{ is a maximal matching in } G\}.$$

Lemma 2.1 (Leconte et al. [19]): For any graph G,

$$\sigma(G) \geq \mu(G)/\nu(G).$$

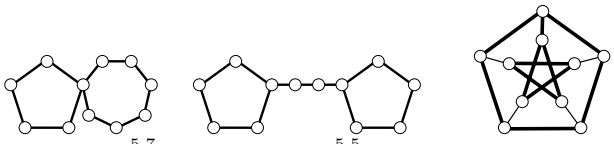
Using Definition 2.4, we provide an alternative proof to this lemma in [2].

To demonstrate the benefits of the σ-OLoP definition, we provide a very simple proof to the fact that GMS achieves 50% throughput in any network graph G (shown in different methods in [8], [21]). First, note that the size of any maximal matching is at least half the size of a maximum matching [25], which means that $\mu(G) \geq \nu(G)/2$, for all G. By Lemma 2.1 and (3), it follows that $\sigma^*(G) \geq 1/2$ for every graph G, and therefore, that $\gamma^* \geq 1/2$.

B. Interference Graphs

We now generalize the model by introducing interference graphs. Based on the network graph and the interference constraints, the interference between network links can be modeled by an *interference graph* (or a *conflict graph*) $G_I = (V_I, E_I)$ [15]. We assign $V_I = E$. Thus, each edge e_k in the network graph is represented by a node v_k in the interference graph, and an edge $v_i v_j$ in the interference graph indicates a conflict between network graph links e_i and e_j (i.e., transmissions on e_i and e_j cannot take place simultaneously). Under primary interference, the interference graph G_I corresponds to the *line graph* of G_N. When network

[4] This definition slightly differs from that in [5] by setting the sum equal to $\mathbf{e}^T$ instead of $c\mathbf{e}^T$, where c is a positive constant.



(a) $D_0^{5,7}$ (b) $D_3^{5,5}$ (c) Petersen graph

Fig. 1. Graphs (a) and (b): examples of graphs from the family $D_k^{p,q}$, all of which fail OLoP under primary interference. Graph (c): the Petersen graph. This graph does not satisfy OLoP [16] because it contains, among other graphs, C_6 and $D_1^{5,5}$ (bold edges) as subgraphs.

and interference graphs are used concurrently, we will use G_N for the former in order to emphasize their distinction.

The model and the LoP theory described so far extend to interference graphs. The nodes of G_I correspond to queues to which packets arrive according to a stochastic process at every time slot. A scheduling algorithm must pick an independent set at each slot so that neighboring nodes will not be activated simultaneously. Each column of the matrix $\mathbf{M}(G_I)$ corresponds to a maximal independent set of G_I. An algorithm which selects the independent set with the largest weights (i.e., solves the Maximum Weight Independent Set) is stable. SLoP corresponds to finding a vector $\boldsymbol{\alpha} \in [0,1]^{|V_I|}$ that assigns a weight $\boldsymbol{\alpha}(u)$ to each node u such that $\sum_{u \in I} \boldsymbol{\alpha}(u) = 1$ for every maximal independent set I in G_I. If such a vector exists, we will call it a *good node weighting*. For OLoP to be satisfied, SLoP must be satisfied by all *induced* subgraphs (i.e., with respect to node removals). σ-SLoP, σ-OLoP and Lemma 2.1 extend to this case in a very similar way.

III. Network graphs that satisfy OLoP under primary interference

Only a small collection of network graphs have been shown to satisfy OLoP under primary interference. Among the known cases are trees [5], [17], and $2 \times n$ bipartite graphs [5]. *The main result of this section is a description of the structure of all network graphs that satisfy OLoP under primary interference. This structure shows that such graphs are relatively easy to construct and, moreover, they can be recognized in linear time.*

Define the following families of graphs. For $k \geq 3$, let C_k be a cycle with k edges (or, equivalently, k nodes). For $k \geq 0$ and $p, q \in \{5, 7\}$, let $D_k^{p,q}$ be the graph formed by the union of two cycles of size p and q joined by a k-edge path (where $k \geq 0$). If $k = 0$, the cycles share a common node (see Fig. 1-(a) and 1-(b)). Let $\mathcal{F} = \{C_k \mid k \geq 6, k \neq 7\} \cup \{D_k^{p,q} \mid k \geq 0; p, q \in \{5, 7\}\}$. For two graphs G and H, we say that G contains H as a subgraph if G has a subgraph that is isomorphic to H. We will say that a graph G is *$\mathcal{F}$-free* if it does not contain any graph $F \in \mathcal{F}$ as a subgraph.

The results in this section are three-fold. First, in Subsection III-B, we will give a structural description of all $\mathcal{F}$-free graphs. Second, in Subsection III-C, we will use this description to prove the following theorem:

Theorem 3.1: A network graph G satisfies OLoP under primary interference if and only if G is $\mathcal{F}$-free.

Theorem 3.1 shows that if a network graph G does not satisfy OLoP under primary interference, then G contains some $F \in \mathcal{F}$ as a subgraph. For example, it was previously known that the Petersen graph (Fig. 1-(c)) fails OLoP [16]. Using Theorem 3.1 we can immediately see this from the fact that it contains, for example, C_6 and $D_1^{5,5}$ as a subgraph.

Testing whether a network graph satisfies SLoP previously required enumerating all maximal matchings (of which there are an exponential number) and solving a Linear Program [9]. To test the OLoP condition, this procedure had to be repeated for every subgraph. A weakness of that approach is that it does not give any insight into the reason why a network fails OLoP. Theorem 3.1 allows us to determine whether a given graph satisfies OLoP by solely examining its structural properties.

Another weakness of the Linear Programming approach is its large computational effort. In Subsection III-D, we present the third result, which uses the structure of $\mathcal{F}$-free graphs to construct an algorithm that decides in linear time whether a graph satisfies OLoP, as described in the following theorem:

Theorem 3.2: It can be decided in $O(|V(G)|)$ time whether a network graph G satisfies OLoP under primary interference.

A. Definitions and notation

We will need some definitions, notation and facts from graph theory. For details, we refer to [12], [28].

Let G be a graph. For $v \in V(G)$, we will write $N(v)$ for the set of all nodes in $V(G)$ that are incident with v. Let $\deg(v) = |N(v)|$ denote the *degree of* v. For $x, y \in V(G)$, we say that x is *a clone of* y if $N(x) = N(y)$. For $x \in V(G)$, we denote by $G - x$ the graph obtained from G by deleting x and all edges incident with it.

We say that G is connected if there exists a path in G between every two distinct nodes u and v. A *connected component of* G is a maximal connected induced subgraph of G. We will focus on connected graphs, because it is easy to see that a graph satisfies OLoP if and only if all its connected components satisfy OLoP. So we may assume without loss of generality that all graphs in this section are connected graphs.

Finally, for $n \geq 1$, we let K_n denote the complete graph on n nodes. For $n \geq 1, t \geq 1$, we let $K_{t,n}$ denote the $t \times n$ complete bipartite graph.

B. The structure of $\mathcal{F}$-free graphs

We will start with a structural description of $\mathcal{F}$-free graphs. The reason for our interest in $\mathcal{F}$-free graphs is the fact (which will be shown in Subsection III-C) that the class of $\mathcal{F}$-free graphs is precisely the class of network graphs that satisfy OLoP under primary interference.

We will describe the structure of $\mathcal{F}$-free graphs in terms of the so-called 'block decomposition'. Let G be a connected graph. We call $x \in V(G)$ a *cut-node of* G if $G - x$ is not connected. We call a maximal connected induced subgraph B of G such that B has no cut-node a *block of* G. Let $B_1, B_2, \ldots, B_q$ be the blocks of G. We call the collection $\{B_1, B_2, \ldots, B_q\}$ the *block decomposition of* G. It is known that the block decomposion is unique and that $E(B_1), E(B_2), \ldots, E(B_q)$ forms a partition of $E(G)$ (e.g., [28]). Furthermore, the node

sets of every two blocks intersect in at most one node and this node is a cut-node of G.

Block decompositions give a tree-like decomposition of a graph in the following sense. Construct the *block-cutpoint graph of* G by keeping the cut-nodes of G and replacing each block B_i of G by a node b_i. Make each cut-node v adjacent to b_i if and only if $v \in V(B_i)$. It is known that the block-cutpoint graph of G forms a tree (e.g., [28]). With this tree-like structure in mind, we say that a block B_i is a *leaf block* if it contains at most one cut-node of G. Clearly, if $q \geq 2$, then $\{B_i\}_{i=1}^{q}$ contains at least two leaf blocks.

It turns out that the block decomposition of an $\mathcal{F}$-free graph is relatively simple in the sense that there are only two types of blocks. The types are defined by the following two families of graphs. Examples of these families appear in Fig. 2.

$\mathcal{B}_1$: Construct $\mathcal{B}_1$ as follows. Let H be a graph with $V(H) = \{c_1, c_2, \ldots, c_k\}$, with $k \in \{5, 7\}$, such that
1) c_1-c_2-$\cdots$-c_k-c_1 is a cycle;
2) if $k = 5$, then the other adjacencies are arbitrary; if $k = 7$, then all other pairs are non-adjacent, except possibly $\{c_1, c_4\}$, $\{c_1, c_5\}$ and $\{c_4, c_7\}$.

Then, $H \in \mathcal{B}_1$.

Now iteratively perform the following operation. Let $H' \in \mathcal{B}_1$ and let $x \in V(H')$ with $\deg(x) = 2$. Construct H'' from H' by adding a node x' such that $N(x') = N(x)$. Then, $H'' \in \mathcal{B}_1$. We say that a graph is *of the* $\mathcal{B}_1$ *type* if it is isomorphic to a graph in $\mathcal{B}_1$.

$\mathcal{B}_2$: Let $\mathcal{B}_2 = \{K_2, K_3, K_4\} \cup \{K_{2,t}, K_{2,t}^{+} \mid t \geq 2\}$, where $K_{2,t}^{+}$ is constructed from $K_{2,t}$ by adding an edge between the two nodes on the side that has cardinality 2. We say that a graph is *of the* $\mathcal{B}_2$ *type*, if it is isomorphic to a graph in $\mathcal{B}_2$.

In plain English, graphs of the $\mathcal{B}_1$ type are constructed by starting with a cycle of length five or seven. Then we may add some additional edges between nodes of the cycle, subject to some constraints. Finally, we may iteratively take a node x of degree 2 and add a clone x' of x. It will turn out that $\mathcal{F}$-free graphs have at most one block of the $\mathcal{B}_1$ type, and all other blocks are of the $\mathcal{B}_2$ type. This means that $\mathcal{F}$-free graphs can be constructed by starting with a block that is either of the $\mathcal{B}_1$ or of the $\mathcal{B}_2$ type, and then iteratively adding a block of the $\mathcal{B}_2$ type by 'glueing' it on an arbitrary node.

Fig. 2 shows an example of an $\mathcal{F}$-free graph. The tree-like structure is clearly visible. The graph has one block of the $\mathcal{B}_1$ type with $k = 7$. Clearly, this block consists of a cycle of length 7 together with two clones. The other blocks are of the $\mathcal{B}_2$ type. Some of them are attached to the block of the $\mathcal{B}_1$ type through a cut-node, others are attached to other blocks of the $\mathcal{B}_1$ type. Notice that trees and $2 \times n$ complete bipartite graphs, which were previously known to satisfy OLoP [5], [17], are, as should be expected, subsumed by this structure.

The goal of this subsection is to prove the following formal version of the characterization given above:

Theorem 3.3: Let G be a connected graph and let $\{B_1, B_2, \ldots, B_q\}$ be the block decomposition of G. Then G is $\mathcal{F}$-free if and only if there is at most one block that is of the $\mathcal{B}_1$ type and all other blocks are of the $\mathcal{B}_2$ type.

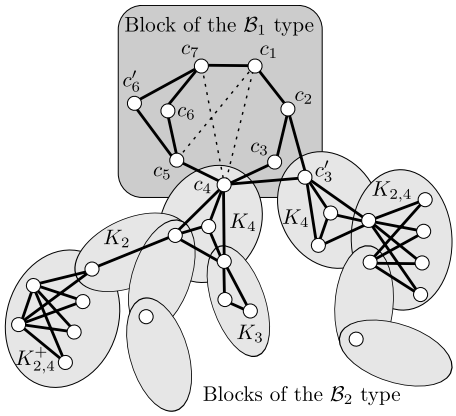


Fig. 2. An example of an $\mathcal{F}$-free graph (the dashed edges may or may not be present). The ellipses show the blocks of the graph.

The proof of the 'if' direction is straightforward. Here, we will give a proof sketch of the 'only-if' direction in a number of steps. For a block B in an $\mathcal{F}$-free graph, its type depends on the size of the longest cycle in B. It will turn out that if B contains a cycle of length 5 or 7, then B is of the $\mathcal{B}_1$ type. Otherwise, B is of the $\mathcal{B}_2$ type. We have the following result on blocks that have a cycle of length five or seven. The proof can be found in [2].

Lemma 3.1: Let G be an $\mathcal{F}$-free graph and let B be a block of G. Let F be a cycle in B that has maximum length. If $|V(F)| \geq 5$, then B is of the $\mathcal{B}_1$ type.

Next, we deal with blocks that do not contain a cycle of length 5 or 7. It follows from the definition of $\mathcal{F}$-free graphs that such blocks do not have cycles of length at least 5. Maffray [22] proved the following theorem:

Theorem 3.1 (Maffray [22]): Let G be a graph. Then the following statements are equivalent:

(1) G does not contain any odd cycle of length at least 5.
(2) For every connected subgraph G' of G, either G' is isomorphic to K_4, or G' is a bipartite graph, or G' is isomorphic to $K_{2,t}^{+}$ for some $t \geq 1$, or G' has a cut-node.

Theorem 3.1 implies the following lemma, the proof of which can be found in [2].

Lemma 3.2: Let G be an $\mathcal{F}$-free graph and let B be a block of G. Suppose that B contains no cycle of length at least 5. Then B is of the $\mathcal{B}_2$ type.

We are now ready to prove Theorem 3.3:

Proof of Theorem 3.3: Let G be an $\mathcal{F}$-free graph and let $\{B_1, B_2, \ldots, B_m\}$ be the block decomposition of G. For every $i \in \{1, 2, \ldots, m\}$, if B_i contains a cycle of length 5 or 7, it follows from Lemma 3.1 that B_i is of the $\mathcal{B}_1$ type. Otherwise, it follows from from Lemma 3.2 that B_i is of the $\mathcal{B}_2$ type. Now suppose that there are $i \neq j$ and $p, q \in \{5, 7\}$ such that B_i contains a cycle T_1 of length p and B_j contains a cycle T_2 of length q. Since G is connected, there exists a path P of length $k \geq 0$ from a node in T_1 to a node in T_2. Since T_1 and T_2 are subgraphs of different blocks, T_1 and T_2

share at most one node. If they share a node, then $k = 0$. Now the edges of T_1, T_2, P form a graph isomorphic to $D_k^{p,q}$, a contradiction. This proves Theorem 3.3. ∎

C. Network graphs satisfy OLoP under primary interference if and only if they are $\mathcal{F}$-free

Now that we have described the structure of all $\mathcal{F}$-free graphs, we use this structure to prove Theorem 3.1 which states that a network graph satisfies OLoP under primary interference if and only if it is $\mathcal{F}$-free. It was shown in [5] (Theorems 2 and 3) that all cycles of length $k \geq 6, k \neq 7$ fail SLoP. [5] Therefore, such cycles do not appear as subgraphs in graphs that satisfy OLoP. The following lemma shows that the same is true for the graphs $D_k^{p,q}$. Its proof is in [2].

Lemma 3.3: $D_k^{p,q}$ fails SLoP for all $p, q \in \{5, 7\}$, $k \geq 0$.

The results from [5] together with Lemma 3.3 imply the following result:

Corollary 3.1: Graphs that satisfy OLoP are $\mathcal{F}$-free.

Proof: Let G be a graph that satisfies OLoP. By the definition of OLoP, every subgraph H of G satisfies SLoP. Since every graph in $\mathcal{F}$ fails SLoP, it follows that G does not contain any graph in $\mathcal{F}$ as a subgraph. ∎

Corollary 3.1 settles the 'only-if' direction of Theorem 3.1. To prove the 'if' direction, we will start with a useful lemma. We will give the idea of the proof, the full proof is in [2].

Lemma 3.4: Let G be a graph and $x, x' \in V(G)$ such that $\deg(x) = 2$ and x' is a clone of x. Then G satisfies SLoP.

Proof idea: Let $\{z_1, z_2\} = N(x)$. Define $\boldsymbol{\alpha} \in [0, 1]^{|E|}$ by letting $\boldsymbol{\alpha}(e) = 1/2$ if e is incident with z_1 or z_2, and $e \neq z_1z_2$, $\boldsymbol{\alpha}(z_1z_2) = 1$ if $z_1z_2 \in E(G)$, and $\boldsymbol{\alpha}(e) = 0$ for all other edges e. Now every maximal matching uses either two edges e_1, e_2 with $\boldsymbol{\alpha}(e_1) = \boldsymbol{\alpha}(e_2) = 1/2$, or edge z_1z_2. ∎

The following lemma is the crucial step in settling the 'only-if' direction of Theorem 3.1. Again, we will give the idea of the proof, the full proof is in [2].

Lemma 3.5: Every connected $\mathcal{F}$-free satisfies SLoP.

Proof idea: Let G be a connected $\mathcal{F}$-free graph and let $\{B_1, B_2, \ldots, B_q\}$ be the block decomposition of G. It follows from Theorem 3.3 that there is at most one block B_i that is of the $\mathcal{B}_1$ type, and all other blocks are of the $\mathcal{B}_2$ type. We will construct a good edge weighting $\boldsymbol{\alpha}$ for G.

Suppose first that G has a leaf block B_i of the $\mathcal{B}_2$ type. If $q = 2$, then let x be the cut-node of G in $V(B_i)$. If $q = 1$, let $x \in V(B_i)$ be arbitrary. There are four cases:

(1) B_i is isomorphic to K_2: let x, v denote the nodes of B_i. Let $\boldsymbol{\alpha}(e) = 1$ for all edges incident with x and $\boldsymbol{\alpha}(e) = 0$ for every other edge e.

(2) B_i is isomorphic to K_3: let $\boldsymbol{\alpha}(e) = 1$ for all $e \in E(B_i)$ and $\boldsymbol{\alpha}(e) = 0$ for every other edge e.

(3) B_i is isomorphic to K_4: let x, v_1, v_2, v_3 denote the nodes of B_i and let $\boldsymbol{\alpha}(v_1v_2) = \boldsymbol{\alpha}(v_1v_3) = \boldsymbol{\alpha}(v_2v_3) = 1$ and $\boldsymbol{\alpha}(e) = 0$ for all $e \in (E(G) \setminus \{v_1v_2, v_1v_3, v_2v_3\})$.

[5] Although the case considered in the reference pertains to interference graphs, the network case is identical since the interference graph (under primary interference) of a cycle is a cycle of the same length.

(4) B_i is isomorphic to $K_{2,t}$ or $K_{2,t}^+$ for some $t \geq 2$: let $V(B_i) = V_1 \cup V_2$ such that $|V_1| = 2$ and V_2 is an independent set. Let $V_1 = \{y_1, y_2\}$ and let $V_2 = \{z_1, z_2, ..., z_t\}$. If B_i is isomorphic to $K_{2,2}^+$ and $x \in V_2$, then assume that $x = z_1$ and set $\boldsymbol{\alpha}(y_1z_2) = \boldsymbol{\alpha}(y_2z_2) = \boldsymbol{\alpha}(y_1y_2) = 1$. Otherwise, B_i contains nodes p, p' such that $\deg(p) = \deg(p') = 2$ and p' is a clone of p, and hence the result follows from Lemma 3.4.

Thus, we may assume that G does not have a leaf block of the $\mathcal{B}_2$ type. Since if $q \geq 2$, G has at least two leaf blocks, and hence at least one leaf block of the $\mathcal{B}_2$ type, we may assume that $q = 1$ and $G = B_1$ is of the $\mathcal{B}_1$ type. Let H, k be as in the definition of $\mathcal{B}_1$. It follows from the definition of H that $|V(H)| = k$. First suppose that $V(G) \setminus V(H) \neq \emptyset$. Then it follows from the definition of $\mathcal{B}_1$ that there exist two nodes x, x' such that $\deg(x) = \deg(x') = 2$ and x' is a clone of x. It follows from Lemma 3.4 that G satisfies SLoP. So we may assume that $V(G) = V(H)$. If $k = 5$, then every maximal matching has size two and hence we may set $\boldsymbol{\alpha}(e) = 1/2$ for all $e \in E(G)$. If $k = 7$, then every maximal matching has size three and hence we may set $\boldsymbol{\alpha}(e) = 1/3$ for all $e \in E(G)$. This proves Theorem 3.1. ∎

We are now in a position to prove Theorem 3.1:

Proof of Theorem 3.1: Corollary 3.1 is the 'only-if' part of the theorem. For the 'if' part, since every subgraph of G is $\mathcal{F}$-free, it follows from Lemma 3.5 that every subgraph of G satisfies SLoP. Therefore, G satisfies OLoP. ∎

D. Recognizing network graphs that satisfy OLoP under primary interference

Having described the structure of graphs that satisfy OLoP, we provide an efficient algorithm for testing whether a given network graph satisfies OLoP under primary interference. A useful observation is the following (see [2] for the proof).

Lemma 3.6: $|E(G)| \leq 2|V(G)|$ for every $\mathcal{F}$-free graph G.

This puts us in a position to prove Theorem 3.2. Again, we will give a proof idea, the details can be found in [2].

Proof idea of Theorem 3.2: We may assume that G is connected. By Theorem 3.1 and Theorem 3.3, it suffices to check whether G admits the structure described in Theorem 3.3. We propose the following algorithm. Let $n = |V(G)|$ and $m = |E(G)|$. First we check that $m \leq 2n$, because otherwise G is not $\mathcal{F}$-free by Lemma 3.6 and we stop immediately. Now, construct the block decomposition $\{B_1, B_2, ..., B_q\}$ of G. Since $m \leq 2n$, this can be done in $O(n + m) = O(n)$ time (see e.g. [12]). For each block B_i, we test in $O(|V(B_i)|)$ time whether B_i is of the $\mathcal{B}_2$ type. If G has more than one block that is not of the $\mathcal{B}_2$ type, then G is not $\mathcal{F}$-free and we stop. If we encounter no such block, then G is $\mathcal{F}$-free and we stop. Next, we check whether B^* is of the $\mathcal{B}_1$ type using multiple applications of Bodlaender's algorithm [3] which, for fixed k, finds a cycle of length at least k in a given graph H, if it exists, in $O(k!2^k|V(H)|)$ time. Checking this can be done in $O(|V(B)|)$ time. Therefore, the overall complexity of the algorithm is $O(n)$. ∎

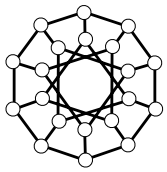


Fig. 3. The Desargues graph $\mathcal{D}$. It satisfies $\sigma(\mathcal{D}) = 0.6$ and it is a subgraph of $K_{10,10}$, showing that $\sigma^*(K_{10,10}) \leq 0.6$.

IV. $t \times n$ SWITCHES WITH $t \leq 7$ SATISFY $\sigma^* \geq 2/3$

In the previous section, we characterized the full set of graphs that satisfy OLoP. It is only natural to ask the question: what happens to graphs that do not satisfy OLoP? In this section, we will show that every bipartite graph G that has one side with at most 7 nodes satisfies $\sigma^*(G) \geq 2/3$.

This result will enable us to prove that every *complete* bipartite graph G with at least three nodes on each side, and one side with at most seven nodes, satisfies $\sigma^*(G) = 2/3$ (i.e., $\sigma^*(K_{t,n}) = 2/3$ for $3 \leq t \leq 7, n \geq 3$). This is close to best possible in a sense. Consider the so-called Desargues graph $\mathcal{D}$ in Fig. 3. $\mathcal{D}$ is a graph on 20 nodes with 30 edges. By solving the Linear Program (2) corresponding to that graph, we find that $\sigma(\mathcal{D}) = \frac{3}{5}$. Since $\mathcal{D}$ is a subgraph of $K_{10,10}$, this implies that $\sigma^*(K_{t,n}) \leq \frac{3}{5}$ for all $t \geq 10, n \geq 10$.

So let us concentrate on subgraphs of $K_{t,n}$ with $t \leq 7, n \geq 1$. We will start with some easy observations that help give a lower bound on $\sigma(G)$.

Lemma 4.1: Let G be a graph.

(a) If there exists $v \in V(G)$ such that every maximal matching in G covers v, then $\sigma(G) = 1$.
(b) If $\deg(v) = 1$ for some $v \in V(G)$, then $\sigma(G) = 1$.
(c) If $\deg(v) = 2$ for some $v \in V(G)$, then $\sigma(G) \geq 2/3$.

Proof: Part (a): let $\boldsymbol{\alpha}(e) = 1$ for all edges incident with v and $\boldsymbol{\alpha}(e) = 0$ for all other edges. Clearly, every maximal matching Z satisfies $\sum_{e \in Z} \boldsymbol{\alpha}(e) = 1$. This proves (a). Part (b) follows immediately because if $\deg(v) = 1$, then every maximal matching covers the unique neighbor u of v. Part (c): let a, b be the neighbors of v. Let $\boldsymbol{\alpha}(av) = \boldsymbol{\alpha}(bv) = 2/3$, $\boldsymbol{\alpha}(ab) = 1$ if $ab \in E(G)$, $\boldsymbol{\alpha}(e) = 1/3$ for all edges $e \notin \{ab, bv, av\}$ that are incident with a or b, and $\boldsymbol{\alpha}(e) = 0$ for all other edges. It is not hard to see that $\sum_{e \in Z} \boldsymbol{\alpha}(e) \geq 2/3$ for every maximal matching Z in G. This proves (c), thus proving Lemma 4.1. ∎

By using the conditions given in Lemma 4.1 and Lemma 2.1, we are able to prove the following lemma, the proof of which is in [2].

Lemma 4.2: Let G be a bipartite graph with $\mu(G) \leq 4$. Then $\sigma(G) \geq 2/3$.

Lemma 4.2 has the following corollaries:

Corollary 4.1: Every bipartite graph G with $\nu(G) \leq 7$ satisfies $\sigma^*(G) \geq 2/3$.

Proof: Let H be a subgraph of G (perhaps $H = G$). Clearly, $\nu(H) \leq \nu(G) \leq 7$. If $\mu(H) \leq 4$, then it follows from Lemma 4.2 that $\sigma(H) \geq 2/3$. Otherwise, $\mu(H) \geq 5$ and hence it follows from Lemma 2.1 that $\sigma(H) \geq 5/7 > 2/3$. Therefore, $\sigma(H) \geq 2/3$ for all subgraphs H of G. It follows that $\sigma^*(G) \geq 2/3$. This proves Lemma 4.2. ∎

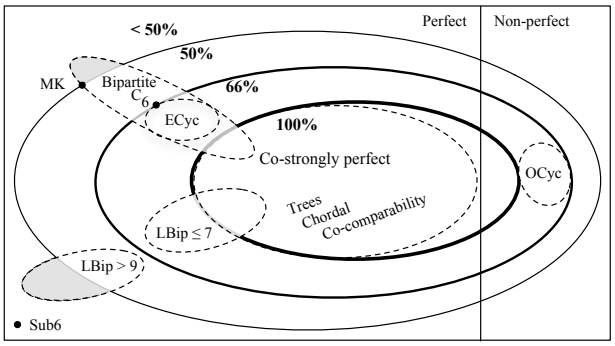


Fig. 4. Throughput guarantees (lower bounds on σ^*) for various graph families: ECyc - cycles C_n with n even and $n \geq 6$, OCyc - cycles C_n with n odd and $n \geq 9$, LBip - line graphs of bipartite graphs, MK - Möbius-Kantor graph, Sub6 - graphs obtained by cycle substitution.

It is already known that $\sigma^*(K_{t,n}) = 1$ for $t \in \{1, 2\}$. For $3 \leq t \leq 7$, we obtain:

Corollary 4.2: $\sigma^*(K_{t,n}) = 2/3$ for all $3 \leq t \leq 7$, $n \geq 3$.

Proof: Let $3 \leq t \leq 7$, $n \geq 3$. It follows from Corollary 4.1 that $\sigma^*(K_{t,n}) \geq 2/3$. Since $K_{t,n}$ contains C_6 as a subgraph and $\sigma(C_6) = 2/3$, it follows that $\sigma^*(K_{t,n}) = 2/3$. This proves Lemma 4.2 ∎

V. INTERFERENCE GRAPHS AND THEIR σ^* VALUES

Our focus so far has been on network graphs and primary interference constraints. We now consider general interference graphs that represent various transmission constraints [15]. The results of this section are summarized in Fig. 4 which illustrates throughput guarantees of several graph families.

Under general interference constraints, a scheduling algorithm has to select an independent set from the interference graph at each slot. An algorithm that solves the Maximum Weight Independent Set problem at every slot is stable [27]. Since this is an NP-complete problem, we are interested in the low-complexity GMS algorithm which greedily picks the nodes with the largest weight (this algorithm is also referred to as the Maximal Weighted Independent Set algorithm).

In the following, we denote by $\overline{\nu}(G)$ and $\overline{\mu}(G)$ the sizes of a maximum independent set (independence number) and a minimum maximal independent set, respectively.

A. OLoP-satisfying Interference Graphs

We first show that the OLoP condition holds in a large subclass of perfect graphs which we will call co-strongly perfect graphs. A graph G is *strongly perfect* if every induced subgraph H of G contains an independent set that intersects every maximal clique in H. We will say that a graph G is *co-strongly perfect* if and only if the complement of G is strongly perfect. The definition of strongly perfectness implies that a graph G is co-strongly perfect if every induced subgraph H of G contains a clique that intersects every maximal independent set of H. An equivalent, and for our purposes more useful definition, is the following.

Definition 5.1 (Co-strongly perfect graph): A graph G is co-strongly perfect if for every induced subgraph H of G there exists $\boldsymbol{\alpha} = \boldsymbol{\alpha}(H) \in \{0,1\}^{|V(H)|}$ such that $\boldsymbol{\alpha}^T \mathbf{M}(H) = \mathbf{e}^T$.

It follows from this definition, and from the interference graph counterparts of Definitions 2.2 and 2.3 that every graph that is co-strongly perfect satisfies OLoP. Note from the above weighting that co-strongly perfect graphs satisfy OLoP with an integer vector $\boldsymbol{\alpha}$. An open question is whether all perfect graphs that satisfy OLoP do so with integer weights $\boldsymbol{\alpha}$. This is not true for imperfect graphs, because C_5 is an imperfect graph that satisfies OLoP, but the unique optimal solution to the Linear Program (2) satisfies $\boldsymbol{\alpha}(v) = 1/2$ for all $v \in V(C_5)$.

It is an open problem as to whether all OLoP-satisfying perfect graphs are co-strongly perfect. Therefore in Fig. 4, co-strongly perfect graphs appear in a dashed circle that almost covers the entire OLoP-satisfying perfect graph region.

The co-strongly perfect class includes a very large number of perfect graph families (some of them were identified individually in [4]). To provide some context about the magnitude of this result, consider the set of simple graphs with 10 nodes. There are 3,063,185 such co-strongly perfect graphs. This can be compared to the 126,768 chordal graphs with 10 nodes (the chordal graphs family is one of the largest previously known families satisfying OLoP) and to the 106 trees.

The family of weakly chordal graphs that was left unresolved in [30] is now known not to be entirely OLoP-satisfying, as we have a counter-example that is weakly chordal, but not co-strongly perfect (Fig. 42 in [14]).

B. σ^*-value for Various Graphs

We now examine the σ^* values of interference graphs that fail OLoP. The contour lines in Fig. 4 indicate a lower bound on the σ^* values of the families that are included in them. For instance, the results on bipartite network graphs from Section IV are shown in the figure as the `LBip` classes (line graphs of bipartite graphs). Line graphs of subgraphs of $K_{t,n}$ with $t \leq 7$ have $\sigma^* \geq 2/3$, while there exist line graphs of subgraphs of $K_{t,n}$ with $t, n > 9$ and $\sigma^* < 2/3$. We proved in Section II that $\sigma^*(G) >= 1/2$ for every line graph G.

The graphs that appear in this subsection are all symmetric. For symmetric graphs, Lemma 2.1 has the following stronger counterpart (the proof of a stronger version is in [2]):

Lemma 5.1: $\sigma(G) = \frac{\overline{\mu}(G)}{\overline{\nu}(G)}$ for every symmetric graph G.

Cycles: The throughput efficiency of the 6-cycle has been examined in [9], [16]. It has also been shown that large cycles ($n \geq 8$) fail OLoP [30]. The following lemma provides the σ^* of cycles, thereby establishing a lower bound on the throughput efficiency of all cycles.

Lemma 5.2: For $n \geq 3$, $\sigma^*(C_n) = \lceil n/3 \rceil / \lfloor n/2 \rfloor$.

Proof: Let $n \geq 3$. Since every proper induced subgraph of C_n is a forest, we have $\sigma(H) = 1$ for every proper induced subgraph H of G. Now consider C_n itself. A maximum independent set in C_n can be constructed by choosing nodes alternatingly on the cycle. This implies that $\overline{\nu}(G) = \lfloor n/2 \rfloor$. A smallest maximal independent set can be constructed by choosing nodes skipping two nodes at a time. This implies that $\overline{\mu}(G) = \lceil n/3 \rceil$. Since C_n is symmetric, it follows from Lemma 5.1 that $\sigma(G) = \lceil n/3 \rceil / \lfloor n/2 \rfloor$. From this and the above, the result follows from the definition of $\sigma^*(G)$. ■

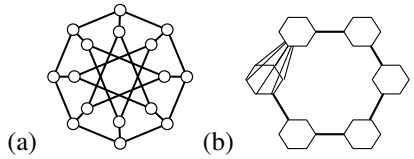


Fig. 5. Graphs that have low σ^* values: (a) Möbius-Kantor graph (b) C_6 substituted for every node of C_6.

To the best of our knowledge, this is the first time an entire family's throughput has been characterized this precisely. This result is shown in Fig. 4 as the classes `OCyc` and `ECyc` for odd and even cycles, respectively. Since $\lceil n/3 \rceil / \lfloor n/2 \rfloor \geq 2/3$ for all $n \geq 3$, Lemma 5.2 has the following corollary.

Corollary 5.1: The efficiency ratio of all cycles is $\gamma^* \geq 2/3$.

Low σ^* values: The current knowledge of σ^* values is limited to a handful graphs in which GMS achieves a large portion of the stability region. The lowest σ^*-value of a specific graph (the Petersen graph, Fig. 1-(c)) is $\sigma^* = 0.6$ [16]. In [17], it was shown that there exists geometric graphs for which $\sigma^* < 1/3$. We now present a specific graph in which σ^* is very low, as obtained by the Linear Progam (2). We also provide a method through which it is possible to create networks with arbitrarily low σ^* values.

Consider the graph shown in Fig. 5-(a). It is a generalised Petersen graph with factors $GP(8,3)$, also known as the Möbius-Kantor graph. Despite its relatively small size of 16 nodes, we showed by solving (2) that $\sigma^* = 0.5$. Hence, GMS can only guarantee 50% throughput.[6] Being a bipartite graph, the Möbius-Kantor allows us to assert that bipartite graphs can have σ^* values as low as 0.5, as illustrated in Fig. 4. Whether bipartite graphs can have $\sigma^* < 0.5$ is still an open question.

Now consider the following family. Let F_1 be a 6-cycle and, for $k \geq 2$, construct F_k from F_{k-1} by substituting a 6-cycle for each node $v \in V(F_k)$. Here by *substituting* C_6 for a node x of the original graph, we mean that we replace x by a 6-cycle H and we make every $v \in V(H)$ adjacent to every neighbor of x. For example, F_2 is shown in Fig. 5-(b), (where the hexagons represent 6-gons). Using Lemma 5.1 and the fact that the F_k are symmetric, we can prove the following:

Observation 5.1: $\sigma^*(F_k) \leq \left(\frac{2}{3}\right)^k$ for all $k \geq 1$.

Since we may choose k arbitrarily large, it follows that there exist graphs with arbitrarily small σ^*. A graph generated by this method appears in Fig. 4 as `Sub6`.

C. Simulation results

When GMS guarantees only low throughput efficiency γ^*, there may exist a specific arrival rate outside of $\gamma^*\Lambda^*$ for which GMS is not stable. In real-life arrival processes, it is sometimes unlikely that such an arrival process would

[6]Note that since this graph contains a claw (i.e., a complete bipartite graph $K_{1,3}$), it cannot be the interference graph of any network under primary interference constraints.

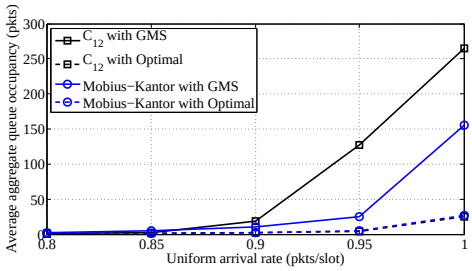


Fig. 6. Average queue sizes as a function of the arrival rate under GMS and the optimal algorithm. The results obtained via simulation in a 12-cycle and a Möbius-Kantor graph.

occur. Hence GMS may behave better than predicted. We used Matlab simulations in order to evaluate the performance of GMS in graphs with low σ^* identified in Section V-B.

We consider i.i.d uniform arrivals to every node at each time slot for a range of normalized loads within the stability region. We tested the GMS and the stable algorithm that solves the Maximum Weight Independent Set problem[7]. For each arrival rate, the simulation was run for 10,000 iterations. For each graph and arrival rate value, the average queue occupancies appear in Fig. 6. The cycle C_{12} has $\sigma^* = 2/3$. In the figure, we see that in a cycle, the queues under GMS become unstable at around load level of 0.85. Although the Möbius-Kantor graph has a $\sigma^* = 1/2$, GMS performs similarly.

VI. Conclusion

The Local Pooling (LoP) conditions provide a new tool for better understanding the performance of Greedy Maximal Scheduling (GMS) algorithms. In this paper, we identified *all* the network graphs in which these conditions hold under primary interference constraints (in these graphs Greedy Maximal Scheduling achieves 100% throughput). In addition, we showed that in all bipartite graphs of size up to $7 \times n$, GMS is guaranteed to achieve 66% throughput. Finally, we studied the performance of GMS in interference graphs and showed that σ^* can be arbitrarily low.

We emphasize that our objective in this paper is to obtain a better *theoretical* understanding of LoP that will assist the development of future algorithms. As such, the paper demonstrates that using graph theoretical methods can significantly contribute to our understanding of greedy scheduling algorithms. From the graph theoretical point of view, LoP raises many interesting open problems. For example, three of the authors [7] are currently working on extending some of the results to the so-called claw-free graphs, which are a generalization of the interference graphs of networks under primary interference. From the networking point of view, there remain many open problems. For example, generalizing the interference model to a model based on SINR and deriving the corresponding LoP conditions remain a major subject for future research.

[7]Although the problem is NP-complete, we obtained numerical solutions in small graphs.

References

[1] M. Ajmone Marsan, E. Leonardi, M. Mellia, and F. Neri, "On the stability of isolated and interconnected input-queueing switches under multiclass traffic," *IEEE Trans. Inform. Th.*, vol. 51, no. 3, pp. 1167–1174, Mar. 2005.

[2] B. Birand, M. Chudnovsky, B. Ries, P. Seymour, G. Zussman, and Y. Zwols, "Using local pooling to characterize networks in which greedy scheduling performs well," Columbia University, EE, Tech. Rep. 2009-07-30. [Online]. Available: http://www.columbia.edu/~bb2408/pdfs/GMSThroughputLoP.pdf

[3] H. Bodlaender, "On linear time minor tests with depth-first search," *J. Algorithms*, vol. 14, no. 1, pp. 1–23, 1993.

[4] A. Brzezinski and E. Modiano, "Greedy weigthed matching for scheduling the input-queued switch," in *Proc. CISS'06*, Mar. 2006.

[5] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks - a partitioning approach," in *Proc. ACM MOBICOM'06*, Sept. 2006.

[6] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," *IEEE Trans. Inform. Th.*, vol. 54, no. 2, Feb. 2008.

[7] M. Chudnovsky, B. Ries, and Y. Zwols, "Claw-free Graphs That Satisfy Local Pooling," 2009, in preparation.

[8] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM'00*, Mar. 2000.

[9] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest queue first scheduling: second order properties using fluid limits," *Adv. Appl. Probab.*, vol. 38, no. 2, pp. 505–521, June 2006.

[10] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*, 2006.

[11] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE J. Select. Areas Commun.*, vol. 21, no. 4, pp. 546–559, May 2003.

[12] J. Gross and J. Yellen, *Graph theory and its applications*. CRC press, 2006.

[13] J.-H. Hoepman, "Simple distributed weighted matchings," Oct. 2004, eprint cs.DC/0410047.

[14] S. Hougardy, "Classes of perfect graphs," *Discrete Mathematics*, vol. 306, no. 19-20, pp. 2529–2571, 2006.

[15] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *ACM/Springer Wireless Networks*, vol. 11, no. 4, pp. 471–487, July 2005.

[16] C. Joo, X. Lin, and N. B. Shroff, "Performance limits of greedy maximal matching in multi-hop wireless networks," in *Proc. IEEE CDC'07*, Dec. 2007.

[17] C. Joo, X. Lin, and N. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proc. INFOCOM'08*, April 2008.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "What cannot be computed locally!" in *Proc. ACM PODC'04*, July 2004.

[19] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," in *Proc. ACM MOBIHOC'09*, May 2009.

[20] B. Li, C. Boyaci, and Y. Xia, "A refined performance characterization of longest-queue-first policy in wireless networks," in *Proc. ACM MOBIHOC'09*, May 2009.

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

[22] F. Maffray, "Kernels in perfect line-graphs," *Journal of Combinatorial Theory Series B*, vol. 55, no. 1, pp. 1–8, 1992.

[23] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

[24] T. Moscibroda and R. Wattenhofer, "The complexity of connectivity in wireless networks," in *Proc. IEEE INFOCOM'06*, Apr. 2006.

[25] R. Preis, "Linear time $1/2$-approximation algorithm for maximum weighted matching in general graphs," ser. LNCS, vol. 1563. Springer, 1999.

[26] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. ACM MOBICOM'06*, Sept. 2006.

[27] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[28] D. West *et al.*, *Introduction to graph theory*. Prentice Hall Upper Saddle River, NJ, 2001.
[29] X. Wu and R. Srikant, "Bounds on the capacity region of multi-hop wireless networks under distributed greedy scheduling," in *Proc. IEEE INFOCOM'06*, Apr. 2006.
[30] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *Proc. IEEE INFOCOM'08*, April 2008.