RICE UNIVERSITY

Beyond Interference Avoidance: Distributed Sub-network Scheduling in Wireless Networks with Local Views

by

Pedro Enrique Santacruz

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Dr. Ashutosh Sabharwal, *Chair*
Professor of Electrical and Computer Engineering

Dr. Behnaam Aazhang
J.S. Abercrombie Professor of Electrical and Computer Engineering

Dr. Edward W. Knightly
Professor of Electrical and Computer Engineering

Dr. Illya V. Hicks
Associate Professor of Computational and Applied Mathematics

HOUSTON, TEXAS
MAY 2013

Abstract

Beyond Interference Avoidance: Distributed Sub-network Scheduling in Wireless Networks with Local Views

by

Pedro Enrique Santacruz

In most wireless networks, nodes have only limited local information about the state of the network, which includes connectivity and channel state information. With limited local information about the network, each node's knowledge is mismatched; therefore, they must make distributed decisions. In this thesis, we pose the following question - if every node has network state information only about a small neighborhood, how and when should nodes choose to transmit? While link scheduling answers the above question for point-to-point physical layers which are designed for an interference-avoidance paradigm, we look for answers in cases when interference can be embraced by advanced code design, as suggested by results in network information theory.

To make progress on this challenging problem, we propose two constructive distributed algorithms, one conservative and one aggressive, which achieve rates higher than link scheduling based on interference avoidance, especially if each node knows more than one hop of network state information. Both algorithms schedule sub-networks such that each sub-network can employ advanced interference-embracing coding schemes to achieve higher rates. Our innovation is in the identification, selection and schedul-

ing of sub-networks, especially when sub-networks are larger than a single link.

Using normalized sum-rate as the metric of network performance, we prove that the proposed conservative sub-network scheduling algorithm is guaranteed to have performance greater than or equal to pure coloring-based link scheduling. In addition, the proposed aggressive sub-network scheduling algorithm is shown, through simulations, to achieve better normalized sum-rate than the conservative algorithm for several network classes. Our results highlight the advantages of extending the design space of possible scheduling strategies to include those that leverage local network information.

Acknowledgements

I would like to begin by expressing my most sincere and deepest gratitude to Professor Ashutosh Sabharwal for being an advisor that has provided support, guidance, motivation, and a great example of what it takes to be an outstanding researcher. His willingness and ability to understand the best approach to maximize my abilities is one of the biggest reasons why I was able to successfully complete this degree. I also would like to thank my committee: Dr. Behnaam Aazhang, Dr. Edward Knightly, and Dr. Illya Hicks. Your guidance, comments, and probing questions added significantly to the work presented in this thesis. A special thanks to Vaneet Aggarwal, who played and integral role in the development of the problem and throughout the quest for solutions. Your collaboration is very much appreciated.

Next, I would like to thank all fellow Ph.D. students who have helped me throughout my studies. I would like to thank Debashis Dash for being a great mentor during my early years at Rice. To David Kao and Evan Everett, thank you for your friendship, support, and insightful discussions. Also, I am extremely grateful to Achaleshwar Sahai for the countless times when he provided key assistance in technical matters, but more importantly for countless times when he shared his friendship and charisma making the graduate student experience thoroughly enjoyable.

Thanks to my family for the unconditional love and support they have

provided all this time. They have always been the foundation that supports all my endeavors.

With the fullness of my heart, I would like to thank Samantha Summerson. Words cannot express the level of gratefulness, appreciation, and love I have for everything you have done for me. You make me better, wiser, stronger, and, above all, happier.

Finally, to the One who guided the beginning of my work, directed its progress, and brought it to its completion, my humble gratitude always.

A

mi Mamá, mi Papá, Humberto y Gilberto

Gracias

Contents

Abstract **ii**

Acknowledgements **iv**

1 Introduction **1**

2 System Model and Problem Formulation **5**
2.1 Network Model 5
2.2 Local View 7
2.3 Normalized Sum-rate 8

3 Related Work **9**
3.1 Interference Avoidance: Link Scheduling 9
3.2 Beyond Interference Avoidance 12

4 Overview **14**
4.1 Approach 14
4.2 Contributions 15
4.3 Overview of Proposed Algorithms 16

5 Step 1: Identification ($G \to G_\rho^-(v)$) **20**

6 Step 2: Selection ($G_\rho^-(v) \to G_\rho(v)$) **24**
6.1 Consevative Selection Algorithm 24
6.2 Aggressive Selection Algorithm 29

7 Step 3: Scheduling **37**
7.1 Local Multicoloring Algorithm 38
7.2 Application 39
7.3 Overhead 40

8 Results **42**
8.1 Conservative Algorithm Guarantee 44
8.2 Numerical Results: Normalized Sum-Rate 46
8.3 Net Sum-rate Comparisons 52
8.4 Numerical Results: Net Sum-rate 63
8.5 Remarks on Overhead 66

9 Conclusion **70**
9.1 Summary of Contributions and Significance 70
9.2 Future Directions 71

References **73**

List of Figures

2.1 Interference Network 6
2.2 Conflict Graph, G, associated with Interference Network in Figure 2.1 6

5.1 Line-clique graph with N nodes, G 21
5.2 $G_1^-(1)$ 21
5.3 $G_1^-(2)$ 21
5.4 $G_1^-(N)$ 22

6.1 $G_1^-(2) \to G_1(2)$ 26
6.2 $G_1^-(1) \to G_1(1)$ 26
6.3 $G_1^-(N) \to G_1(N)$ 27
6.4 G_ρ^- for $\rho = 1$ 31
6.5 Cliques that appear twice elsewhere are removed 31
6.6 Final graph, G_1, after algorithm 32
6.7 r-cliques, $r = 0, 1, 2$ 33
6.8 Cliques are removed 33
6.9 Final graph, G_2, after aggressive algorithm 33

8.1 N-node Line-star Graph 46
8.2 Conservative vs. Agressive Algorithms - Sample Graphs 47
8.3 Sample Random Graph with Low Connectivity, $p = 0.1$ 49
8.4 Sample Random Graph with High Connectivity, $p = 0.9$ 49
8.5 Sample Scale-free Graph 50
8.6 Sample Geometric Graph 50

8.7 Normalized sum-rate performance comparison in random graphs with low connectivity, $p = 0.1$ 51
8.8 Normalized sum-rate performance comparison in random graphs with medium connectivity, $p = 0.5$ 52
8.9 Normalized sum-rate performance comparison in random graphs with high connectivity, $p = 0.9$ 53
8.10 Normalized sum-rate performance comparison in scale-free graphs . . 54
8.11 Normalized sum-rate performance comparison in geometric graphs, $d = 0.25$ 55
8.12 Normalized sum-rate performance comparison in geometric graphs, $d = 0.5$ 56
8.13 Two cliques 57
8.14 Two cliques after conservative algorithm, G_1 57
8.15 Two cliques after aggressive algorithm, G_1 58
8.16 Clique-star 60
8.17 Clique-star after conservative algorithm, G_1 60
8.18 Clique-star after aggressive algorithm, G_1 61
8.19 Net sum-rate performance comparison in random graphs with low connectivity, $p = 0.1$ 64
8.20 Net sum-rate performance comparison in random graphs with medium connectivity, $p = 0.5$ 65
8.21 Net sum-rate performance comparison in random graphs with high connectivity, $p = 0.9$ 66
8.22 Net sum-rate performance comparison in scale-free graphs 67
8.23 Net sum-rate performance comparison in geometric graphs, $d = 0.25$. 68
8.24 Net sum-rate performance comparison in geometric graphs, $d = 0.5$. 69

List of Tables

8.1 Table of Results for Two Example Topologies 62

Chapter 1

Introduction

The shared nature of wireless communication networks results in the fundamental problem of dealing with interference from other simultaneous transmissions by co-located flows. The most commonly used technique of managing interference is to avoid it by scheduling transmissions such that the co-located flows do not transmit simultaneously. Link scheduling inherently assumes that the underlying physical layer architecture is designed to decode a single packet. Link scheduling, both centralized and distributed, has a rich history and continues to be an active area of research, see [1, 2, 3, 4, 5, 6] and references therein. We pose and study the scheduling problem for the case when the physical layer architecture can embrace interference by using advanced coding methods.

While interference-avoidance continues to be the near de-facto strategy in wireless networks, it has been known for some time that avoiding interference is not a capacity maximizing strategy for many networks. For example, techniques like multi-user detection [7], Han-Kobayashi coding for 2-user interference channel [8] and interference alignment for general interference networks [9] are known to yield higher capacity by embracing, not avoiding, interference; e.g., see the book-length exposition [10] for details. These new ideas have also inspired new standardization activity, like Coor-

dinated Multipoint (COMP) [11], which uses network MIMO to improve capacity at the edge of the cells. However, almost all such advanced techniques assume extensive knowledge about the network topology, channel statistics, and in many cases, instantaneous channel information, to achieve capacity gains from embracing interference [10]. A direct impact of requiring such extensive knowledge at each node is that the resulting network architecture poses scalability limitations – as the network size grows, the amount of network state information needed at every node also grows proportional to the number of users in the network.

In this thesis, we pose the following problem. If each node in the network has limited information about the network state (connectivity and channel states), say it only knows the network state information about channels and links $h \geq 1$ hops away from it, then what is the capacity maximizing transmission scheme. Note that limited local information problems have been extensively studied in distributed scheduling [3, 4, 5, 6]. However, as mentioned above, all of them assume interference avoidance as their underlying architecture. In our model, the physical layer architecture is not restricted a priori and is allowed to be any feasible scheme, including those which embrace interference [10]. However, unlike network information theory formulations [8, 9, 12], we are explicitly studying only scalable architectures by limiting network state information at each node.

The new problem turns out to be extremely hard, and is the generalized version of the distributed capacity problem studied recently in [13, 14]. The formulation in [13, 14] shared full network connectivity information with all nodes but only h-hop information about the channel state. The key (and surprising) result in [13] was that a generalized form of scheduling is information-theoretically optimal for many networks. The general scheduling, labeled Maximum Independent Graph Scheduling (MIG Scheduling), schedules *connected sub-networks* larger than a single link. This

is especially true if $h > 1$, i.e, nodes know more than one hop of channel information. The size of the sub-networks is such that within each sub-network, an advanced coding scheme can be used since all nodes have enough channel information to operate optimally. Since MIGS assumed that the nodes have full connectivity information, they can coordinate their time of transmission. Thus MIGS is akin to centralized scheduling, but of connected sub-networks potentially bigger than a single link.

In our work, no global connectivity information is available at any node and thus all decisions have to be truly distributed. To make progress on the challenging new network capacity problem, we use MIGS as our starting point and focus on how sub-networks can be *identified*, *selected* and *scheduled* in a distributed manner. Thus, our contributions in this thesis are two distributed sub-network scheduling algorithms, both of which achieve better network performance compared to interference-avoidance, especially when more than one-hop of network state information is made available at each node.

The first step is in identification of sub-networks which can operate optimally with limited local information. That is, if the rest of the network was switched off beyond a small sub-network, there exists an advanced interference-embracing coding scheme which will achieve the maximum possible capacity in that sub-network. We limit our attention to a smaller set of sub-graphs, ρ-cliques (defined later), where ρ depends on the amount of available network information. We note that identification of maximal independent sub-graph schedules [13], which are a generalization of independent set scheduling, is an open problem. By limiting our attention to finding only ρ-cliques, we ensure that each sub-network by itself can operate optimally with limited local network view at each node, thereby circumventing the open problem and still guaranteeing the sufficient condition of sub-networks to be used by MIGS.

In the second step, we select a subset of identified ρ-cliques in order to maximize

network sum-rate. The challenge, like in any distributed problem, is for nodes to reach consensus *locally* such that the global rate is optimized. Towards that end, we propose two selection algorithms labeled *conservative* and *aggressive*. Both selection algorithms prune the identified ρ-cliques in order to decrease the maximum degree of the graphs made from the identified ρ-cliques, which is directly related to increasing the network sum-rate. Thus, the two algorithms use only local graph properties in their decision making. The conservative algorithm is guaranteed to produce schedules which will never achieve rates below interference-avoiding link scheduling, but ends up making conservative decisions for many networks. The aggressive algorithm does not provide provable guarantees but is shown to achieve better sum-rates for several network classes. The last step of scheduling sub-networks is performed using Kuhn's local multicoloring algorithm [15], applied to the more general graph structure induced by sub-networks.

We note that once the connected sub-networks are decided and scheduled, each sub-network uses the appropriate coding scheme based on their topology. Thus, like traditional link scheduling which is concerned with *when* links transmit and not *how* actual channel coding/decoding is performed, our contributions are to identify the sub-networks and not what actual codes are used. The decoupling is possible because of the optimality condition required in MIGS for allowed sub-networks [13].

Chapter 2

System Model and Problem Formulation

2.1 Network Model

In this work, we consider a wireless network in an interference channel model scenario. The interference network model consists of N source-destination pairs. Each source-destination pair is considered a user in the network. The source nodes, labeled S_i, are connected to a subset of the destination nodes, labeled D_j, $(i, j \in \{1, ..., N\})$ if the received power at destination D_j from S_i is above some threshold. The set of transmitters that are connected to receiver D_j is labeled $\mathcal{I}_j$. We assume there is always a connection between S_i and D_j, for all $i = j$. The channel gain between S_i and D_j is denoted H_{ij}. The received signal at receiver D_j is

$$Y_j = \sum_{i:S_i \in \mathcal{I}_j} H_{ij} X_i + W, \tag{2.1}$$

where X_i is the transmit signal from S_i subject to its average power constraint P_i and W is complex gaussian noise $\mathcal{CN}(0, 1)$.

Associated with the interference network, there is a *conflict graph* $G(V, E)$. In this conflict graph, a vertex $v \in V$ represents a user in the interference network and

an edge $e \in E$ represents interference between those two users, that is, $\{i, j\} \in E$ if S_i is connected to D_j or if S_j is connected to D_i. Figure 2.1 depicts an example interference network and its corresponding conflict graph is illustrated in Figure 2.2. It is important to note that, since our conflict graph is undirected, there are several interference networks that result in the same conflict graph. While all our results hold for arbitrary conflict graphs, we will use the conflict graph in Figure 2.2 as an illustrative example in the rest of the thesis. Hence, we label the N-node graph (in Figure 2.2) with a line of size $N-3$ and an attached clique of size 3 as the *line-clique* graph.

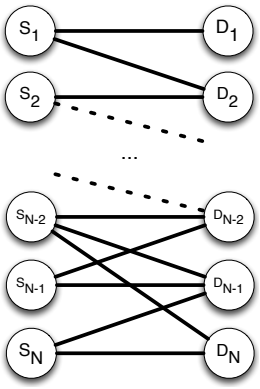


Figure 2.1: Interference Network

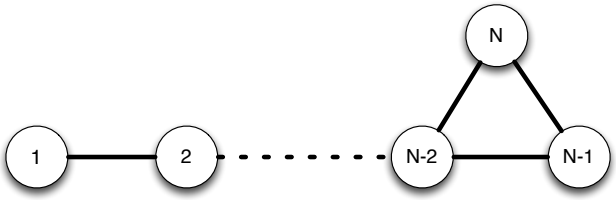


Figure 2.2: Conflict Graph, G, associated with Interference Network in Figure 2.1

We note some of the differences between our model and the network model commonly known as the k-hop interference model in the network scheduling literature ([3, 16, 4, 5] and others). In the k-hop interference model, each node in the interference network can be either a transmitter or receiver while in our model each node

has an assigned role. Since in our model interference only occurs if a transmitter directly interferes with another receiver, our model is more closely related to the 1-hop, or node-exculsive, interference model. All traffic in our model is assumed to be single-hop traffic and there exist interference connections in the interference network that are not data links (users) in the network, i.e., connections between S_i and D_j for $i \neq j$ do not represent data links, yet a connection exists and interference behavior is from the interference network to the conflict graph.

2.2 Local View

In [13], local knowledge at any node was modeled as h-hops of channel information with that node as the center. However, all nodes were assumed to have full connectivity information. We will study a more detailed model of network information, giving each node only limited network connectivity and channel state knowledge, as described below. For convenience, we will describe them in the conflict graph representation of the network.

A user is said to have τ hops of connectivity information if it knows all vertices and edges τ hops away from it in the conflict graph G. Similarly, a user has η hops of channel information if it has knowledge of all channel gains in the interference network for all users η hops away in the conflict graph. Notice that η hops of channel knowledge in conflict graph equals $h = 2\eta + 1$ hops of channel knowledge in the interference network. The same holds true for connectivity information. Also, we assume that $\tau \geq \eta$ since, in general, channel information is more difficult to obtain than connectivity information.

2.3 Normalized Sum-rate

Our metric of network performance is a slight modification of the *normalized sum-rate*, α, introduced in [13], which is the information-theoretic sum-rate achieved normalized by the sum-capacity with full network state information. More precisely, a normalized sum-rate of $\alpha(\eta)$ with η hops of channel state information in the conflict graph is said to be achievable if there exists a strategy that allows transmission at rates R_i for each flow $i \in \{1, 2, ..., N\}$ with error probabilities going to zero, and satisfying

$$\sum_{i=1}^{N} R_i \geq \alpha(\eta) C_{sum} - \epsilon \tag{2.2}$$

for all topologies consistent with the local view information, regardless of the realization of the channel gains. Here C_{sum} is the sum capacity of the network with full information and ϵ is some non-negative constant independent of channel gains.

In this work, we extend the concept of normalized sum-rate by removing the assumption that full connectivity information is available at every node. Instead, we quantify both the hops of channel *and* connectivity information available at each node by η and τ, respectively. We propose as our metric of performance the more general form of normalized sum-rate, denoted $\tilde{\alpha}(\eta, \tau)$, as a function of η and τ, such that the following is satisfied

$$\sum_{i=1}^{N} R_i \geq \tilde{\alpha}(\eta, \tau) C_{sum} - \epsilon. \tag{2.3}$$

The problem we address in this work is to characterize the achievable performance in a network with η hops of available channel information and τ hops of available connectivity information. We tackle this problem by creating schemes that use only (η, τ) hops of information and analyze their performance.

Chapter 3

Related Work

Interference management in wireless networks has been widely studied at both the network and physical layers. From the networking point of view, prior research has focused on developing schemes and algorithms that reduce the computational complexity and allow optimal throughput strategies to be executed in a distributed fashion. At the physical layer, the focus has been set on finding schemes that approach the holy grail of network capacity. Approaches at both the networking and physical layer have acknowledged and tried to address the issue of local information in some form or another. In this section, we review some of the works related to our goal of characterizing the effects of local information in wireless networks.

3.1 Interference Avoidance: Link Scheduling

In the seminal work by Tassiulas and Ephremides [1], the capacity of a constrained queueing system for an interference-avoiding physical layer was derived and characterized. The problem was shown to be solved by finding a maximum-weight independent set of nodes in the network graph in each transmission time-slot. While this schemes yields optimal throughput under the interference-avoidance paradigm, it re-

quires complete information about the connectivity and queue states of the network by all members of the network or, alternatively, a centralized entity that computes the optimal schedule and communicates it to the members of the network in each time-slot. Furthermore, even when complete information is available, the optimization problem that needs to be solved has high complexity. The need for complete information and the high computation complexity of the optimal solution make maximum weight-scheduling unfeasible for most practical purposes, especially in wireless scenarios. There has been a significant amount of work on reducing the amount of network information required at each node, developing algorithms that are implementable in a distributed manner, and reducing the computational complexity of the algorithms. Some prior work aims to maximize performance in terms of utility [17, 18, 19] while others are concerned with improving throughput. Most state-of-the-art approaches that aim to maximize throughput can be arranged into four categories.

1. The first category of link scheduling algorithms is denoted *Maximal Matching*. The algorithms proposed in this class, [6], [3], [4], [5], attempt to produce a maximal schedule in each time-slot. A maximal schedule is defined as a set of links such that no two links interfere with each other and such that there are no other links that could be added to the set without creating interference. These algorithms produce throughput guarantees and are suitable for distributed implementation, yet they require global knowledge about the connectivity of the network, or at least some predetermined global ordering that allows nodes to avoid conflicts during algorithm execution. Moreover, the amount of overhead in each time-slot requires $O(\log^2 N)$ rounds of message passing in the worst-case scenario.

2. The second category consists of the so-called *Pick-and-compare* policies where, at each time-slot, the current schedule serves as a building block for the next

schedule. Some examples include the work in [20] where schedules are augmented in order to improve performance, and the work in [21] where the current schedule is mixed with a new random schedule. Both of these algorithms make a comparison between the current schedule and the potential new one by choosing the schedule that results in higher throughput. Once again, these algorithms require global connectivity information and large number of rounds of execution. In addition, the long convergence time to the optimal schedule can result in increased delay.

3. Another class of algorithms under the paradigm of interference-avoidance are the policies that require a *constant-time overhead*. These policies are all based on separating each time frame into a scheduling time-slot and data transmission time-slot. The policies in this class include the random-access-type algorithms in [16], [22], and [23]. The design of these algorithms presents an explicit tradeoff between performance and overhead. Constant-time overhead algorithms also include those that employ Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). The schemes presented in [24], [25], and [26], assume carrier-sensing capabilities by the nodes in the network.

4. Finally, recent work ([27] and[28]) has further developed algorithms that introduce the idea of network locality in the process of scheduling. These algorithms provide *local greedy scheduling* schemes that approximate Greedy Maximal Scheduling [3] with nodes in the network using only information about themselves and their neighbors. The information being exchanged before each data communication session include channel, connectivity, and queue state information.

All algorithms described in this section, and other approaches not categorized here [29, 30, 31, 32, 33, 34, 35, 36] have the predetermined assumption that receivers in the network decode an incoming transmission successfully only if none of the other links within reception range transmit concurrently. Some of the implementation issues of the optimal solution presented in [1] are addressed by the works described above, yet the underlying interference-avoidance physical-layer architecture remains. This results in more practical and more easily implementable schemes that can guarantee a fraction of the performance of the scheduling capacity region described in [1], but leaves the possibility of leveraging advanced physical layers with local information open for exploration.

3.2 Beyond Interference Avoidance

Interference-avoidance strategies are often not the optimal approach from a sum-capacity perspective and developments in network information theory describe advanced coding techniques that achieve higher sum-capacity [10]. In fact, the capacity region of a network with interference-avoidance as the assumed physical layer must lie completely inside the capacity region where all physical layer techniques are available. Naturally, the major drawback of these results is the need for complete (or almost complete) network information, including connectivity and channel information.

Several works have addressed this prohibitive requirement by analyzing the performance of networks with only limited information. In [37], the author considers an interference channel where transmitters do not have channel information and only connectivity information is available. Network performance under different example topologies is analyzed in terms of degrees of freedom (DoF). Techniques and results used in the problem of wired networks with linear network coding are applied to the wireless problem. The work in [38] produces results in a similar scenario that assumes

connectivity information but no channel information is available to the transmitters. A study of the capacity region for the 2-user interference channel when each transmitter knows only a subset of the channel gains in the network is presented in [39]. These results along with the work by Aggarwal et al. [13] motivate the argument that it is possible to use advanced coding techniques with limited local knowledge. We will use the work by Aggarwal et al. as the launchpad for our work and expand it to capture a more detailed analysis of the impact of local information on network performance.

Chapter 4

Overview

4.1 Approach

In [13], the authors examine optimal schemes with local channel state and global connectivity knowledge, and propose a strategy called Maximal Independent Graph (MIG) Scheduling which is information-theoretically optimal for various classes of networks. Information-theoretic optimality means that there exist no other physical layer coding strategy which can achieve higher sum-rates given the amount of knowledge available. In MIG Scheduling, every node in the interference network knows the complete topology of the network and each node is assumed to have h hops of channel information. The network is separated into sub-networks that can achieve a normalized sum-capacity of 1 with the h hops of channel information (i.e., sub-networks with enough local knowledge to simultaneously transmit in an optimal way). Thus, the important result from [13] is that a generalized form of network scheduling is information-theoretically optimal in many cases. Our approach finds simple distributed algorithms that use only local connectivity information *and* local channel information to find key sub-networks and implement this generalized form of network scheduling.

In MIG Scheduling, the sub-networks that are able to achieve normalized sum-capacity of 1 are labeled *independent sub-graphs*. MIG scheduling divides the network into t independent sub-graphs, $\mathcal{A}_1, \ldots, \mathcal{A}_t$ (not all distinct, for some t), and each user i belongs to d_i independent sub-graphs. An important result in [13] is that this scheme of dividing the network into independent sub-graphs achieves a normalized sum-rate of

$$\alpha(h) = \min_{i \in 1,2,\ldots N} \frac{d_i}{t}. \tag{4.1}$$

The set of independent sub-graphs, $\mathcal{A}_1, \ldots, \mathcal{A}_t$, that maximizes the value of $\alpha(h)$ is called the MIG schedule.

The problem of finding the MIG schedule for an arbitrary network is a difficult task, even with complete connectivity knowledge, and particularly challenging with only local connectivity information at each node. The optimal independent sub-graphs are only known for few topologies and small number of users. In this work, to answer our posed capacity problem, we focus on identification of independent sub-graphs in a distributed fashion with only (τ, η)-hops of knowledge about network state.

4.2 Contributions

The contributions of the work presented in this thesis are threefold. The first contribution is associated with our problem formulation. While previous work has looked into network performance using normalized sum-rate [13, 14, 40], the formulation in that literature has assumed that only channel state information is locally available and connectivity information is available globally to all users in the network. In this work we remove the assumption of globally available connectivity information and

formulate the problem to characterize a more general form of normalized sum-rate with η hops of channel information and τ hops of connectivity information.

The second contribution of this thesis is the design of two constructive distributed sub-network scheduling algorithms to improve normalized sum-rate performance using local network views. The two algorithms are denoted *conservative* and *aggressive*. These are one-shot algorithms that are based on simple heuristics and use η hops of channel state information and τ hops of connectivity information.

Finally, our third contribution is the analysis of the two proposed algorithms in terms of normalized sum-rate performance. We show that the performance of the conservative sub-network scheduling algorithm is guaranteed to be a non-decreasing function of the number of hops of information available to each node. Through simulations, we also show that the aggressive algorithm achieves significant performance gains over several important network classes.

4.3 Overview of Proposed Algorithms

In this section, we give a general overview our algorithms to distributedly find independent sub-graphs as required by Independent Graph Scheduling. We propose two algorithms labeled as *conservative* and *aggressive*. Both algorithms consist of three major steps: 1) identification, 2) selection, and 3) scheduling. In the first step, we use the available channel and topology information to identify all sub-networks of diameter at-most ρ such that each sub-network independently achieves a normalized sum-capacity of 1. In the second step, we strategically select a subset of these sub-networks. The selected subset of sub-networks will be the only sub-networks that will be transmitting. Finally, in the third step we arrange several of these connected sub-networks into independent sub-graphs that still achieve normalized sum-capacity of 1. The creation of independent graph is done by using a distributed coloring al-

gorithm that assigns a single color to groups of sub-networks. Our algorithms are parametrized by ρ, which is the maximum diameter of the connected sub-networks being identified. Given a ρ, we assume that each node has at least $\eta = \rho + 1$ hops of channel knowledge and either $\tau = 3\rho + 3$ or $\tau = 3\rho + 1$ hops of connectivity information, depending on the algorithm used. For simplicity, we also denote the generalized normalized sum-rate, $\tilde{\alpha}(\eta, \tau)$, by a single parameter, ρ, and use the symbol $\tilde{\alpha}(\rho)$.

In Step 1, we leverage the local knowledge available at each node by finding r-cliques for $r \leq \rho$ which is defined as follows:

Definition 1. *A r-clique in a graph $G = (V, E)$ is a subgraph, $G[S]$, induced by a subset of nodes $S \subset V$ that satisfies the following three conditions:*

1. *Every node in $G[S]$ is at most a distance of r hops away from all other nodes in $G[S]$.*

2. *The diameter of $G[S]$ is r.*

3. *There is no $S' \subset V$ that also satisfies Conditions 1 and 2 and such that $S \subset S'$. In other words, $G[S]$ is a maximal subgraph.*

Note that a single node is a graph by itself and a 0-clique according to the above definition.

Step 1 consists of identifying r-cliques, for $r = 0, ..., \rho$, in the conflict graph, G. After the r-cliques have been identified, Step 2 consists of selecting a subset of the identified r-cliques and consolidating the selected r-cliques into single vertices to generate a consolidated graph, G_ρ, where each vertex represents an r-clique, $r = 0, ..., \rho$, from the conflict graph, G. An edge exists between two vertices in the consolidated graph if there exists an edge between members of the two cliques in the original conflict graph.

Step 3 of the general procedure is performed by applying the distributed multi-coloring algorithm by Kuhn [15] to the consolidated graph, G_ρ, which results in the assignment of time slots to each one of the cliques. The set of cliques with the same color are defined as an *independent clique set*. An independent clique set achieves $\alpha(\eta) = 1$ because each clique achieves $\alpha(\eta) = 1$ and the cliques do not interfere with each other. When we assign a time slot to each of the independent clique sets we create a scheme for Independent Graph Scheduling. We have chosen Kuhn's multi-coloring algorithm because it requires only one round of communication and ensures that each node in the graph being colored receives at least a fraction $1/(\Delta + 1)$ of the total colors assigned. We note that our metric of normalized sum-rate is directly related to the time slots assigned to the worst-case user [13]. Thus, given a fixed number of cliques containing a specific node, it is desirable to use the multicoloring algorithm in consolidated graphs which have smaller maximum degree, Δ.

With the objective to find the consolidated graph, G_ρ, with a smaller maximum degree, Δ, our major innovations occur in Step 2 to convert G to G_ρ. Here, we summarize the steps of our proposed algorithms.

1. Identification: Each node identifies all the potential cliques it can belong to. This step is mainly governed by the amount of network information available to identify r-cliques (for $r \leq \rho$) that achieve individual $\tilde{\alpha}(\eta, \tau) = 1$. With $\tau = \rho+1$ hops of connectivity information r-cliques ($r \leq \rho$) can be identified and with $\eta = \rho + 1$ hops of channel information, we ensure the normalized sum-rate of each r-cliques is 1.

2. Selection: Select which of the potential cliques from Step 1 will become vertices in the consolidated graph, G_ρ. This step is crucial to ensure that agreement in the distributed coloring process of Step 3 and overall improvement in normalized sum-rate are possible. We will describe two different algorithms for this step,

needing $\tau = 3\rho+3$ and $\tau = 3\rho+1$ hops of connectivity information respectively.

3. Scheduling: The consolidated graph, G_ρ, is colored using Kuhn's distributed multicoloring algorithm. Each sub-network receives a series of colors that represent time-slots. A group of sub-networks with the same color represents an independent sub-graph, and thus achieve normalized sum-rate equal to 1. The overall performance of the algorithm depends on the number of time-slots when the worst-case sub-network is active (as described in Chapter 3).

We describe the Identification, Selection, and Scheduling steps in more detail in the following chapters. We will propose two different methods for Step 2. The first method presents a conservative approach that ensures that the normalized sum-rate does not decrease when more knowledge is available with respect to the traditional distributed scheduling with complete interference avoidance. The conservative approach ensures improvement, but its conservative nature may limit gains. We also present an alternative method which is an aggressive approach where we do not ensure improvements for all graphs. The normalized sum-rate performance with this method is significantly better, on average, than the conservative approach for practical classes of graphs, such as random graphs and scale-free graphs, especially with smaller amount of local information.

Finally, we note that to derive our results we do not need to state the form of optimal coding methods used by each node in the identified sub-networks. The fact that we can analyze sum-rate without explicitly defining coding methods is possible due to the characteristics of normalized sum-rate.

Chapter 5

Step 1: Identification ($G \rightarrow G_\rho^-(v)$)

Let us begin by describing in detail the procedure followed in the Identification step. Consider a conflict graph G and a parameter ρ. We assume each user in the network has $\eta = \rho + 1$ hops of channel information and $\tau = \rho + 1$ hops of connectivity information. In Step 1, for a given ρ, we identify the r-cliques, $r = 0, ..., \rho$ which can be done with $\rho + 1$ hops of connectivity information. We are interested in these r-cliques because, with the available channel information, it is ensured that each r-clique can achieve $\tilde{\alpha}(\rho) = 1$. These potential cliques are the candidates to ultimately be represented by a vertex in the consolidated graph G_ρ.

Since each node has a different local view of the conflict graph, G, the potential cliques discovered by each node will be different. Thus, in Step 1, each node will generate a temporary graph where the potential cliques it sees are turned into vertices. We will denote the temporary graph from the point to view of node v as $G_\rho^-(v) = (W^-(v), F^-(v))$, which is described as follows. The set of vertices $W^-(v)$ represents all the r-cliques ($r \leq \rho$) in the part of the graph known to node v with $\rho + 1$ hops of connectivity information. Each node in $w \in W^-(v)$ maps to a set of nodes in the original conflict graph; we denote that set of nodes in the conflict graph represented by vertex w as $nodes(w)$. An edge exists between two vertices in $G_\rho^-(v)$, w_1 and w_2,

if there is an edge between a member of $nodes(w_1)$ and a member of $nodes(w_2)$ in G or if a member of $nodes(w_1)$ is also a member of $nodes(w_2)$.

The following example shows the construction $G \rightarrow G_\rho^-(v)$ with the parameter $\rho = 1$ using the example original conflict graph G in Figure 2.2.

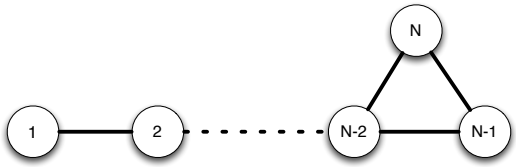


Figure 5.1: Line-clique graph with N nodes, G

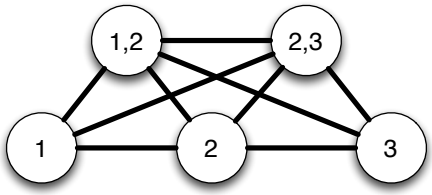


Figure 5.2: $G_1^-(1)$

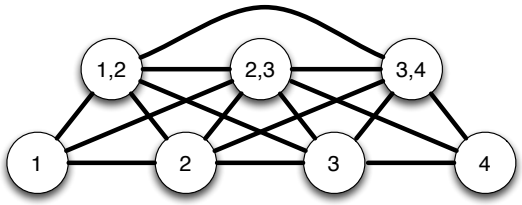


Figure 5.3: $G_1^-(2)$

Figure 5.2 shows what the graph of all potential vertices looks like from the point of view of node 1, which has 2 hops of connectivity knowledge. The vertices are labeled according to their corresponding set of nodes from the original conflict graph (in other words, the label of node w is $nodes(w)$). As we can see, node 1 observes 5 potential cliques, three 0-cliques ($\{1\}, \{2\}, \{3\}$) and two 1-cliques ($\{1, 2\}, \{2, 3\}$). There exists an edge between the vertices labeled $\{1\}$ and $\{1, 2\}$ because 1 is present

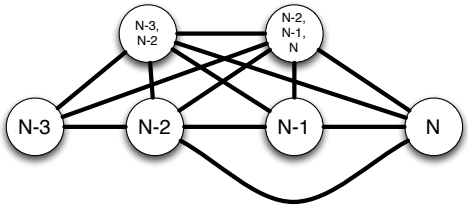


Figure 5.4: $G_1^-(N)$

in both vertices and because there is an edge between 1 and 2 in the original conflict graph. Similarly, there are edges between $\{1,2\}$ and $\{2\}$ and between $\{1\}$ and $\{2\}$ and so on.

Figure 5.3 depicts the graph $G_1^-(2)$. In this case, node 2 in G sees four 0-cliques: $\{1\}$, $\{2\}$, $\{3\}$, and $\{4\}$. Also, node 2 sees three different 1-cliques $\{1,2\}$, $\{2,3\}$ and $\{3,4\}$ that could be formed so there is a total of 7 vertices in the graph $G_1^-(2)$. The edges are generated following the rules explained earlier. Similarly, Figure 5.4 describes the graph $G_1^-(N)$. In this last case, notice that the vertex $\{N-2, N-1, N\}$ was created since it forms a 1-clique and node N belongs to it. An important point is the exclusion of the sets $\{N-1, N\}$ and $\{N-2, N\}$. These sets are not included as vertices in $G_1^-(N)$ because we have defined an r-clique as a maximal subset and both $\{N-1, N\}$ and $\{N-2, N\}$ are 1-cliques superseded by the 1-clique $\{N-2, N-1, N\}$, therefore $\{N-2, N-1, N\}$ is the only set that becomes a vertex in $G_1^-(N)$.

The clique identification process is easily extended for any $\rho > 1$ by identifying all r-cliques, for $r = 0, ..., \rho$. For example, if $\rho = 2$, $G_2^-(v)$ would consist of the full $G_1^-(v)$ plus all the 2-cliques in the 2-neighborhood of v, along with their respective edges. As we have mentioned, a larger ρ would increase the minimum amount of information required at each node. Also, finding maximal r-cliques is in general a hard problem, but since our goal is to leverage local information, we primarily concentrate on the cases of small r.

Up to this point, we have identified cliques that are made up of nodes that can

transmit simultaneously in an optimal manner with the available local knowledge. However, in order to assign time slots to each one of these cliques, there are two problems that need to be addressed. First, there is the issue that each node now has a graph with a maximum degree which is significantly higher the maximum degree of the original conflict graph. As we have described before, the maximum graph degree and normalized sum-rate achieved by our scheme are intimately related, so our goal is a consolidated graph, G_ρ, with smallest degree. The second issue is the fact that we need to ensure that a distributed coloring algorithm does not lead to coloring conflicts especially since the graphs seen by different nodes differ so much from each other. In the next section we will select which vertices of the graph $G_\rho^-(v)$ should remain and which should be pruned in order to reduce the maximum degree of the final consolidated graph, G_ρ, and to ensure that there will be no conflict in the use of a distributed coloring algorithm.

Chapter 6

Step 2: Selection ($G_\rho^-(v) \to G_\rho(v)$)

In this chapter, we will describe the Selection step, which consists of selecting which of the potential vertices in $G_\rho^-(v)$ identified by each node in Step 1 will be pruned and which will be kept in their own view of the final consolidated graph $G_\rho(v)$. We propose two different approaches to Step 2. The first approach is a conservative selection algorithm that allows each user to be represented by only one vertex in the consolidated graph and ensures that the normalized sum-rate of the network never decreases. The second approach allows each user to be represented by more then one vertex in the consolidated graph, and while strict guarantees cannot be provided, the gains in normalized sum-rate are significant in most classes of graphs, especially with small amounts of local knowledge.

6.1 Consevative Selection Algorithm

In Step 1, we created a graph $G_\rho^-(v)$ that consists of vertices that represent cliques from the original conflict graph and can simultaneously transmit in an optimal way. As discussed above, graphs $G_\rho^-(v)$ can have the maximum degree which is higher than G. The increase in maximum degree is expected; since cliques of nodes are now

transmitting simultaneously, their joint interference footprint is expected to grow. To guarantee improvement in normalized sum-rates, the simplest way is a conservative selection algorithm that satisfies two properties:

1. Each node v from the conflict graph G is represented by *only one* node in the consolidated graph $G_\rho(v)$

2. The degree of the vertex that represent v in the consolidated graph $G_\rho(v)$ is less than or equal to the degree of v in the conflict graph G.

Please note that we say user v in G is represented by vertex w in $G_\rho(v)$ if $v \in nodes(w)$. The two simple properties described above ensure that the procedure will achieve a normalized sum-rate of $\tilde{\alpha}(\rho) = 1/(\Delta_{G_\rho} + 1)$, where Δ_{G_ρ} is the maximum over all maximum degrees of the $G_\rho(v)$ graphs.

In Step 1, we assumed $\rho + 1$ hops of connectivity information. In this algorithm, we will assume $3\rho+3$ hops of connectivity knowledge. Generally speaking, the reason behind the significant amount of connectivity information required is that each node needs to know not only its own consolidated graph but also the consolidated graphs of its neighbors because the distributed multicoloring algorithm is a process with a 1 hop footprint. By requiring $3\rho + 3$ hops of connectivity knowledge we ensure that there will be no coloring conflicts in the Scheduling step (Step 3) of the algorithm. We label the conservative selection algorithm as Algorithm $\mathcal{A}_1(3\rho + 3)$

6.1.1 Example

Carrying on with the previous example, we illustrate the heuristics of the conservative selection algorithm. Let $\rho = 1$ and consider the temporary graphs of Figures 5.2, 5.3 and 5.4. For illustrative purposes, let us begin with graph $G_1^-(2)$ and node 2's conservative selection process. According to our required properties, described in

Section 6.1, to guarantee improvement in normalized sum-rate, node 2 can only be represented by one vertex in the graph $G_\rho(2)$. Node 2 must choose one of the three options $\{1,2\}$, $\{2,3\}$ or $\{2\}$ and the others must be pruned. When possible, we want to keep vertices that represent more nodes since that could result, intuitively, in consolidated graphs with less vertices and with smaller maximum degree. The first step is, therefore, to check if there exists a vertex that represents a unique maximum ρ-clique, meaning a unique ρ-clique with the maximum number of members. In this case, there are two cliques with two members each. Since we do not have a heuristic to prefer clique $\{1,2\}$ over clique $\{2,3\}$, to avoid conflicts with neighboring nodes about the clique chosen, node 2 concludes that neither of the two ρ-cliques will be selected to represent it in the consolidated graph. Therefore node 2 chooses the vertex $\{2\}$ as its vertex to keep. Similarly, it chooses all the single node vertices for all other nodes in the graph as shown in Figure 6.1.

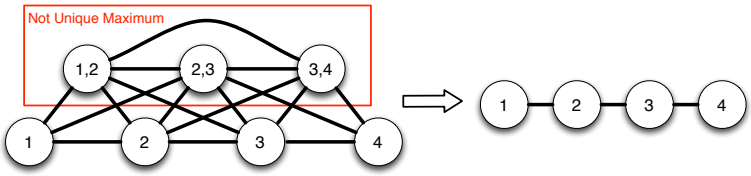


Figure 6.1: $G_1^-(2) \rightarrow G_1(2)$

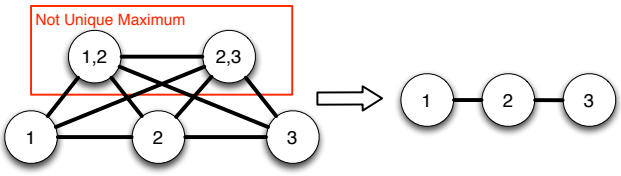


Figure 6.2: $G_1^-(1) \rightarrow G_1(1)$

Now, consider the graph $G_1^-(1)$ and which vertices node 1 will select to be kept in the consolidated graph $G_\rho(1)$. Node 1 sees that node 2 does not have a unique ρ-clique, therefore vertices $\{1,2\}$ and $\{2,3\}$ cannot be kept, so all the single nodes

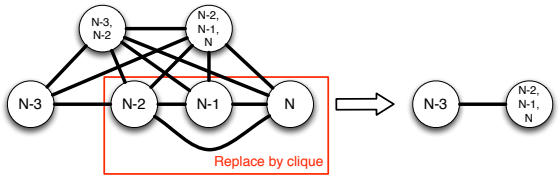


Figure 6.3: $G_1^-(N) \rightarrow G_1(N)$

vertices are chosen shown by Figure 6.2.

Finally, let us look at the graph $G_1^-(N)$ and node N's selection process. Node N has a unique maximum clique $\{N-2, N-1, N\}$ and, in this case, it is also the maximum clique for nodes $N-1$ and N. Also, the degree of the vertex representing that clique in $G_\rho(v)$ would be 1, which is smaller than the degree of nodes $N-2$, $N-1$ and N in G. Therefore, node N decides that it will be represented by the vertex $\{N-2, N-1, N\}$ in the consolidated graph, $G_\rho(N)$, shown in Figure 6.3. After the conservative selection process with $\rho = 1$, the graphs $G_\rho(v)$, for all $v \in G$ have, at most a maximum degree of 2, hence $\Delta_{G_\rho} = 2$. This means that the conservative selection algorithm achieves normalized sum-rate of $1/3$ which is an improvement over the simple distributed Independent Set Scheduling which achieves a normalized sum-rate of $1/4$.

In general, in the conservative selection algorithm, each node will be represented either by the vertex associated to a ρ-clique or a vertex associated to the single node. When the single node vertex is chosen, the algorithm is essentially taking the conservative approach and reverting back to the original conflict graph. In other words, a ρ-clique will be kept only when it is guaranteed to help. Throughout the whole process we have only been concerned with ρ-cliques and single nodes vertices (0-cliques). In cases where $\rho > 1$, all r-cliques in $G_\rho^-(v)$, for $0 < r < \rho$ are automatically pruned. Also note that if the diameter of the graph G is $3\rho + 3$ or less, we no longer have incomplete connectivity information and we can use the techniques in [13] or [40].

6.1.2 Conservative Selection Algorithm Description

Now that we have given a heuristic about the conservative selection process, we go ahead and provide a formal description. We begin with a given ρ and the assumption that each node knows $3\rho + 3$ hops of connectivity in the conflict graph $G = (V, E)$. After each node $v \in V$ performs the conservative selection algorithm, they will have generated a graph $G_\rho(v) = (W(v), F(v))$, where each $w \in W(v)$ represents a subset of nodes from the graph G.

To initialize the algorithm, each node v in G finds the maximum ρ-clique it belongs to and checks to see if it is unique. This is easily performed by inspection of the graph $G_\rho^-(v)$. If the unique maximum ρ-clique exists, we call that vertex representing that clique $w^*(v)$, otherwise node v will be represented by vertex $\{v\}$ in the consolidated graph. Given the existence of a unique maximum ρ-clique, node v needs to know the vertices representing every $u \in nodes(w^*(v))$, i.e., the $w^*(u)$ of every member in $nodes(w^*(v))$.

With this knowledge, node v must find the degree of $w^*(v)$ if it were to be kept in the final consolidated graph. Node v knows also the clique representing every neighbor of every node in $nodes(w^*(v))$ since it has $3\rho + 3$ hops of knowledge. That is, node v can create a set of these potential neighboring vertices $\mathcal{U}^-(v) = \{\bigcup_{z \in Z} w^*(z)\}$, where $Z = \bigcup_{u \in nodes(w^*(v))} \Gamma_G(u)$ and $\Gamma_G(u)$ is the set of neighbors of u in G, u inclusive.

Furthermore, v estimates which cliques from $\mathcal{U}^-(v)$ will be kept in the consolidated graph based on the $3\rho + 3$ hops of knowledge since with this amount of knowledge v has access to all potential cliques neighboring each one of the members of $\mathcal{U}^-(v)$. Therefore, v is able to generate the set $\mathcal{U}(v)$ which consists of the vertices that v considers will be present in the consolidated graphs. The degree of $w^*(v)$ in $G_\rho(v)$ if it were selected is the number of members of $\mathcal{U}(v)$, $\delta_{w^*(v)} = |\mathcal{U}(v)|$.

If $\delta_{w^*(v)} \leq \delta_u$ for all $u \in nodes(w^*(v))$, then $w^*(v)$ is formed and it will appear

as a clique in the consolidated graph $G_\rho(v)$. Otherwise, each $u \in nodes(w^*(v))$ will be represented by the 0-clique $\{u\}$ in $G_\rho(v)$. The summary of this algorithm can be found in Algorithm 1.

Algorithm 1 Conservative Selection $\mathcal{A}_1(3\rho + 3)$

Input: Graphs $G_\rho^-(v)$ for each node $v \in V$ and $3\rho + 3$ hops of connectivity information

Every node $v \in V$ find its unique maximum ρ-clique
if a unique maximum ρ-clique does not exists **then**
  The vertex labelled $\{v\}$ is a vertex in the final graph $G_\rho(v)$
else
  if $w^*(u) \neq w^*(v)$ for some $u \in nodes(w^*(v))$ **then**
    Every $u \in nodes(w^*(v))$ is represented by a vertex $\{u\}$ in the consolidated graph $G_\rho(v)$
  else
    if $\delta_{w^*(v)} > \delta_u$ for some $u \in w^*(v)$ **then**
      Every $u \in nodes(w^*(v))$ is represented by a vertex $\{u\}$ in the consolidated graph $G_\rho(v)$
    else
      $w^*(v)$ will be a vertex in the final graph $G_\rho(v)$
    end if
  end if
end if
The graph $G_\rho(v)$ consists of all the vertices representing cliques selected to be kept by node v.

6.2 Aggressive Selection Algorithm

In this subsection, we will present a second approach to selecting which vertices from the graphs $G_\rho^-(v)$ should be carried over to graphs $G_\rho(v)$. The conservative selection algorithm in the previous section ensures that $\alpha_1(\rho) \geq \alpha(0)$, for all ρ and for any arbitrary graph. However, since it must provide this strict guarantee, it tends to be overly conservative and loses potential gains in large classes of graph. To address this issue, we propose a second clique selection algorithm that is more aggressive.

The Aggressive Selection Algorithm relaxes the two major constraints of the conservative algorithm: 1) it allows nodes from the original conflict graph to be represented by more than one vertex in the consolidated graph and 2) the degrees of the vertices being kept for the consolidated graphs are allowed to be larger than the degrees in the conflict graph of the nodes that make up the vertices. We have mentioned that graphs with larger maximum degrees are undesirable, so the aggressive algorithm provides a heuristic that balances the maximum degree of the consolidated graphs with the number of vertices representing each node. Recall that the normalized sum-rate of a network is the fraction of active time slots of the worst-case node in the network. Using our proposed distributed procedure, this is simply $\min_{v \in V} a(v)/\Delta_{G_\rho}$, where $a(v)$ is the number of vertices in $G_\rho(v)$ representing node v from the original conflict graph. As long as the number of vertices representing each node in the network increases enough to counteract for the increase in the maximum degree of the consolidated graph, gains in normalized sum-rate can be achieved. We now describe the heuristic of the Aggressive Selection Algorithm.

We begin with some assumptions about the amount of network information available to each node in the network. For purposes of exposition, we will describe the general idea of the aggressive algorithm by assuming complete connectivity information and later show that only $3\rho + 1$ hops of connectivity information is needed. We denote this centralized aggressive selection algorithm by $\mathcal{A}_2(Full)$. The distributed form of the aggressive selection is denoted $\mathcal{A}_2(3\rho + 1)$. Also, we consider the idea of a temporary graph G_ρ^-. In Step 1 we described the process of each node obtaining $G_\rho^-(v)$, the graph G_ρ^- can be described as a "centralized temporary graph" with full topology information, but still forming cliques of at most diameter ρ. The graph G_ρ^- is a single graph that contains all the possible r-cliques, $r = 0, ..., \rho$, in the original conflict graph G. We show the aggressive algorithm's heuristic by using an example.

6.2.1 Example

Once again we present an example using the N-node line-clique graph depicted in Figure 2.2. We begin with the case where $\rho = 1$. Consider the graph G_ρ^- and the process the centralized entity uses to decide which vertices to keep and which vertices to prune to make the consolidated graph G_ρ.

The basis for the aggressive selection algorithm is quite simple. We wish to keep as many vertices from G_ρ^- as possible but there are some vertices that create a lot of interference and are somewhat redundant. For this reason, a vertex, w, representing a set of nodes, $nodes(w)$, is removed if every member of $nodes(w)$ appears more twice somewhere else in the vertices of graph G_ρ^-. For example, consider the 0-clique, $\{2\}$, since node 2 appears in the cliques consisting of $\{1, 2\}$ and $\{2, 3\}$, the 0-clique $\{2\}$ is removed. The intuition behind the selection process is that we wish to remove nodes to avoid interference and increasing degrees, but at the same time let every user in the original graph have enough contributions in order to have increasing normalized sum-rate. The aggressive selection step is illustrated in Figure 6.5. With these nodes removed, the final graph that is scheduled is shown in Figure 6.6.

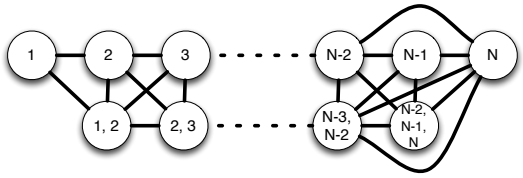


Figure 6.4: G_ρ^- for $\rho = 1$

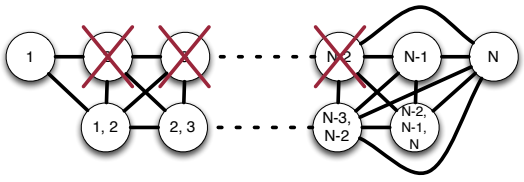


Figure 6.5: Cliques that appear twice elsewhere are removed

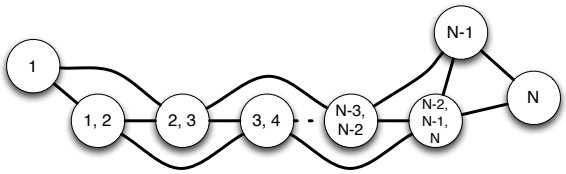


Figure 6.6: Final graph, G_1, after algorithm

Now, consider the algorithm when we begin by letting $\rho = 2$. First, all 1-cliques and 2-cliques are generated as in Figure 6.7. For clarity, we omit the edges is these graphs. Now, starting with the 0-cliques (i.e., the single nodes), they are removed if every member is present 2 times in any of the 1-cliques or 2-cliques. There is an exception to the removal of 0-cliques that establishes that all 0-cliques representing a node of degree 1 remain in the graph even if they are appear twice elsewhere. This is to avoid pathological cases of possible starvation of end nodes. In this case, clique 1 is not removed. We define the set of nodes with degree 1 in the conflict graph G as O_Λ.

Next, the 1-cliques are removed if they appear 2 times in the set of 2-cliques. It is important to note that i-cliques are removed only if they appear twice in the set of j-cliques, for all $j > i$. After performing this operation, illustrated in Figure 6.8, the final graph is shown in Figure 6.9.

Notice that the achievable normalized sum-rate in our example line-clique with $\rho = 0$ was $\alpha = 1/4$, with the aggressive selection algorithm when $\rho = 1$ then $\alpha_2(\rho) = 2/5$, and when $\rho = 2$ then $\alpha_2(\rho) = 3/7$. Also, compare to the conservative algorithm with achieves $\alpha_1(\rho) = 1/3$, for both $\rho = 1$ and $\rho = 2$. This increase in normalized sum-rate exemplifies the advantages of the aggressive algorithm.

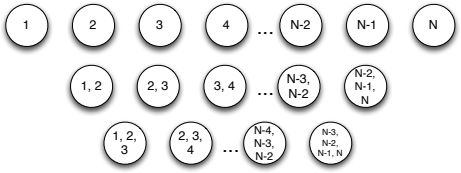


Figure 6.7: r-cliques, $r = 0, 1, 2$

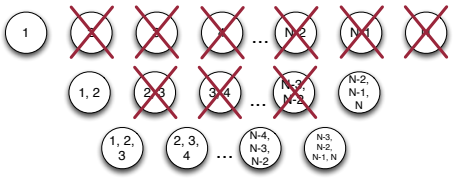


Figure 6.8: Cliques are removed

6.2.2 Distributed Aggressive Algorithm

We now show that the centralized aggressive selection algorithm described above can be performed in a distributed manner. Let G_ρ be the consolidated graph of the centralized algorithm and let $G_\rho(v)$ be the consolidated graph from node v's point of view of a distributed algorithm to be described in this subsection. The requirement that must be fulfilled is that each node performing Kuhn's multicoloring algorithm on $G_\rho(v)$ is equivalent to applying the multicoloring algorithm to G_ρ. Because Kuhn's algorithm only requires knowledge about the 1-hop neighbors, our individual nodes are only interested in knowing 1-hop information about G_ρ. In other words, our objective is to show that the neighborhood of vertices containing each node v in $G_\rho(v)$ is identical to their neighborhood in G_ρ. We now describe the distributed

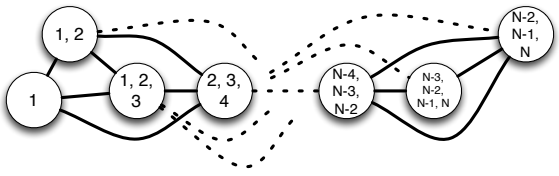


Figure 6.9: Final graph, G_2, after aggressive algorithm

aggressive selection algorithm.

We assume that each node has only $3\rho+1$ hops of topology information and that each node will be performing independent actions. Based on the removal heuristic of the centralized algorithm, each node will remove nodes that appear twice somewhere else in the graph they observe using the algorithm described in Algorithm 2. We define the set $O_{3\rho+1}(v)$ as the set of nodes of degree 1 in the $3\rho+1$-neighborhood of node v in the original conflict graph G. Also define $A_u^v(s)$ as the number of s-cliques representing node u in $G_\rho^-(v)$. The summary of the Distributed Aggressive Selection Algorithm is shown in Algorithm 2.

Algorithm 2 Ditributed Aggressive Selection $\mathcal{A}_2(3\rho+1)$

Input: Graphs $G_\rho^-(v)$ for each node $v \in V$ with $3\rho+1$ hops of topology information

1: The 0-cliques representing members of $O_{3\rho+1}(v)$ are ensured remain in $G_\rho(v)$.
2: **for** $r = 0$ **to** $\rho - 1$ **do**
3: Consider vertex $w \in W^-(v)$ (except those representing a node in $O_{3\rho+1}(v)$) that represents an r-clique in G. A vertex $w \in W^-(v)$ is removed from $G_\rho^-(v)$ if for every $u \in nodes(w)$

$$A_u^v(s) \geq 2, \text{for } s > i \text{ if } u \notin O_{3\rho+1}(v)$$
$$A_u^v(s) \geq 1, \text{for } s > i \text{ if } u \in O_{3\rho+1}(v)$$

4: **end for**
5: The graph $G_\rho^-(v)$ is updated by removing all identified nodes and their connecting edges. The final result is a graph $G_\rho(v)$.

6.2.3 Consistency of $\mathcal{A}_2(3\rho+1)$

We wish to establish that the $\mathcal{A}_2(3\rho+1)$ algorithm is a valid distributed implementation of algorithm $\mathcal{A}_2(Full)$. Since our final objective is to schedule G_ρ, all we have to do to establish the validity of $\mathcal{A}_2(3\rho+1)$ is to show that the neighborhood of cliques containing each node v in $G_\rho(v)$ is identical to their neighborhood in G_ρ. To do this we present the following theorem.

Theorem 1. *Let each node $v \in V$ have $3\rho + 1$ hops of knowledge. The neighborhood of every $v \in nodes(w)$ for every $w \in G_\rho(v)$ is identical to the neighborhood of w in G_ρ.*

Proof. We will break up the proof into two parts. First, we show that, from the point of view of a node v, the neighborhoods of every vertex representing v in G_ρ^- and $G_\rho^-(v)$ are identical, then we show that the cliques removed from each neighborhood are the same.

> **Lemma 1.** *Let each node $v \in V$ have $3\rho+1$ hops of connectivity knowledge. The neighborhood of every $w \in G_\rho^-(v)$ such that $v \in nodes(w)$ is identical to the neighborhood of w in G_ρ^-.*
>
> *Proof.* (Lemma 1) Let $\Gamma_{G_\rho^-}(w)$ be the neighborhood of vertex w in G_ρ^-. Also, consider $G_\rho^-(v)$, where v is a member of $nodes(w)$. There exists a vertex w' in $G_\rho^-(v)$ such that $nodes(w) = nodes(w')$, since there are at least ρ hops of topology knowledge. Now, let $\Gamma_{G_\rho^-(v)}(w')$ be the neighborhood of w' in $G_\rho^-(v)$. Because v has $3\rho+1$ hops of knowledge and the cliques being made are of, at most, diameter ρ, v has perfect knowledge of all cliques of up to diameter ρ that include nodes that are at most $2\rho + 1$ hops away. Now, every vertex in $\Gamma_{G_\rho^-}(w)$ represents nodes that are at most $2\rho+1$ hops away from v since each vertex in $\Gamma_{G_\rho^-}(w)$ has at most diameter ρ. Therefore $\Gamma_{G_\rho^-}(w) = \Gamma_{G_\rho^-(v)}(w')$. □

Now that we have shown all the correct neighborhood cliques are generated, we continue by showing that the correct cliques are removed as well.

> **Lemma 2.** *A vertex in $\Gamma_{G_\rho^-(v)}(w')$ is removed if and only if it is also removed from $\Gamma_{G_\rho^-}(w)$.*

Proof. (Lemma 2) We begin the proof by proving the forward direction, i.e., if a vertex is removed from $\Gamma_{G_\rho^-(v)}(w')$, it is also removed from $\Gamma_{G_\rho^-}(w)$. A vertex is removed from $\Gamma_{G_\rho^-(v)}(w')$ if and only if all its members appear two or more times somewhere else in the graph. By construction, for every u in a clique in $\Gamma_{G_\rho^-(v)}(w')$, $A_u^v(s) \leq A_u(s)$ for every $s = 1, ..., \rho$. This is because the vertices in $G_\rho^-(v)$ are a subset of the vertices in G_ρ^-. Therefore, a vertex is removed from $\Gamma_{G_\rho^-(v)}(w')$, it is also removed from $\Gamma_{G_\rho^-}(w)$.

In the other direction, we prove that if a clique is removed from $\Gamma_{G_\rho^-}(w)$, it is also removed from $\Gamma_{G_\rho^-(v)}(w')$. A clique is removed from $\Gamma_{G_\rho^-}(w)$ if and only if all its members appear twice elsewhere in the graph. All cliques in $\Gamma_{G_\rho^-}(w)$ are composed by nodes at most $2\rho + 1$ hops away from v. Since every formed clique is at most diameter ρ, every appearance of nodes that compose the $\Gamma_{G_\rho^-}(w)$ can only occur in cliques with members that are at most $2\rho + 1 + \rho = 3\rho + 1$ from v. Since v has this amount of knowledge, if a clique is removed from $\Gamma_{G_\rho^-}(w)$, it is also removed from $\Gamma_{G_\rho^-(v)}(w')$. □

Since $\Gamma_{G_\rho^-}(w) = \Gamma_{G_\rho^-(v)}(w')$ and a node is removed from $\Gamma_{G_\rho^-(v)}(w')$ if and only if it is also removed from $\Gamma_{G_\rho^-(v)}(w')$, for any arbitrary w and v, then we have that the neighborhood of every $v \in nodes(w)$ for every $w \in G_\rho(v)$ is identical to the to the neighborhood of w in G_ρ. □

In general, once the algorithm has been shown to be consistent we can describe the graph G_ρ as the union of all $G_\rho(v)$ and each $G_\rho(v)$ is a local view subgraph of the G_ρ graph centered around node v. The algorithm is finalized by each node applying Kuhn's multicoloring to the graph $G_\rho(v)$. The normalized sum-rate achieved by the Aggressive Selection Algorithm $\mathcal{A}_2(3\rho + 1)$ with parameter ρ is $\alpha_2(\rho) = \min_{v \in V} a(v)/\Delta_{G_\rho}$, where $a(v)$ is the number of vertices in $G_\rho(v)$ representing node v and $\Delta_{G_\rho} = \max_{v \in V} \Delta_{G_\rho(v)}$.

Chapter 7

Step 3: Scheduling

In Step 2 of the algorithms, each user v in the network finishes with a graph $G_\rho(v)$ that is composed of nodes representing r-cliques of at most diameter ρ and at least one of those nodes includes user v. The resulting r-cliques from Step 2 indicate that whenever user v transmits, it will do so along with all the other users that are members of the r-cliques that include user v using the optimal physical layer scheme. In the third step of the proposed algorithms, users in the network determine the time-slots when their respective r-cliques have been assigned to transmit. In other words, in Step 3, the graphs formed in Step 2 are scheduled. One approach to schedule nodes in a graph is to use graph coloring [41]. The problem of minimizing the number of required colors to color a graph has been a widely studied [42, 43, 44]. In [45] it was shown that coloring a graph with the optimal number of colors, defined as the *chromatic number*, is an NP-hard problem, even for a centralized algorithm with full connectivity information. Also, some distributed algorithms, including [43] and [44] can achieve coloring with $O(\Delta)$ colors in $O(\log N)$ number of rounds, where Δ is the maximum degree of the graph and N is the number of nodes in the graph. Other variations of distributed algorithms exploit specific graph structures to reduce complexity [46, 47].

To perform scheduling in our algorithms, we use a local multicoloring algorithm introduced by Kuhn in [15] since it is a one-shot algorithm and does not add to the number of hops of network information required for execution. First, we describe the local multicoloring algorithm in terms of a normal graph and explain the performance that can be expected, then we go into the detail of how this algorithm is used in our Step 3 of our algorithms.

7.1 Local Multicoloring Algorithm

Consider a graph $G = (V, E)$ with N nodes. We assume each node knows the number of users, N, and a parameter, k. The local multicoloring algorithm proceeds in three steps:

1. Each node $v \in V$ generates a vector $L_v = [l_{v,1}, l_{v,2}, \ldots, l_{v,k}]$ of k random numbers, where each $l_{v,i}$ is chosen uniformly from the set $\{1, 2, \ldots, kN^4\}$.

2. Each node v sends the vector L_v to all its neighbors. We call the set of neighbors of node v, $\Gamma(v)$. Each node v also receives the vectors L_u, for all $u \in \Gamma(v)$

3. Each node v acquires all colors i for which $l_{v,i} < l_{u,i}$, for all $u \in \Gamma(v)$.

The results in [15] show that if k is chosen to be greater that or equal to $6(\Delta + 1)\ln(N)/\varepsilon^2$, then each node v will acquire at least a fraction $\frac{1-\varepsilon}{\delta_v+1}$ of the k colors available, with high probability. It is important to note that one could relax the assumption of having to know N and k (which requires knowledge of Δ) and only require knowledge of an upper bound on N, denoted as $\overline{N}$, and a predetermined, network-wide ε. With this information, each node would know that the length of the random number vector it has to choose is $k = 6(\overline{N} + 1)\ln(\overline{N})/\varepsilon$ and each random number would be uniformly chosen from the set $\{1, 2, \ldots, k\overline{N}^4\}$.

7.2 Application

The local multicoloring algorithm can be used to schedule the sub-networks that have been formed in Step 2. We begin by assuming each node v in the original conflict graph G generates a random number vector $L_v = [l_{v,1}, l_{v,2}, \ldots, l_{v,k}]$ of k random numbers, where each $l_{v,i}$ is chosen uniformly from the set $\{1, 2, \ldots, k\overline{N}^4\}$ and

$$k = 6(\overline{N} + 1)\ln(\overline{N})/\varepsilon^2. \tag{7.1}$$

Consider the graph $G_\rho(v) = (W(v), F(v))$ from the point of view of node v. We have assumed that each node has $\tau = 3\rho + 3$ hops of connectivity information for the conservative algorithm and $\tau = 3\rho + 1$ hops of connectivity information for the aggressive algorithm. We now assume that in the process of information exchange to learn the connectivity of the network, the random vectors from all nodes τ hops away are also learned by each node. This means that every user v in the original conflict graph, G, knows all the random number vectors for all users in its r-clique(s) and all the random number vectors for all members of its neighboring r-cliques in $G_\rho(v)$.

Each node v finds the node with the smallest ID in each $w \in W(v)$. Node v assumes that the random number vector for vertex $w \in W(v)$ is the random number vector corresponding to the node with the smallest ID in that r-clique. In other words,

$$L_{w \in W(v)} = L_{\min\{x : x \in nodes(w)\}}. \tag{7.2}$$

Once the random vectors for each vertex in $G_\rho(v)$ have been identified, we can apply the local multicoloring algorithm such that node v knows the colors assigned to all r-cliques to which it belongs. A vertex $w \in G_\rho(v)$ will be assigned time-slot i if $l_{w,i} < l_{z,i}$, for all $z \in \Gamma(w)$. If node v is represented by vertex w, then node v knows L_w and L_z for all $z \in \Gamma(w)$ because we assume each node knows the random number

vectors of nodes τ hops away. This also ensures that all L_w are consistent over all $G_\rho(v)$.

Using the result from [15], it can be concluded that each r-clique represented by some vertex w will be assigned a fraction at least

$$\frac{1-\varepsilon}{\delta_w+1} \tag{7.3}$$

of the total k time-slots assigned with high probability, where δ_w is the degree of vertex w in the graph G_ρ.

7.3 Overhead

Let us analyze the overhead of the Step 3 in our algorithms in more detail. First, we note that in terms of hops of information, this step does not require any extra hops beyond the τ hops of connectivity we assumed in Step 2. We have assumed that as the connectivity information is being exchanged the random number vectors required for local multicoloring are also being communicated. The sharing of these vectors requires communication by each node of $6(\overline{N}+1)\ln(\overline{N})/\varepsilon^2$ random numbers, each with a possible magnitude of up to $k\overline{N}^4$. This results in generation and communication of $\mathcal{O}(\overline{N}\log^2(\overline{N})/\epsilon^2)$ random bits by each node. We can use the same non-trivial probabilistic argument mentioned in [15] to claim the same results with only $\mathcal{O}(\log(\overline{N}))$ bits required.

Another important consideration to keep in mind is that we have assumed that the parameter k can be chosen arbitrarily at the cost of complexity and amount of random bits to be exchanged. Since k represents the number of time-slots to be assigned assuming a static graph G_ρ, in practical applications, the value of k might be constrained by the coherence time of the network. In the description of our algorithms

we have assumed that k is smaller than the connectivity and channel coherence time of the network.

Chapter 8

Results

We first characterize the normalized sum-rate, $\tilde{\alpha}(\rho)$, achieved by the proposed algorithms with (η, τ) hops of network information. Both algorithms conclude with a set of graphs, $G_\rho(v)$, $\forall v \in G$, and the performance of both proposed algorithms can be described in terms of the topology characteristics of the consolidated final graph, G_ρ. The graph G_ρ is the union over all graphs $G_\rho(v)$. The normalized sum-rate performance of the algorithms is described in the following theorems.

Theorem 2. *Consider a conflict graph, G, where each user has η hops of channel information and τ hops of connectivity information. Using the conservative sub-network scheduling algorithm, the achievable normalized sum-rate is*

$$\tilde{\alpha}_1(\eta, \tau) = \tilde{\alpha}_1(\rho) = \frac{1 - \varepsilon}{\Delta_{G_\rho} + 1}, \tag{8.1}$$

with high probability, where Δ_{G_ρ} is the maximum degree of graph G_ρ and $\varepsilon > 0$.

Proof. We use the result from [13] regarding the normalized sum-rate of independent graph scheduling. The results says that if a network is divided into t sub-graphs, $\mathcal{A}_1, ...\mathcal{A}_t$ (not all distinct, for some t) and each user i belongs to d_i independent

sub-graphs, then the normalized sum-rate of the network is

$$\min_{i\in 1,2,\dots N} \frac{d_i}{t}. \tag{8.2}$$

In our sub-network scheduling algorithms, we have generated k independent sub-graphs. A set of sub-networks that share one of the k colors is an independent sub-graph since is composed by a set of sub-networks, each with a normalized sum-rate of 1, that do not interfere with each other.

By properties of the local multicoloring algorithm, each sub-network $w \in G_\rho$ will be assigned $\left(\frac{1-\varepsilon}{\delta_w+1}\right) k$ colors in total. Since in the conservative algorithm each user can only be represented by one sub-network, a user j in sub-network w appears in $d_j = \left(\frac{1-\varepsilon}{\delta_w+1}\right) k$ sub-graphs. Therefore,

$$\tilde{\alpha}_1(\rho) = \min_{i\in 1,2,\dots N} \frac{d_i}{t} \tag{8.3}$$

$$= \min_{w\in G_\rho} \frac{\left(\frac{1-\varepsilon}{\delta_w+1}\right) k}{k} \tag{8.4}$$

$$= \frac{1-\varepsilon}{\Delta_{G_\rho}+1}. \tag{8.5}$$

□

Similarly, we also describe the performance of the aggressive sub-network scheduling algorithm.

Theorem 3. *Consider a conflict graph, G, where each user has η hops of channel information and τ hops of connectivity information. Using the aggressive sub-network*

scheduling algorithm, the achievable normalized sum-rate is

$$\tilde{\alpha}_2(\eta, \tau) = \tilde{\alpha}_2(\rho) = \min_{v \in G} \sum_{v \in nodes(w)} \frac{1-\varepsilon}{\delta_w + 1}, \tag{8.6}$$

with high probability.

Proof. In the case of the aggressive algorithm, each user can be represented in more than one sub-network. Each sub-network will be active a total of $\left(\frac{1-\varepsilon}{\delta_w+1}\right) k$ time-slots. Hence, the number of time-slots each user will be active is the sum of all the time-slots the sub-networks to which it belongs are active, in other words,

$$d_i = \sum_{i \in w} \left(\frac{1-\varepsilon}{\delta_w + 1}\right) k \tag{8.7}$$

The worst-case node in terms of active time-slots gives us the normalized sum-rate of the network:

$$\tilde{\alpha}_2(\rho) = \min_{i \in 1,2,\ldots N} \frac{d_i}{t} \tag{8.8}$$

$$= \min_{v \in G} \frac{\sum_{v \in w} \left(\frac{1-\varepsilon}{\delta_w+1}\right) k}{k} \tag{8.9}$$

$$= \min_{v \in G} \sum_{v \in nodes(w)} \frac{1-\varepsilon}{\delta_w + 1}. \tag{8.10}$$

□

8.1 Conservative Algorithm Guarantee

The key objective of the proposed work is to provide techniques that harness the availability of local connectivity and channel information to improve the performance of a network in terms of generalized normalized sum-rate. This goal is achieved by

the proposed Conservative Algorithm. The main characteristic of the conservative algorithm is that, by leveraging local information when $\rho \geq 1$, the normalized sum-rate is ensured to be greater than or equal than the normalized sum-rate achieved by local multicoloring of the original network, G.

Theorem 4. *Let $\tilde{\alpha}_1(\rho)$ be the normalized sum-rate of a network after applying Algorithm $\mathcal{A}_1(3\rho+3)$ to the original graph G. If $\alpha(0)$ is the normalized sum-rate achieved by distributed multicoloring of the original network, G, then $\alpha(0) \leq \alpha_1(\rho)$, for $\rho \geq 1$.*

Proof. Since our overall distributed scheduling algorithm will conclude with the use of Kuhn's algorithm, the normalized sum-rate of the network is governed by the maximum degree of the final graph being scheduled. Using Kuhn's distributed multicoloring, $\alpha(0) = 1/(\Delta_G + 1)$, where Δ_G is the maximum degree of the original graph, G. Now, Algorithm $\mathcal{A}_1(3\rho + 3)$ ensures that, for every $v \in V$, the maximum degree of graph $G_\rho(v)$, $\Delta_{G_\rho(v)}$, is less than or equal to Δ_G. Since the proposed algorithm is an instance of Independent Graph Scheduling, the achievable normalized sum-rate is the fraction of active time slots of the worst-case user. Since Kuhn's multicoloring assigns a fraction of at least $1/(\Delta_{G_\rho(v)} + 1)$ to each clique in $G_\rho(v)$ and each v only appears once in $G_\rho(v)$, the worst-case user is active a fraction $1/(\Delta_{G_\rho} + 1)$, where Δ_{G_ρ} is the largest maximum degree over all $\Delta_{G_\rho(v)}$. Therefore, for every $\rho \geq 1$,

$$\alpha(0) = \frac{1}{\Delta_G + 1} \leq \frac{1}{\Delta_{G_\rho(v)} + 1} = \tilde{\alpha}_1(\rho). \tag{8.11}$$

□

We compare our algorithm's performance to the distributed multicoloring algorithm of the original graph to highlight the advantages of leveraging local information. The distributed multicoloring algorithm serves as a reasonable baseline of performance for one-shot algorithms. In contrast, other algorithms such as distributed greedy

scheduling [5] or randomized maximal schedulers [21] consist of rounds of exchanges to make decisions. By making our algorithm a one-shot algorithm, we ensure that the amount of knowledge required is constrained to $3\rho + 3$ hops. This quantifiable guarantee cannot be made under algorithms that involve several rounds as knowledge about the network propagates with each round. We address the performance, overhead, and complexity of several algorithms more in detail in Chapter **??**.

8.2 Numerical Results: Normalized Sum-Rate

In this section, we present results that compare the performance of the Conservative and the Aggressive Selection Algorithms. First, we present the performance of both algorithms in two example graphs for several values of the parameter $\rho = 0, 1, 2, 3$. In these results, $\rho = 0$ reflects the case when there is no topology information and Kuhn's multicoloring algorithm is performed directly on the conflict graph, G. The two sample graphs being compared are the N-node line-clique, which has been presented as an example throughout this paper, and the N-node line-star graph shown in Figure 8.1.

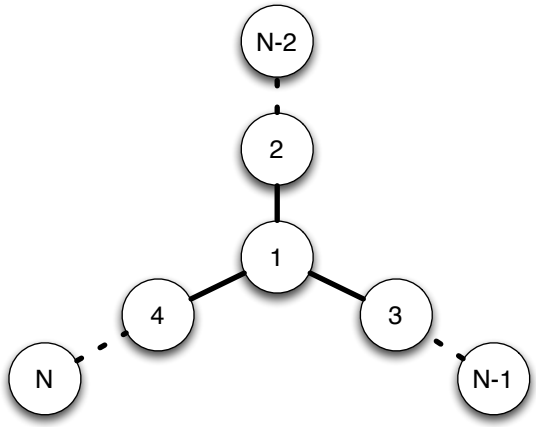


Figure 8.1: N-node Line-star Graph

The results shown in Figure 8.2 show that in both of these example graphs, the Aggressive Selection Algorithm outperforms the Conservative Selection Algorithm and the gain increases as the diameter of the cliques being formed increases. The results on these sample graphs expose some of the limitations of the conservative algorithm, namely, the need for a *unique* maximum ρ-clique to exist in order to form cliques. In highly symmetrical graph such as the ones in these examples, the conservative algorithm provides marginal gains. On the other hand, it is precisely in these situations where the aggressive algorithm displays its strengths.

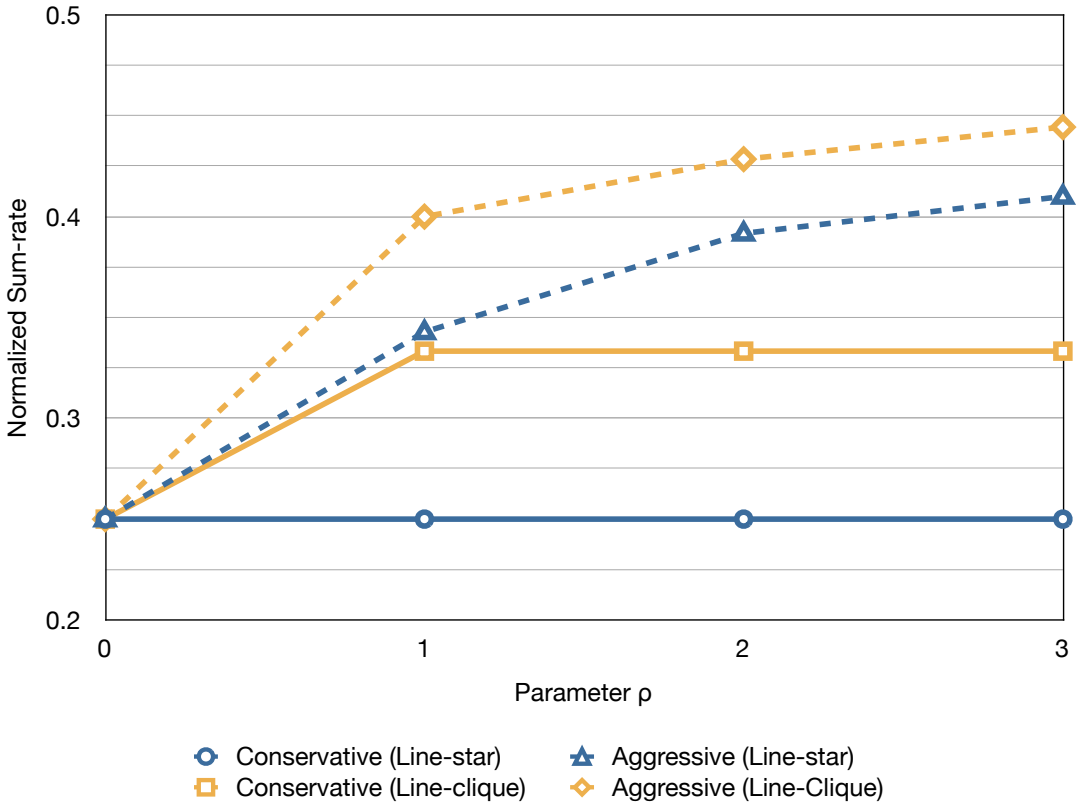


Figure 8.2: Conservative vs. Agressive Algorithms - Sample Graphs

Given that our work builds on the difficulty of obtaining global information, we are especially concerned with small amounts of local information. We are also interested in the algorithms' performance for classes of graphs that are representative

of wireless network scenarios. We present a comparison between the performance of four different algorithms for different classes of graphs and with parameter $\rho = 1$. The algorithms selected for comparison are distributed coloring, greedy scheduling (maximal scheduler), our conservative algorithm, and our aggressive algorithm. The greedy distributed scheduling algorithm that produces maximal schedules is described as follows:

1. Assign a randomized ordering to the nodes in the network
2. Following the assigned order, a node is added to the schedule if it has packets to send and none of its interfering nodes have been scheduled

Note that the greedy algorithm described here requires full network information. There are distributed implementations of similar greedy scheduling algorithms that require rounds of communication with neighbors in the network.

The simulations are presented for three different classes of graphs. We first look at Random Graphs, $\mathcal{G}(n, p)$, with n nodes and edge probability p. A graphical demonstration of a sample random graph with low connectivity ($p = 0.1$) is shown in Figure 8.3 and with high connectivity ($p = 0.1$) in Figure 8.4. Then we simulate algorithm performance in random scale-free graphs generated using the B-A algorithm [48]. The degree distribution of scale-free graphs follows a power scale law and is a good representation of sensor networks. An example of a scale-free graph with 20 nodes is shown in Figure 8.5. The third class of graphs is random geometric graphs, in which n transmitter-receiver pairs are randomly placed with uniform distribution in a unit square and interference occurs if any transmitter is within a diameter d of a receiver from another user. An example of a geometric graph is shown in Figure 8.6.

To evaluate algorithm performance, for each class of graph and each parameter setting 100 independent random graphs are generated and the average normalized

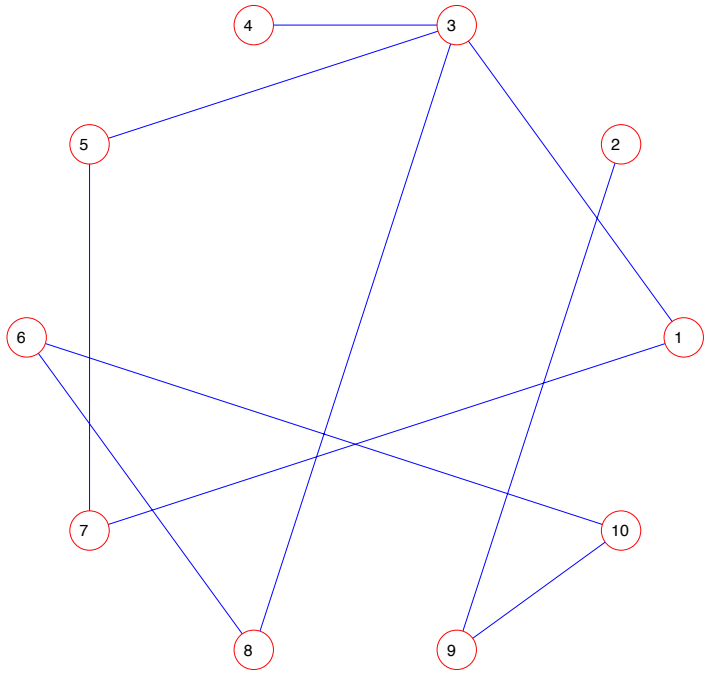


Figure 8.3: Sample Random Graph with Low Connectivity, $p = 0.1$

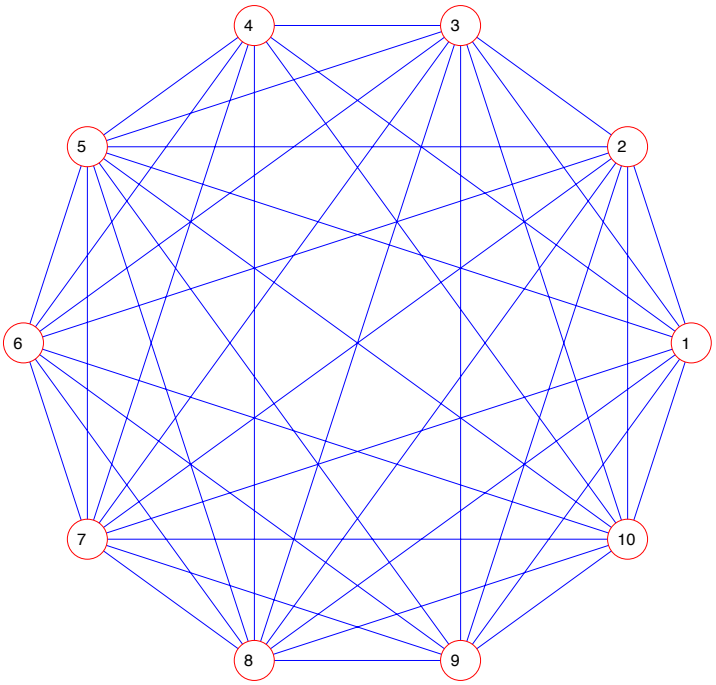


Figure 8.4: Sample Random Graph with High Connectivity, $p = 0.9$

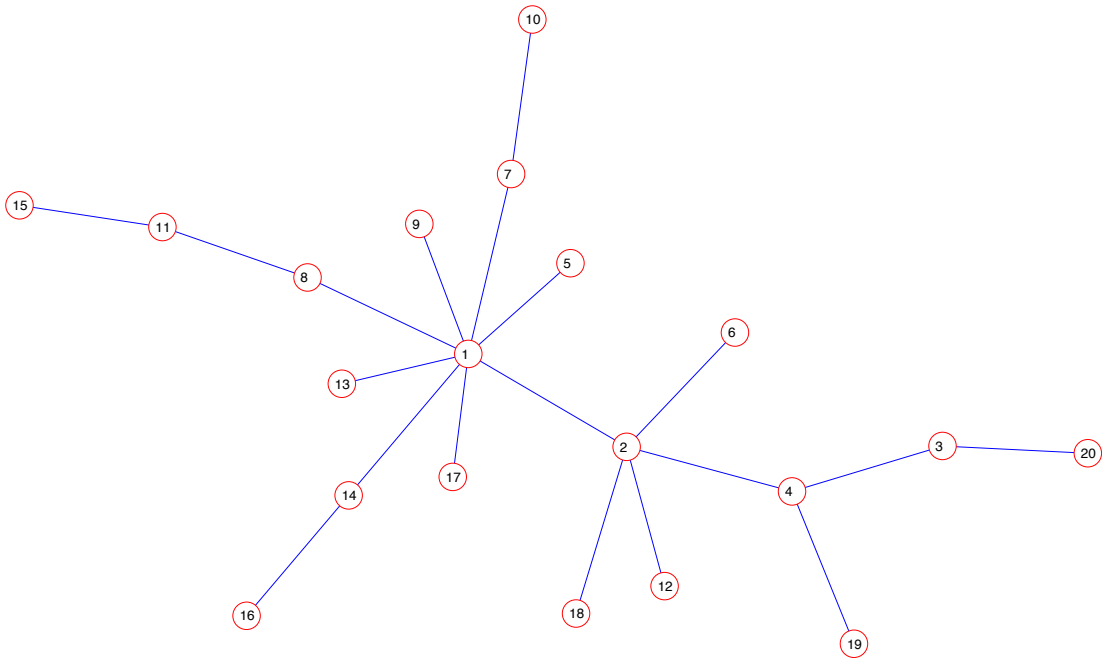


Figure 8.5: Sample Scale-free Graph

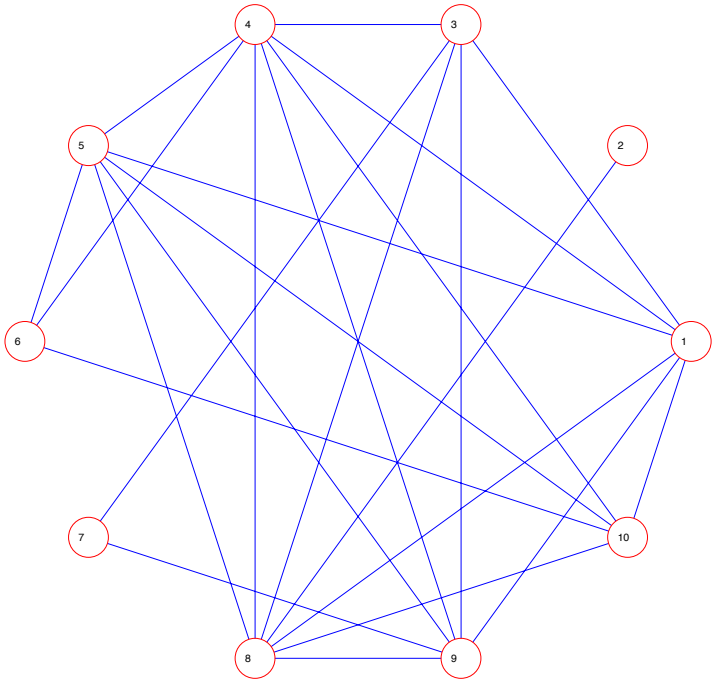


Figure 8.6: Sample Geometric Graph

sum-rate of each algorithm is reported. Figure 8.7 shows the performance comparison of random graph with parameters $\mathcal{G}(n, 0.1)$, Figure 8.8 describes performance for

graphs with $\mathcal{G}(n, 0.5)$, and Figure 8.9 for graphs with $\mathcal{G}(n, 0.9)$. In Figure 8.10 we report algorithm performance for the class of scale-free graphs with 25, 50, and 100 users. Finally, in Figure 8.11 we see the performance comparison in the class of geometric graphs with interference diameter $d = 0.25$ and in Figure 8.12 the performance in geometric graphs with interference diameter is $d = 0.5$.

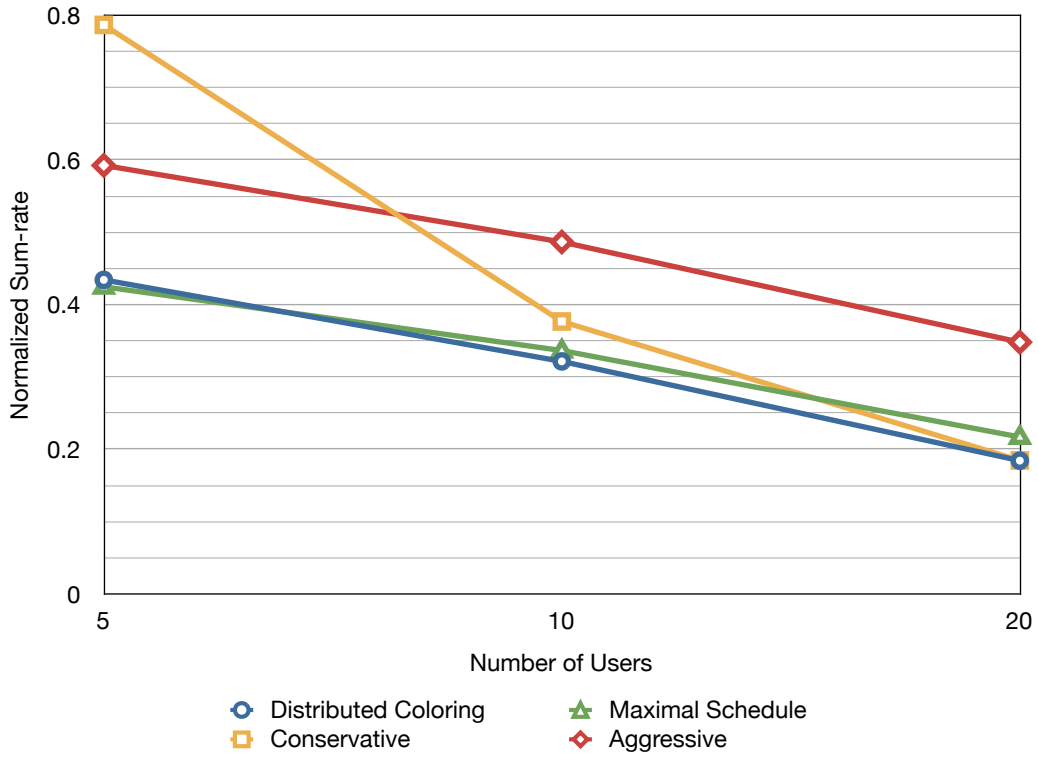


Figure 8.7: Normalized sum-rate performance comparison in random graphs with low connectivity, $p = 0.1$

The aggressive algorithm outperforms the distributed coloring and greedy (maximal schedule) algorithms in all this cases. Some notable points about the results include the performance of the conservative algorithm in the highly connected random graphs ($p = 0.9$) where cliques with large number of nodes are readily present and the conservative algorithm is able to outperform distributed coloring and maximal scheduling. Also, note the performance in the case of scale-free graphs, where

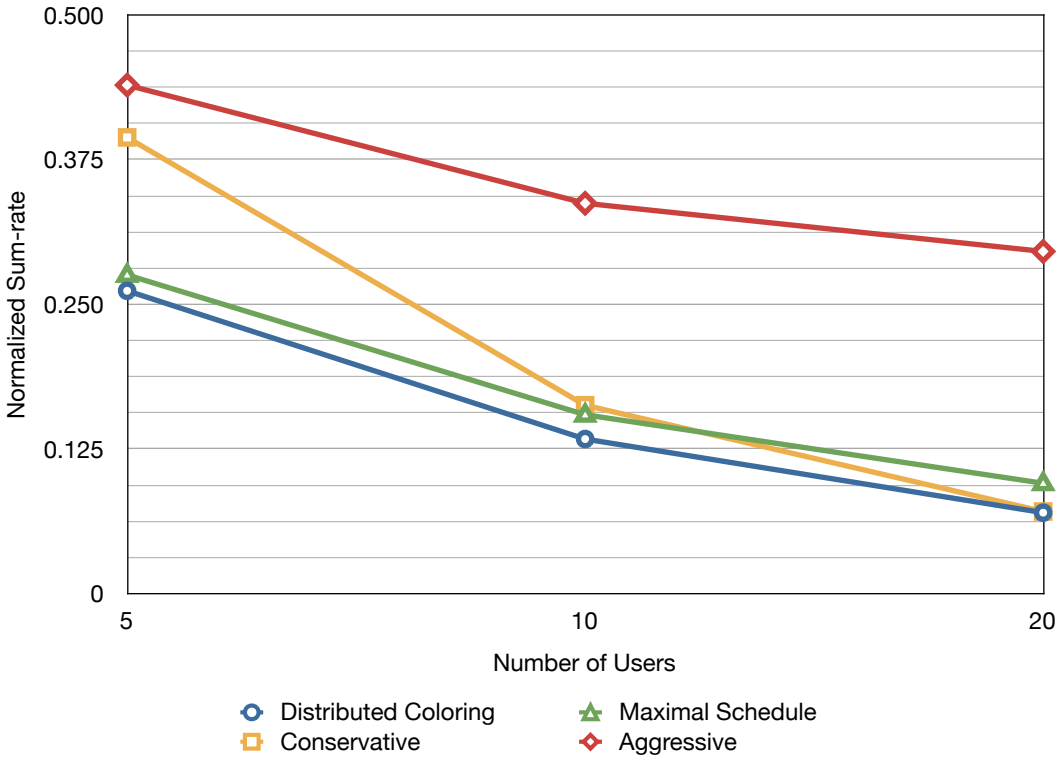


Figure 8.8: Normalized sum-rate performance comparison in random graphs with medium connectivity, $p = 0.5$

the conservative algorithm could often not ensure gains and so it remained conservative, and close in performance to distributed coloring, while the aggressive algorithm formed cliques for gains in normalized sum-rate.

8.3 Net Sum-rate Comparisons

We have presented two distributed sub-network scheduling algorithms and analyzed their normalized sum-rate performance. The algorithms show performance improvements and provide schemes that can be implemented with limited local information in terms of connectivity and channel states. Performance in terms of normalized sum-rate is an important result because it represents the guaranteed fraction of what

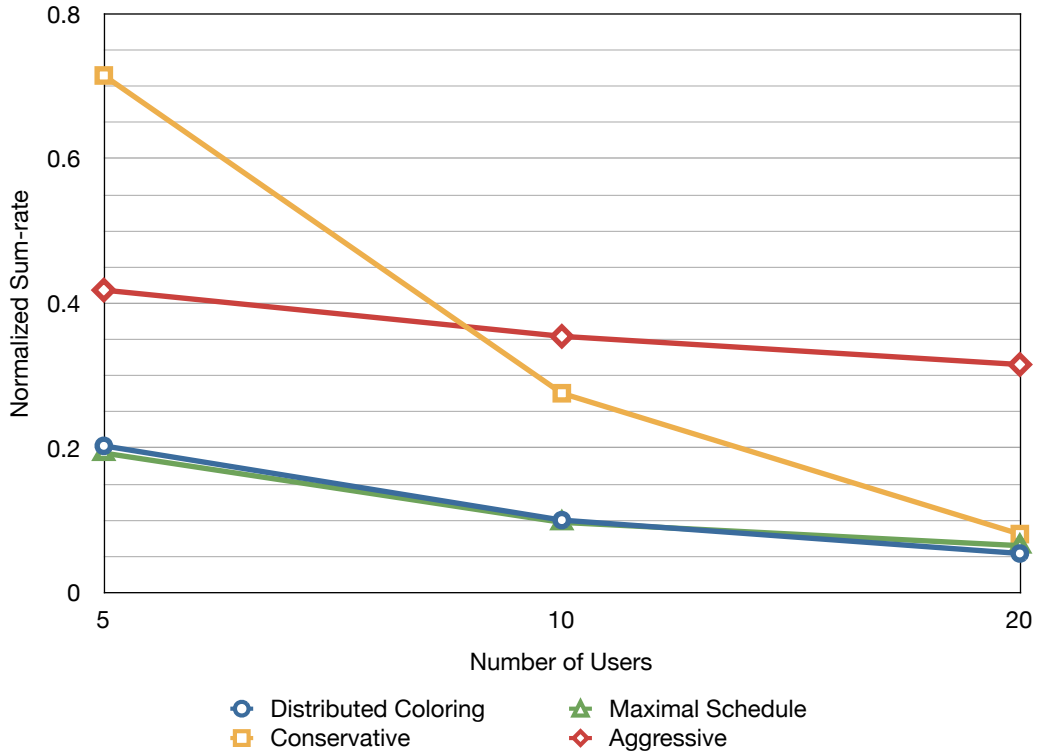


Figure 8.9: Normalized sum-rate performance comparison in random graphs with high connectivity, $p = 0.9$

could be achieved with the optimal strategy and full knowledge. Nevertheless, since it provides a guarantee, the normalized sum-rate provides worst-case performance behavior which could be significantly different from average performance in terms of other metrics of performance, for example *net* sum-rate. Along with consideration of net sum-rate, it is also important to compare the proposed algorithms to other known distributed algorithms in terms of overhead and the amount of information required to execute them. We comment on the performance of our conservative and aggressive algorithms and compare them to distributed coloring and maximal scheduling in terms of net sum-rate. Also, we discuss the amount and type of overhead of our algorithms and of state-of-the-art distributed scheduling algorithms.

Net sum-rate describes network performance without taking into account what

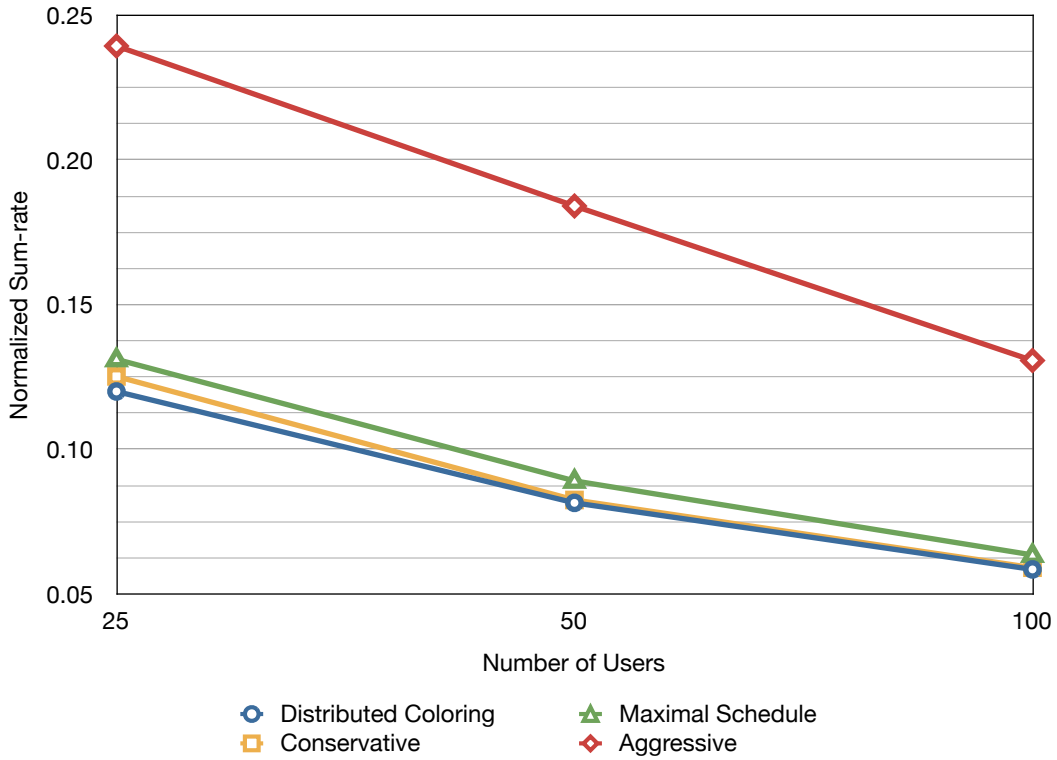


Figure 8.10: Normalized sum-rate performance comparison in scale-free graphs

the optimal, full-knowledge capacity of the network is. While net sum-rate provides a practical measure of net throughput performance, characterization for arbitrary networks is unavailable due to the open problem of the capacity of the general interference channel. Furthermore, net sum-rate performance using our algorithms is dependent on the topology of the network, making direct comparisons with existing scheduling algorithms problematic.

Here, we provide a couple of examples that illustrate what type of performance we can expect from our algorithm when compared to a well-known distributed independent set scheduling algorithm of queue-based systems in terms of net sum-rate.

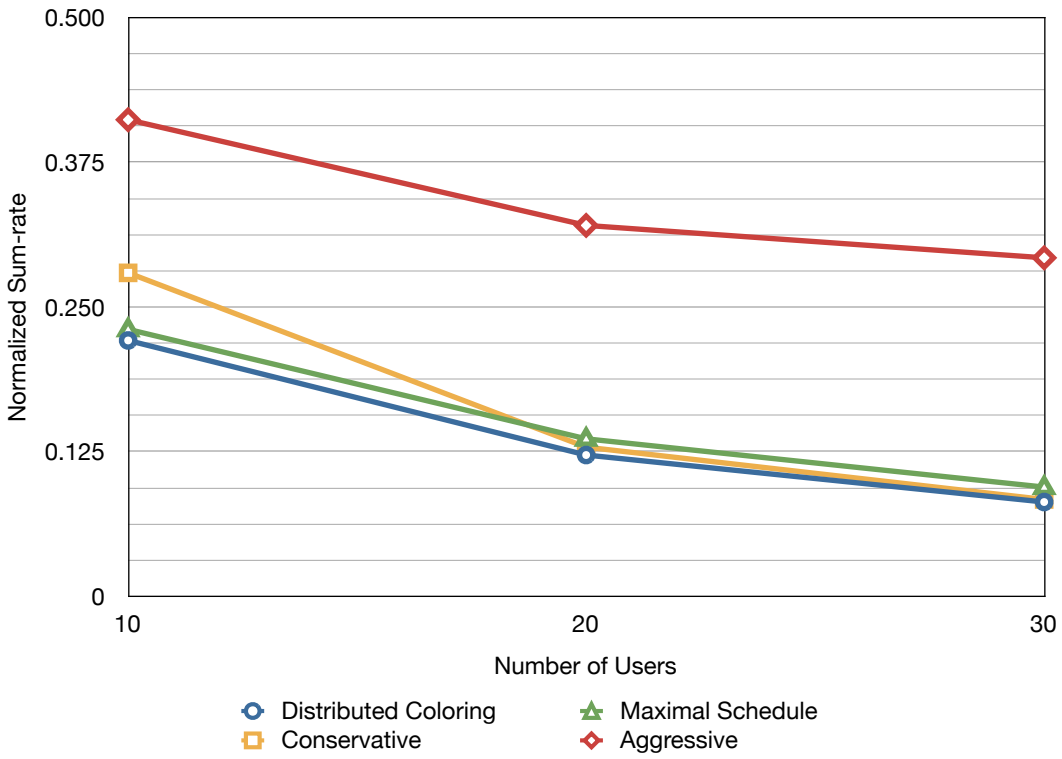


Figure 8.11: Normalized sum-rate performance comparison in geometric graphs, $d = 0.25$

8.3.1 Example 1

Consider the topology in Figure 8.13, which represents a 6-node network arranged in two cliques of size three that are connected by an single edge.

Assume each user i has a capacity C_i when no interference is present and the arrival rate to user i is λ_i. Using the greedy distributed scheduling, if the original ordering is randomized uniformly over the 6 nodes and $\lambda_i = 1$ for $i = 1, 2, ..., 6$, nodes 1, 2, 5, and 6 are scheduled 3/8 of the time and nodes 3 and 4 are active 1/4 of the time, with high probability. This results in an achievable sum-rate using the greedy algorithm of

$$R_{sum}^{greedy} = \frac{3}{8}(C_1 + C_2 + C_5 + C_6) + \frac{1}{4}(C_3 + C_4) \tag{8.12}$$

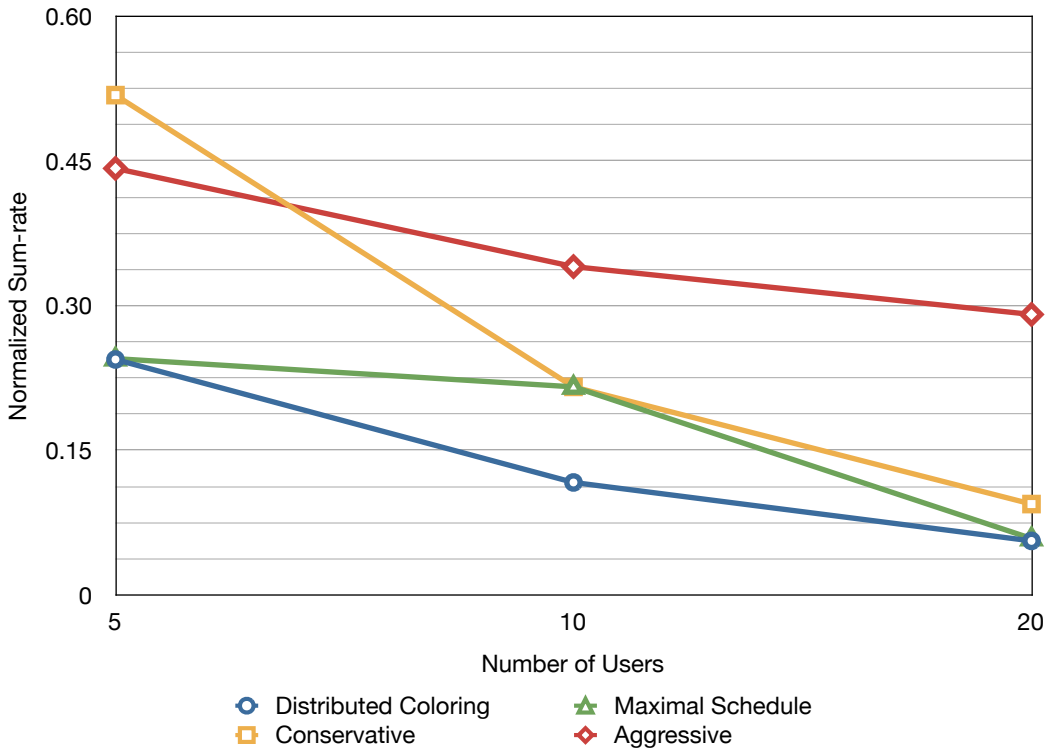


Figure 8.12: Normalized sum-rate performance comparison in geometric graphs, $d = 0.5$

and when we assume symmetric capacities, $C_i = 1$, for all $i = 1, 2, ..., 6$, $R_{sum}^{greedy} = 2$.

We now consider the use of our algorithms with parameter $\rho = 1$. Using our conservative algorithm, the two cliques, each consisting of three nodes are identified and selected, and the resulting G_ρ is depicted in Figure 8.14.

Each clique is assigned half of the time slots by the multicoloring algorithm. The achievable sum rate, depends on the interference properties of the network. In order to compute the achievable net sum-rate we must know the interference channel gains and the scheme being employed by each sub-network attempting to use advanced physical layer strategies. For example, in the low interference regime, we can assume treating interference as noise while in the high interference regime we can assume interference cancellation. In general, computing the achievable sum-rate for a set of

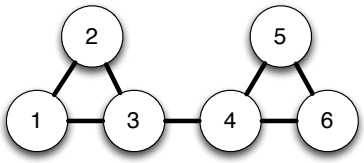


Figure 8.13: Two cliques

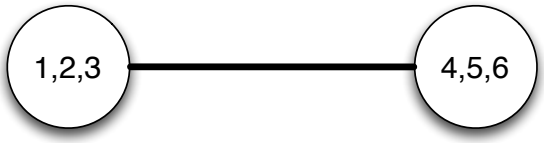


Figure 8.14: Two cliques after conservative algorithm, G_1

interfering users is an open problem. Nevertheless, within each one of the selected sub-networks, the achievable rate by each user can be lower and upper bounded. In the worst case scenario, each member within each sub-network has to time-share the medium with the other users in the sub-network. In this case, a user i in a sub-network with m users will achieve rate C_i/m every time the sub-network is active, in average. In the best case scenario, all interference is manageable and each user i achieves C_i every time the sub-network is active. Therefore, the achievable sum-rate by the conservative algorithm is bounded as follows:

$$\frac{1}{6}\sum_{i=1}^{6} C_i \leq R_{sum}^{cons} \leq \frac{1}{2}\sum_{i=1}^{6} C_i \tag{8.13}$$

Our lower bound is achieved by each one of the three users in each of the selected cliques using time-division multiplexing (TDM) and obtaining rate $(1/3)C_i$ each time their respective sub-network is active, which is $1/2$ of the time. In the best-case scenario, each user achieves C_i every time their sub-network is active, again $1/2$ of the time. Assuming unit capacities, $C_i = 1$ for all i, the achievable net sum-rate using

the conservative algorithm is bounded such that

$$1 \leq R_{sum}^{cons} < 3, \tag{8.14}$$

for $\rho = 1$.

The next step is to bound net sum-rate performance when using the aggressive algorithm. Now, each user is allowed to be part of more than one sub-network and the resulting G_ρ is depicted in Figure 8.15

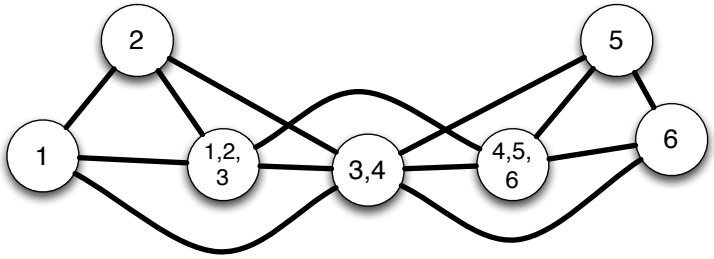


Figure 8.15: Two cliques after aggressive algorithm, G_1

First, consider the fraction of the time each sub-network will be active. The sub-networks $\{1\}, \{2\}, \{5\}$, and $\{6\}$ will be active $1/4$ of the time, sub-networks $\{1, 2, 3\}$ and $\{4, 5, 6\}$ will be active $1/5$ of the time, and sub-network $\{3, 4\}$ will be active $1/7$ of the time. The fraction of the time each sub-network is active is a result of the degree of each sub-network in the graph G_1. Using the bounding arguments described above, the rates of each of the users is described as follows:

$$\left(\frac{19}{60}\right) C_i \leq R_i \leq \left(\frac{9}{20}\right) C_i, \text{ for } i = 1, 2, 5, 6, \tag{8.15}$$

and

$$\left(\frac{29}{210}\right) C_i \leq R_i \leq \left(\frac{17}{35}\right) C_i, \text{ for } i = 3, 4. \tag{8.16}$$

With these bounds on each user's rate we can bound the net sum-rate achieved by the aggressive algorithm.

$$\left(\frac{19}{60}\right)\sum_{i\in\{1,2,5,6\}} C_i + \left(\frac{29}{210}\right)(C_3+C_4) \leq R_{sum}^{agg} \leq \left(\frac{9}{20}\right)\sum_{i\in\{1,2,5,6\}} C_i + \left(\frac{17}{35}\right)(C_3+C_4). \tag{8.17}$$

When we assume the symmetric capacity point $C_i = 1$ for all i, then the net sum rate becomes

$$\left(\frac{54}{35}\right) \leq R_{sum}^{agg} < \left(\frac{97}{35}\right), \tag{8.18}$$

for $\rho = 1$.

The fact that the upper bounds of the proposed algorithms' performance is higher than the sum-rate of the greedy scheduling algorithm means that when interference is manageable, our algorithm can perform better in terms of net sum-rate. While this is encouraging, the actual net sum-rate remains dependent on network channel states and direct comparison becomes unclear. On the other hand, normalized sum-rate gives us a metric that is independent of channel state and guarantees that in the worst-case over all possible channel realizations, we can guarantee a fraction of the capacity. In this case, the normalized sum-rate of the greedy scheduling is 3/8 while our conservative algorithm achieves 1/2 and our aggressive algorithm achieves 12/35.

8.3.2 Example 2

As expected, our algorithm's ability to present gains is not only based on interference properties, but on topology and the amount of information available. Consider the topology in Figure 8.16.

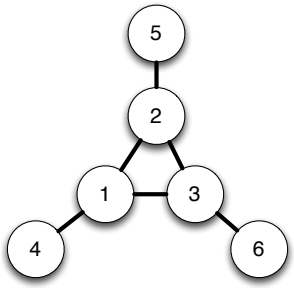


Figure 8.16: Clique-star

For this topology, the greedy distributed scheduling achieves

$$R_{sum}^{greedy} = \frac{7}{24}(C_1 + C_2 + C_3) + \frac{17}{24}(C_4 + C_5 + C_6), \tag{8.19}$$

and when we assume all capacities are equal to 1, $R_{sum}^{greedy} = 3$. For the same topology, our conservative algorithm produces the G_ρ depicted in Figure 8.17.

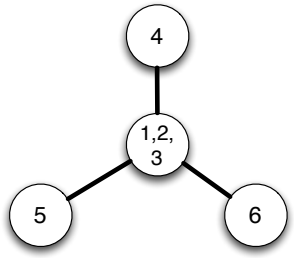


Figure 8.17: Clique-star after conservative algorithm, G_1

The conservative algorithm achieves a net sum-rate that is bounded as follows

$$\frac{1}{12}\sum_{i=1}^{3} C_i + \frac{1}{2}\sum_{j=4}^{6} C_j \leq R_{sum}^{cons} < \frac{1}{4}\sum_{i=1}^{3} C_i + \frac{1}{2}\sum_{j=4}^{6} C_j \tag{8.20}$$

and when we assume unit capacities

$$1.75 \leq R_{sum}^{cons} < 2.25. \tag{8.21}$$

In this case, even with manageable interference, our algorithm cannot achieve a higher

rate than the greedy distributed scheduling with only $\rho = 1$.

We also consider the performance of the aggressive algorithm. The G_ρ obtained after performing the aggressive algorithm is depicted in Figure 8.18.

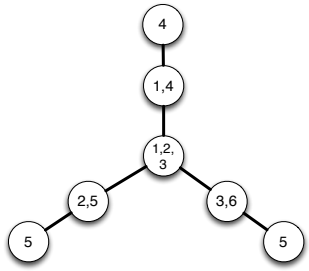


Figure 8.18: Clique-star after aggressive algorithm, G_1

The achievable rate by each user after the aggressive algorithm is bounded as follows

$$\left(\frac{31}{210}\right) C_i \leq R_i \leq \left(\frac{12}{35}\right) C_i, \text{ for } i = 1, 2, 3, \tag{8.22}$$

and

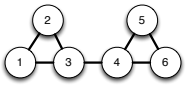
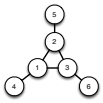
$$\left(\frac{13}{30}\right) C_i \leq R_i \leq \left(\frac{8}{15}\right) C_i, \text{ for } i = 4, 5, 6. \tag{8.23}$$

With these bounds on each user's rate we can bound the net sum-rate using the aggressive algorithm.

$$\left(\frac{31}{210}\right) \sum_{i\in\{1,2,3\}} C_i + \left(\frac{13}{30}\right) \sum_{j\in\{4,5,6\}} C_j \leq R_{sum}^{agg} \leq \left(\frac{12}{35}\right) \sum_{i\in\{1,2,3\}} C_i + \left(\frac{8}{15}\right) \sum_{j\in\{4,5,6\}} C_j. \tag{8.24}$$

When we assume the symmetric capacities $C_i = 1$ for all i, then the achievable

Table 8.1: Table of Results for Two Example Topologies

Topology	Algorithm	Norm. sum-rate	Net sum-rate
	Distributed Coloring	0.25	1.833
	Greedy	0.25	2
	Conservative	0.5	$1 \leq R_{sum} \leq 3$
	Aggressive	0.343	$1.54 \leq R_{sum} \leq 2.48$
	Distributed Coloring	0.25	2.25
	Greedy	0.292	3
	Conservative	0.25	$1.75 \leq R_{sum} \leq 2.25$
	Aggressive	0.343	$1.74 \leq R_{sum} \leq 2.62$

net sum-rate becomes

$$\left(\frac{61}{35}\right) \leq R_{sum}^{agg} < \left(\frac{92}{35}\right), \tag{8.25}$$

for $\rho = 1$. We summarize the results of the previous two examples in Table 8.1.

One important note is that the greedy scheduling algorithm is *not* a one-shot algorithm that results in maximal schedules in each time slot. Not using distributed scheduling algorithms, such as greedy scheduling, certifies that the amount of information needed to perform our algorithm is limited to $3\rho + 3$ or $3\rho + 1$, which would

not be the case otherwise since information propagates with the number of rounds. A discussion regarding how locality constraints affect algorithms performance can be found in [49] and general locality-sensitive approaches to distributed algorithms are described in [50].

8.4 Numerical Results: Net Sum-rate

Just as in the case of normalized sum-rate, we also present net sum-rate performance comparison for simulation in three classes of graphs: random graphs, scale-free graphs, and geometric graphs. In this section, we report net sum-rate performance for four algorithms: distributed coloring, greedy distributed algorithm (maximal schedule), our conservative algorithm, and our aggressive algorithm. As discussed in Section 8.3, we assume each user i has a capacity C_i when no interference is present and compute the sum-rate achievable by each of the four algorithms. Also, we note that in our proposed algorithms net sum-rate is not directly computable, therefore in our results we report lower and upper bounds.

Figure 8.19 shows the net sum-rate performance of random graph with parameters $\mathcal{G}(n, 0.1)$, Figure 8.20 describes performance for graphs with $\mathcal{G}(n, 0.5)$, and Figure 8.21 for graphs with $\mathcal{G}(n, 0.9)$. In Figure 8.22, we report the net sum-rate achievable for the class of scale-free graphs with 25, 50, and 100 users. Finally, in Figure 8.23 we see the performance comparison in the class of geometric graphs with interference diameter $d = 0.25$ and in Figure 8.12 in geometric graphs with interference diameter is $d = 0.5$.

The results in Figures 8.19 - 8.24 show that in terms of net sum-rate, our aggressive algorithm outperforms the conservative algorithm in the majority of the cases. This trend agrees with the performance in terms of normalized sum-rate. An interesting observation from the numerical results is that the upper bound on the

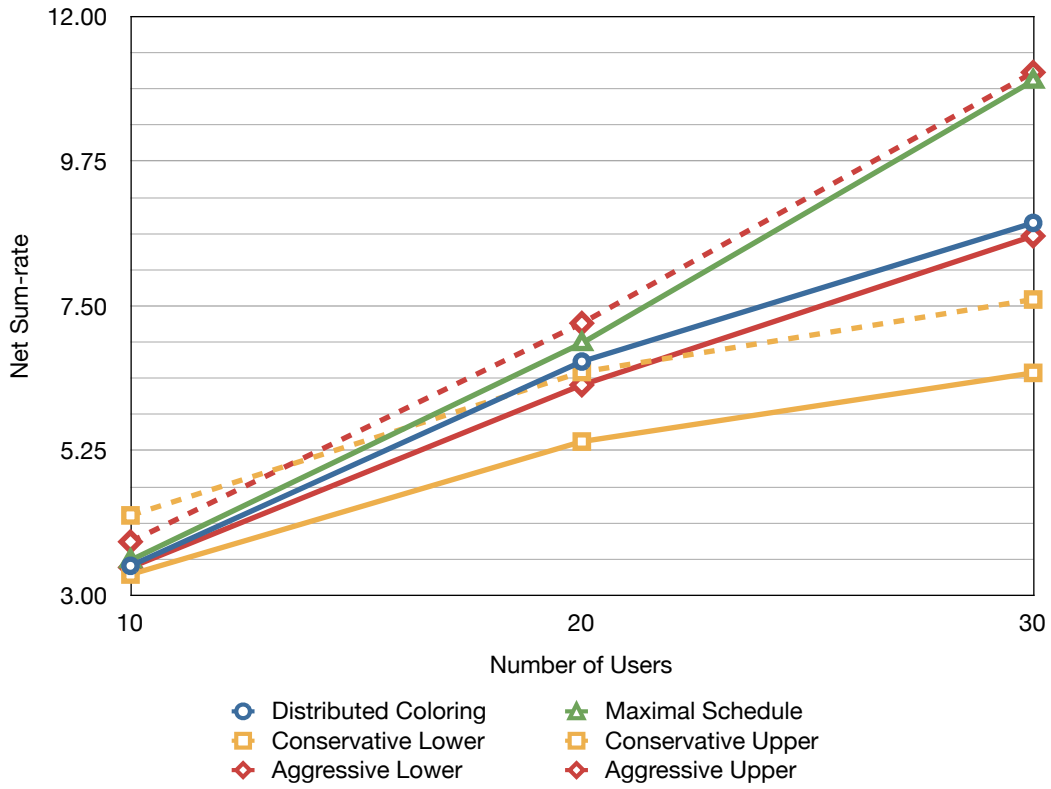


Figure 8.19: Net sum-rate performance comparison in random graphs with low connectivity, $p = 0.1$

sum-rate achievable by our aggressive algorithm is consistently higher than the sum-rate achieved by distributed scheduling and the maximal scheduler for the simulated random graphs and geometric graphs. This validates the assertion that when interference is manageable, our aggressive algorithm can perform better in terms of net sum-rate. The only case where neither of our algorithms could surpass maximal scheduling was the class of scale-free graphs. As we discussed in Section 8.3, the net sum-rate performance is dependent on topology and amount of knowledge available. The structure of scale-free networks is particularly unfavorable to the aggressive algorithm since the spoke-hub nature of the graphs results in large degree increase when $\rho = 1$. Nevertheless, in scale-free graphs, increasing the diameter of the cliques being formed to $\rho = 2$ produces significant improvement since large sections of each

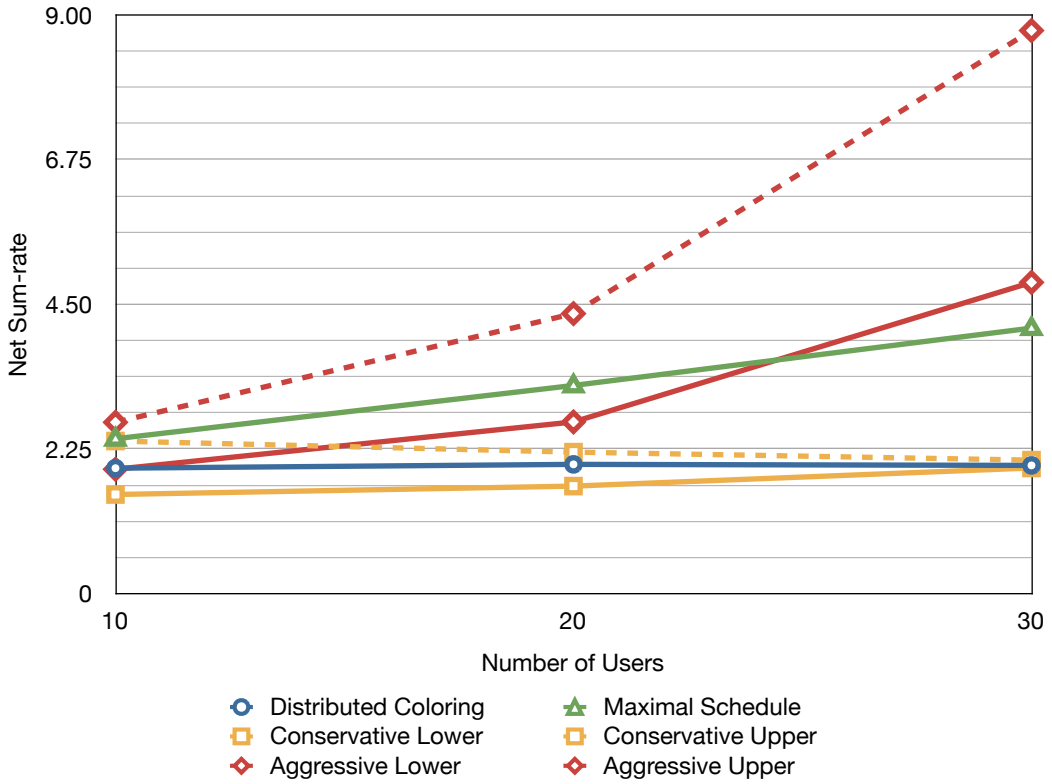


Figure 8.20: Net sum-rate performance comparison in random graphs with medium connectivity, $p = 0.5$

spoke-hub are identified as a single sub-network. Once again, these observations are in agreement with the examples in the previous sections where we showed that in terms of net sum-rate there are cases where maximal scheduling can outperform the aggressive algorithm. The numerical results for net sum-rate performance highlight the advantages of the aggressive algorithm because, with only $\rho = 1$, it is able to outperform maximal scheduling in random and geometric graphs. In cases where higher net sum-rate is not achieved, such as in scale-free graphs, it has the flexibility to leverage more knowledge (e.g., let $\rho = 2$) and improve performance.

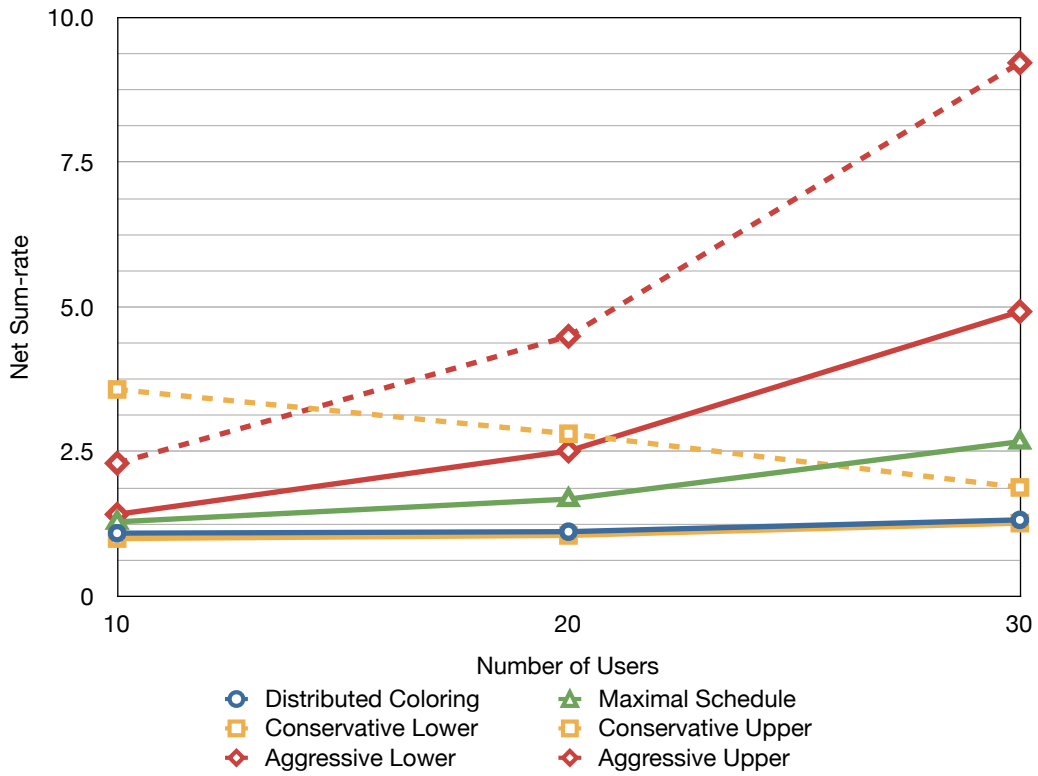


Figure 8.21: Net sum-rate performance comparison in random graphs with high connectivity, $p = 0.9$

8.5 Remarks on Overhead

The key feature of the algorithms we have proposed is that they can be executed with $\rho+1$ hops of channel information and $3\rho+3$ hops of connectivity information for the conservative algorithm or $3\rho+1$ hops of connectivity information for the aggressive algorithm. In order to be able to make comparisons with other distributed scheduling algorithms, it is important to understand what is the overhead to required obtain this amount of information. In this section we make some general remarks about the type and amount of overhead induced by our algorithms and how it compares to state-of-the-art distributed scheduling algorithms.

The total amount of overhead required for execution depends on how often the system needs to renew its knowledge. Consider three different time-scales in the life of

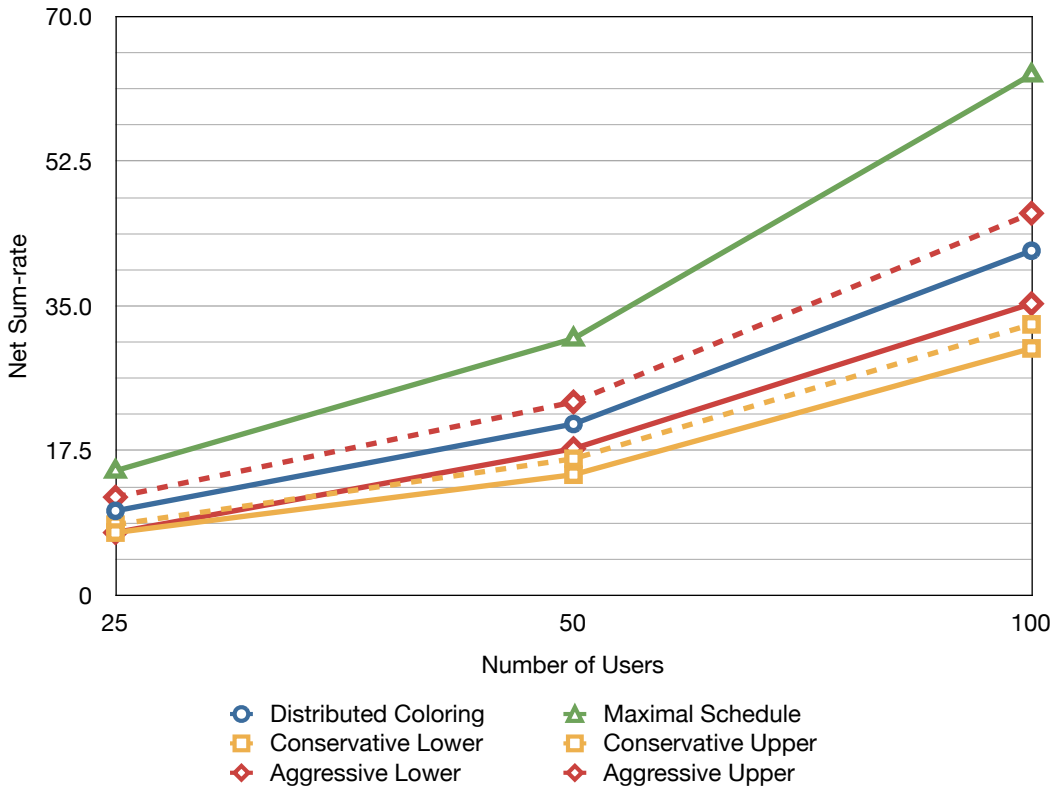


Figure 8.22: Net sum-rate performance comparison in scale-free graphs

the network. First, we define *topology coherence time*, T_{topo}, as the amount of time the network remains static in terms of topology. In other words, the amount of time the network is correctly described by the graph G. Second, the amount of time the channel remains constant before changing is denoted *channel coherence time*, $T_{channel}$. Finally, the amount of time required for a single communication transmission is denoted *data transmission time*, T_{data}. The length of each one of these time-scales and their ratios depend on network properties such as mobility, fading environments, and packet lengths. Due to the nature of wireless networks and their physical properties, it is reasonable to assume that in most cases $T_{topo} \leq T_{channel} \leq T_{data}$.

When we describe our proposed algorithms as one-shot algorithms, we are referring to the fact that they need to exchange information only after some time T_{topo} or $T_{channel}$. This important feature contrasts with a large section of distributed link

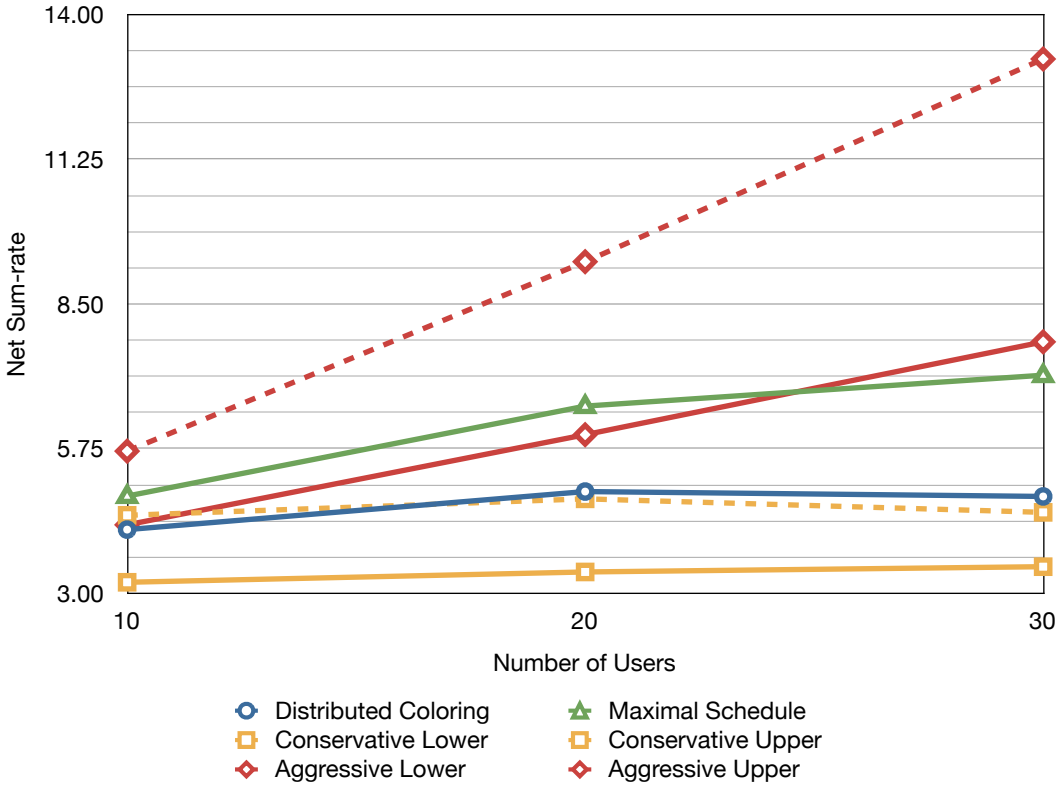


Figure 8.23: Net sum-rate performance comparison in geometric graphs, $d = 0.25$

scheduling algorithms which require several rounds of message passing or dedicated control sub-frames each time a data communication is established, i.e., every T_{data}. The distributed link scheduling algorithms described in Chapter 3 require rounds of message passing per data transmission time because they are based on queue state information. The queue state information of each user is updated after each data transmission and must be communicated every T_{data}. While some of these algorithms only require communication from each node with neighbors only one hop away, they require large amount of rounds per T_{data}. A sequence of message exchanges implicitly propagates information beyond one hop.

Direct overhead comparisons between the proposed algorithms in this work and state-of-the-art distributed scheduling algorithms is non-trivial. The primary objective of the proposed algorithms is a reduction in the number of hops of network

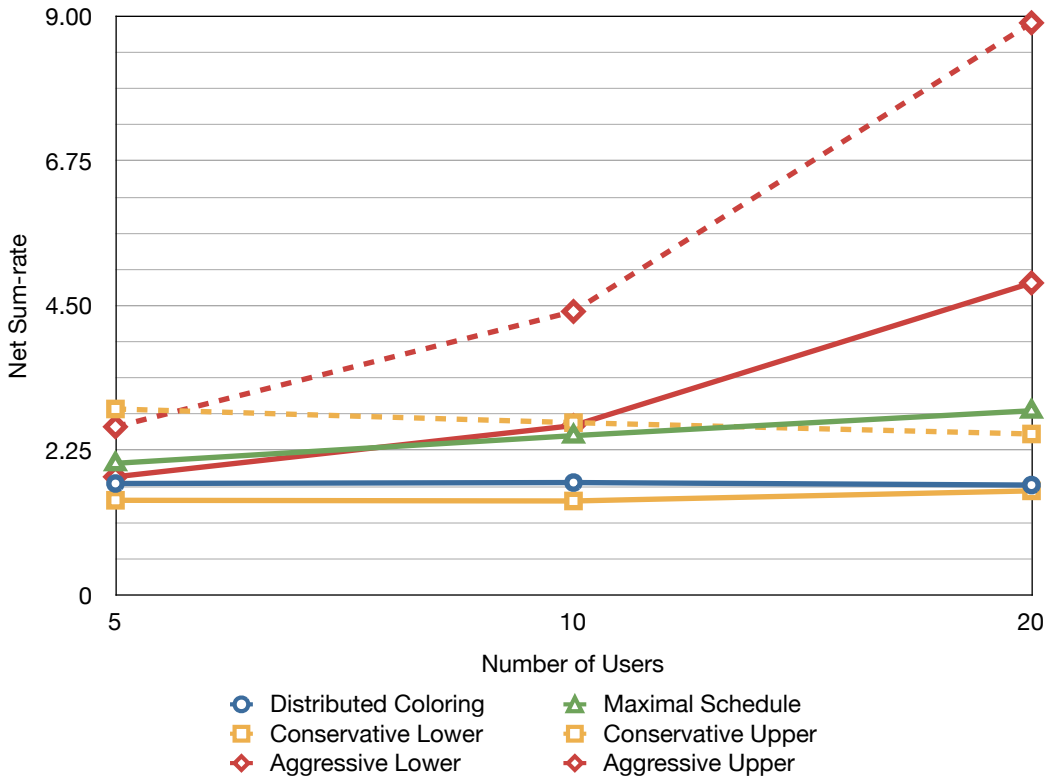


Figure 8.24: Net sum-rate performance comparison in geometric graphs, $d = 0.5$

information required for algorithm execution. In other words, we are concerned with information locality. On the other hand, distributed scheduling algorithms have their main priority set on reducing computational complexity [21, 20, 4, 5, 16, 22]. Because the emphasis of efficiency is different in both scenarios, the relative cost of overhead also changes. Therefore, while several rounds of message passing with neighbors is typical in distributed scheduling algorithms and considered low overhead, in terms of information locality, it can represent significant overhead since each round of message passing can implicitly provide an extra hop of network information.

Chapter 9

Conclusion

In this chapter, we present a summary of our contributions and a list of possible future directions in which the work presented can be extended.

9.1 Summary of Contributions and Significance

The work presented in this thesis was motivated by the original question of how and when users in a wireless network should transmit if only local information is available. To take steps towards solving this challenging problem, we have developed two distributed algorithms that use only local connectivity information to coordinate sub-networks that have enough channel information to communicate in an optimal manner. The key idea behind the algorithms presented is to identify independent sub-graphs such that, at each time slot, the active users in the network have all the information required to use physical layers beyond interference avoidance.

The proposed conservative algorithm is guaranteed to have a normalized sum-rate performance that is greater than or equal to distributed graph coloring. The aggressive algorithm also shows significant improvement for important graph classes over distributed graph coloring, maximal scheduling, and the conservative algorithm.

These two algorithms are constructive and provide more feasible schemes that can leverage realistic assumptions of local information.

An implication of the results presented in this thesis is that the algorithms proposed provide a lower bound on the normalized sum-capacity with local information. The results in this thesis highlight the benefits that can be gained by using local connectivity and channel information, and going beyond interference avoidance.

9.2 Future Directions

The study of the work we have presented opens the door to several future directions. The first direction of interest is the idea of exploring sub-network scheduling when the physical layer is fixed to known achievable schemes. In the work we have presented, we have concentrated on the fact that the sub-networks being identified have full information to engage in optimal communication strategies, yet the optimal communication strategy is not known for many scenarios. If we consider only a set of physical layer strategies, the identification process and selection of sub-networks to be scheduled needs to be properly adapted. This direction would generate more practical algorithms that utilize feasible physical layers and are more easily implementable.

In the third step of the algorithms presented, a distributed multicoloring algorithms was used to schedule sub-networks. Another direction to extend our work is the use of other distributed scheduling algorithms to schedule the sub-networks identified and selected in the first two steps of the proposed algorithms. In particular, an attractive option for scheduling sub-network are algorithms that use random access and carrier sensing to find schedules. These algorithms can improve performance since the normalized sum-rate will not be directly dependent on the maximum degree of the sub-network graph. Random access scheduling algorithms also have the added advantage of preserving the amount of local information since the only information

they need is from their one-hop network.

Finally, the third possible extension for our work is the addition of queues to the network model. With queue information in our system we can provide queue stability analysis of our algorithms. With a queue stability analysis of the proposed algorithms, direct comparison to existing state-of-the-art distributed scheduling algorithms will become more evident. This type of analysis has the potential to highlight the value of local information, not just in terms of normalized sum-rate but also in terms of quality of service and other typical networking performance metrics.

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, December 1992. 1, 3.1, 3.1

[2] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, March 2000. 1

[3] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, March 2005, pp. 1804–1814 vol. 3. 1, 2.1, 1, 4

[4] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, ser. MobiCom '06, New York, NY, USA, 2006, pp. 227–238. 1, 2.1, 1, 8.5

[5] X. Wu, R. Srikant, and J. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 6, pp. 595–605, June 2007. 1, 2.1, 1, 8.1, 8.5

[6] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," *Information Theory, IEEE Transactions on*, vol. 54, no. 2, pp. 572–594, Feb. 2008. 1, 1

[7] S. Verdu, *Multiuser Detection*, 1st ed. New York, NY, USA: Cambridge University Press, 1998. 1

[8] T. Han and K. Kobayashi, "A new achievable rate region for the interference channel," *Information Theory, IEEE Transactions on*, vol. 27, no. 1, pp. 49–60, Jan. 1981. 1

[9] V. Cadambe and S. Jafar, "Interference alignment and spatial degrees of freedom for the k user interference channel," in *Communications, 2008. ICC '08. IEEE International Conference on*, 2008, pp. 971–975. 1

[10] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge, MA: Cambridge University Press, 2011. 1, 3.2

[11] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H. P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint: Concepts, performance, and field trial results," *Communications Magazine, IEEE*, vol. 49, no. 2, pp. 102–111, Feb. 2011. 1

[12] H. Sato, "The capacity of the gaussian interference channel under strong interference (corresp.)," *Information Theory, IEEE Transactions on*, vol. 27, no. 6, pp. 786–788, 1981. 1

[13] V. Aggarwal, A. Avestimehr, and A. Sabharwal, "On achieving local view capacity via maximal independent graph scheduling," *Information Theory, IEEE Transactions on*, vol. 57, no. 5, pp. 2711–2729, May 2011. 1, 2.2, 2.3, 3.2, 4.1, 4.2, 4.3, 6.1.1, 8

[14] V. Aggarwal, Y. Liu, and A. Sabharwal, "Sum capacity of interference channels with a local view: Impact of distributed decisions," *Information Theory, IEEE Transactions on*, vol. 58, no. 3, pp. 1630–1659, March 2012. 1, 4.2

[15] F. Kuhn, "Local Multicoloring Algorithms: Computing a Nearly-Optimal TDMA Schedule in Constant Time," in *26th International Symposium on Theoretical Aspects of Computer Science*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 3, Dagstuhl, Germany, 2009, pp. 613–624. 1, 4.3, 7, 7.1, 7.2, 7.3

[16] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 1258–1263. 2.1, 3, 8.5

[17] L. Jiang, D. Shah, J. Shin, and J. Walrand, "Distributed random access algorithm: Scheduling and congestion control," *Information Theory, IEEE Transactions on*, vol. 56, no. 12, pp. 6182–6207, 2010. 3.1

[18] J. Lee, J. Lee, Y. Yi, S. Chong, A. Proutiere, and M. Chiang, "Implementing utility-optimal CSMA," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, 2009, pp. 102–111. 3.1

[19] Y. Yi, A. Proutière, and M. Chiang, "Complexity in wireless scheduling: impact and tradeoffs," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '08. New York, NY, USA: ACM, 2008, pp. 33–42. 3.1

[20] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 313–324, Jun. 2007. 2, 8.5

[21] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '06/Performance '06, New York, NY, USA, 2006, pp. 27–38. 2, 8.1, 8.5

[22] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1846–1859, Dec. 2009. 3, 8.5

[23] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1481–1493, Oct. 2009. 3

[24] P. Marbach and A. Eryilmaz, "A backlog-based CSMA mechanism to achieve fairness and throughput-optimality in multihop wireless networks," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, 2008, pp. 768–775. 3

[25] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010. 3

[26] 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, June 2012. 3

[27] C. Joo and N. Shroff, "Local greedy approximation for scheduling in multihop wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 3, pp. 414–426, March 2012. 4

[28] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 3, pp. 709–720, June 2011. 4

[29] D. Baker, J. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network," in *IEEE ICC*, vol. 82, 1982, p. 2F. 3.1

[30] D. Shah, P. Giaccone, and B. Prabhakar, "Efficient randomized algorithms for input-queued switch scheduling," *Micro, IEEE*, vol. 22, no. 1, pp. 10–18, 2002. 3.1

[31] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 411–424, 2005. 3.1

[32] G. Sharma, C. Joo, and N. B. Shroff, "Distributed scheduling schemes for throughput guarantees in wireless networks," in *the 44th Annual Allerton Conference on Communications, Control, and Computing*, 2006. 3.1

[33] S. Sarkar and K. Kar, "Achieving 2/3 throughput approximation with sequential maximal scheduling under primary interference constraints," in *Proceedings of 44th Annual Allerton Conference on Communication, Control and Computing*, 2006, pp. 27–29. 3.1

[34] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 1139–1147. 3.1

[35] L. Ying and S. Shakkottai, "Scheduling in mobile ad hoc networks with topology and channel-state uncertainty," in *INFOCOM 2009, IEEE*, 2009, pp. 2347–2355. 3.1

[36] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "Flashlinq: A synchronous distributed scheduler for peer-to-peer ad hoc networks," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010, pp. 514–521. 3.1

[37] S. A. Jafar, "Topological interference management through index coding," *arXiv CoRR*, vol. abs/1301.3106, 2013. 3.2

[38] N. Naderializadeh and A. S. Avestimehr, "Interference networks with no csit: Impact of topology," *arXiv CoRR*, vol. abs/1302.0296, 2013. 3.2

[39] D. T.-H. Kao and A. Sabharwal, "On capacity regions of interference channels with mismatched local views," *arXiv CoRR*, vol. abs/1110.0886, 2011. 3.2

[40] K. Sutuntivorakoon, V. Aggarwal, A. Avestimehr, and A. Sabharwal, "Maximal k-clique scheduling: A simple algorithm to bound maximal independent graph scheduling," in *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, 2011, pp. 816–823. 4.2, 6.1.1

[41] R. Diestel, "Graph Theory," *Graduate Texts in Mathematics. Springer-Verlag, Heidelberg*, vol. 173, 2005. 7

[42] N. Linial, "Locality in distributed graph algorithms," *SIAM Journal on Computing*, vol. 21, no. 1, pp. 193–201, 1992. 7

[43] M. Luby, "Removing randomness in parallel computation without a processor penalty," in *Foundations of Computer Science, 1988., 29th Annual Symposium on*, 1993, pp. 162–173. 7

[44] K. Kothapalli, C. Scheideler, M. Onus, and C. Schindelhauer, "Distributed coloring in $O(\sqrt{(}\log n))$ bit rounds," in *Proceedings of the 20th international conference on Parallel and distributed processing*, ser. IPDPS'06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 44–44. 7

[45] R. Karp, "Reducibility among combinatorial problems," in *50 Years of Integer Programming 1958-2008*. Springer Berlin Heidelberg, 2010, pp. 219–241. 7

[46] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM Journal on Computing*, vol. 15, no. 4, pp. 1036–1053, 1986. 7

[47] M. Wattenhofer and R. Wattenhofer, "Distributed weighted matching," in *Distributed Computing*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed. Springer Berlin Heidelberg, 2004, vol. 3274, pp. 335–348. 7

[48] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002. 8.2

[49] A. Korman, J.-S. Sereni, and L. Viennot, "Toward more localized local algorithms: removing assumptions concerning global knowledge," in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, ser. PODC '11, New York, NY, USA, 2011, pp. 49–58. 8.3.2

[50] D. Peleg, *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, 1987, vol. 5. 8.3.2