An Efficient DoF Scheduling Algorithm for Multi-hop MIMO Networks

Huacheng Zeng Yi Shi Y. Thomas Hou Wenjing Lou
Virginia Polytechnic Institute and State University, USA

Abstract—**Degree-of-Freedom (DoF)-based model is a simple yet powerful tool to analyze MIMO's spatial multiplexing (SM) and interference cancellation (IC) capabilities in a multi-hop network. Recently, a new DoF model was proposed and was shown to achieve the same rate region as the matrix-based model (under SM and IC). The essence of this new DoF model is a novel node ordering concept, which eliminates potential duplication of DoF allocation for IC. In this paper, we investigate DoF scheduling for a multi-hop MIMO network based on this new DoF model. Specifically, we study how to perform DoF allocation among the nodes for SM and IC so as to maximize the minimum rate among a set of sessions. We formulate this problem as a mixed integer linear programming (MILP) and develop an efficient DoF scheduling algorithm to solve it. We show that our algorithm is amenable to local implementation and has polynomial time complexity. More importantly, it guarantees the feasibility of final solution (upon algorithm termination), despite that node ordering establishment and adjustment are performed locally. Simulation results show that our algorithm can offer a result that is close to an upper bound found by CPLEX solver, thus showing that the result found by our algorithm is highly competitive.**

I. Introduction

MIMO is a powerful physical layer technology that exploits multiple transmit and receive antennas to increase bit rate within a given bandwidth. In recent years, there is a growing interest in employing MIMO to increase network throughput in a multi-hop ad hoc network. In particular, there is an active line of research of multi-hop MIMO network that built upon the so-called degree-of-freedom (DoF) model [1], [2], [4], [7], [11].

The number of DoFs of a node is typically assumed to be the same as the number of antennas at the node and represents the total available resources at the node for *spatial multiplexing* (SM) and *interference cancellation* (IC) [5], [8], [13]. SM refers to the use of one or multiple DoFs (both at transmit and receive nodes) for data transport, with each data stream corresponding to one DoF. IC refers to the use of one or multiple DoFs to cancel interference, which can be done at either transmit node (to cancel its interference to another node) or receive node (to cancel interference from another node). For example, consider two links in Fig. 1. To transmit z_1 data streams on link (T_1, R_1), both nodes T_1 and R_1 need to consume z_1 DoFs for SM. Similarly, to transmit z_2 data streams on link (T_2, R_2), both nodes T_2 and R_2 need to consume z_2 DoFs for SM. The interference from T_2 to R_1 can be cancelled by either R_1 or T_2. If R_1 cancels this interference, it needs to consume z_2 DoFs. If T_2 cancels this interference, it needs to consume z_1 DoFs.

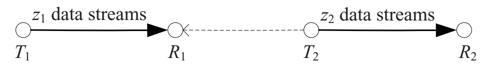

Fig. 1. DoF allocation for SM and IC.

A significant advantage of DoF-based model is that it only requires simple numeric computation (addition/subtraction) to keep track of SM and IC at a node. Although a DoF-based model is not able to completely capture all the physical layer capabilities of MIMO, it offers a simple and effective tool to analyze MIMO in a multi-hop network. Since its inception, a DoF-based model has been applied to solve throughput optimization problems [1], [4] and to design MAC protocols [2], [11].

As shown in the example in Fig. 1, since an interference can be cancelled by either a transmit node or a receive node, a question that arises is which node should take the responsibility for IC? The lack of a systematic rule in assigning IC responsibility is likely to lead to sub-optimal or infeasible solution and is the main limitation in some of the prior efforts [1], [2], [4], [11]. In our recent work in [10], we addressed the important problem of which node should be responsible for IC by developing a DoF-based link layer model for multi-hop MIMO networks. A novel concept in this model is an *ordering* relationship among all the nodes in the network for DoF allocation (both SM and IC). It was shown in [10] that once DoF allocation is performed at each node following this node ordering, potential duplication in IC can be completely eliminated and no DoF resource will be wasted. The details of this ordering concept will be reviewed in Section II.

Inspired by the new DoF model in [10], in this paper, we employ this model to study a throughput maximization problem in a multi-hop MIMO network. Specifically, we study a DoF scheduling problem with the goal of maximizing the minimum session rate for a group of sessions. We show that this problem is in the form of mixed integer linear programming (MILP) and is likely NP-hard. Since this problem formulation cannot be solved in a reasonable amount of time, we develop an efficient and fast DoF scheduling algorithm. Our developed algorithm is an iterative greedy algorithm and includes three modules: *link selection module* (LSM), *resource allocation module* (RAM), and *local reordering module* (LRM). Some of the highlights of our algorithm include:

- It is amenable for local implementation. We will show that each module in our algorithm can be implemented in a distributed manner.

- The final DoF allocation solution (upon algorithm termination) is feasible at a global level; there exists a global node ordering for the final solution. This is not trivial, as neither the RAM nor the LRM performs DoF allocation and node ordering adjustment with a global knowledge.
- Its performance is highly competitive. Simulation results show that the objectives by our algorithm are close to upper bounds by CPLEX solver, indicating that the objective by our algorithm is very close to the optimum.
- It has a polynomial time complexity, and offers a solution rather quickly (in contrast to exponential complexity of CPLEX).

The remainder of this paper is organized as follows. In Section II, we give a review of the new DoF model introduced in [10]. In Section III, we outline a throughput maximization problem formulation based on the new DoF model. Section IV introduces our DoF scheduling algorithm, with the details of its three modules given in Sections V to VII. Section VIII gives a proof that the final solution is feasible. Section IX presents simulation results and Section X concludes this paper.

II. A Primer: A MIMO Link Layer Model

We consider a multi-hop network consisting of a set of nodes, each of which is equipped with multiple antennas. Assume that the channel matrix between any two nodes is i.i.d. Then the number of DoFs available for a node is equal to the number of its antennas. A node can use some or all of its DoFs for either SM or IC, as long as the number of consumed DoFs does not exceed its total available DoFs. According to [10], the DoF allocation at a node for IC depends on its "ordering" in the node list. For a given ordered list of nodes, the DoFs at a node for IC are allocated in the following manner.

- *Transmit Node.* A transmit node only needs to cancel its interference to those receive nodes that are before itself in the ordered list of nodes. It does not need to expend DoFs to cancel its interference to those receive nodes that are after itself in the ordered list. Interference from this transmit node to those receive nodes will be cancelled by those receiving nodes.
- *Receive Node.* A receive node only needs to cancel the interference from those transmit nodes that are before itself in the ordered list. It does not need to cancel the interference from those transmit nodes that are after itself in the ordered list. Interference from those transmit nodes will be cancelled by those nodes.

Note that the "ordering" concept is crucial to avoid any duplication in IC among the nodes in the network [10]. In the rest of this section, we give a mathematical model for this MIMO link layer model.

A. A Link Layer Model

Suppose that there are N nodes in the network and for each node i, there are A_i antennas. Assume that a time frame is divided into T time slots. Denote a binary variable $x_i(t)$ as whether node i is a transmitter in time slot t. Similarly, denote $y_i(t)$ as whether node i is a receiver in time slot t. Let $\mathcal{L}_i^{\text{in}}$ and $\mathcal{L}_i^{\text{out}}$ be the set of possible incoming and outgoing links at node i, respectively. For simplicity, we assume that one data stream corresponds to one unit of data rate [5], [10]. Denote $z_l(t)$ as the number of data streams on link l in time slot t. Then we have

$$x_i(t) \le \sum_{l \in \mathcal{L}_i^{\text{out}}} z_l(t) \le A_i \cdot x_i(t), \quad (1 \le i \le N, 1 \le t \le T), \tag{1}$$

$$y_i(t) \le \sum_{l \in \mathcal{L}_i^{\text{in}}} z_l(t) \le A_i \cdot y_i(t), \quad (1 \le i \le N, 1 \le t \le T). \tag{2}$$

Denote $\pi(t)$ as the order of nodes in the network in time slot t and denote $\pi_i(t)$ as the position of node i in order $\pi(t)$. Then we have

$$1 \le \pi_i(t) \le N, \quad (1 \le i \le N, 1 \le t \le T). \tag{3}$$

Denote binary variable $\theta_{ji}(t)$ as the relative position of node i and node j in order $\pi(t)$ as follows: $\theta_{ji}(t) = 1$ if node i is after node j in order $\pi(t)$ and 0 otherwise. Then we have

$$\pi_i(t) - N \cdot \theta_{ji}(t) + 1 \le \pi_j(t) \le \pi_i(t) - N \cdot \theta_{ji}(t) + N - 1, \quad (1 \le i \le N, j \in \mathcal{I}_i, 1 \le t \le T), \tag{4}$$

where $\mathcal{I}_i$ is the set of nodes within node i's interference range.

If node i is transmitting or receiving, then the number of its DoFs consumed by SM and IC should be less than or equal to A_i; otherwise (i.e., an idle node), the number of its DoFs consumed by SM should be 0 and there is no constraint on the number of its DoFs consumed by IC. Thus we have

$$\sum_{l \in \mathcal{L}_i^{\text{out}}} z_l(t) + \sum_{j \in \mathcal{I}_i} \theta_{ji}(t) \sum_{k \in \mathcal{L}_j^{\text{in}}}^{\text{Tx}(k) \ne i} z_k(t) \le A_i x_i(t) + (1 - x_i(t)) B_i, \quad (1 \le i \le N, j \in \mathcal{I}_i, 1 \le t \le T), \tag{5}$$

and

$$\sum_{k \in \mathcal{L}_i^{\text{in}}} z_l(t) + \sum_{j \in \mathcal{I}_i} \theta_{ji}(t) \sum_{k \in \mathcal{L}_j^{\text{out}}}^{\text{Rx}(k) \ne i} z_k(t), \le A_i y_i(t) + (1 - y_i(t)) B_i, \quad (1 \le i \le N, j \in \mathcal{I}_i, 1 \le t \le T), \tag{6}$$

where B_i is a large enough number to ensure that there is no restriction on DoF consumption when $x_i = 0$ in (5) and $y_i = 0$ in (6) (e.g., $B_i = \sum_{j=1}^{N} A_j$).

B. Linearization

Among the above constraints, only (5) and (6) are nonlinear. We employ *reformulation-linearization technique* (RLT) [9] to linearize those two constraints. By defining $\lambda_{ji}(t) = \theta_{ji}(t) \sum_{k \in \mathcal{L}_i^{\text{in}}}^{\text{Tx}(k) \ne i} z_k(t)$, (5) can be replaced by

$$\sum_{l\in\mathcal{L}_i^{\text{out}}} z_l(t) + \sum_{j\in\mathcal{I}_i} \lambda_{ji}(t) \le A_i x_i(t) + (1-x_i(t))B_i, \quad (1\le i\le N, 1\le t\le T), \tag{7}$$

$$\lambda_{ji}(t) \le \sum_{k\in\mathcal{L}_i^{\text{in}}}^{\text{Tx}(k)\neq i} z_k(t), \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T), \tag{8}$$

$$\lambda_{ji}(t) \le A_j\cdot\theta_{ji}(t), \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T), \tag{9}$$

$$\lambda_{ji}(t) \ge A_j\cdot\theta_{ji}(t) + \sum_{k\in\mathcal{L}_i^{\text{in}}}^{\text{Tx}(k)\neq i} z_k(t) - A_j, \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T). \tag{10}$$

Similarly, by defining $\mu_{ji}(t) = \theta_{ji}(t)\sum_{k\in\mathcal{L}_i^{\text{out}}}^{\text{Rx}(k)\neq i} z_k(t)$, (6) can be replaced by

$$\sum_{k\in\mathcal{L}_i^{\text{in}}} z_l(t) + \sum_{j\in\mathcal{I}_i} \mu_{ji}(t) \le A_i y_i(t) + (1-y_i(t))B_i, \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T), \tag{11}$$

$$\mu_{ji}(t) \le \sum_{k\in\mathcal{L}_i^{\text{out}}}^{\text{Rx}(k)\neq i} z_k(t), \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T), \tag{12}$$

$$\mu_{ji}(t) \le A_j\cdot\theta_{ji}(t), \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T), \tag{13}$$

$$\mu_{ji}(t) \ge A_j\cdot\theta_{ji}(t) + \sum_{k\in\mathcal{L}_i^{\text{out}}}^{\text{Rx}(k)\neq i} z_k(t) - A_j, \quad (1\le i\le N, j\in\mathcal{I}_i, 1\le t\le T). \tag{14}$$

III. Problem Formulation

Suppose there is a set of F unicast sessions in the network. The route of each session is given *a priori,* which can be computed by some routing protocol. In this setting, we are interested in exploring an optimal DoF scheduling so that some throughput objective can be maximized. Denote $r(f)$ as the throughput of session f. Then a plausible optimization objective is to maximize the minimum (bottleneck) throughput among all sessions. Note that the choice of this specific objective does not limit our solution to be extended to other throughput objectives.

Half Duplex. We assume that a wireless transceiver is half-duplex. Then we have

$$x_i(t) + y_i(t) \le 1, \quad (1\le i\le N; 1\le t\le T). \tag{15}$$

OPT-DoF:	
Max	$r_{\min}$
S.t.	throughput objective: (20);
	link layer constraints: (1)–(4) and (7)–(14);
	link capacity constraints: (16);
	half-duplex constraints: (15);
	flow balance constraints: (17) and (18).

Fig. 2. A problem formulation for DoF scheduling.

Link Capacity Constraint. Denote $\text{src}(f)$ and $\text{dst}(f)$ as the source and destination nodes of session f, respectively. Denote $r_l(f)$ as the amount of data rate on link l that is attributed to session f. Then the average rate of link l over T time slots is $\frac{1}{T}\sum_t z_l(t)$. Thus, we have

$$\sum_f r_l(f) \le \frac{1}{T}\sum_t z_l(t), \quad (1\le l\le L), \tag{16}$$

where L is the number of links in the network.

Flow Balance at Each Node. At each node, flow conservation must be observed. Then at a source node, we have

$$\sum_{l\in\mathcal{L}_i^{\text{out}}} r_l(f) = r(f), \quad (i=\text{src}(f), 1\le f\le F). \tag{17}$$

At an intermediate node, we have

$$\sum_{l\in\mathcal{L}_i^{\text{out}}} r_l(f) = \sum_{l\in\mathcal{L}_i^{\text{in}}} r_l(f), \quad (1\le i\le N, 1\le f\le F, i\neq\text{src}(f), i\neq\text{dst}(f)). \tag{18}$$

At a destination node, we have

$$\sum_{l\in\mathcal{L}_i^{\text{in}}} r_l(f) = r(f), \quad (i=\text{dst}(f), 1\le f\le F). \tag{19}$$

It can be easily verified that if (17) and (18) are satisfied, then (19) is also satisfied. Therefore, it is sufficient to include only (17) and (18) in the problem formulation.

Throughput Objective. Denote $r_{\min}$ as the throughput rate of the bottleneck session. Then we have

$$r_{\min} \le r(f), \quad (1\le f\le F). \tag{20}$$

With the above constraints and the MIMO link layer model described in Section II, our optimization problem can be formulated in Fig. 2. This formulation is in the form of mixed integer linear programming (MILP), which is NP-hard in general. The goal of this paper is to develop a distributed and highly competitive solution to this problem.

IV. A DoF Scheduling Algorithm

A. Node Ordering in a Distributed Environment

Recall that in Section II, we described a node ordering concept for DoF scheduling in a multi-hop MIMO network. The relative ordering between two nodes directly determines DoF scheduling behavior at each node for IC and ensures no duplication (and thus no waste of DoF resource) in IC. For a centralized optimization problem, an optimal node ordering can be found by putting the ordering constraints (3) and (4) into the problem formulation (see Fig. 2).

However, in a distributed multi-hop network environment, establishing and maintaining an “explicit” ordering among all the nodes in the network does not appear to be feasible. The main difficulty here is that each node is only able to maintain information of its neighboring nodes. Although it is possible to maintain some relative ordering of a node with its neighboring nodes, it is not clear how such local ordering can lead to some explicit global ordering.

A major contribution in our distributed algorithm is that, through proper design, it is possible to have a per-node based local node ordering match to some “implicit” global ordering of all nodes in the network, achieving the *same* effect as that in a centralized problem. Specifically, we will show that the establishment of initial per-node based local node ordering and re-adjustment of neighboring node ordering during each iteration lead to an implicit but feasible global node ordering. This important result is stated in Theorem 1.

B. Algorithm Overview

In this section, we offer an overview of the proposed DoF scheduling algorithm to solve the optimization problem in Fig. 2. In essence, it is an iterative greedy algorithm that attempts to increase the minimum data rate among all links during each iteration. A flow chart of the algorithm is illustrated in Fig. 3, which includes three key modules: *link selection module* (LSM), *resource allocation module* (RAM), and *local reordering module* (LRM). Basically, the scheduling algorithm tries to increase the data rate of one of the links for the sessions during each iteration. Once such a link is identified, the RAM is invoked to see how the DoFs for SM can be increased among the time slots while all local and neighboring interference constraints are satisfied. If RAM is not able to yield a feasible increment, then we explore whether altering the local ordering of some nodes can yield a feasible increment. This is done by LRM.

In the rest of this section, we give an overview of each module. Detailed descriptions for the three modules are given in Sections V to VII.

- **LSM.** The goal of this module is to identify a link for rate increment in each iteration. We propose a *session-independent* link selection approach by establishing a list of all links in the network based on their potential “interference burden”. We show that this link selection approach is equivalent to the session-dependent link selection approach in terms of increasing the minimum rate among all sessions. We also show that this link selection approach can be implemented in a distributed environment.
- **RAM.** The goal of this module is to allocate DoF resource to increase the rate of the selected link. We first introduce local node ordering and global node ordering as well as the data structure that should be maintained at each node. Then, we explore the conditions under which the rate of the selected link can be increased in a given time slot and how DoFs should be allocated for the rate increment if the conditions are satisfied. We further show that if the DoF allocation *before* the rate increment is feasible based on a global node ordering, then the DoF allocation *after* the rate increment is also feasible based on a global node ordering. Based on the outcome for the rate increment in a given time slot, we explain how to allocate DoF resource for the rate increment in a time frame.
- **LRM.** When RAM fails to increase a data stream on the selected link in all time slots, we use LRM to alter some local node ordering so that some DoFs can be relieved from some nodes to accommodate one more data stream on the selected link. In a given time slot, we first identify the set of nodes $\mathcal{D}$ that are in shortage of DoF resource for the rate increment, and then explain how to adjust the local ordering for a node in $\mathcal{D}$ so that its remaining DoFs can be increased. We further show that the operations of local node ordering adjustment can preserve global feasibility of a solution and the existence of a global node ordering. Based on the outcome of the local ordering adjustment in a given time slot, we explain how to perform the local ordering adjustment in a time frame.

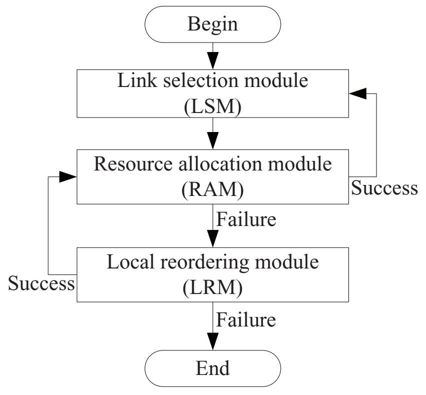
Begin
Link selection module (LSM)
Resource allocation module (RAM)
Success
Failure
Local reordering module (LRM)
Success
Failure
End

Fig. 3. A flow chart of proposed DoF scheduling algorithm.

V. Link Selection Module

Our distributed algorithm is a greedy algorithm that attempts to increase the minimum rate among all sessions iteratively. Several approaches may be considered to achieve this purpose. A straightforward approach is to identify a session with the minimum rate in the network and then try to increase the DoFs for SM on each link by one unit along the session's path. Unfortunately, our simulation results reveal that an algorithm based on this approach does not perform well. The failure of such a *session-dependent* link selection approach may be attributed to the fact that it ignores the significance of potential “interference burden” of each link in the network. By “interference burden” of a link, we mean the number of DoFs required at both the link's transmitter and receiver for IC. This consideration motivates us to pursue a *session-independent* link selection approach based on the potential interference burden of each active link in the network. Under this new approach, we first quantify the potential interference burden through a link priority definition as follows.

Definition 1: For a link (i, j), denote the number of nodes within the interference range of nodes i and j as q_i and q_j, respectively. Then, the priority of link (i, j), denoted as $q_{(i,j)}$, is defined as $q_{(i,j)} = q_i + q_j$.

In our approach, we sort all active links in the network based on *non-increasing* order of their link priorities into a list, which we denote as $\mathcal{B}$. A tie in link priority can be handled by any tie-breaking rule, e.g., giving a higher priority to the link with smaller source node ID. A small but important detail in $\mathcal{B}$ is the representation of a link that is traversed by multiple sessions. In our design, we would like to have session-independent link based approach to achieve the same effect as the session-dependent link based approach in term of increasing the minimum rate among all sessions iteratively. To do this, it is necessary to represent a link multiple times in list $\mathcal{B}$ if it is traversed by multiple sessions.

Based on this link list $\mathcal{B}$, we select link sequentially for rate increment (by one data stream). The reason why we consider links with higher priorities (and thus larger interference burden) first is because resource allocation task for these nodes is more demanding than those links with lower interference burden. Once these most demanding links are taken care of first, it would be easier to perform resource allocation for those less demanding links with the remaining network resource.

Distributed Implementation. We assume that the nodes in the network operate in a time-slotted frame structure (e.g., IEEE 802.16j mobile multi-hop relay (MMR) networks [3]), in which there is a dedicated control channel for scheduling. The dedicated control channel is divided into a set of sub-channels for information exchange among the nodes for scheduling. Time synchronization is achieved at the nodes in the network by some distributed algorithms (see e.g., [12]). For this DoF scheduling algorithm, each iteration is executed in a time slot.

To implement our session-independent link selection approach in a distributed manner, each link (its transmitter and receiver) needs to know in which time slot it shall execute RAM and LRM for rate increment.[1] If each link l knows its rank (position index), denoted as $\mathrm{rank}(l)$, in the list $\mathcal{B}$, then it can execute RAM and LRM for rate increment in the $[\mathrm{rank}(l)+kL]$-th time slot, where $k = 0, 1, 2, \cdots$, and L is the total number of links. Doing this is equivalent to increasing the rate of all sessions sequentially under the session-dependent link selection approach. If a link fails to increase its rate (in both RAM and RLM), then it broadcasts an "END" message to the other links and the algorithm terminates. Note that this broadcast operation is acceptable since it is used on an extremely limited basis (only once in our algorithm) [6].

Given that list $\mathcal{B}$ is invisible to each link in a distributed network, the question to ask is how each link can obtain its rank in list $\mathcal{B}$. This problem can be solved by using the distributed ranking algorithm in [14]. To apply the distributed ranking algorithm in our problem, we can have the transmitter of each link maintain the priority of that link and then execute the distributed ranking algorithm by treating the reciprocal priority of that link (i.e., $1/q_{(i,j)}$) as its initial value.[2] At the end of the ranking algorithm, the transmitter of each link can obtain the rank of that link.

VI. Resource Allocation Module

The goal of the RAM is to increase the rate of the selected link by one data stream for SM and cancel its interference to other nodes appropriately in a given time slot. To do this, we first discuss the relationship between per-node based local ordering and global node ordering. Based on this understanding, we introduce the data structure that should be maintained at each node, and then explore the condition under which a link rate can be successfully increased by one data stream in a given time slot.

Per-node based Local Ordering vs. Global Node Ordering. Recall that in a centralized environment, a global node ordering plays a key role in a feasible and efficient DoF scheduling [10]. This is because such ordering determines the IC responsibility of each node and its neighboring nodes, i.e., who needs to cancel interference to/from another node and how many DoFs are needed. However, in a distributed environment it is impractical to establish and maintain such a global node ordering in the network. Instead of pursuing a global node ordering, we propose to have each node establish and maintain a relative ordering with its neighboring nodes in a distributed environment. This can be done by having each node i maintain two sets of its neighboring nodes: (1) $\mathcal{I}_i(t)$ the set of nodes for which node i has allocated DoFs for IC: these nodes are considered before node i in the local ordering; and (2) $\mathcal{J}_i(t)$ the set of nodes that have allocated their DoFs to cancel interference either from or to node i: these nodes are considered after node i in the local ordering. We will show how these two sets can be established and maintained through a distributed mechanism. More importantly, we will show that by properly updating and maintaining these two sets at each node, one can determine which node is responsible for the cancellation of a particular interference. In other words, as far as the responsibility of IC is concerned, such a per-node based local ordering (based on $\mathcal{I}_i(t)$ and $\mathcal{J}_i(t)$ at each node i) achieves the same effect as a global node ordering. We show in Theorem 1 that based on these two sets, one can indeed identify a corresponding global node ordering in the network, despite that none of the nodes is aware of it.

Data Structure at a Node. Table I lists state information that we maintain at each node in the network. In the table, $\mathcal{I}_i(t)$ and $\mathcal{J}_i(t)$ are the two sets representing the local ordering at node i. Since $\mathcal{I}_i(t)$ contains the set of nodes for which node

[1] The T time slots considered in RAM and LRM is for scheduling in the next time frame.

[2] This distributed ranking algorithm can handle the case where multiple nodes have the same initial value. The reason why we also treat $1/q_{(i,j)}$ as the initial value is that the original algorithm [14] is for non-decreasing ranking while we desire a non-increasing ranking result here.

TABLE I
STATE INFORMATION AT EACH NODE i.

Symbol	Definition
$s_i(t)$	The status of node i (transmit, receive, or idle) in time slot t
$\mathcal{I}_i$	The set of nodes within node i's interference range
$\mathcal{I}_i^T(t)$	Transmitters in $\mathcal{I}_i$ in time slot t
$\mathcal{I}_i^R(t)$	Receivers in $\mathcal{I}_i$ in time slot t
$\mathcal{L}_i$	The set of incoming and outgoing links at node i
$\{z_l(t) : l \in \mathcal{L}_i\}$	The number of data streams on the incoming or outgoing links of node i
$\lambda_i^{\text{SM}}(t)$	The number of DoFs at node i allocated for SM in time slot t
$\lambda_i^{\text{IC}}(t)$	The number of DoFs at node i allocated for IC in time slot t
$\mathcal{T}_i(t)$	The set of nodes to which node i has established links in time slot t
$\mathcal{I}_i(t)$	The set of nodes for which node i has allocated DoFs for IC in time slot t
$\mathcal{J}_i(t)$	The set of node i's neighboring nodes that have allocated their DoFs to cancel interference either to or from nodes i in time slot t

i has allocated DoFs for IC in time slot t, we consider that these nodes are before node i in the local ordering. Similarly, since $\mathcal{J}_i(t)$ contains the set of node i's neighboring nodes that have allocated their DoFs to cancel interference either to or from nodes i in time slot t, these nodes are after node i in the local ordering. $\lambda_i^{\text{SM}}(t)$ is the number of DoFs allocated for SM at node i in time slot t. $\lambda_i^{\text{IC}}(t)$ is the number of DoFs allocated for IC at node i in time slot t. We denote $\overline{\lambda}_i(t)$ as the number of remaining DoFs at node i in time slot t. Thus, we have $\overline{\lambda}_i(t) = A_i - \lambda_i^{\text{SM}}(t) - \lambda_i^{\text{IC}}(t)$. $s_i(t)$ denotes the status of node i in time slot t, which is defined as follows: $s_i(t) =$ "T" if node i is a transmitter; $s_i(t) =$ "R" if node i is a receiver; and $s_i(t) =$ "I" if node i is idle.

During the initialization stage, each node is set to idle status (i.e., $s_i(t) =$ "I" for $i \in \mathcal{N}$, $t = 1, 2, \cdots, T$); each node allocates zero DoF for SM and IC (i.e., $\lambda_i^{\text{SM}}(t) = 0$ and $\lambda_i^{\text{IC}}(t) = 0$, $z_l(t) = 0$, $\mathcal{T}_i(t) = \emptyset$, $\mathcal{I}_i(t) = \emptyset$, $\mathcal{J}_i(t) = \emptyset$ for $i \in \mathcal{N}$, $l \in \mathcal{L}$, $t = 1, 2, \cdots, T$).

Rate Increment in a Given Time Slot. To increase the rate of link (i, j) by one data stream in time slot t, node i and node j first check their current status (transmit, receive, idle) in time slot t. If the status for both nodes meet the requirements (in Case I or Case II below), then nodes i and j as well as their neighboring nodes will check whether they have enough remaining DoFs for IC under their current local orderings. If yes, nodes i and j and relevant neighboring nodes update their state information to accommodate this one data stream increment on link (i, j).

Case I: In this case, link (i, j) is not active and we wish to add one data stream on this link if the following conditions are satisfied: (i) node i is idle; (ii) all of the receivers within node i's interference range have at least one remaining DoF to cancel interference from node i; (iii) node j is idle and its total DoFs are more than the sum of DoFs for SM at those nodes that are interfering node j (assuming node j will become the last node in the local ordering).

If the above conditions are satisfied, then node i and node j as well as their neighboring nodes do the following:

- At node i, its status is changed from idle to transmit. The number of DoFs consumed for SM at node i is updated to one. The rate of link (i, j) is increased to one. To update the local ordering at node i, we define node i to be the first node in its local ordering. To do this, we update $\mathcal{J}_i(t) = \mathcal{I}_i^R(t)$.
- At each of i's neighboring nodes $a \in \mathcal{J}_i(t)$, node a adds node i into set $\mathcal{I}_a(t)$ and increases its DoF consumption for IC by one.
- At node j, its status is updated from idle to receive. The number of DoFs consumed for SM is updated to one. The rate of link (i, j) is updated to one. Correspondingly, to update the local ordering at node j, we define node j to be the last node in its local ordering. To do this, we update $\mathcal{I}_j(t) = \mathcal{I}_j^T(t)$. The number of DoFs consumed for IC is updated to be the sum of data streams of its neighboring transmitters except node i.
- At each of node j's neighboring nodes $b \in \mathcal{I}_j(t)$, node b adds node j into set $\mathcal{J}_b(t)$.

Case II: In this case, link (i, j) is already active and we wish to add one more data stream on this link if the following conditions are satisfied: (i) node i is a transmitter and has at least one remaining DoF for SM; (ii) each node in $\mathcal{J}_i(t)$ has at least one remaining DoF to cancel interference from node i; (iii) node j is a receiver and has at least one remaining DoF for SM; (iv) each node in $\mathcal{J}_j(t)$ has at least one remaining DoF to cancel its interference to node j.

If the above conditions are satisfied, then nodes i and j as well as their neighboring nodes do the following:

- At node i, the number of DoFs consumed for SM is increased by 1. The rate of link (i, j) is increased by 1.
- Each node in $\mathcal{J}_i(t)$ increases its DoF consumption by 1.
- At node j, the number of DoFs consumed for SM is increased by 1. The rate of link (i, j) is increased by 1.
- Each node in $\mathcal{J}_j(t)$ increases its DoF consumption for IC by 1.

It is easy to see that rate increment (as described in Cases I and II) is a local operation and also feasible (in terms of DoF allocation) for those nodes involved in this operation. A natural question to ask is how such local operation will affect feasibility at a global level among all nodes. We now state an important property for the rate increment operation, which says that if the DoF allocation is feasible among all the nodes in the network, then this local operation will result in a new DoF allocation that is also globally feasible. Formally, denote $\pi(t)$ as a global ordering for all nodes in the network. Based on $\pi(t)$, suppose $\varphi(t)$ is a feasible DoF scheduling for SM and IC at all nodes in the network. Denote $\hat{\varphi}(t)$ as the new DoF scheduling after the rate increment operation on $\varphi(t)$. Then we have the following lemma.

Lemma 1: $\hat{\varphi}(t)$ is a globally feasible DoF scheduling. Further, there exists a global ordering $\hat{\pi}(t)$ that corresponds to $\hat{\varphi}(t)$.

We offer a proof sketch here. We show that this lemma holds by constructing a global ordering $\hat{\pi}(t)$ for all nodes in the network after the rate increment operation. This can be done by letting $\hat{\pi}(t) = [i\ \pi(t)\ j]$ for Case I and letting $\hat{\pi}(t) = \pi(t)$ for Case II. Based on the given conditions in the corresponding case, we find that every node in $\hat{\pi}(t)$ has enough DoFs for SM and IC after rate increment operation in Case I and II. Therefore, $\hat{\varphi}(t)$ is a globally feasible DoF scheduling and $\hat{\pi}(t)$ is a global ordering that corresponds to $\hat{\varphi}(t)$.

Resource Allocation in a Time Frame. Recall that there are T time slots in a time frame. If the rate increment operation described above fails in the first time slot, we try it again in the second time slot and so forth, until a rate increment is successful in a time slot or fails after all T time slots.

VII. Local Re-adjustment Module

Following the flow chart in Fig. 3, when RAM fails to increase one data stream on a given link in a time frame, we enter the *local re-adjustment module* (LRM). The goal of this module is to adjust the local ordering for the nodes associated with the underlying link so that IC responsibilities can be transferred from one node to another, thereby relieving some DoF resources for some nodes so as to accommodate a new data stream on the underlying link.

A. Local Ordering Adjustment in a Given Time Slot

Given that RAM fails to increase a data stream on a given link (i, j), we conclude that there is a lack of DoF resources at a subset of nodes among i, j, or their neighboring nodes based on the current local ordering at these nodes. This subset of nodes can be easily identified in a hypothesized scenario by looking for those nodes that would use more DoFs than their total DoFs should one more data stream were added on link (i, j). Denote this subset of bottleneck nodes as $\mathcal{D}$.

For each node $a \in \mathcal{D}$, we perform local ordering adjustment, with the goal of relieving one DoF (already used for IC) from a node so that a new DoF can become available. To avoid race condition in a distributed system, we use a token and let it pass from one node to the next in $\mathcal{D}$ so that at any time, only one node is allowed to perform local ordering adjustment. The initiation of this taken can be done by node i and then passed on to node j. The token is passed to the next node in $\mathcal{D}$ only if the local ordering adjustment in the previous node in $\mathcal{D}$ is successful (resulting in one free DoF at that node). Otherwise, the token will not be passed to the next node and we move on to the next time slot in a frame (see Section VII-B).

Now for a given node $a \in \mathcal{D}$ that currently holds the token, we first need to identify a set of a's neighboring nodes $\mathcal{E}$ that can relieve some of a's DoF consumption for IC. First, nodes in $\mathcal{E}$ should not include any node of i, j, and their neighboring nodes. Otherwise, we may run into the risk of a loop of changing local node ordering without yielding any net improvement. Second, nodes in $\mathcal{E}$ must be ahead of a in a's local ordering, i.e., $b \in \mathcal{I}_a(t)$, since a is using its DoFs to cancel interference from nodes in $\mathcal{E}$. Third, nodes in $\mathcal{E}$ should have enough remaining DoF resources to relieve a's DoF's consumption for IC to b, i.e., $\overline{\lambda}_b(t) \geq \lambda_a^{\text{SM}}(t)$. Finally, we need to ensure that there does not exist another node, say c, that is in a higher local order than node $b \in \mathcal{E}$ but in a lower local order than a. This will ensure that a local node ordering swap between a and b will not violate the local ordering between b and c.

For the set of candidate nodes in $\mathcal{E}$ for node a, we only need one node to swap its local ordering with a. In our algorithm, we choose a node in $\mathcal{E}$ that has the most remaining DoFs. A tie can be broken by selecting the node with a smaller node ID. Denote this node in $\mathcal{E}$ as b^*. For nodes a and b^*, we perform the following operation: (i) node a moves b^* from its $\mathcal{I}_a(t)$ to $\mathcal{J}_a(t)$, indicating node b^*'s new order is now behind node a; (ii) node a no longer needs to cancel interference from node b^* and its remaining DoFs are increased, i.e. $\lambda_a^{\text{IC}}(t) := \lambda_a^{\text{IC}}(t) - \lambda_{b^*}^{\text{SM}}(t)$, $\overline{\lambda}_a(t) := \overline{\lambda}_a(t) + \lambda_{b^*}^{\text{SM}}(t)$; (iii) node b^* moves a from its $\mathcal{J}_{b^*}(t)$ to $\mathcal{I}_{b^*}(t)$, indicating node a's new order is now before node b^*; (iv) node b^* now needs to cancel interference from node a and its remaining DoFs are decreased, i.e., $\lambda_{b^*}^{\text{IC}}(t) := \lambda_{b^*}^{\text{IC}}(t) + \lambda_a^{\text{SM}}(t)$, $\overline{\lambda}_{b^*}(t) := \overline{\lambda}_{b^*}(t) - \lambda_a^{\text{SM}}(t)$.

A question to ask is how such a local node reordering operation will affect feasibility at a global level among all the nodes. We now state an important property, which says that if the DoF scheduling is feasible among all the nodes in the network, then the LRM operation will result in a new DoF scheduling that is also globally feasible. Formally, denote $\pi(t)$ as a global ordering for all the nodes in the network. Based on $\pi(t)$, suppose $\varphi(t)$ is a feasible DoF scheduling for SM and IC for all nodes in the network. Denote $\hat{\varphi}(t)$ as the DoF scheduling for all the nodes after LRM is performed at some nodes a and b^* under $\varphi(t)$. Then we have the following lemma:

Lemma 2: $\hat{\varphi}(t)$ is a globally feasible DoF scheduling. Further, there exists a global ordering $\hat{\pi}(t)$ that corresponds to $\hat{\varphi}(t)$.

We offer a proof sketch here. A proof can be based on construction. Denote D as the set of nodes between a and b^* in $\pi(t)$. Denote B as the set of nodes before b^* in $\pi(t)$ and denote C as the set of nodes after a in $\pi(t)$. Then we have $\pi(t) = [B\ b^*\ D\ a\ C]$. Further, denote Γ as the set of nodes that are in a lower local ordering than node a. We construct a new global ordering $\hat{\pi}(t)$ for all nodes in the network after the rate increment operation as follows: $\hat{\pi}(t) = [B\ \ D \cap \Gamma\ \ a\ \ b^*\ \ D \cap \Gamma^c\ \ C]$, where Γ^c is the complement of set Γ. Based on the conditions given in LRM, we find that every node in $\hat{\pi}(t)$ has enough DoFs for SM and IC after the rate increment operation. Therefore, $\hat{\varphi}(t)$ is a globally feasible scheduling and $\hat{\pi}(t)$ is a global ordering that corresponds to $\hat{\varphi}(t)$.

B. Local Ordering Adjustment in a Time Frame

Recall that there are T time slots in a time frame. If the local ordering adjustment described above fails in the first time slot, we try again in the second time slot and so forth, until local ordering adjustment is successful in a time slot or fails after all T time slots.

VIII. Global Feasibility of Final Solution

Recall that both RAM and LRM modules in our algorithm perform local operations and are amenable for distributed implementation. A natural question to ask is whether the final DoF scheduling at all nodes in the network through these iterative local algorithms is still feasible at a global level. The following theorem answers this question.

Theorem 1: Suppose that $\varphi(t)$ is the final DoF scheduling for SM and IC at all nodes in the network. Then, $\varphi(t)$ is a globally feasible solution. Further, there exists a global ordering $\pi(t)$ that corresponds to $\varphi(t)$.

Proof. We prove it by induction. Since there are L links in the network, the maximum rate of each link (i, j) is A_i data streams, and there are T time slots in a frame, the algorithm in Fig. 3 will terminate in at most LTA iterations, where $A = \max\{A_i : 1 \le i \le N\}$. Denote $\varphi_n(t)$ as the DoF scheduling at all nodes in the network at the end of n-th iteration.

Base case: We first show that the theorem holds for $n = 1$. To see this, note that before the first iteration, none of the DoFs at any node in the network is allocated. So the LSM selects the link with the highest priority, say link (i, j). Since none of the DoFs on nodes i and j has been allocated, we perform RAM and obtain $\varphi_1(t)$, which is also a global feasible solution. For $\varphi_1(t)$, there exists a trivial global node ordering $\pi_1(t)$.

Inductive step: We now show that if $\varphi_n(t)$ is a global feasible solution with a global ordering $\pi_n(t)$, then at the end of the next iteration $(n + 1)$, $\varphi_{n+1}(t)$ is a also global feasible solution with a global ordering $\pi_{n+1}(t)$. From $\varphi_n(t)$ to $\varphi_{n+1}(t)$, the operation may be a successful RAM at first try or first LRM and then RAM. From Lemma 2, we know that the LRM will preserve global feasibility of a solution as well as the existence of a global node ordering. From Lemma 1, we know that RAM will also preserve the global feasibility of a solution as well as the existence of a global node ordering. Therefore, if $\varphi_n(t)$ is a feasible solution with a global ordering $\pi_n(t)$, then $\varphi_{n+1}(t)$ is a feasible solution with a global ordering $\pi_{n+1}(t)$.

Combining the base case and the inductive step, the proof is complete. □

Theorem 1 shows that the solution found by the proposed DoF scheduling algorithm is feasible. We can also verify that the complexity of the algorithm is polynomial time.

IX. Performance Evaluation

In this section, we present simulation results to demonstrate the performance of the proposed DoF scheduling algorithm. Ideally, the best performance benchmark would be the optimal solution to the OPT-DoF problem in Fig. 2. However, OPT-DoF formulation is MILP and thus NP-hard in general. Since an optimal solution is unlikely to be available in reasonable amount of time, we will compare the performance of the proposed algorithm against an upper bound of OPT-DoF, which can be obtained by CPLEX (for a given termination time, e.g., 3 hours).[3] Note that the optimal solution lies between the upper bound and the feasible solution found by our algorithm. Therefore, if simulation results show that the objective by our algorithm is close to the upper bound by CPLEX, then we can infer that the result by the proposed algorithm is very close to the optimal solution (thus highly competitive).

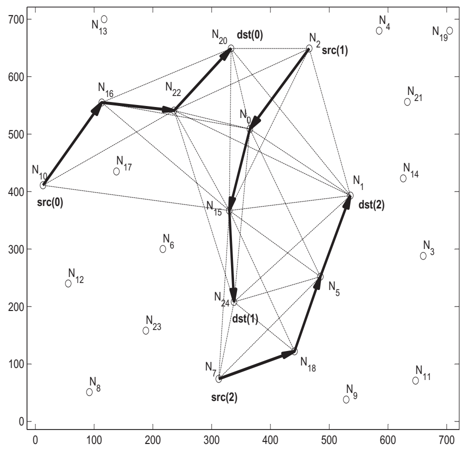

Fig. 4. A 25-node network instance.

A. Simulation Setting

For ease of exposition, we normalize all units for distance, time, bandwidth, and data rate with appropriate dimensions. We consider networks of three sizes: (i) 25 nodes in a 750×750 area with 3 sessions; (ii) 50 nodes in a 1000×1000 area with 4 sessions; and (iii) 100 nodes in a 1500×1500 area with 5 sessions. We assume that all transmit nodes have the same transmission range 180 and the same interference range 360. For each network size, 100 randomly generated network instances are studied. For each network instance, the source and destination nodes of each session are randomly selected, with the route between them being shortest path route. We assume that each node is equipped with four antennas and there are four time slots in a time frame.

B. A Case Study

Before we present the complete simulation results, we first show a case study of a 25-node network instance in Fig. 4. In this figure, solid line with arrow represents a link while dashed line represents a potential interference. There are three sessions in this network: session 1 is from N_{10} to N_{20}; session 2 is from N_2 to N_{24}; and session 3 is from N_7 to N_1.

Figure 5 and Table II give the details of the solution found by our algorithm in the first time slot. For example, the set of active links are (N_{10}, N_{16}), (N_{22}, N_{20}), (N_2, N_0), (N_{15}, N_{24}), (N_7, N_{18}), and (N_5, N_1). The two local node sets at each

[3]When using CPLEX to solve MILP problems, it always yields a lower bound (a best-known and feasible solution) and an upper bound for the objective value. The gap between the lower bound and the upper bound become smaller and smaller if enough computation time is provided. If the lower bound hits the upper bound, then an optimal solution is found.

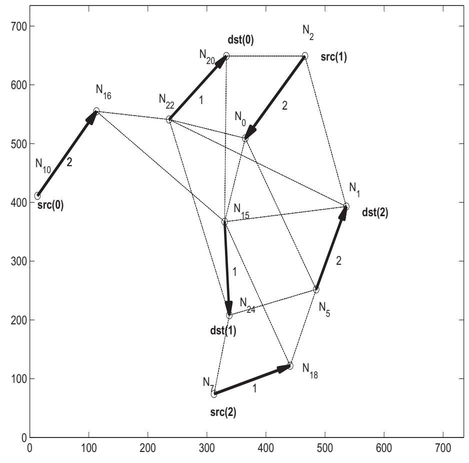

Fig. 5. Active links in the first time slot.

TABLE II
LOCAL NODE ORDERING AND DoF ALLOCATION AT EACH NODE IN THE FIRST TIME SLOT.

Node i	$\mathcal{I}_i(t)$	$\mathcal{J}_i(t)$	$\lambda_i^{\text{SM}}(t)$	$\lambda_i^{\text{IC}}(t)$
N_0	$\{N_{22}\}$	$\{N_{15}, N_5\}$	2	1
N_1	$\{N_{15}, N_{22}\}$	$\{N_2\}$	2	2
N_2	$\{N_1\}$	$\{N_{20}\}$	2	2
N_5	$\{N_0\}$	$\{N_{18}, N_{24}\}$	2	2
N_7	$\{N_{24}\}$	$\emptyset$	1	1
N_{10}	$\emptyset$	$\emptyset$	2	0
N_{15}	$\{N_0\}$	$\{N_{18}, N_1, N_{16}, N_{20}\}$	1	2
N_{16}	$\{N_{15}, N_{22}\}$	$\emptyset$	2	2
N_{18}	$\{N_{15}, N_5\}$	$\emptyset$	1	3
N_{20}	$\{N_2, N_{15}\}$	$\emptyset$	1	3
N_{22}	$\emptyset$	$\{N_0, N_1, N_{16}, N_{24}\}$	1	0
N_{24}	$\{N_5, N_{22}\}$	$\{N_7\}$	1	3

node are given in Table II. For example, for node N_0, it considers N_{22} before itself (i.e., $\mathcal{I}_{N_0}(1) = \{N_{22}\}$) while N_{15} and N_5 after itself (i.e., $\mathcal{J}_{N_0}(1) = \{N_{15}, N_5\}$). Node N_0 uses two DoFs for SM to receive two data streams from N_2 (i.e., $\lambda_{N_0}^{\text{SM}}(1) = 2$) and uses one DoF to cancel interference from N_{22} (i.e., $\lambda_{N_0}^{\text{IC}}(1) = 1$).

We can piece up a global ordering among the nodes in the network based on each node's local ordering. For the first time slot, a global ordering among the nodes in the network is $[N_{10}, N_{22}, N_0, N_5, N_{15}, N_{16}, N_{18}, N_{24}, N_1, N_2, N_7, N_{20}]$. It is easy to verify that the global ordering is consistent with the local orderings and the DoF allocation at each node is feasible.

The objective found by our algorithm is 0.75, corresponding to 3 data streams in 4 time slots for a bottleneck session. The upper bound by CPLEX is also 0.75, which shows that our solution is optimal in this case study.

C. Complete Simulation Results

We now present complete simulation results (100 network instances for each network size). The average ratios between the results by our algorithm and the upper bound by CPLEX are 84.4%, 83.6%, and 82.9% for the 25-, 50-, and 100-node networks. Since the optimal solution lies between the feasible solution by our algorithm and the upper bound by CPLEX, we conclude that our solutions are very close to the optimum.

X. CONCLUSIONS

In this paper, we studied DoF scheduling for a multi-hop MIMO network. Specifically, we studied how to perform DoF allocation among the nodes for SM and IC so as to maximize the minimum rate among a set of sessions. We formulated this problem as a mixed integer linear program (MILP). Subsequently, we developed an efficient DoF scheduling algorithm. Some highlights of this DoF scheduling algorithm include: (i) amenable to local implementation; (ii) polynomial time complexity; (iii) feasibility of final solution (upon algorithm termination). The performance of our algorithm was substantiated by simulation results, which showed that our algorithm can offer results close to upper bounds found by CPLEX solver, indicating that the results of our algorithm are highly competitive.

ACKNOWLEDGMENT

The work of Y.T. Hou and W. Lou was supported in part by NSF grants 1102013 (Hou), 0925719 (Hou), 1156311 (Lou) and 1156318 (Lou).

REFERENCES

[1] R. Bhatia and L. Li, “Throughput optimization of wireless mesh networks with MIMO links,” in *Proc. IEEE INFOCOM,* pp. 2326–2330, Anchorage, AK, May 2007.

[2] D. Blough, G. Resta, P. Santi, R. Srinivasan, and L.M. Cortes-Pena, “Optimal one-shot scheduling for MIMO networks,” in *Proc. IEEE SECON,* pp. 377–385, Salt Lake City, Utah, June 2011.

[3] V. Genc, S. Murphy, Y. Yang, and J. Murphy, “IEEE 802.16J relay-based wireless access networks: an overview,” *IEEE Wireless Communications,* vol. 15 , no. 5, pp. 56–63, Oct. 2008.

[4] B. Hamdaoui and K.G. Shin, “Characterization and analysis of multi-hop wireless MIMO network throughput,” in *Proc. ACM MobiHoc,* pp. 120–129, Montreal, Quebec, Canada, Sep. 2007.

[5] S. Jafar and M. Fakhereddin, “Degrees of freedom for the MIMO interference channel,” *IEEE Trans. on Information Theory,* vol. 53, no. 7, pp. 2637–2642, July 2007.

[6] N.A. Lynch, *Distributed Algorithms,* Chapter 15, Morgan Kaufmann Publishers, Inc., San Francisco, CA 1996.

[7] J. Mundarath, P. Ramanathan, and B.D. Van Veen, “Exploiting spatial multiplexing and reuse in multi-antenna wireless ad hoc networks,” *Elsevier Ad Hoc Networks,* vol. 7, no. 2, pp. 281–293, March 2009.

[8] Q.H. Spencer, A.L. Swindlehurst, and M. Haardt, “Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels,” *IEEE Trans. on Signal Processing,* vol. 52, no. 2, pp. 461–471, Feb. 2004

[9] H.D. Sherali and W.P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems,* Chapter 8, Kluwer Academic Publishers, 1999.

[10] Y. Shi, J. Liu, C. Jiang, C. Gao, and Y.T. Hou, “An optimal MIMO link model for multi-hop wireless networks,” in *Proc. IEEE INFOCOM,* pp. 1916–1924, Shanghai, China, April 2011.

[11] K. Sundaresan, R. Sivakumar, M. Ingram, and T-Y. Chang, “Medium access control in ad hoc networks with MIMO links: Optimization considerations and algorithms,” *IEEE Trans. on Mobile Computing,* vol. 3, no. 4, pp. 350–365, Oct. 2004.

[12] R. Solis, V.S. Borkar, and P.R. Kumar, “A new distributed time synchronization protocol for multihop wireless networks,” in *Proc. IEEE Conference on Decision and Control,* pp. 2734–2739, San Diego, CA., Dec. 2006.

[13] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Chapter 7, Cambridge University Press, Cambridge, UK, 2005.

[14] S. Zaks, “Optimal distributed algorithms for sorting and ranking,” *IEEE Trans. on Computers,* vol. 5, no. 1, pp. 376–379, April 1985.