Joint Flow Routing and DoF Allocation in Multihop MIMO Networks

Xiaoqi Qin, *Student Member, IEEE*, Xu Yuan, *Student Member, IEEE*, Yi Shi, *Senior Member, IEEE*, Y. Thomas Hou, *Fellow, IEEE*, Wenjing Lou, *Fellow, IEEE*, and Scott F. Midkiff, *Senior Member, IEEE*

***Abstract*—Recently, degree-of-freedom (DoF)-based models have been widely used to study MIMO network performance. Existing DoF-based models differ in their interference cancellation (IC) behavior and many of them suffer from either loss of solution space or possible infeasible solutions. To overcome these limitations, a new DoF-based model, which employs an IC scheme based on node-ordering was proposed. In this paper, we apply this new DoF IC model to study a throughput maximization problem in a multihop MIMO network. The problem formulation involves joint consideration of flow routing and DoF allocation and falls in the form of a mixed-integer linear program (MILP). Our main contribution is an efficient polynomial time algorithm that offers a competitive solution to the MILP through a series of linear programs (LPs). The algorithm employs a sequential fixing framework to obtain an initial feasible solution and then improves the solution by exploiting: 1) the impact of node ordering on DoF consumption for IC at a node and 2) route diversity in the network. Simulation results show that the solutions obtained by our proposed algorithm are competitive and feasible.**

***Index Terms*—Network throughput optimization, algorithm design, multi-hop MIMO networks.**

I. Introduction

It is well known that MIMO technology delivers significant performance gain in terms of channel capacity and reliability [1]–[4]. To date, MIMO has become an essential element of wireless communications including wireless LAN (802.11n) [5], WiMAX (802.16e) [6] and LTE (3GPP) [7]. Despite of rapid advances of physical layer techniques for single-hop communications [8]–[14], our knowledge on how to exploit the advantages of MIMO in multi-hop networks has been relatively limited. The main difficulty here is that mathematical characterization of MIMO's physical layer signal processing involves complex matrix operations (due to the use of multiple antennas). Such matrix manipulations pose a technical barrier in the design and analysis of network algorithms and protocols. Therefore, a tractable and accurate MIMO model is needed for multi-hop MIMO network research.

Manuscript received April 30, 2015; revised August 29, 2015; accepted October 15, 2015. Date of publication November 2, 2015; date of current version March 8, 2016. This work was supported in part by NSF grants 1343222, 1247830, 1064953, 1102013 and ONR grant N000141310080. Part of W. Lou's work was completed while she was serving as a Program Director at the NSF. Any opinion, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not reflect the views of the NSF. An abridged version of this paper was presented at IEEE MASS, Hangzhou, China, October 2013 [34]. The associate editor coordinating the review of this paper and approving it for publication was Q. Li.

X. Qin, X. Yuan, Y. T. Hou, W. Lou, and S. F. Midkiff are with Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: xiaoqi@vt.edu; xuy10@vt.edu; thou@vt.edu; wjlou@vt.edu; midkiff@vt.edu).

Y. Shi is with the Intelligent Automation Inc., Rockville, MD 20855 USA (e-mail: yshi@vt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2015.2496961

Realizing the difficulties, researchers in the networking community extended the concept of degree-of-freedom (DoF), which was originally defined by the information theory (IT) community to represent the multiplexing gain of a MIMO channel [15]–[17], to characterize the spatial freedom provided by multiple antennas at a MIMO node. Since then, various DoF-based models have been developed to analyze the spatial multiplexing (SM) and interference cancellation (IC) capabilities of MIMO [18]–[22]. Instead of complex matrix operations, these DoF-based models only require simple numeric computations on the number of DoFs to identify a feasible DoF region. The main idea of these DoF-based models is as follows: (i) The number of available DoFs at a node is equal to the number of its antennas. (ii) A node consumes DoFs for SM. Specifically, a transmit node consumes DoFs to transmit its data streams while a receive node consumes DoFs to receive its desired data streams. (iii) A node consumes DoFs for IC. Specifically, a transmit node may cancel its interference to its neighboring unintended receive nodes by consuming its DoFs; likewise, a receive node may cancel the interference from its neighboring unintended transmit nodes by consuming its DoFs. (iv) A node can use some or all of its DoFs for SM and IC, as long as the total number of DoFs consumed for SM and IC does not exceed its available DoFs.

Based on different IC schemes, the DoF-based models can be classified into two major types: *conservative models* and *optimistic models*. The conservative models ensure feasibility by employing different kinds of predefined IC rules, such as IC at both transmit and receive nodes, restrictions on IC at receiver nodes, among others. As a result, a conservative model [18], [21], [22] loses some feasible solutions. The optimistic models, on the other hand, tend to incorrectly enlarge the feasible solution space due to the lack of a systematic IC scheme. As a result, an optimistic model [19], [20] may offer infeasible solutions. To address the problems associated with the conservative and optimistic models, a new DoF-based MIMO model was developed by Shi *et al.* [23], [24]. The novelty of this model is a disciplined IC scheme based on an ordered node list. Under this model, each node in the network is associated with an order and it is only responsible for canceling interference to/from those nodes that are before itself in the ordered list. That is, a transmit node only needs to cancel its interference to neighboring receive nodes ahead itself in the ordered node list, instead of all its neighboring receive nodes in the network. Similarly, a receive node only needs to cancel interference from neighboring transmit nodes that are before itself in the ordered node list, instead of all its neighboring transmit nodes in the network. By employing the novel "node ordering" concept, IC at each node can be performed in a systematic and disciplined manner. As a result, the potential waste of DoF resources is avoided by eliminating possible duplication in IC (in conservative models). Further, DoF allocation solution is guaranteed to be feasible in contrast to the optimistic models [25].

With the new DoF model in [23], one can study MIMO multi-hop networks without the limitations associated with the conservative and optimistic DoF models. In this paper, we explore how to apply this model to study a throughput maximization problem in a multi-hop MIMO network. Note that the scope of this paper and [23] is fundamentally different. In [23], the authors proposed a new interference cancellation (IC) scheme that can be used for DoF allocation in a multi-hop MIMO network. Although they presented a case study involving a throughput maximization problem, the problem is much simpler than the one that we study in this paper. For example, routing for each session was assumed to be given *a priori* in [23] rather than being considered as part of the optimization problem in this paper. Further, there was no discussion on how to solve such an optimization problem. It merely relied on a commerical solver (CPLEX) to demonstrate results. This was OK as the main objective of [23] was to introduce a new MIMO IC scheme rather than pursuing an efficient solution procedure for a throughput maximization problem. In contrast, the objective of this paper is to pursue an efficient solution procedure for a throughput maximization problem – a problem in a more complex (and general) form than that in [23]. Specifically, in this paper, we consider a set of unicast sessions in a MIMO network. The objective is to maximize the minimum achievable session rate (in DoFs). For simplicity, we assume that fixed modulation and coding scheme (MCS) is used for each data stream and each data stream corresponds to one unit data rate. Through joint formulation of flow routing and DoF allocation, we obtain a mixed-integer linear program (MILP), which is NP-hard in general. Although a commercial solver (CPLEX) may solve our problem for small-sized networks, a more efficient (and polynomial time) algorithm is needed to handle networks of larger sizes.

The main contribution of this paper is an efficient algorithm for the joint flow routing and DoF allocation problem. The proposed solution is an iterative greedy algorithm that solves a series (but limited number) of LPs, each of which has polynomial time complexity. The algorithm consists of two stages. In the first stage, the algorithm employs a *sequential fixing* (SF) technique [26, Chapter 10] to obtain an initial feasible solution. The SF technique is itself based on a series of LPs. In the second stage, the algorithm improves the initial feasible solution. By identifying a "bottleneck" link, the algorithm attempts to increase DoF allocation (for SM) on the bottleneck link by altering the ordering of the node list. This idea exploits the unique property in the DoF model where a node's DoF consumption for IC depends on its position in the node-ordering list. When altering node-ordering can no longer improve the solution, our algorithm tries to make improvement by altering routes. It tries to find a parallel route to bypass the bottleneck link. Throughout the iterations, our algorithm ensures that DoF constraints for SM and IC at each node are satisfied. Simulation results show that our proposed algorithm can offer a competitive solution to the MILP. Further, the solution by our proposed algorithm is guaranteed to be feasible.

The reminder of this paper is organized as follows. In Section II, we develop a problem formulation for the throughput maximization problem by joint consideration of flow routing and DoF allocation. In Section III and Section IV, we present our algorithm to solve the throughput maximization problem in two stages. Section V gives a complexity analysis of the proposed algorithm. Section VI presents simulation results. Section VII concludes this paper.

TABLE I
NOTATION

Symbol	Definition
$\mathcal{N}$	The set of nodes in the network
$\mathcal{F}$	The set of sessions in the network
T	The number of time slots in a frame
$\mathcal{T}_i$	The set of nodes within the transmission range of node i
$\mathcal{R}_i$	The set of nodes within the interference range of node i
A_i	Number of antennas at node i
$x_i[t]$	=1 if node $i \in \mathcal{N}$ is a transmitter in time slot t, and is 0 otherwise
$y_i[t]$	=1 if node $i \in \mathcal{N}$ is a receiver in time slot t, and is 0 otherwise
$z_{ij}[t]$	The number of data streams transmitted from node i to node j in time slot t, $i, j \in \mathcal{N}$
$\pi[t]$	An ordered node list for IC in time slot t
$\pi_i[t]$	The position of node $i \in \mathcal{N}$ in $\pi[t]$
$\theta_{ji}[t]$	Binary indicator showing the relationship between nodes i and j in $\pi[t]$, $i, j \in \mathcal{N}$
$r(f)$	The data rate of session $f \in \mathcal{F}$
$r_{ij}(f)$	The data rate attributed to session $f \in \mathcal{F}$ on link (i, j) in time slot t
$r_{\min}$	The minimum data rate among all sessions

II. Modeling and Formulation

In this section, we study a throughput maximization problem for multi-hop MIMO networks. For a set of sessions in the network, our objective is to maximize the achievable session rate (in DoFs) among the sessions by optimizing variables in flow routing (network layer) and DoF allocation (link layer).

A. Mathematical Modeling

Table I lists notation in this paper. We consider a multi-hop MIMO network with a set of $\mathcal{N}$ nodes, with $N = |\mathcal{N}|$ being the number of nodes. Each MIMO node i has A_i antennas. Denote $\mathcal{F}$ as a set of sessions in the network. For each session $f \in \mathcal{F}$, denote $s(f)$ and $d(f)$ as its source and destination nodes, respectively. Denote $r(f)$ as the achieved throughput (in DoFs) of session $f \in \mathcal{F}$. We assume that time slot based scheduling is used at the link layer. Denote T as the number of time slots in a frame.

Half-duplex Constraint. We assume that wireless transceivers are half-duplex.[1] In a time slot based system, a half-duplex transceiver operates in one of three modes in a time slot: transmit, receive, or idle. To model this behavior in a time slot, we define two binary variables $x_i[t]$ and $y_i[t]$ to indicate whether node i is a transmit node or a receive node in time slot t, respectively. That is, $x_i[t] = 1$ if node i is a transmit node in time slot t and 0 otherwise (either idle or receive); $y_i[t] = 1$ if node i is a receive node in time slot t and 0 otherwise (either idle or transmit); Then the half-duplex constraint can be modeled as

$$x_i[t] + y_i[t] \le 1, (i \in \mathcal{N}, 1 \le t \le T). \tag{1}$$

We assume the total number of DoFs at a node i is equal to its number of antenna elements A_i. Denote $z_{ij}(t)$ as the number of data streams from node i to node j in time slot t. If node i is not an active transmitter in time slot t, then no data stream is transmitted from this node, i.e., $\sum_{j \in \mathcal{T}_i} z_{ij}[t] = 0$ if $x_i[t] = 0$, where $\mathcal{T}_i$ is the set of nodes within the transmission range of node i. Otherwise, the total number of DoFs used for transmission cannot exceed the total number of antennas A_i at this node, i.e., $1 \le \sum_{j \in \mathcal{T}_i} z_{ij}[t] \le A_i$ if $x_i[t] = 1$. These two cases can be formulated as

$$x_i[t] \le \sum_{j \in \mathcal{T}_i} z_{ij}[t] \le A_i \cdot x_i[t], (i \in \mathcal{N}, 1 \le t \le T). \tag{2}$$

Similarly, considering whether or not node i is a receive node in time slot t, we have

$$y_i[t] \le \sum_{j \in \mathcal{T}_i} z_{ji}[t] \le A_i \cdot y_i[t], (i \in \mathcal{N}, 1 \le t \le T), \tag{3}$$

where we assume node j and node i have the same transmission range, i.e., if $i \in \mathcal{T}_j$ then $j \in \mathcal{T}_i$, and vice versa.

Ordering and DoF Constraints. The "node ordering" concept was proposed in [23] to ensure feasibility at the physical (PHY) layer while avoiding duplication in canceling the same interference by both the transmitter and receiver in a MIMO network. An optimal ordering can be found by incorporating ordering variables in the problem formulation.

Denote $\pi[t]$ as a list of length N, with each element containing a node in the network. The position of the node in the list defines the "order" of that node. Denote $\pi_i[t]$ as the order (position) of node i in $\pi[t]$. For example, if $\pi_i[t] = 5$, then it means that node i is in the fifth element in the ordered list. Therefore, we have

$$1 \le \pi_i[t] \le N, (i \in \mathcal{N}, 1 \le t \le T). \tag{4}$$

We use a binary variable $\theta_{ji}[t]$ to indicate the relative position between two nodes i and j in $\pi[t]$. i.e., $\theta_{ji}[t] = 1$ if node j is before node i in $\pi[t]$ and 0 otherwise. Based on the definition of $\theta_{ji}[t]$, it can be easily verified that the following relationships hold between $\pi_i[t]$ and $\pi_j[t]$:

$$\pi_i[t] - N \cdot \theta_{ji}[t] + 1 \le \pi_j[t] \le \pi_i[t] - N \cdot \theta_{ji}[t] + N - 1,$$
$$(i, j \in \mathcal{N}, i \ne j, 1 \le t \le T). \tag{5}$$

A node can use its DoFs for either SM or IC, as long as the number of consumed DoFs does not exceed its total available DoFs. Depending on whether the node is a transmit or receive node in time slot t, it has different IC behavior as follows:

- If the node is a transmit node, in addition to SM, the node should use its DoFs to cancel its interference to all the unintended neighboring receive nodes (within its interference range) that are before itself in the ordered node list $\pi[t]$. The number of DoFs consumed by this transmit node for IC is equal to the sum of intended data streams received by those unintended receive nodes.
- If the node is a receive node, in addition to SM, the node should use its DoFs to cancel interference from all unintended neighboring transmit nodes (whose interference range covers this receive node) that are before itself in the ordered node list $\pi[t]$. The number of DoFs consumed by this node for IC is equal to the sum of data streams transmitted by those unintended transmit nodes.

Note that in either case (transmitter or receiver), the node only needs to consider the nodes before itself in $\pi[t]$ for IC. Interference to/from nodes that are after it in the ordered node list $\pi[t]$ will be taken care of by those nodes later when we go through the list. This is the key in allocating DoF resources for IC. Denote $\mathcal{R}_i$ as the set of nodes within the interference range of node i. We now model both cases mathematically as follows:

$$\text{If } x_i[t] = 1, \text{ then} \sum_{j \in \mathcal{T}_i} z_{ij}[t] + \sum_{j \in \mathcal{R}_i} (\theta_{ji}[t] \sum_{k \in \mathcal{T}_j}^{k \ne i} z_{kj}[t]) \le A_i,$$
$$(i \in \mathcal{N}, 1 \le t \le T), \tag{6}$$

where on the left side of the inequality, the first and second terms represent the number of DoFs consumed by node i for SM and IC, respectively.

$$\text{If } y_i[t] = 1, \text{ then} \sum_{j \in \mathcal{T}_i} z_{ji}[t] + \sum_{j \in \mathcal{R}_i} (\theta_{ji}[t] \sum_{k \in \mathcal{T}_j}^{k \ne i} z_{jk}[t]) \le A_i,$$
$$(i \in \mathcal{N}, 1 \le t \le T), \tag{7}$$

where on the left side of the inequality, the first and second terms represent the number of DoFs consumed by node i for SM and IC, respectively.

Flow Balance Constraints. For flexibility and better load balancing, we allow flow splitting in the network. That is, the flow of a session (with granularity of one DoF) may split and merge inside the network in whatever manner as long as it can help to achieve a higher throughput. Denote $r_{ij}(f)$ as the data rate (in DoFs) on link (i, j) that is attributed to session

[1]Although full-duplex MIMO node has been demonstrated in recent research literatures [27], [28], its near-term availability remains unclear. Should it become widely available in the future, only some minor extensions are needed in our formulation and proposed algorithm.

$f \in \mathcal{F}$, where $i \in \mathcal{N}$ and $j \in \mathcal{T}_i$. Then we have the following flow balance constraints:

- If node i is the source node of session f (i.e., $i = s(f)$), then

$$\sum_{j\in\mathcal{T}_i} r_{ij}(f) = r(f), \quad (f \in \mathcal{F}). \tag{8}$$

- If node i is an intermediate relay node for session f (i.e., $i \neq s(f), i \neq d(f)$), then

$$\sum_{j\in\mathcal{T}_i}^{j\neq s(f)} r_{ij}(f) = \sum_{k\in\mathcal{T}_i}^{k\neq d(f)} r_{ki}(f), \quad (f \in \mathcal{F}, i \in \mathcal{N}). \tag{9}$$

- If node i is the destination node for session f (i.e., $i = d(f)$), then

$$\sum_{j\in\mathcal{T}_i} r_{ji}(f) = r(f), \quad (f \in \mathcal{F}). \tag{10}$$

It can be easily verified that once (8) and (9) are satisfied, (10) must also be satisfied. As a result, it is sufficient to just include (8) and (9) in the formulation.

We assume that fixed modulation and coding scheme (MCS) is used for each data stream and each data stream corresponds to one unit data rate. For each link (i, j), the sum of the data rates over all sessions that traverse this link cannot exceed the average data rate that can be supported by link (i, j) over T time slots. Then on each link (i, j), we have:

$$\sum_{f\in\mathcal{F}} r_{ij}(f) \le \frac{1}{T}\sum_{t=1}^{T} z_{ij}[t], \quad (i \in \mathcal{N}, j \in \mathcal{T}_i), \tag{11}$$

where T is the number of time slots in a frame.

B. Formulation

In multi-hop networks, a fundamental problem is to maximize the achievable end-to-end session throughput. Given that there are multiple sessions in the network, we also need to consider fairness issue when maximizing throughput among competing sessions. Fairness can be defined in different ways. In this study, we choose the objective of maximizing the minimum throughput among all the active sessions in the network ($r_{\min}$). This objective aims to achieve both throughput maximization and fairness among the sessions. The problem can be formulated as follows:

OPT-raw	
max	$r_{\min}$
s.t	$r_{\min} \le r(f), \quad f \in \mathcal{F}$;
	half duplex constraints: (1)–(3);
	ordering and DoF constraints: (4)–(7);
	flow balance constraints: (8), (9), (11);

In this formulation, A_i, N and T are constants, $x_i[t]$, $y_i[t]$, $z_{ij}[t]$, $\pi_i[t]$ and $\theta_{ji}[t]$ are integer variables, and $r_{\min}$, $r(f)$ and $r_{ij}(f)$ are continuous variables. Note that the two sets of constraints in (6) and (7) are stated in the form of sufficient conditions rather than in the form of mathematical programming. Therefore, a reformulation of (6) and (7) is needed.

Reformulation For constraint (6), if $x_i[t] = 1$, then we have $\sum_{j\in\mathcal{T}_i} z_{ij}[t] + \sum_{j\in\mathcal{R}_i}(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]) \le A_i$. On the other hand, if $x_i[t] = 0$, then no DoF is consumed. Constraint (6) can be reformulated by incorporating binary variable $x_i[t]$ into the expression as follows:

$$\sum_{j\in\mathcal{T}_i} z_{ij}[t] + \sum_{j\in\mathcal{R}_i}\left(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]\right) \le A_i x_i[t] + (1 - x_i[t])B_i \quad (i \in \mathcal{N}, 1 \le t \le T), \tag{12}$$

where $B_i = \sum_{j\in\mathcal{R}_i} A_j$ is an upper bound of $\sum_{j\in\mathcal{R}_i}(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t])$.

Similarly, constraint (7) can be reformulated as follows:

$$\sum_{j\in\mathcal{T}_i} z_{ji}[t] + \sum_{j\in\mathcal{R}_i}\left(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{jk}[t]\right) \le A_i y_i[t] + (1 - y_i[t])B_i \quad (i \in \mathcal{N}, 1 \le t \le T). \tag{13}$$

Note that constraints (12) and (13) have nonlinear terms $\sum_{j\in\mathcal{R}_i}(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t])$ and $\sum_{j\in\mathcal{R}_i}(\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{jk}[t])$, respectively. To remove the nonlinear terms in the formulation, we employ the *Reformulated-Linearization Technique* (RLT) [26, Chapter 6]. For constraint (12), we introduce a new variable $\lambda_{ji}[t] = \theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]$. Then constraint (6) can be replaced by the following linear constraint:

$$\sum_{j\in\mathcal{T}_i} z_{ij}[t] + \sum_{j\in\mathcal{R}_i}\lambda_{ji}[t] \le A_i x_i[t] + (1 - x_i[t])B_i, \quad (i \in \mathcal{N}, 1 \le t \le T). \tag{14}$$

Now we need to add constraints for $\lambda_{ji}[t]$. Since $\theta_{ji}[t]$ is a binary variable (therefore $0 \le \theta_{ji}[t] \le 1$) and $0 \le \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t] \le A_j$, then the following constraints must hold:

$$[\theta_{ji}[t] - 0] \cdot \left[\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t] - 0\right] \ge 0, \quad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \le t \le T), \tag{15}$$

$$[\theta_{ji}[t] - 0] \cdot [A_j - \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]] \ge 0, \quad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \le t \le T), \tag{16}$$

$$[1 - \theta_{ji}[t]] \cdot \left[\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t] - 0\right] \ge 0, \quad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \le t \le T), \tag{17}$$

$$[1 - \theta_{ji}[t]] \cdot \left[A_j - \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]\right] \ge 0, \quad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \le t \le T). \tag{18}$$

Substituting $\lambda_{ji}[t]$ for $\theta_{ji}[t]\sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t]$ in the above constraints, we obtain

$$\lambda_{ji}[t] \geq 0, \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (19)$$

$$\lambda_{ji}[t] \leq A_j \cdot \theta_{ji}[t], \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (20)$$

$$\lambda_{ji}[t] \leq \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t], \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (21)$$

$$\lambda_{ji}[t] \geq A_j \cdot \theta_{ji}[t] + \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{kj}[t] - A_j, \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T). \quad (22)$$

Similarly, for constraint (7), we introduce a new variable $u_{ji}[t] = \theta_{ji}[t] \cdot \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{jk}[t]$. Then, constraint (7) can be replaced by the following set of linear constraints:

$$\sum_{j\in\mathcal{T}_i} z_{ji}[t] + \sum_{j\in\mathcal{R}_i} u_{ji}[t] \leq A_i y_i[t] + (1 - y_i[t])B_i, \qquad (i \in \mathcal{N}, 1 \leq t \leq T), \quad (23)$$

$$u_{ji}[t] \geq 0, \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (24)$$

$$u_{ji}[t] \leq A_j \cdot \theta_{ji}[t], \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (25)$$

$$u_{ji}[t] \leq \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{jk}[t], \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T), \quad (26)$$

$$u_{ji}[t] \geq A_j \cdot \theta_{ji}[t] + \sum_{k\in\mathcal{T}_j}^{k\neq i} z_{jk}[t] - A_j, \qquad (i \in \mathcal{N}, j \in \mathcal{R}_i, 1 \leq t \leq T). \quad (27)$$

The problem is now reformulated as follows:

OPT
max $r_{\min}$
s.t $r_{\min} \leq r(f) \quad f \in \mathcal{F}$;
half duplex constraints: (1)–(3);
ordering and DoF constraints: (4), (5), (14), (19)–(27);
flow balance constraints: (8), (9), (11);

In this formulation, A_i, B_i, N and T are constants, $x_i[t]$, $y_i[t]$, $z_{ij}[t]$, $\pi_i[t]$ and $\theta_{ji}[t]$ are integer variables, and $r_{\min}$, $r(f)$, $r_{ij}(f)$, $\lambda_{ji}[t]$ and $\mu_{ji}[t]$ are continuous variables. This optimization problem is in the form of a mixed-integer linear program (MILP), which is NP-hard in general. The theoretical worst-case complexity of solving a general MILP problem is exponential [29]. It cannot be solved by commercial solvers for moderate sized networks. Thus, we need to develop an efficient algorithm to find a competitive solution.

III. Construction of an Initial Feasible Solution

A. Overview

The proposed solution consists of two stages. In the first stage, we obtain an initial feasible solution through a customized algorithm. The algorithm is based on the so-called *sequential fixing* (SF) technique [26, Chapter 6]. Another technique to find an initial feasible solution is to first fix route of each session (e.g., by using *Shortest Path* routing). After sessions' routes are given, the number of integer variables in the original MILP is reduced. However, the reduced problem is still MILP, which is NP-hard in general. On the other hand, SF finds a feasible solution by solving a series of LPs, each of which has a polynomial-time complexity (see Section V). Since our goal in this paper is to develop a polynomial time algorithm to solve the MILP, we employ SF to find an initial feasible solution.

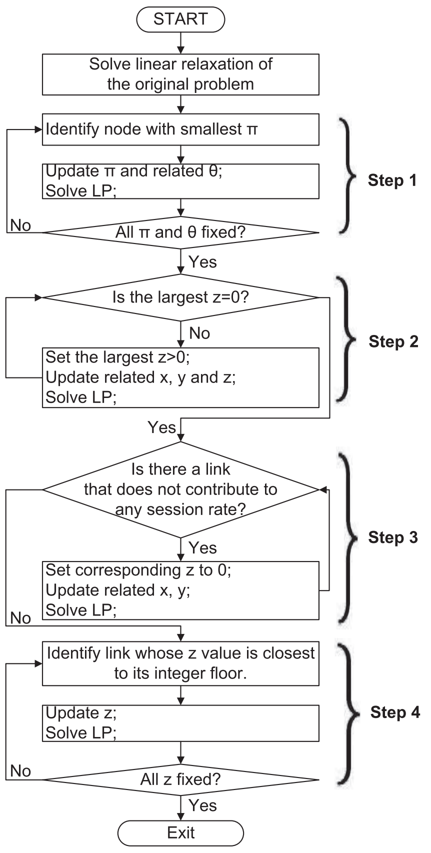

START
Solve linear relaxation of the original problem
Identify node with smallest π
Update π and related θ; Solve LP;
All π and θ fixed?
No
Yes
Step 1
Is the largest z=0?
No
Set the largest z>0; Update related x, y and z; Solve LP;
Step 2
Yes
Is there a link that does not contribute to any session rate?
Yes
Set corresponding z to 0; Update related x, y; Solve LP;
No
Step 3
Identify link whose z value is closest to its integer floor.
Update z; Solve LP;
All z fixed?
No
Yes
Step 4
Exit

Fig. 1. A flow chart for finding a good initial feasible solution.

The idea of SF is as follows. For a mixed integer linear program, if the integer variables are fixed, then the MILP would be reduced to an LP, which can be solved by commercial softwares in polynomial time. Therefore, the key challenge here is how to fix the values for all integer variables. This can be done by solving the linear relaxation of the original problem, where all integer variables are relaxed to continuous variables. Then we can fix the values of integer variables in iterations. Once we fix some integer variables' values in an iteration, we build a new MILP for the remaining integer variables and solve the new relaxed LP. A naive application of SF will fix the values of the relaxed variables solely based on the closeness to integers. However, the performance of such an application may be an issue due to the fact that it does not consider the relationships among the variables. In the first stage of our algorithm, the decision of which variables to fix is carefully designed according to the roles they play in the optimization problem. Figure 1 shows the flow chart of finding a good initial feasible solution.

To obtain an initial feasible solution, we need to fix the integer values for $\mathbf{x}[t]$, $\mathbf{y}[t]$, $\mathbf{z}[t]$, $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$, where $\mathbf{x}[t]$, $\mathbf{y}[t]$ and $\boldsymbol{\pi}[t]$ represent vectors $[x_1[t], x_2[t], \ldots, x_N[t]]$, $[y_1[t], y_2[t], \ldots, y_N[t]]$ and $[\pi_1[t], \pi_2[t], \ldots, \pi_N[t]]$. $\mathbf{z}[t]$ and $\boldsymbol{\theta}[t]$ represent matrices $[z_{ij}[t]]_{N\times N}$ and $[\theta_{ji}[t]]_{N\times N}$. In our problem formulation, the ordering variables $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$ play a key role in the solution since they determine how IC is performed at each node. Therefore, in Step 1, we first fix the values of $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$.

After Step 1, we obtain an ordered node list for each time slot, which can be used to calculate DoF consumption for IC at each node. Therefore, in Step 2, we can determine whether a link can be active or not based on the DoF constraints (6) and (7). If link (i, j) can be active in time slot t, we set $z_{ij}[t] > 0$, otherwise, we set $z_{ij}[t] = 0$. Then, we can fix the values of corresponding scheduling variables $\mathbf{x}[t]$ and $\mathbf{y}[t]$. Note that in this step, we do not fix the exact integer value of $\mathbf{z}[t]$ for active links.

After Step 2, all active links are identified. Note that since the route for each session is not predetermined, there may exist some active links that do not contribute to session throughput. Although these links do not affect the feasibility of the solution, however, some other nodes in the network may have to consume their DoFs to do IC for the transmit and receive nodes of these links. To avoid wasting DoF resources, we release DoF allocation on such "dummy" links in Step 3. Then the route of each session is now identified by the remaining active links.

In Step 4, we fix the integer values of $\mathbf{z}[t]$ according to their closeness to integer floor. After Step 4, all integer variables are fixed and we obtain an initial feasible solution for the MILP.

B. Algorithm Details

Now we give detailed descriptions of each step in finding an initial feasible solution.

Step 1: Fixing π and θ variables. Since the ordering of a node determines its DoF consumption for IC, in Step 1, we fix the integer values of ordering variables $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$. In each iteration, for $1 \le t \le T$, we fix one $\pi_i[t]$ variable and related $\theta_{ij}[t]$ variables. Since the values of $\pi_i[t]$ and $\theta_{ij}[t]$ variables in one time slot are independent of their values in another time slot, they may be fixed in parallel. Then we build a new MILP for the remaining integer variables and solve the new relaxed LP. Since there are N nodes in the network, we need $(N-1)$ iterations.

Specifically, as Fig. 1 shows, at the beginning of the first stage, we solve the linear relaxation of the original problem, where all integer variables in our model are relaxed to continuous variables. That is, before Step 1, we obtain a solution in which $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$ are continuous variables. In the first iteration of Step 1, we identify node i with the smallest $\pi_i[t]$ value and set $\pi_i[t] = 1$, which means that we put node i at position 1 in the ordered node list. Once the position of node i is fixed, we know that all the other nodes are behind node i, and therefore, we set $\theta_{ji}[t] = 0$ and $\theta_{ij}[t] = 1$ for $j \in \mathcal{N}$, $j \ne i$ (according to Eq. (5)). With the integer values of $\pi_i[t]$ and corresponding $\boldsymbol{\theta}[t]$ fixed, we build a new MILP and solve its linear relaxation. In each iteration, we fix the π-value for one node. After $(N-1)$ iterations, all the nodes in the network have their positions fixed in the ordered node list.

Step 2: Fixing x and y variables. In Step 1, we determine an order value for each node so that IC can be performed according to this order. In Step 2, we determine the status of each link based on the DoF consumptions at its transmit and receive nodes. Then we fix the values of corresponding scheduling variables $\mathbf{x}[t]$ and $\mathbf{y}[t]$. Note that in this step, we do not fix the exact number of data streams on each active link ($\mathbf{z}[t]$). The reason behind this is that leaving the exact value of $\mathbf{z}[t]$ to be fixed later can help achieve a better solution when the algorithm terminates.

Specifically, in the first iteration, for $1 \le t \le T$, we identify link (i, j) that has the largest $z_{ij}[t]$ value and fix its status as active (by adding constraint $z_{ij}[t] > 0$). Once we set link (i, j) as active in time slot t, we can fix $x_i[t] = 1$ and $y_j[t] = 1$ (according to constraints (2) and (3)). To accelerate the algorithm, we identify all links that cannot be active simultaneously with link (i, j) and fix as many $\mathbf{x}[t]$, $\mathbf{y}[t]$ and $\mathbf{z}[t]$ values as possible by the following rules.

- **Links associated with nodes i and j.** Since we already identified link (i, j) as an active link in time slot t, according to constraint (1), we can fix $y_i[t] = 0$ for transmit node i and $x_j[t] = 0$ for receive node j. Then we can fix the status of all incoming links to node i as inactive (according to constraint (3)) and the status of all outgoing links from node j as inactive (according to constraint (2)). That is, we set $z_{ki}[t] = 0$ for each node $k \in \mathcal{T}_i$ and $z_{jk}[t] = 0$ for each node $k \in \mathcal{T}_j$.
- **Links associated with those nodes that should perform IC for node i or node j.** Since link (i, j) is identified as active in time slot t, there is at least one data stream on it. For transmit node k, if its interference range covers receive node j and node k is after node j in the ordered node list (i.e., $\pi_k[t] > \pi_j[t]$), then node k has to consume DoFs to cancel its interference to node j. According to constraint (2) and (6), node k can be an active transmitter only if it has enough DoFs for IC and also at least one remaining DoF for SM. To check if node k meets the requirement, we first assume that there is only one data stream on link (i, j), then we calculate the minimum number of DoFs node k uses to perform IC by constraint (6). If the minimum DoF consumption for IC equals to or is larger than the number of DoFs at node k, then it cannot be an active transmitter (set $x_k[t] = 0$). Therefore, all outgoing links from node k cannot be active (set corresponding $\mathbf{z}[t]$ to zero) based on constraint (2). Similarly, for transmit node i, some nodes in the network cannot be active receivers in time slot t based on constraint (7). Therefore, all the incoming links to these nodes cannot be active in time slot t based on constraint (3). Thus, we can set the corresponding $\mathbf{y}[t]$ and $\mathbf{z}[t]$ to zero.
- **Links associated with those nodes that require node i or node j to perform IC.** Since link (i, j) is active in time slot t, transmit node i and receive node j both have to consume DoFs for IC. For receive node j, in addition to SM, it has to use its DoFs to cancel the interference from all unintended neighboring transmit nodes

that are before itself in the ordered node list. We calculate the DoF consumption at node j. If it is equal to its total number of DoFs, then node j does not have any spare DoFs to perform IC for additional transmit nodes. As a result, an inactive node h, where $h \in \mathcal{R}_j$ and $\pi_h[t] < \pi_j[t]$, cannot be activated as a transmitter (set $x_h[t] = 0$). Moreover, all outgoing links from node h cannot be active (set corresponding $\mathbf{z}[t]$ to zero) based on constraint (2). Similarly, for transmit node i, some of the nodes in the network cannot be receivers. Moreover, based on constraint (3), all the incoming links to these nodes cannot be active. Thus, we set the corresponding $\mathbf{y}[t]$ and $\mathbf{z}[t]$ to zero.

Based on the above discussions, we can identify all links that cannot be active simultaneously with link (i, j). After fixing the integer values of $\mathbf{x}[t]$ and $\mathbf{y}[t]$, we build a new MILP for the remaining integer variables and solve a new relaxed LP. If the largest $z_{ij}[t]$ value among the to-be-determined links is 0, it means that all possible active links have been fixed. Then we fix the remaining links as inactive links (i.e., set corresponding $\mathbf{z}[t]$ to zero). For the transmit and receive nodes of these links, if their $\mathbf{x}[t]$ (or $\mathbf{y}[t]$) values are not yet fixed, we set them to zero.

Step 3: Release DoF allocation on links that do not carry any data flow. In Step 2, we reserve DoF resources for all links that are allowed to be active. During each iteration, for link (i, j) with the largest $z_{ij}[t]$ value, we reserve DoF resources for it by adding constraint $z_{ij}[t] > 0$. However, since the route of each session is not predetermined, it may change throughout iterations in Step 2. Therefore, after Step 2, link (i, j) may not be used in any session's route. Although there are DoF resources allocated for SM on link (i, j), the link may not carry any data flow. Such DoF allocation is feasible, but it wastes DoF resources in the network since some other nodes in the network have to use their precious DoFs to perform IC due to these active links. These wasted DoF resources could have been used more productively for SM to achieve a higher session throughput. So it is necessary to release DoF allocation on links that do not carry any data flow.

Specifically, if we find link (i, j) with $r_{ij}(f) = 0$ for every session f traversing link (i, j), then we set $z_{ij}[t] = 0$ for every time slot t. If all the outgoing links from node i are inactive in time slot t, we set the status of node i to inactive (i.e., $x_i[t] = 0$) based on constraint (2). Similarly, if all incoming links to node j are inactive in time slot t, we set the status of node j as inactive (i.e., $y_j[t] = 0$) based on constraint (3). Then we update $\mathbf{x}[t]$, $\mathbf{y}[t]$ and $\mathbf{z}[t]$ and solve the LP. The process repeats until all such links are cleared.

Step 4: Fixing z variables. In steps 2 and 3, we set $\mathbf{z}[t]$ to 0 for inactive links. However, for the active links, the exact values of $\mathbf{z}[t]$ remain unknown. In this step, we fix the integer values of $\mathbf{z}[t]$ for active links. To do this, we run the following process in each iteration. For each time slot, among all the active links with unfixed $\mathbf{z}[t]$, we identify link (i, j) whose $z_{ij}[t]$ value is closest to its integer floor $\lfloor z_{ij}[t] \rfloor$ and set $z_{ij}[t] = \lfloor z_{ij}[t] \rfloor$. Then we update $z_{ij}[t]$ and solve the LP. The iteration continues if there still exists fractional $\mathbf{z}[t]$ value. Eventually, all integer variables are fixed and we obtain an initial feasible solution.

Lemma 1: A solution obtained by the first stage of our algorithm is feasible.

Proof: In the first stage of our algorithm, we wish to obtain an initial feasible solution for the MILP. We first reduce the original MILP to LP by fixing the values of integer variables ($\boldsymbol{\pi}[t]$, $\boldsymbol{\theta}[t]$, $\mathbf{x}[t]$, $\mathbf{y}[t]$ and $\mathbf{z}[t]$) through four steps. Then we obtain the values of other variables by solving the LP to its optimality. The feasibility of the solution can be verified by checking whether all the integer variables are indeed integers, and whether all the constraints are satisfied after the integer variables are fixed. Specifically, in Step 1, we fix the integer values of ordering variables $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$ through iterations. In each iteration, we fix one $\pi_i[t]$ by constraint (4) and fix its related $\boldsymbol{\theta}[t]$ by constraint (5). After these variables are fixed, we obtain values of other variables by solving a relaxed LP to its optimality. Therefore, after Step 1, $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$ are fixed and all the constraints remain satisfied.

In Step 2, we identify the status of each link and fix all the scheduling variables $\mathbf{x}[t]$ and $\mathbf{y}[t]$. In each iteration, we identify one link (i, j) to be active (set $z_{ij}[t] > 0$) only if constraints (6) and (7) are satisfied. Then we fix corresponding scheduling variables $\mathbf{x}[t]$ and $\mathbf{y}[t]$ according to constraints (1)–(3). The values of other variables are obtained by solving a relaxed LP to its optimality. After Step 2, $\mathbf{x}[t]$ and $\mathbf{y}[t]$ are fixed and all the constraints remain satisfied.

In Step 3, we set $\mathbf{z}[t] = 0$ for "dummy" links and update corresponding scheduling variables $\mathbf{x}[t]$ and $\mathbf{y}[t]$ so that constraints (2) and (3) are satisfied. Since constraints (6) and (7) set an upper bound for DoF consumption at each active node, they remain satisfied. Since these links do not contribute to any session's throughput, the flow balance constraints (8), (9) and (11) are satisfied.

In Step 4, we fix the values of $\mathbf{z}[t]$ for active links. We set $\mathbf{z}[t]$ to their integer floors so that the sum of DoF consumption for SM and IC at each node will not exceed its available DoFs. That is, constraints (6) and (7) are satisfied at each node. After Step 4, $\mathbf{z}[t]$ are fixed and all the constraints remain satisfied.

As a result, all integer variables in our original MILP are fixed. The MILP is reduced to a LP, and values of other variables are obtained by solving this LP to its optimality. Therefore, all constraints in the original MILP are satisfied under the obtained solution. ■

C. An Example

As discussed in Section III-A, compared to other approaches such as setting sessions' routes first (e.g., with shortest path routing), SF is able to find a feasible solution in polynomial time. In addition, since SF technique preserves the flexibility of multipath routing for each session, the throughput for each session in the solution tends to be larger. We use an example to illustrate this point. Consider an example network with 20 nodes and 2 sessions in Figure 2, where $s(1)$, $d(1)$, $s(2)$ and $d(2)$ represent the source and destination nodes for sessions 1 and 2, respectively. We assume that each node in the network is equipped with four antennas and a node's transmission and interference ranges are 30 and 50, respectively. For scalability, we normalize all units for distance and time with appropriate

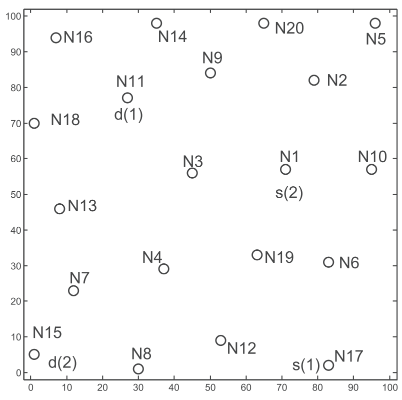


Fig. 2. An example of a 20-node network.

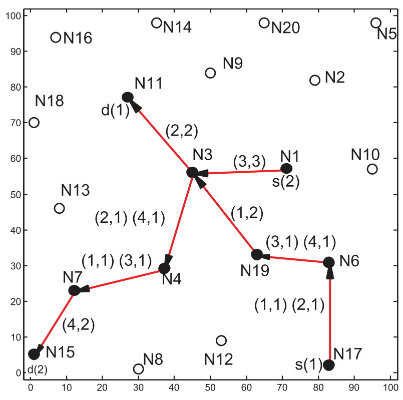


Fig. 4. An initial feasible solution for flow routing and scheduling by first setting the route of each session to be shortest path.

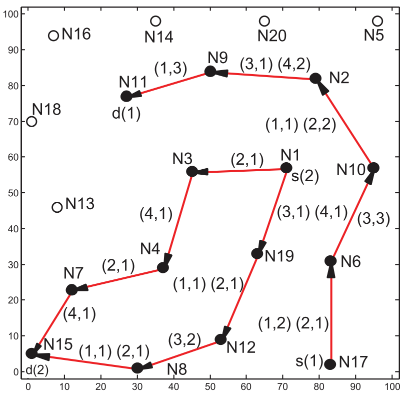


Fig. 3. An initial feasible solution for flow routing and scheduling through SF.

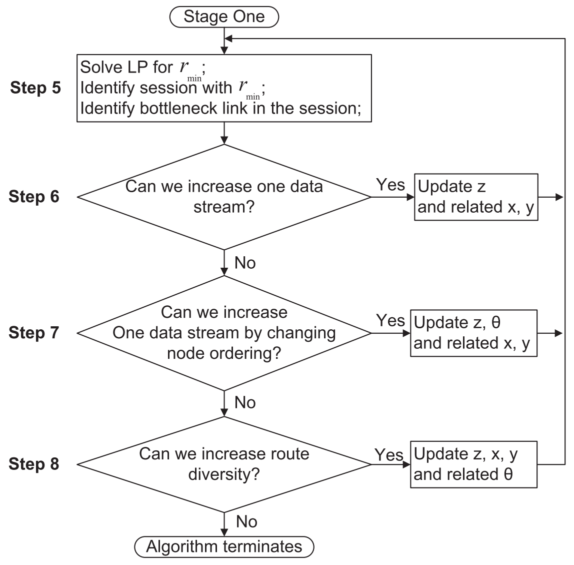

Stage One
Step 5
Solve LP for r_{min};
Identify session with r_{min};
Identify bottleneck link in the session;
Step 6
Can we increase one data stream?
Yes
Update z and related x, y
No
Step 7
Can we increase One data stream by changing node ordering?
Yes
Update z, θ and related x, y
No
Step 8
Can we increase route diversity?
Yes
Update z, x, y and related θ
No
Algorithm terminates

Fig. 5. A flow chart for the second stage of our proposed algorithm.

dimensions. There are four time slots in a frame. The objective value (r_{min}) found by SF is 0.75, while the objective value found by first setting each session's route with shortest path is 0.5. Figure 3 and Figure 4 show the routing topologies and scheduling for each session by SF and shortest path, respectively. The tuple next to each link represents the time slot index of a frame in which the number of data streams are transmitted. For example, in Figure 3, $(1, 3)$ next to link (N_9, N_{11}) denotes that in time slot 1, there are 3 data streams on this link. In the case when there are multiple such tuples next to a link, it means that this link is active over multiple time slots in a frame. Comparing Figure 3 and Figure 4, SF has the following two advantages. First, SF allows flow splitting. In Figure 3, the route of session 2 splits from the source node N_1 into two branches and then merge at the destination node N_{15}. On the other hand, in Figure 4, the route of session 2 obtained by shortest path is only one of the two branches in Figure 3. SF uses DoF resources at more nodes, and therefore has a larger session throughput. Second, SF can make more efficient use of the DoF resources in the network. In Figure 4, node N_3 is the intersection of two sessions. Since both of these two sessions need to use the DoF resources at node N_3, it becomes the bottleneck node. However, in Figure 3, SF finds another route for session 1 to avoid intersection with session 2. Therefore, the initial feasible solution obtained by SF can utilize DoF resources among the nodes in the network more efficiently than the technique with shortest path routing.

IV. Improving the Initial Feasible Solution

A. *Overview*

After the first stage, we have an initial feasible solution. In the second stage of our algorithm, we identify bottleneck link and improve throughput in each iteration until no further increment is possible. In essence, the second stage of the proposed solution is an iterative greedy algorithm. Figure 5 shows its flow chart. In Step 5, we find the current $r_{\min}$ and identify the session with this bottleneck throughput. For the session with this minimum throughput,[2] we identify the link associated with this bottleneck throughput among all links traversed by this session.[3]

In Step 6, we try to enlarge the "pipe" of the bottleneck link by increasing the number of data streams (DoFs) on this link. This is done by examining: (i) at the transmit and receive nodes of this link, if there is any remaining DoFs (in any time slot);

[2]In case of a tie, we choose the session with the smallest session number (index). This will eliminate any randomness in choosing a session.

[3]In case of a tie, we may randomly choose one from the bottleneck links.

and (ii) for nodes that are behind these two nodes in the ordered node list, if there are enough remaining DoFs for IC should a new data stream is added to the bottleneck link.

If Step 6 is not successful, then in Step 7, we try to see if any change in node ordering can change the DoF consumption at a node. This step is motivated by the fact that the relative position of a node in the ordered node list affects its IC behavior and DoF consumption.

If Step 7 is not successful, we try to modify the current routes for the bottleneck session. The design space here is large and will have different complexity and performance trade-off. We adopt a simple approach in this step. Specifically, for the underlying bottleneck link (for which steps 6 and 7 fail), we check if we can find a nearby relay node so that we can add a new route in parallel to this link.

The algorithm continues as long as any of steps 6 to 8 is successful, in which case we update $\mathbf{z}[t]$ and corresponding $\mathbf{x}[t]$, $\mathbf{y}[t]$, $\boldsymbol{\pi}[t]$ and $\boldsymbol{\theta}[t]$ variables as needed. The algorithm terminates when none of steps 6 to 8 is successful.

B. Algorithm Details

Now we give detailed descriptions of each step in the second part of our algorithm.

Step 5: Finding bottleneck link. In this step, we are given a feasible solution which gives values for $\mathbf{x}[t]$, $\mathbf{y}[t]$, $\mathbf{z}[t]$ and $\boldsymbol{\theta}[t]$. With these fixed values for $\mathbf{x}[t]$, $\mathbf{y}[t]$, $\mathbf{z}[t]$, problem OPT degenerates into a LP. By solving this LP, we can find the current $r_{\min}$. Subsequently, we identify a session with rate $r_{\min}$, which we denote as f. A tie is broken by choosing the session with the smallest session number (index). Such deterministic tie-breaking mechanism ensures that we keep working on the same session before moving on to the next one. For the chosen session, we find a bottleneck link, i.e., constraint (11) is binding. A tie among multiple bottleneck links may be broken arbitrarily.

Step 6: Adding a data stream. Denote (i, j) as the bottleneck link that we have identified for session f in Step 5, where i is the transmit node and j is the receive node. In Step 6, we try to increase one data stream on this link in some time slot over a frame. This increment is a successful one if the following conditions are satisfied:

- **(C-1):** Both transmit and receive nodes i and j have at least one remaining DoF.
- **(C-2):** For receive nodes after i (and within i's interference range) in the ordered node list, there is at least one DoF available for IC. Likewise, for transmit nodes after j (and have j in their interference range) in the ordered list, there is at least one DoF available for IC.

In the case when the above increment is successful, then $z_{ij}[t]$ is incremented by 1, and $x_i[t]$ and $y_j[t]$ are updated to 1. Since we have T time slots in a frame, we will check each time slot in Step 6.

Step 7: Adjusting node ordering. Step 6 will fail if condition (C-1) or (C-2) cannot be satisfied in the same time slot. Note that based on our DoF IC model, the ordering of nodes has a profound impact on each node's DoF consumption for IC. Therefore, in Step 7, we will try to adjust the node ordering π in each time slot to see if both conditions (C-1) and (C-2) can be satisfied. We propose a two-phase ordering change, denoted (A-1) and (A-2), to address the requirements in (C-1) and (C-2), respectively.

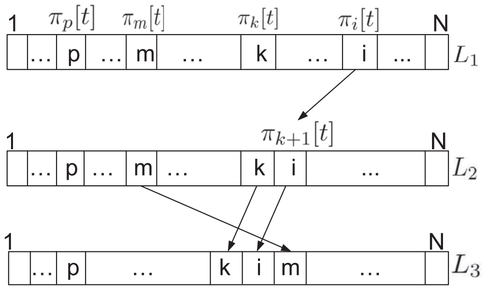


Fig. 6. A schematic illustrating the process of adjusting node ordering.

(A-1): Since condition (C-1) is not satisfied, we have that either transmit node i or receive node j does not have any remaining DoF. We consider the case for transmit node i first. The case for receive node j is similar.

The ordered list L_1 in Figure 6 shows the current ordering of nodes in the network, in which we have shown the position of transmit node i in this order as well as those nodes (i.e., p, m and k in this example) that are receive nodes before node i in L_1 and are within node i's interference range. We do not identify other receive nodes (except p, m and k) before node i in L_1 because they are outside the interference range of i. Among receive nodes p, m and k, k is closest to node i in L_1. Our idea of adjusting node ordering for node i is as follows. Since transmit node i has run out of its DoFs, it is likely that it is using some of its DoFs for IC to nodes p, m, and k. If we could move node i before one of these nodes, then the IC burden on node i will be reduced, allowing some DoFs to be freed up for SM of one more data stream. The outcome of such reordering (successful or not) depends on the DoF consumption on each node (after reordering) and whether the DoF constraints (6) and (7) can be met. There are many ways to move up node i (before p, m or k) in the ordered list. In the following, we present an algorithm that we have designed for this purpose.

To reduce the IC burden on node i, we want to choose a receive node (i.e., p, m and k in this example) and put it after node i. This move will add IC burden on the chosen node (p, m or k) as it will be responsible for IC for more transmit nodes (transmit nodes that are among the nodes between itself and node i in L_1). To reduce the number of these new transmit nodes for IC, we first move node i to position $\pi_k[t] + 1$. This will cause the set of nodes between node k and node i in L_1 to be shifted to the right by one position, as shown in L_2 in Figure 6. Note that this operation will not change DoF consumption at any node in the network and DoF constraints at all nodes remain satisfied.

Now we need to choose a receive node among p, m and k in L_2 and move it after node i. A receive node is eligible for selection if it has enough DoFs available to cancel interference from all the interfering transmit nodes before itself in the node list after it is moved behind node i. We check nodes p, m, and

k individually for its eligibility. Among the eligible nodes,[4] we choose the one (say m) that has the most remaining DoFs after this move.[5] L_3 in Figure 6 shows the ordered node list after m is moved after node i, where all the nodes between nodes m and i in L_2 have been shifted by one position to the left. In L_3, transmit node i is before receive node m and is no longer responsible for canceling its interference to node m. As a result, node i has at least one DoF available for SM to transmit one more data stream on the bottleneck link (i, j).

For receiver node j, it is not hard to see that a similar approach can be applied to increase its available DoFs. To conserve space, we omit its discussion here.

(A-2): Since condition (C-2) is not satisfied, we know that either (i) for a receive node after transmit node i (and within i's interference range) in the ordered node list, there is no DoF left for IC; or (ii) for a transmit node after receive node j (and has j in its interference range) in the ordered node list, there is no DoF left for IC.

For (i), denote h as such a receive node. Then we will try to change the order for node h so that it will have at least one DoF available. But this is precisely the same reordering problem that we would have done for receive node j in (A-1). Therefore, the same node reordering procedure can be applied to node h. For (ii), again the reordering problem is precisely the same as that for transmit node i in (A-1) and therefore the same node reordering procedure for i can be applied.

Both (A-1) and (A-2) are performed in each time slot until a data stream can be added or they fail in all time slots.

Step 8: Improving route diversity. In steps 6 and 7, we try to increase one data stream on the bottleneck link (i, j). When both steps fail, it suggests that it may be futile to add one more data stream on this bottleneck link (i, j). A plausible approach is to open up some other routes (i.e., multiple parallel paths) between nodes i and j so that the extra data stream can be diverted over the new path. There has been extensive research on finding multiple paths between two nodes [30], [31]. For the purpose of this paper, we show one simple algorithm that only employs one extra relay node to create a second path between nodes i and j.

A node k can be considered as a relay node only if k can serve as node i's receive node in one time slot and node j's transmit node in a different time slot. For node k, we need to check whether both links (i, k) and (k, j) can support one more data stream. For either link (i, k) or link (k, j), we are addressing the same problem for link (i, j) in steps 6 and 7. Therefore, procedures in steps 6 and 7 can be applied. If both links (i, k) and (k, j) can support one more data stream, then we update $\mathbf{z}[t]$, $\mathbf{x}[t]$, $\mathbf{y}[t]$ and $\boldsymbol{\theta}[t]$, and return to Step 5. Otherwise, the algorithm terminates.

Lemma 2: A solution following the successful outcome of Step 6, 7, or 8 is feasible.

Proof: The feasibility of the solution following the successful exit of Step 6, 7, or 8 can be verified by checking whether the DoF constraints (6) and (7) are satisfied at each node. Specifically, during Step 6, one more data stream can be added to the bottleneck link only if (C-1) and (C-2) are satisfied. If (C-1) and (C-2) are satisfied, then the DoF constraints (6) and (7) must remain satisfied at each node after the extra data stream is added to the bottleneck link. Therefore, if Step 6 is successful, then the DoF constraints (6) and (7) must be satisfied at each node.

If Step 6 fails, it indicates that either (C-1) or (C-2) cannot be satisfied under current node ordering. Then in Step 7, we try to alter the node ordering by using (A-1) and (A-2). (A-1) and (A-2) address the requirements in (C-1) and (C-2), respectively. If operations in Step 7 are successful, (C-1) and (C-2) are satisfied and therefore the DoF constraints (6) and (7) must remain satisfied at each node.

If Step 7 fails, it indicates that either (C-1) or (C-2) cannot be satisfied. In Step 8, we try to employ a relay node to create a second path between the transmit and receive nodes of the bottleneck link. A node can be chosen as a relay node only if constraint (1) is satisfied. Then the same algorithms in Step 6 and Step 7 are applied to these two new links. Thus, Step 8 is successful only if (C-1) and (C-2) are satisfied and therefore the DoF constraints (6) and (7) must remain satisfied at each node. ■

[4]When there is no eligible node, we move on to the next time slot.

[5]In case of a tie, we choose the node with the smallest node index.

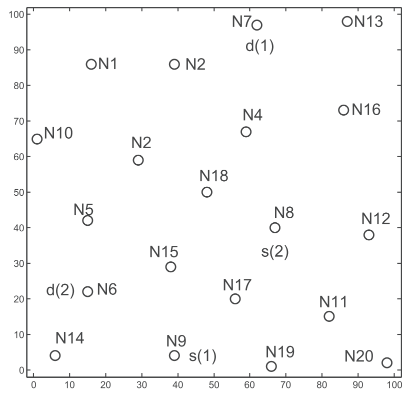


Fig. 7. An example of a 20-node network.

C. An Example

In our DoF IC model, the ordering of nodes has a profound impact on each node's DoF consumption for IC. In Step 7, we exploit this unique property of our model to improve a solution. Now we use an example to give the details of this process. Consider an example network with 20 nodes and 2 sessions shown in Figure 7, where $s(1)$, $d(1)$, $s(2)$ and $d(2)$ represent the source and destination nodes for sessions 1 and 2, respectively. We assume that each node in the network is equipped with four antennas and a node's transmission and interference ranges are 30 and 50, respectively. There are four time slots in a frame. The objective value (r_{min}) obtained after the first stage is 0.25, and is increased to 0.5 after the second stage. Figures 8 and 9 show the routing topologies and scheduling for each session at the end of the first and second stages, respectively. Note that the flow routing is not changed in the second stage, but the scheduling behavior is changed significantly.

After the first stage, in Figure 8, we can see that the first link of session 2 (link (N_8, N_{18})) is active in time slot 1 with one

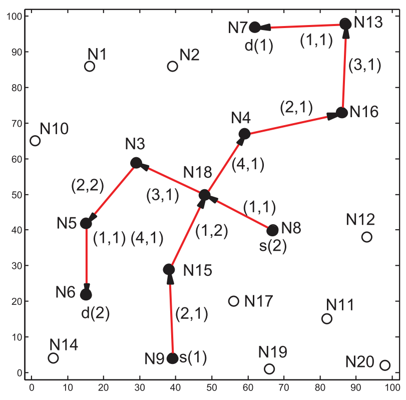


Fig. 8. Flow routing and scheduling solution for the example network after the first stage of our algorithm.

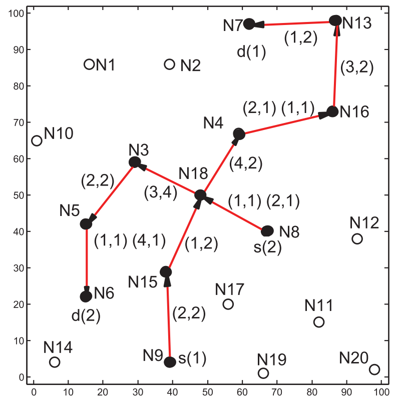


Fig. 9. Flow routing and scheduling solution for the example network after the second stage of our algorithm.

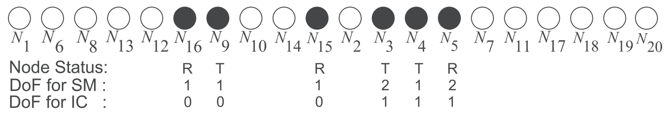

N_1 N_6 N_8 N_{13} N_{12} N_{16} N_9 N_{10} N_{14} N_{15} N_2 N_3 N_4 N_5 N_7 N_{11} N_{17} N_{18} N_{19} N_{20}
Node Status: R T R T T R
DoF for SM : 1 1 1 2 1 2
DoF for IC : 0 0 0 1 1 1

Fig. 10. DoF allocation at each active node in time slot 2 at the end of the first stage of our algorithm.

data stream. Since there are four time slots and the minimum session throughput is 0.25, link (N_8, N_{18}) is a bottleneck link (constraint (11) is binding). After the second stage, as shown in Figure 9, link (N_8, N_{18}) can transmit one more data stream in time slot 2. Therefore, its link throughput is increased to 0.5 and is no longer a bottleneck link. This increment is made possible by adjusting the node ordering list in time slot 2. We now show the details of this adjustment. Figure 10 shows the set of active nodes and their DoF allocation in time slot 2 after the first stage of our algoirthm (SF). Since link (N_8, N_{18}) is not active in time slot 2, we first check if we can activate it without changing the node ordering. The current node ordering list in time slot 2 is $\{N_1, N_6, N_8, N_{13}, N_{12}, N_{16}, N_9, N_{10}, N_{14}, N_{15}, N_2, N_3, N_4, N_5, N_7, N_{11}, N_{17}, N_{18}, N_{19}, N_{20}\}$ (see Fig. 11). For transmit node N_8, if it is active, it does not need to consume any DoF for IC since there is no active receive node before it. Therefore, it has 4 DoFs for SM. For receive node N_{18}, if it is active, it has to consume four DoFs to cancel interference from nodes N_9, N_3 and N_4. Then it does not have remaining DoFs to receive

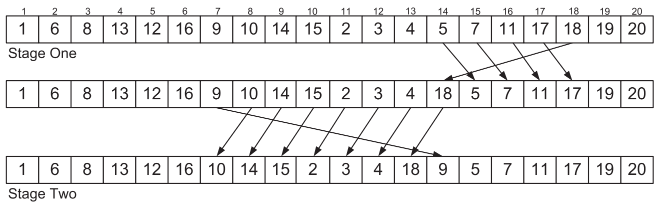


Fig. 11. A schematic illustrating the process of adjusting the ordering for node 18.

any data streams from node N_8. Therefore, to activate link (N_8, N_{18}), we move node N_{18} before one of the three nodes (N_9, N_3 or N_4) so that node N_{18} no longer needs to use its DoFs to cancel interference from that node. Figure 11 shows the details of this move. First, We move node N_{18} from position 18 to position 14 (after transmit node N_4). Note that this operation will not change DoF consumption at any node. Then we choose a node from N_9, N_3 and N_4 and move it after node N_{18}. As described in Section IV-B, the selected node should have the most remaining DoFs after this move. According to Figure 10, in time slot 2, node N_9 consumes one DoF for SM, node N_3 consumes three DoFs (two for SM and one for IC) and node N_4 consumes two DoFs (one for SM and one for IC). Now we check how many additional DoFs these nodes need for IC if they are moved to position 14. For nodes N_3 or N_4, if it is moved to position 14, the number of receive nodes ahead them does not change (N_{16} and N_{15}). For node N_9, if it is moved to position 14, there is one more receive node ahead it (node N_{15}). However, node N_{15} is the intended receiver of node N_9, as shown in Figure 8, then there is no interference to be canceled. Therefore, if being moved to position 14, none of the three nodes needs to consume additional DoFs for IC and node N_9 has the most remaining DoFs (3). As Figure 11 shows, we move node N_9 from position 7 to position 14, which is the current position for node N_{18}. This move will shift nodes between positions 8 to 14 to the left by one position. After this move, node N_{18} is before N_9 in the node ordering list. If node N_{18} is active, it no longer needs to consume one DoF to cancel interference from node N_9. Instead, it can use this available DoF to receive one data stream from node N_8. Meanwhile, node N_9 can use one DoF to cancel its interference to node N_{18}. Now, link (N_8, N_{18}) can be active in time slot 2 with one data stream.

V. Complexity Analysis

We now show that the proposed algorithm has a polynomial time complexity. For each stage of the algorithm, we analyze the number of required iterations and the complexity of each iteration.

In the first stage, for each step, the complexity of each iteration involves solving an LP, searching for the largest/smallest values, and fixing the integer values for selected variables. The complexity of solving an LP is $O(V^3)$ [32], where V is the number of variables. It is not hard to see that the complexity of searching for the largest values and fixing integer variables are much lower than solving an LP. Therefore, the complexity of an iteration is $O(V^3)$.

We now analyze the total number of iterations in the first stage. Note that in each iteration, we fix some variables for all time slots. Since there are N nodes in the network, we need $(N-1)$ iterations for Step 1 as we discussed. Since there are $O(N^2)$ links, we need $O(N^2)$ iterations to determine status of each link (Step 2), identify wasted links (Step 3), and fix the z values for active links (Step 4). Therefore, we have no more than $O(N+3N^2)=O(N^2)$ iterations. As a result, the complexity of the first stage of our algorithm is $O(N^2V^3)$.

In the second stage, the complexity of each step involves solving an LP and identifying a bottleneck link (Step 5), increasing data streams on this link (Step 6 and 7) and adding a parallel path for this link (Step 8). It is not hard to see that solving an LP has the highest complexity among the four steps in an iteration. Therefore, the complexity of an iteration is $O(V^3)$.

We now analyze the total number of iterations in the second stage of our algorithm. Since the links in the network is upper bounded by $O(N^2)$, and for each link, we can increase its data streams by at most A_i times in each time slot. Therefore, the total number of iterations is $O(N^2AT)$, where $A=\max_{i\in\mathcal{N}}\{A_i\}$. As a result, the second stage of our algorithm has an overall complexity of $O(N^2AT\cdot V^3)$.

In our algorithm, three sets of variables, $\theta_{ji}[t]$, $z_{ij}[t]$ and $r_{ij}(f)$, dominate the total amount of variables. Both $\theta_{ji}[t]$ and $z_{ij}[t]$ have $O(N^2\cdot T)$ variables while $r_{ij}(f)$ has $O(N^2\cdot|\mathcal{F}|)$ variables. Therefore, $V=O(N^2\cdot\max\{T,|\mathcal{F}|\})$. In summary, our algorithm has a polynomial time complexity of $O(N^2V^3)+O(N^2AT\cdot V^3)=O(N^2AT\cdot V^3)$, where $V=O(N^2\cdot\max\{T,|\mathcal{F}|\})$.

VI. Simulation Results

In this section, we present simulation results to demonstrate the performance and complexity of the proposed algorithm. We also use a case study to validate the feasibility of a final solution.

A. Simulation Setting

We consider a multi-hop ad hoc network, with nodes being randomly deployed in a 100×100 area. For scalability, we normalize all units for distance and time with appropriate dimensions. We assume that each node in the network is equipped with four antennas and a node's transmission and interference ranges are 30 and 50, respectively. There are four time slots in a frame.

B. Performance and Complexity

To demonstrate the complexity and performance of our proposed algorithm, we use a commercial optimization solver, CPLEX [33], as a benchmark. The computer we use to run the simulation results has 64GB of RAM and a E5-2687w CPU.

First, we compare the complexity (in terms of running time) of the proposed algorithm and CPLEX. We increase the number of nodes in this study. For each network setting, we randomly generate 50 network instances and obtain the average running time required by CPLEX and our proposed algorithm, respectively. Figure 12 shows the trend of average running time required by CPLEX and the proposed algorithm as the number of nodes in the network increases from 4 to 50. Note that the y-axis in Figure 12 is in log-scale, indicating *exponential* running time of CPLEX. On the other hand, the running time of the proposed algorithm is orders of magnitude smaller than the time needed by CPLEX. In Figure 12, in the case of 25 nodes, the average running time required by CPLEX exceeds 10^5 seconds, while it only requires 10^2 seconds by our algorithm. As shown, solving MILP by CPLEX has the disadvantage of requiring an exponentially growing run-time (due to the NP-hard nature of the problem), on the other hand, our proposed algorithm only requires to solve a series of LPs, each of which has polynomial time complexity.

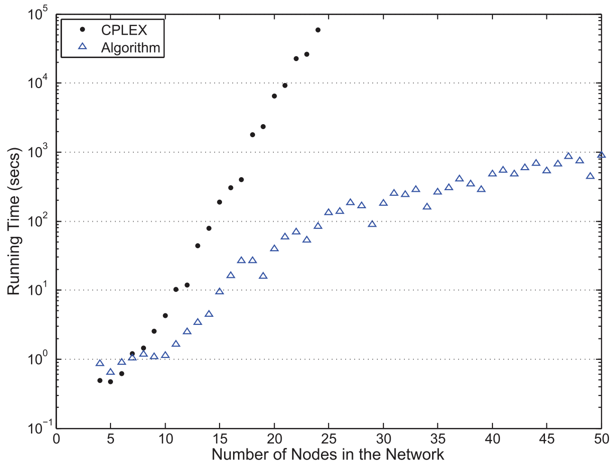


Fig. 12. Running time required by the proposed algorithm and CPLEX.

For performance comparison, we want to demonstrate that the results obtained by the proposed scheme are competitive when compared with the optimal results from CPLEX. Such benchmark comparison is only meaningful when the size of the problem can be handled by CPLEX. Since the average running time required by CPLEX exceeds 24 hours when there are 25 nodes in the network as shown in Fig. 12, we choose network size to be 20 nodes for performance comparison. We present comparison study for 50 random network instances, each with 2 sessions. For each network instance, the node positions are randomly generated and the source and destination nodes for each session are randomly selected. Table II shows the following two set of results for 50 network instances: (i) the ratio between the objective values obtained after the first stage of our algorithm and those from CPLEX, (ii) the ratio between the objective values obtained after the second stage of our algorithm and those from CPLEX. As shown in Table II, among 50 network instances, the objective values in 19 instances are improved by employing the second stage of our algorithm. With only the first stage algorithm, the average ratio is 75.3%, with a standard deviation of 0.19. When the second stage algorithm is employed, the average ratio is improved to 85.6%, with a standard deviation of 0.12. That is, the overall performance of our proposed algorithm is within 85.6% of the optimal solution that can be computed by CPLEX within a reasonable amount of time.

TABLE II
RATIO BETWEEN OBJECTIVE VALUES OBTAINED BY OUR ALGORITHM AND THOSE FROM CPLEX FOR 50 NETWORK INSTANCES

Network Instance	Objective Value Ratio		Network Instance	Objective Value Ratio	
	$\frac{\text{Stage1}}{\text{CPLEX}}$	$\frac{\text{Stage2}}{\text{CPLEX}}$		$\frac{\text{Stage1}}{\text{CPLEX}}$	$\frac{\text{Stage2}}{\text{CPLEX}}$
1	1	1	26	1	1
2	0.5	0.75	27	0.6	0.8
3	0.6	0.8	28	0.5	0.75
4	1	1	29	1	1
5	0.8	0.8	30	0.5	0.75
6	1	1	31	0.5	0.75
7	0.6	0.8	32	0.8	0.8
8	1	1	33	0.75	0.75
9	0.33	0.67	34	0.67	1
10	1	1	35	0.75	0.75
11	0.33	0.67	36	1	1
12	0.6	0.8	37	0.88	0.88
13	0.5	0.83	38	0.5	0.75
14	1	1	39	0.75	0.75
15	0.67	0.67	40	0.86	0.86
16	0.88	0.88	41	0.88	0.88
17	0.67	0.67	42	0.86	0.86
18	0.5	1	43	0.83	0.83
19	1	1	44	0.6	0.8
20	0.67	0.67	45	1	1
21	0.5	1	46	0.75	0.75
22	1	1	47	0.6	0.8
23	0.83	0.83	48	0.75	1
24	0.67	0.75	49	0.83	0.83
25	0.83	0.83	50	1	1

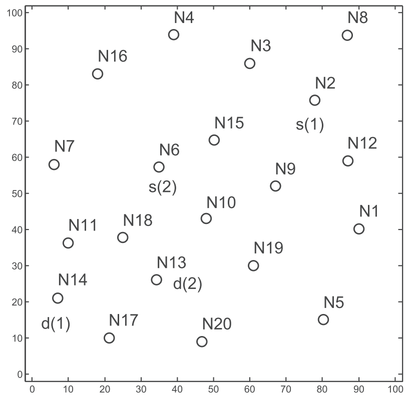


Fig. 13. An instance of a 20-node network.

C. Fesibility of Our Solution

To validate the feasibility of the solution obtained by our proposed algorithm, we randomly pick a network instance (the 33-th) from the above 50 network instances and examine its solution details. Figure 13 shows the locations of the 20 nodes, where $s(1)$, $d(1)$, $s(2)$ and $d(2)$ represent the source and destination nodes for sessions 1 and 2. The objective values ($r_{\min}$) found by our algorithm and CPLEX are both 0.75, indicating the optimality of our solution for this network instance.

Although the two objective values by our algorithm and CPLEX coincide for this network instance, the flow routing and scheduling behavior under the two solutions are different. Figures 14 and 15 show the routing topologies and scheduling for each session by our algorithm and CPLEX, respectively, where the tuple next to each link represents the time slot index of a frame in which the number of data streams are transmitted. For example, in Figure 14, (1, 3) next to link (N_2, N_3) denotes that in time slot 1, there are 3 data streams on this link. In the case when there are multiple such tuples next to a link, it means that this link is active over multiple time slots in a frame.

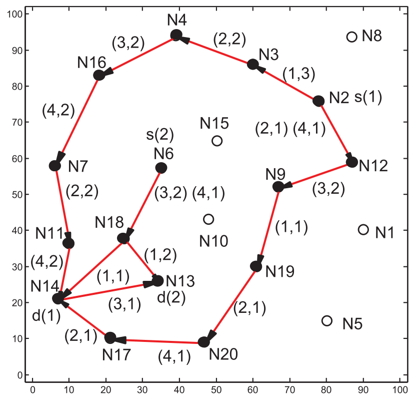


Fig. 14. Solution for flow routing and scheduling by our algorithm.

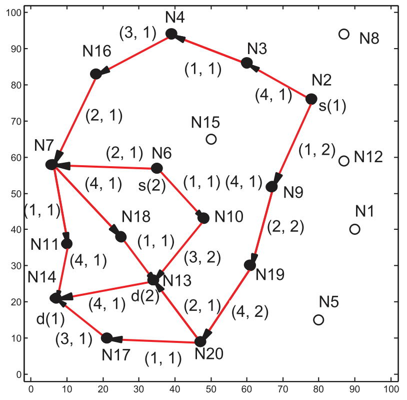


Fig. 15. Solution for flow routing and scheduling by CPLEX.

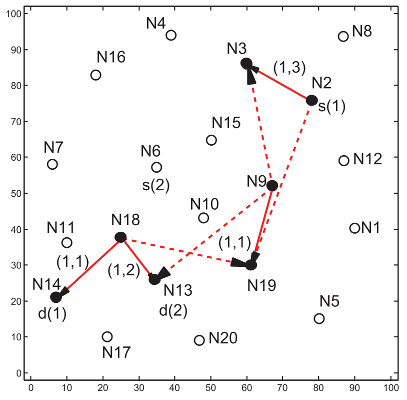
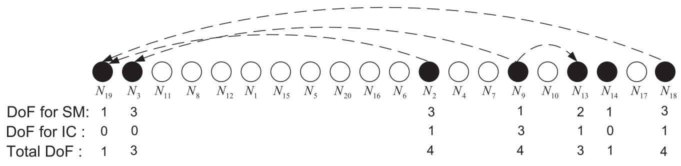


Fig. 16. Scheduling in time slot 1 by our algorithm.

Now let's examine our solution. Table III shows the set of active nodes in each time slot and the DoF allocation for SM and IC at these nodes. As an example, consider the set of active nodes in time slot 1 in Table III, which is shown in Figure 16. The interference relationships among these transmit and receive nodes are shown by the dashed arrows, i.e., node N_2 interferes N_{19}, node N_9 interferes nodes N_3 and N_{13}, and node N_{18} interferes N_{19}. It can be easily verified that by following the relative ordering of these 7 nodes in time slot 1, i.e., $N_{19}, N_3, N_2, N_9, N_{13}, N_{14}, N_{18}$, the DoF constraints in (6) and (7) are satisfied at each of these 7 nodes.

TABLE III
DoF Allocation at Each Active Node in Each Time Slot for a 20-Node Network Instance

Time Slot 1				
Node Ordering	Active Node	Node Status	DoF for SM	DoF for IC
1	N_{19}	receive	1	0
2	N_3	receive	3	0
12	N_2	transmit	3	1
15	N_9	transmit	1	3
17	N_{13}	receive	2	1
18	N_{14}	receive	1	0
20	N_{18}	transmit	3	1
Time Slot 2				
Node Ordering	Active Node	Node Status	DoF for SM	DoF for IC
3	N_3	transmit	2	0
5	N_{20}	receive	1	0
6	N_{19}	transmit	1	0
8	N_2	transmit	1	0
9	N_4	receive	2	1
11	N_7	transmit	2	2
14	N_{11}	receive	2	0
15	N_{12}	receive	1	3
16	N_{14}	receive	1	2
19	N_{17}	transmit	1	3
Time Slot 3				
Node Ordering	Active Node	Node Status	DoF for SM	DoF for IC
4	N_4	transmit	2	0
7	N_{18}	receive	2	0
8	N_{12}	transmit	2	0
9	N_9	receive	2	0
10	N_6	transmit	2	2
15	N_{13}	receive	1	2
16	N_{14}	transmit	1	2
18	N_{16}	receive	2	2
Time Slot 4				
Node Ordering	Active Node	Node Status	DoF for SM	DoF for IC
3	N_{11}	transmit	2	0
4	N_{18}	receive	1	0
6	N_{20}	transmit	1	1
7	N_{17}	receive	1	2
8	N_{16}	transmit	2	1
10	N_7	receive	2	2
12	N_6	transmit	1	3
15	N_2	transmit	1	0
18	N_{12}	receive	1	0
19	N_{14}	receive	2	2



N_{19} N_3 N_{11} N_8 N_{12} N_1 N_{15} N_5 N_{20} N_{16} N_6 N_2 N_4 N_7 N_9 N_{10} N_{13} N_{14} N_{17} N_{18}
DoF for SM: 1 3 3 1 2 1 3
DoF for IC : 0 0 1 3 1 0 1
Total DoF : 1 3 4 4 3 1 4

Fig. 17. A schematic illustrating DoF consumption at each node in time slot 1.

Now we discuss the details of DoF consumption at each active node following the order of their positions in the ordered list in time slot 1. Figure 17 shows the ordered node list, the interference relationship among active nodes (e.g., N_2 interferes N_{19}), and the summary of DoF consumption at each active node.

- N_{19}: the first active node in the ordered node list. It is a receive node. For SM, node N_{19} consumes one DoF to receive one data stream from node N_9. For IC, since it is the first node in the node list, it does not consume any DoF for IC.
- N_3: the second active node in the ordered node list. It is a receive node. For SM, node N_3 consumes three DoFs to receive three data streams from node N_2. For IC, since there is no active transmit node before node N_3 in the node list, it does not consume any DoF for IC.
- N_2: the third active node in the ordered node list. It is a transmit node. For SM, node N_2 consumes three DoFs to transmit three data streams to node N_3. For IC, only receive node N_{19} is within its interference range and before it in the ordered node list. So node N_2 needs to consume one DoF to cancel its interference to node N_{19}.
- N_9: the fourth active node in the ordered node list. It is a transmit node. For SM, node N_9 consumes one DoF to transmit one data stream to node N_{19}. For IC, only transmit node N_3 is within its interference range and before it in the ordered list. So node N_9 needs to consume three DoFs to cancel its interference to node N_3.
- N_{13}: the fifth active node in the ordered node list. It is a receive node. For SM, node N_{13} consumes two DoFs to receive two data streams from node N_{18}. For IC, only transmit node N_9 is within its interference range and before it in the ordered node list. So node N_{13} needs to consume one DoF to cancel its interference to node N_9.
- N_{14}: the sixth active node in the ordered node list. It is a receive node. For SM, node N_{14} consumes one DoF to receive one data stream from node N_{18}. For IC, all the active transmit nodes before node N_{14} in the ordered node list are out of its interference range, it does not consume any DoF for IC.
- N_{18}: the seventh active node in the ordered node list. It is a transmit node. For SM, node N_{18} consumes two DoFs to transmit two data streams to node N_{13} and one DoF to transmit one data stream to node N_{14}. For IC, only receive node N_{19} is within its interference range and before it in the ordered node list. So node N_{18} needs to consume one DoF to cancel its interference to node N_{19}.

Note that the DoF constraints for SM and IC at each node are satisfied in time slot 1. Based on Table III, the readers can easily verify that the DoF constraints at each node are also satisfied in time slot 2, 3, and 4.

VII. Conclusions

DoF based IC model is a powerful tool to study network performance of multi-hop MIMO networks. In this paper, we employed a new DoF IC model in the literature [23] to study a throughput maximization problem in a multi-hop MIMO network. Given the multi-hop network environment, the problem formulation involves joint consideration of multi-path flow routing at network layer and DoF allocation per node at link layer. Since the problem formulation is in the form of a mixed-integer linear program, we proposed to develop an efficient polynomial time algorithm to solve it. Our algorithm design consists of two stages: the first stage is to find a quality initial feasible solution and the second stage is to improve the initial feasible solution. Specifically, in the first stage, we employed the sequential fixing technique to handle integer variables. We showed that this approach has polynomial time complexity and can offer a better initial feasible solution than some other approaches. In the second stage, we improved the initial feasible solution by exploiting the impact of node ordering on

DoF consumption at a node and route diversity in the network. Simulation results showed that our solution offers competitive performance and polynomial time complexity.

Acknowledgments

The authors would like to thank Virginia Tech's Advanced Research Computing for giving them access to the BlueRidge computer cluster.

References

[1] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, Jan. 2007.

[2] D. Gesbert, M. Shafi, D. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[3] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bolcskei, "An overview of MIMO communications—A key to gigabit wireless," *Proc. IEEE*, vol. 92, no. 2, pp. 198–218, Feb. 2004.

[4] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[5] *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks-Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Standard 802.11n-2009, Oct. 2009.

[6] *Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Standard 802.16-2004, Oct. 2004.

[7] *UTRA-UTRAN Long Term Evolution (LTE) and 3GPP System Architecture Evolution (SAE)* [Online]. Available: ftp://ftp.3gpp.org/Inbox/2008_web_files/LTA_Paper.pdf/

[8] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[9] R. S. Blum, "MIMO capacity with interference," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 5, pp. 793–801, Jun. 2003.

[10] G. Caire and S. Shamai, "On the achievable throughput of a multiantenna Gaussian broadcast channel," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1691–1706, Jul. 2003.

[11] S. Cartreux, L. J. Greenstein, and P. F. Dressen, "Simulation results for an interference-limited multiple-input multiple-output cellular system," *IEEE Commun. Lett.*, vol. 4, no. 11, pp. 334–336, Nov. 2000.

[12] B. Chen and M. J. Gans, "MIMO communications in ad hoc networks," *IEEE Trans. Signal Process.*, vol. 54, no. 7, pp. 2773–2783, Jul. 2006.

[13] M. F. Demirkol and M. A. Ingram, "Power-controlled capacity for interfering MIMO links," in *Proc. IEEE Veh. Technol. Conf.*, Atlantic City, NJ, USA, Oct. 2001, pp. 187–191.

[14] M. F. Demirkol and M. A. Ingram, "Stream control in network with interfering MIMO links," in *Proc. IEEE Wireless Commun. Netw. Conf.*, New Orleans, LA, USA, Mar. 2003, pp. 343–348.

[15] S. A. Jafar and M. Fakhereddin, "Degrees of freedom for the MIMO interference channel," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2637–2642, Jul. 2007.

[16] L. Zheng and D. Tse, "Communication on the Grassmann manifold: A geometric approach to the noncoherent multiple-antenna channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 359–383, Feb. 2002.

[17] L. Zheng and D. Tse, "Diversity and multiplexing: A fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

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

[19] D. M. Blough, G. Resta, P. Santi, R. Srinivasan, and L. M. Cortes-Pena, "Optimal one-shot scheduling for MIMO networks," in *Proc. IEEE SECON*, Salt Lake City, UT, USA, Jun. 2011, pp. 377–385.

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

[21] J.-S. Park, A. Nandan, M. Gerla, and H. Lee, "SPACE-MAC: Enabling spatial reuse using MIMO channel-aware MAC," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Seoul, Korea, May 16–20, 2005, pp. 3642–3646.

[22] 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. Mobile Comput.*, vol. 3, no. 4, pp. 350–365, Oct. 2004.

[23] Y. Shi, J. Liu, C. Jiang, C. Gao, and Y. T. Hou, "A DoF-based link layer model for multi-hop MIMO networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, pp. 1395–1408, Jul. 2014.

[24] H. Zeng, Y. Shi, Y. T. Hou, R. Zhu, and W. Lou, "A novel MIMO DoF model for multi-hop networks," *IEEE Netw.*, Vol. 28, no. 5, pp. 81–85, Oct. 2014.

[25] H. Zeng, Y. Shi, Y. T. Hou, and W. Lou, "An efficient DoF scheduling algorithm for multi-hop MIMO networks," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 1564–1554.

[26] Y. T. Hou, Y. Shi, and H. D. Sherali, *Applied Optimization Methods for Wireless Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[27] D. Bhatia and S. Katti, "Full duplex MIMO radios," in *Proc. USENIX NSDI*, Seattle, WA, USA, Apr. 2014, pp. 359–372.

[28] A. C. Cirik, Y. Rong, and Y. Hua, "Achievable rates of full-duplex MIMO radios in fast fading channels with imperfect channel estimation," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3874–3886, Aug. 2014.

[29] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 1999.

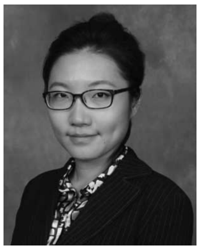
[30] S. Murthy and J. J. Garcia-Luna-Aceves, "Congestion-oriented shortest multi-path routing," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, May 1996, pp. 1038–1036.

[31] P. Papadimitratos, Z. J. Haas, and E. G. Sirer, "Path set selection in mobile ad hoc networks," in *Proc. ACM MobiHoc*, Lausanne, Switzerland, Jun. 2002, pp. 1–11.

[32] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, 4th ed. Hoboken, NJ, USA: Wiley, 2010, ch. 8.

[33] *IBM ILOG CPLEX* [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex/

[34] X. Qin, X. Yuan, Y. Shi, Y. T. Hou, W. Lou, and S. F. Midkiff, "On throughput maximization for a multi-hop MIMO network," in *Proc. IEEE 10th Int. Conf. Mobile Ad-Hoc Sens. Syst. (MASS)*, Hangzhou, China, Oct. 2013, pp. 37–45.

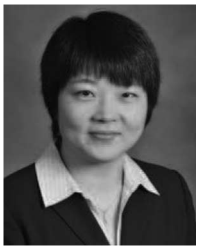


Xiaoqi Qin (S'13) received the B.S. and M.S. degrees in computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2011 and 2013, respectively. Since Fall 2013, she has been pursuing the Ph.D. degree in electrical and computer engineering at Virginia Tech. Her research interests include algorithm design and cross-layer optimization for wireless networks.

Xu Yuan (S'13) received the B.S. degree in computer science from Nankai University, Tianjin, China, in 2009. Since the Fall 2010, he has been pursuing the Ph.D. degree in electrical and computer engineering at Virginia Tech, Blacksburg, VA, USA. His research interests include algorithm design and optimization for cognitive radio networks.

Yi Shi (S'02–M'08–SM'13) received the Ph.D. degree in computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2007. He is currently a Senior Research Scientist with Intelligent Automation Inc., Rockville, MD, USA, and an Adjunct Assistant Professor at Virginia Tech. His research interests include optimization and algorithm design for wireless networks. He was the recipient of the IEEE INFOCOM 2008 Best Paper Award and the only best paper award runner-up of the IEEE INFOCOM 2011.

Y. Thomas Hou (F'14) received the Ph.D. degree from NYU Polytechnic School of Engineering (formerly Polytechnic University). He is a Bradley Distinguished Professor of Electrical and Computer Engineering with Virginia Tech, Blacksburg, VA, USA. He has authored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). His research interests include developing innovative solutions to complex cross-layer optimization problems in wireless and mobile networks. He is the Steering Committee Chair of the IEEE INFOCOM Conference and a member of the IEEE Communications Society Board of Governors.



Wenjing Lou (F'15) received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA. She is a Professor with the Department of Computer Science, Virginia Tech, Blacksburg, VA, USA. Her research interests include wireless networks, with special emphases on wireless security and cross-layer network optimization. Since August 2014, she has been serving as a Program Director with the National Science Foundation. She is the Steering Committee Chair of the IEEE Conference on Communications and Network Security (CNS).

Scott F. Midkiff (S'82–M'85–SM'92) is a Professor and the Vice President for Information Technology and the Chief Information Officer with Virginia Tech, Blacksburg, VA, USA. From 2009 to 2012, he was the Department Head of the Bradley Department of Electrical and Computer Engineering, Virginia Tech. From 2006 to 2009, he served as the Program Director with the National Science Foundation. His research interests include wireless and ad hoc networks, network services for pervasive computing, and cyber-physical systems.