MAXIMIZING THROUGHPUT IN MIMO NETWORKS WITH VARIABLE RATE STREAMS

Ramya Srinivasan[*], *Douglas Blough*[*], *Luis Miguel Cortes-Peña*[*], and *Paolo Santi*[**]

[*] School of ECE, Georgia Institute of Technology, Atlanta, Georgia, USA

[**] IIT-CNR, Pisa, Italy

ABSTRACT

The problem that we consider is that of maximizing throughput in a MIMO network while accounting for variable rate streams on MIMO links. The stream rates on a link depend on the channel conditions of the link, and the manner in which the diversity-multiplexing tradeoff is handled. In this work, we use the dependence of stream rates on the channel to develop methods of link selection and stream allocation that approximately maximize the aggregate throughput. Maximizing throughput is closely tied to the problem of allocating streams based on the stream rates of the selected links. Doing this optimally is very complex even for networks with 10 or fewer links. We develop a stream allocation heuristic that approximately maximizes the throughput over a given set of links. Simulation results for single collision domain networks show that our stream allocation heuristic is within 6% of optimal in networks with up to 10 links (in a typical case where the maximum concurrency allowed is 15 links). The algorithm also cuts the difference between heuristic and optimal results in half, compared to a simple greedy algorithm. Our research has also identified the feasibility checking problem for general MIMO networks as being a computationally hard problem. However, we also identify several practical special cases, e.g. when interference suppression is done only at the receiver side, for which feasibility checking remains a polynomial-time operation.

1. INTRODUCTION

Deployments of all-wireless networks are increasing rapidly due to the emergence of wireless mesh networks and WiMax. These networks are expected to compete with all wired networks and combination wired/wireless networks in terms of performance. Thus, the problem of maximizing throughput in all-wireless networks is extremely important. One of the most promising approaches to throughput improvement in wireless networks is the use of multiple-input multiple-output, or MIMO, technology. Applying MIMO on individual links can provide an immediate throughput boost to these networks. However, optimizing the use of MIMO resources across the entire network has the potential to improve throughput by an even greater amount. For example, our prior work has demonstrated that the total number of concurrent streams that can be supported across a network can be doubled if network-wide optimization techniques are employed, as opposed to link-by-link optimization [13].

In this paper, we consider the problem of maximizing throughput in a MIMO network while accounting for variable-rate streams on individual MIMO links, as well as across different links. This represents the first attempt to characterize and solve a MIMO network optimization problem with a general rate model, since prior work assumed link rates that increase linearly with the number of streams on a link. Within this more accurate framework, we decompose the maximum throughput problem into simpler problems, the solutions of which can be combined into a throughput optimization procedure. The sub-problems we study are referred to as feasibility checking, stream allocation, and one-shot link scheduling, and are formally defined in Section 4.

We first show that the feasibility checking problem for MIMO links is a variation of the Boolean satisfiability problem and is, therefore, very likely to be NP-complete. This is in contrast to feasibility for non-MIMO links, which is polynomial in complexity. A popular greedy approach to scheduling algorithms for wireless networks, e.g. [1, 2, 4, 12], relies on the efficiency of feasibility checking and this result could be a significant impediment to applying this popular scheduling approach to MIMO networks. However, we also show several practical special cases of the MIMO network problem, which do still have polynomial-time feasibility, e.g. when interference suppression is done only on the receiver side but not on the transmitter side. In terms of the overall throughput maximization problem, the algorithm we present is shown through simulation to be within 6% of optimal for networks with up to 10 links. Due to the computational complexity of finding optimal solutions, comparison against optimal was not possible for larger networks. However, over the same range of network sizes, our approach was shown to reduce the difference between heuristic and optimal results by half, compared to a greedy algorithm.

2. RELATED WORK

While a vast body of literature has been devoted to the investigation of MIMO channel capacity in typical configurations such as one-to-one, one-to-all, all-to-one, etc., the issue of characterizing capacity of MIMO-equipped *networks* has been approached only recently. The difficulty in achieving such characterizations is that usage of MIMO links introduces additional optimization knobs into an already very complex optimization problem involving routing, transmit power control, and scheduling in the most general formulation. For these reasons, researchers have typically introduced some simplification in the models and/or problem formulation.

For instance, in [9] the authors provide *upper bounds* to the achievable throughput in a MIMO equipped wireless multi-hop network, under the assumption that perfect CSI is available at both transmitter and receiver, and that only spatial multiplexing and interference suppression is used. In [3], the authors characterize the benefits of cross-layer optimizations in interference-limited MIMO-equipped wireless mesh networks providing an LP-based formulation of the joint routing and link scheduling problem, and a heuristic to solve the resulting throughput optimization problem subject to fairness constraints. Note that both these approaches provide only *bounds* to the achievable network throughput. A few other papers attempted at characterizing the *optimal* throughput achieved in quite restricted network scenarios. For instance, in our previous work [13] we characterized the optimal throughput achievable in a single collision domain network, under the assumption that only spatial multiplexing and interference suppression are considered and that all streams have the same data rate. In [6], the author approaches maximum throughput characterization in a single-hop network through presenting a joint scheduling and MIMO stream allocation problem, and characterizes the optimal solution for the case of two interfering links.

The approaches described above are based on a simplified model of MIMO systems, in which the aggregate rate on a link increases *linearly* with the number of spatially multiplexed streams. Indeed, sub-linear increase of aggregate rate is expected in a practical system, due to the different gains experienced at the involved transmit and received antenna elements. A few papers have accounted for non-linear aggregate rates when attempting to characterize throughput of MIMO-equipped networks, considering so-called *variable rate* stream control in the problem formulation. Variable-rate stream control for CSMA-based MAC layer has been discussed in [14]. Our results, and other optimization-based approaches, target TDMA-based MACs. The approaches of [15] and [5] consider variable rate stream control but only provide upper bounds complemented with feasible heuristic approaches [15], or simply heuristic solutions [5]. Thus, to the best of our knowledge, the problem addressed in this paper of characterizing *optimal* throughput with variable-rate stream control has not yet been investigated.

3. BACKGROUND AND NETWORK MODEL

3.1 MIMO Degrees of Freedom Model

While MIMO links can be used in a variety of ways to achieve different performance and/or reliability goals, the primary aspects that we consider herein are spatial multiplexing and interference suppression. These aspects are often represented by a degrees of freedom (DOF) model. In a DOF model, a node with k elements in its antenna array has up to k DOFs, which it can use for either spatial multiplexing and/or suppressing interference between its link and other concurrently transmitting links.[1] In the absence of interference, a link with k_t DOFs at the transmitter and k_r DOFs at the receiver can support up to $\min(k_t, k_r)$ spatially multiplexed streams. If DOFs are used for interference suppression, then the number of spatially multiplexed streams that can be supported on a link will be reduced.

With a MIMO link, interference suppression can be done by the transmitter or by the receiver or both. To completely eliminate interference requires channel state information (CSI). Receivers can measure channels during transmission of probe sequences in order to collect CSI necessary both for interference suppression and for performance optimization of the channel. CSI can be fed back from receivers to transmitters. It is commonly assumed that channels are symmetric, thereby also allowing transmitters to measure CSI by exchanging roles with receivers. In general, we assume that interference suppression can be done by both transmitters and receivers. However, it is common to have interference suppression done only at receivers.gains on the links. We refer to this as the "receiver-side suppression only" case. Although it is not used in actual networks as far as we know, for completeness we also consider the "transmitter-side suppression only" case.

The number of DOFs needed by a transmitter to suppress interference on a concurrent receiver is equal to the number of streams that are spatially multiplexed on the receiver's link. Similarly, the number of DOFs needed by a receiver to suppress interference from a concurrent transmitter is equal to the number of streams that are spatially multiplexed on the transmitter's link. Assume that a node i has k DOFs, spatially multiplexes s_i streams on its link, and suppresses interference with other nodes $j_1, \ldots, j_n$, carrying s_{j_l} streams respectively, then the following inequality must be satisfied:

$$s_i + \sum_{l=1}^{n} s_{j_l} \leq k$$

3.2 Network Model

In this paper, we focus primarily on single collision domain networks. By this, we mean that all links in the network interfere with each other sufficiently so that no two links can be used concurrently without suppressing interference between them. A single collision domain network can result from a set of users in one area who want to engage in pair-wise communications with each other. A single collision domain can also be considered as a single contention region within a larger network [14].

Our results on the feasibility problem (see Sections 4.1 and 5) apply to arbitrary multi-hop networks, and so we loosen the single collision domain assumption when considering this problem. For this case only, we adopt a simple and symmetric binary interference model wherein two links either interfere completely or not at all. Thus, we define a conflict graph $G_c = (V_c, E_c)$, where V_c is the set of links in the multi-hop network and $e = (l_i, l_j) \in E_c$ if and only if links l_i and l_j interfere with each other.

A basic constraint on concurrency of transmissions is that each node can participate in one transmission at a time, either as transmitter or as receiver.[2] A set of links is said to be *primary-interference-free* if and only if every node in the network appears at most once in the set.

3.3 Rates and Streams

In prior work, we considered how to maximize the total number of streams that can be concurrently transmitted in the single collision domain scenario [13]. However, we are really interested in maximizing the aggregate rate of data transmissions. Since streams on different links can have different rates and also because rate on a single MIMO link does not necessarily increase linearly with the number of streams on

[1]Due to possible correlations in the channel, the usable DOFs might be lower. The actual DOFs that a node can use are often referred to as the *effective* DOFs. Throughout the paper, we will assume the total DOFs usable by a node are its effective DOFs and, in most cases, we will drop the use of the term "effective" for simplicity.

[2]Here, we assume each node is equipped with only a single radio.

the link, maximizing the number of streams does not necessarily maximize the aggregate transmission rate. The rate on an individual link is determined by the characteristics of the channel in between the transmitter and receiver, as well as how many DOFs are used at the transmitter and the receiver for multiplexing. We model this with a rate function $R(t_i, r_i, \mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$, which gives the rate on the link (t_i, r_i) when ADOF_{t_i} DOFs are available at t_i and ADOF_{r_i} DOFs are available at r_i. We do not make any assumption on the rate function, i.e. it can be an arbitrary function. Note that, if the channel between t_i and r_i is random (as is the case with Rayleigh fading channels, for example), the rate on the link is also a random variable depending on the channel characteristics. In this case, we interpret R as the expected data rate, which can also be thought of as the long-term rate on the link if its channel characteristics change dynamically and at random.

While allocating more streams on a link l increases the aggregate rate on that specific link, more radio resources (DOFs) need to be used in the network to cancel the interference generated by the transmitter of link l. So, how to optimally allocate streams and DOFs becomes a complex optimization problem even in a single collision domain MIMO network. For a fixed DOF assignment on a link l, the optimal number of streams carried by l depends on the channel characteristics and the available DOFs at the transmitter and receiver. As with the rate function, we can represent the number of streams by a function $s(t_i, r_i, \mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i}) \leq \min(\mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$. In this paper, we make the simplifying assumption that the number of streams is exactly $\min(\mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$. The only situation in which a smaller number of streams is optimal is when the channel between t_i and r_i is so weak that using DOFs to boost signal strength is preferable to increasing the number of spatially multiplexed streams. In the single collision domain scenario, where all nodes are within a small geographical area, it is reasonable to assume that all channels are sufficiently strong so that maximizing streams on a link also maximizes the link's rate.

4. PROBLEM DEFINITION

4.1 Feasibility

The problem of feasibility is to determine if a set of transmissions can be undertaken concurrently such that all individual transmissions are successful, under a given interference model. Many existing scheduling algorithms assume that feasibility can be determined efficiently [1, 2, 4, 12]. These algorithms iteratively add transmissions to slots in a frame, checking a slot for feasibility whenever a new transmission is proposed to be added to a slot by the algorithm. Thus, feasibility is considered to be a very simple operation that can be repeated many times during execution of the scheduling algorithm. One of the interesting findings of this research is that with MIMO links, in certain cases, feasibility checking becomes a very expensive operation. In these cases, the scalability with increasing network size could be quite limited for this iterative schedule construction with repeated feasibility checking approach. Thus, alternative scheduling approaches will likely have to be developed for the most general MIMO link scheduling problems.

Without MIMO links, checking feasibility of a set of transmissions amounts to simply checking the transmissions against the given interference model. However, MIMO links have the capability to suppress interference. Thus, whether a set of transmissions is feasible depends both on the interference model and on how the links choose to use their degrees of freedom (DOFs) in suppressing interference. As detailed earlier, the number of DOFs necessary for the transmitter of a link i to suppress interference on the receiver of a link j is given by the number of streams carried on link j. (The same is true if the receiver of i is suppressing interference from the transmitter of j.) Thus, to determine feasibility, we must know both which links are transmitting *and* how many streams are carried on each of the links.

Let the set of links under consideration be denoted by $L = \{(t_1, r_1), \ldots, (t_m, r_m)\}$. Let $S = [s_1, \ldots, s_m]$ be the stream allocation vector, i.e. the number of streams carried by (t_i, r_i) is $s_i > 0$.Let $K^t = [k^t_1, \ldots, k^t_m]$ be the vector of (effective) DOFs of the transmitters, Similarly, let $K^r = [k^r_1, \ldots, k^r_m]$ be the vector of (effective) DOFs of the receivers. Let $a^t_{ij} = 1$ if the transmitter of link i suppresses interference on the receiver of link j and let $a^t_{ij} = 0$, otherwise. Furthermore, let $a^t_{ii} = 1$, for all i. Denote by A^t the matrix of a^t_{ij} values. Similarly, let $a^r_{ij} = 1$ if the receiver of link i suppresses interference from the transmitter of link j, let $a^r_{ij} = 0$, otherwise, and let $a^r_{ii} = 1$, for all i. Denote by A^r the matrix of a^r_{ij} values.

The feasibility problem is defined as follows:

Input: A set $L = \{(t_1, r_1), \ldots, (t_m, r_m)\}$ of links, a stream allocation vector S for L, and a conflict graph $G_c = (L, E_c)$.

Output: **True** if S and L are feasible and **False** otherwise. S and L are defined to be feasible if L is free of primary interference and there exist A^t and A^r such that:

1. $A^t S \leq K^t$,
2. $A^r S \leq K^r$, and
3. for all $i \neq j$ such that $(l_i, l_j) \in E_c$, $a^t_{ij} + a^r_{ji} \geq 1$.

Conditions 1 and 2 ensure that a node does not use more DOFs than it has available. For each t_i, we have that:

$$s_i + \sum_{j: j \neq i \text{ and } (l_i, l_j) \in E_c} a^t_{ij} s_j \leq k^t_i$$

In other words, t_i uses s_i DOFs for spatially multiplexing its streams and uses s_j DOFs for every receiver on which it suppresses its interference, and the total of these values cannot exceed the size of t_i's antenna array. Condition 1 states this inequality in matrix form over all transmitters and Condition 2 is the equivalent for receivers. Condition 3 ensures that all interference is cancelled, i.e. for every pair of links i and j that interfere with each other, either the transmitter of i or the receiver of j (or both) suppresses the interference from i to j.

Note that the cases of receiver-side suppression only and transmitter-side suppression only can easily be handled by this general problem statement. For receiver-side suppression only, a^t_{ij} is set to zero for all $i \neq j$, and for transmitter-side suppression only, a^r_{ij} is set to zero for all $i \neq j$.

Note also that Conditions 1 and 2 above are a variation of the basic DOF inequality in which a set of boolean variables, the a^t_{ij}'s and the a^r_{ij}'s, are included. These boolean variables indicate which nodes are suppressing interference on which other nodes. Thus, one way of stating the feasibility problem is to ask the question: "Does there exist an assignment of interference suppressions to nodes (a^t_{ij} and a^r_{ij} values) that satisfy the DOF inequalities at every node and together suppress all interference?". This formulation makes it clear that the feasibility problem is a special type of Boolean satisfiability problem.

4.2 Stream Allocation

The achievable rate $R(t_i, r_i, \mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$ on a link depends on the numbers of available degrees of freedom (DOFs) at both ends of the link. The number of available DOFs at t_i is given by $k_i^t - ks_{t_i}$, where ks_{t_i} denotes the number of DOFs that t_i uses to suppress its streams on receivers other than r_i. Note that $ks_{t_i} = \sum_{i \neq j} a_{ij}^t s_j$, i.e. ks_{t_i} is the sum of the numbers of streams received by all receivers on which t_i suppresses its interference. Similarly, the number of available DOFs at r_i is given by $k_i^r - ks_{r_i}$, where $ks_{r_i} = \sum_{i \neq j} a_{ij}^r s_j$.

The stream allocation problem, formally defined below, is to find an optimal stream vector, given a set of links that could be scheduled concurrently. An optimal stream vector is defined as a feasible stream vector with maximum aggregate transmission rate. Links that could be scheduled concurrently means that the links are free of primary interference.

Input: A set $L = \{(t_1, r_1), \ldots, (t_m, r_m)\}$ of primary-interference-free links, DOF vectors K^t and K^r, and rate function $R(t_i, r_i, \mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$.

Output: A stream allocation vector S and matrices A^t and A^r that make S feasible, where (S, A^t, A^r) has maximum aggregate rate over all feasible stream vectors.

4.3 One-Shot Link/Stream Scheduling

In the stream allocation problem, a set of primary-interference-free links is given. However, in classical one-shot link scheduling, the problem is to determine which links, when scheduled together concurrently, will maximize the aggregate rate. In other words, this is a version of link scheduling in which the goal is to squeeze as much out of a single scheduling slot as possible. Repeatedly scheduling a maximum-rate set of links over and over will yield a maximum throughput solution. However, such a schedule obviously does not meet any fairness criteria and, therefore, this approach cannot be considered a solution to an overall network scheduling problem. Nevertheless, one-shot scheduling algorithms can be adapted in various ways to address fairness and can therefore still form a core component of an overall scheduling approach.

We can generalize the stream allocation problem into a one-shot scheduling problem. In this case, we simply need to start with an arbitrary set of links (rather than being given a primary-interference-free set of links) while maintaining the same goal of maximizing the aggregate rate. Thus, in this situation, the problem can be defined as follows:

Input: An arbitrary set $L = \{(t_1, r_1), \ldots, (t_m, r_m)\}$ of links, DOF vectors K^t and K^r, and rate function $R(t_i, r_i, \mathrm{ADOF}_{t_i}, \mathrm{ADOF}_{r_i})$.

Output: A set of primary-interference-free links L_{pif}, a stream assignment vector S for L_{pif}, and matrices A^t and A^r that make S feasible, where (L_{pif}, S, A^t, A^r) has maximum aggregate rate over all sets of primary-interference-free links and feasible stream vectors.

5. RESULTS ON FEASIBILITY

5.1 Complexity of MIMO Feasibility

In this section, we study the complexity of checking the feasibility of a stream allocation vector in a MIMO network. As is made clear in the formal problem definition given in Section 4.1, feasibility is a special type of Boolean satisfiability problem. It is well known that many variations of Boolean satisfiability are NP-complete. Due to the specialized constraints in the MIMO feasibility problem, it is unlikely that a proof of NP-completeness will be found, but we conjecture that MIMO feasibility is NP-complete. However, certain special cases of the feasibility problem are solvable in polynomial time and we provide proofs of this for several cases in this section.

The fact that feasibility can no longer be trivially solved with MIMO links has important implications for scheduling algorithms. As mentioned earlier, many greedy scheduling algorithms attempt to assign links to the first slot in which they are feasible. This common approach assumes that feasibility can be efficiently tested, so that repeated execution of feasibility checks does not negatively impact the execution time of the scheduling algorithm. Since this assumption is not valid for the general MIMO link scheduling case, alternative approaches to building schedules will have to be developed. For example, approaches that build schedules, which are provably feasible by the manner in which they are constructed and thus do not have to employ feasibility tests, could be developed.

When CSI is available only at the receivers and not at the transmitters, then only receiver side interference suppression can be done. Theorem 1 states that, in this special case, the feasibility problem is polynomial time in complexity.

Theorem 1 *Checking feasibility of a stream allocation vector S and a link set L over an arbitrary MIMO network with receiver-side-suppression only can be done in polynomial time.*

Proof: Let $S = [s_1, \ldots, s_m]$ and recall that $s_i > 0$, for all i (otherwise, we can simply remove link i from L and consider feasibility of the smaller vector and link set). Denote the conflict graph of the network by $G_c = (L, E_c)$. Note that Condition 3 of feasibility (see Section 4.1) says that for all distinct links i and j that conflict with each other, either $a_{ij}^t = 1$ (meaning t_i suppresses its interference on r_j) or $a_{ji}^r = 1$ (meaning r_j suppresses interference from t_j). Since only receiver-side suppression is done in this case, $a_{ij}^t = 0$ and in order to satisfy Condition 3, it is required that $a_{ji}^r = 1$. Thus, every receiver must necessarily suppress interference from every transmitter whose link conflicts with the receiver's link.

Checking feasibility amounts to checking whether all of the conditions from the feasibility problem definition are true. In light of the fact that $A^t = 0$ with receiver-side-suppression only, the following procedure suffices to check feasibility.

Step 1: Check that L is primary-interference-free. This can be done by simply scanning through all links and counting the number of occurrences of every node. If any node appears more than once, L is not primary-interference-free and S and L are not feasible. If all nodes appear at most once in L, then continue to Step 2.

Step 2: Check whether there exist A^t and A^r that satisfy Conditions 1-3 of feasibility. From the above discussion A^t is the matrix of all zeroes and therefore Condition 1 ($A^tS \leq K^t$) is trivially satisfied. Since A^t is all zeroes, A^r is therefore fixed by Condition 3. Therefore, it is only necessary to check that Condition 2 is satisfied for every link. In the case under consideration, for a given receiver r_i, Condition 2 becomes:

$$s_i + \sum_{j:j \neq i \text{ and } (l_i,l_j) \in E_c} s_j \leq k_i^r$$

Since all s_i's are given by the input stream allocation vector S, checking this condition amounts to simply checking the above inequality for every receiver. This can be easily done in $O(l^2)$ time.

∎

The essence of what makes feasibility polynomial time in the special case of receiver-side-suppression only is that the choice of how to suppress interference (either by the transmitter of the interfering link or by the receiver of the interfered with link) is removed. In the general MIMO case, for every pair of interfering links, there is a choice as to how to suppress the interference, and combining these choices over all pairs of interfering links yields an enormous number of possibilities that are all potential ways to make a stream allocation vector feasible. The case of transmitter-side-suppression only also removes the choice of how to suppress interference and, therefore, it has the same effect on problem complexity, i.e. feasibility for the transmitter-side-suppression only case is also of polynomial complexity.

Another interesting special case is when the DOFs of all nodes are small. In particular, when every node in the network has $k = 2$ DOFs, even when interference suppression can be done at *both* transmitter side and receiver side, then the feasibility problem is polynomial time in complexity. This result is stated in Theorem 2.

Theorem 2 *Checking the feasibility of a stream allocation vector S and a link set L over an arbitrary MIMO network where every node has $k = 2$ degrees of freedom is a polynomial time operation.*

Proof: See Appendix.

5.2 Feasibility Heuristics

Given that the general MIMO feasibility problem is quite possibly NP-complete, heuristics for checking feasibility are necessary. Perhaps the most obvious heuristic is to see whether all interference can be suppressed by greedily allocating DOFs for interference suppression. The algorithm works as follows. Sort the links in order of non-increasing number of allocated streams. Begin with the first link and use its DOFs to suppress interference on the links with which it interferes one by one until all its DOFs are used. Then, move onto the next link and continue until all interference is suppressed or all DOFs are used up, whichever comes first. If all interference can be removed wiht the available DOFs in the network, the allocation vector is declared to be feasible.

In experimenting with Algorithm Simple Greedy, we found that it tends to concentrate DOFs among small groups of nodes, rather than more evenly distributing those resources across links in the network, and this causes it to frequently label feasible vectors as infeasible. To remedy this problem,

```
Input: Stream allocation vector $S$, link set $L$, $K^t$, $K^r$, conflict graph $G_c = (V_c, E_c)$
Output: feasible $\in \{\text{true}, \text{false}\}$, $A^t$, $A^r$

1: Order $S$ in non-increasing fashion
2: $A^t = A^r = I_{|L| \times |L|}$
3: for $i = 1 \rightarrow |L|$
4:   if $< A^t(i, 1:i), S^{1:i} > \; \leq K_i^t$, distribute $1's$ in $A^t(i, G_c(i, i+1:l))$ greedily, giving equal priority to columns of equal weight such that $< A_i^t, S > \; \leq K_i^t$
5:   if $< A^r(i, 1:i), S^{1:i} > \; \leq K_i^r$, distribute $1's$ in $A^r(i, G_c(i, i+1:l))$ greedily, giving equal priority to columns of equal weight such that $< A_i^r, S > \; \leq K_i^r$
6:   $A^r(m,i) = 1 - A^t(i,m)$ and $A^t(m,i) = 1 - A^r(i,m)$ $\forall m \geq i+1 : (i,m) \in E_c$
7: end for
8: feasible = true if $A^tS \leq K^t$ and $A^rS \leq K^r$, else feasible = false
```

Figure 1: Algorithm Extended Greedy

we developed the algorithm in Figure 1, which we refer to as Algorithm Extended Greedy. This algorithm, when considering multiple candidate links, all carrying equal number of streams, on which to suppress interference, chooses a target link uniformly at random from the candidates. This tends to produce a better distribution of resources and outperforms Algorithm Simple Greedy. In Figure 1, note that the standard notation $< V, W >$ is used to represent the inner product of vectors V and W and that I is the identity matrix.

Both Algorithm Simple Greedy and Algorithm Extended Greedy are safe, in the sense that they always label infeasible vectors as infeasible. However, they are both non-optimal in that they each label some feasible vectors as infeasible. The accuracy of the two heuristics is evaluated in Section 5.3), in terms of the percentage of feasible vectors that are labeled infeasible.

5.3 Accuracy of Feasibility Heuristics

The scalability of the heuristics for verifying feasibility of a stream allocation vector in a MIMO network is studied experimentally by calculating the entire feasible space for values of $K^t = K^r = K = 8, 12, 16$ and network sizes up to 15 links. The results are shown in the graph of Figure 2. Note that the Extended Greedy heuristic is significantly more accurate than Simple Greedy. Extended Greedy is inaccurate at most 5% of the time with $k = 8$ and $k = 12$ and at most 10% of the time with $k = 16$, for the network sizes studied here.

We have also developed a more accurate feasibility heuristic based on a sum-product algorithm. However, the feasibility heuristics are not the main emphasis of this paper. Rather, we are interested in applying the heuristics within an overall one-shot link scheduling approach. Since, over the range of network sizes studied in Section 7.2, the better accuracy of the sum-product algorithm does not have a substantial impact on the overall results, we choose not to take the extra space to describe and evaluate this additional heuristic herein.

6. STREAM ALLOCATION AND ONE-SHOT SCHEDULING ALGORITHMS

Consider the general one-shot link scheduling problem of maximizing the aggregate throughput over an arbitrary set of

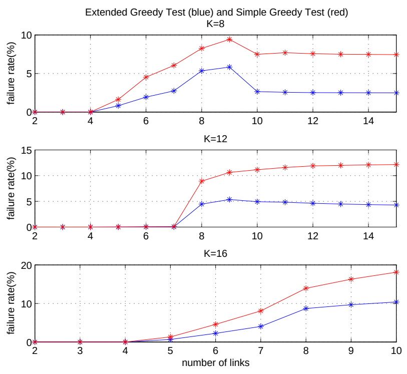

Figure 2: Failure Rates of Simple Greedy and Extended Greedy Heuristics

links (that are not necessarily primary-interference-free). We approach the problem by splitting it into two subproblems. In the first problem (stream allocation), an algorithm determines a stream allocation vector that approximately maximizes the throughput, given a set of primary-interference-free links. The second problem considers how to select a "good" set of primary-interference-free links to provide as input to the stream allocation algorithm. When solving the overall one-shot link scheduling problem, we first run the primary-interference-free link selection algorithm, then run the stream allocation algorithm using the output of the link selection algorithm.

6.1 Stream Allocation Algorithms

A simple heuristic for the stream allocation problem, which we will use as a baseline for comparison, is to adopt a greedy approach. Streams are scheduled successively from highest to lowest rate and the allocation vector is tested for feasibility at each step. We refer to this as the "upward construction" approach, because it simply keeps adding streams in a greedy fashion until no more streams can be added without making the allocation vector infeasible.

We now present an algorithm that provides a better approximation to the stream allocation problem (compared to the simple upward construction approach). For a given set of primary-interference-free links, the value of the stream allocation vector S is initialized so as to maximize the aggregate throughput while satisfying the following two constraints: (1) interference between every pair of links is suppressed and (2) weight of the stream allocation vector is bounded by $\left\lfloor \frac{2kl}{l+1} \right\rfloor$, where l is the number of non-zero entries in the vector and k is the median value of the vector resulting from taking the minimum of the elements of K^t and K^r.[3] Note that, since the initialization only checks pairwise interference between links, the initial vector might not be feasible. However, since all pairwise interference constraints can be checked in polynomial time, this produces an initial vector with high throughput that provides a minimum level of interference suppression. Pseudo-code is shown for the initialization procedure in Figure 3.

```
1: Let $S^0 = [0,\dots,0]_{|L|}$
2: repeat
3: add the highest rate stream not already in $S^0$ that maintains the following condition: $\forall l_i, l_j \in L$, at least one of the following holds: $s_i^0 + s_j^0 \le \min(K_i^t, K_i^r)$; $s_i^0 + s_j^0 \le \min(K_j^t, K_j^r)$; $s_i^0 + s_j^0 \le \min(K_i^t, K_j^r)$; $s_i^0 + s_j^0 \le \min(K_j^t, K_i^r)$
4: until $\sum_{i=1}^{|L|} s_i^0 = \left\lfloor \frac{2kl}{l+1} \right\rfloor$, where $l =$ no. of non-zero entries in $S^0$ and $k =$ median of $\min(K^t, K^r)$
```

Figure 3: Initialization Procedure for Algorithm StreamMaxRate

Input: Primary-interference-free link set L, K^t, K^r, $R(t_i, r_i, ADOF_{t_i}, ADOF_{ri})$
Output: Feasible stream allocation vector S for L, A^t and A^r that make S feasible

```
1: Initialization: Choose $S^0$ to satisfy pairwise interference constraints and approximately maximize aggregate rate as detailed in Figure 3
2: $S = S^0$
3: (feasible, $A^t$, $A^r$) = ExtGr($S, L, K^t, K^r$)
4: while not feasible
5:     (feasible, $S, A^t, A^r$) = UpdateRule($S, L, K^t, K^r, A^t, A^r$)
6: end while
```

Figure 4: Algorithm StreamMaxRate

Once an inital stream allocation vector is determined, it is tested for feasibility using any feasibility checking algorithm (in the description presented herein, we assume the Extended Greedy heuristic of Section 5).[4] If the initial vector is feasible, then it becomes the the final output of the algorithm. In most cases, the initial stream allocation vector will not be feasible and the algorithm will then adjust it to try to make it feasible. This is done by removing streams from the initial vector until it becomes "more feasible" and then trying to add more streams in where possible without reducing feasibility. The stream allocation vector is adjusted by an update rule procedure, which is guaranteed to increase the number of valid rows of the LHS in each of Conditions 1 and 2 that are feasible by at least one. Thus, repeated calls to the update rule will eventually produce a stream allocation vector that is completely feasible. Pseudo-code is shown for the overall StreamMaxRate algorithm in Figure 4 and for the allocation vector updating procedure in Figure 5.

6.2 One-Shot Link Scheduling Algorithms

As mentioned earlier, our approach to one-shot link scheduling is to first pick a "good" set of primary-interference-free links and then apply Algorithm StreamMaxRate to optimize stream allocation among those links. Any set of links making up a matching of the communication graph is primary-interference-free and is therefore an eligible candidate for the input to Algorithm StreamMaxRate. Clearly, the optimal solution will use a set of links corresponding to some maximal matching.

[3] If $K_i^t = K_i^r = k, \forall i$, then $\left\lfloor \frac{2kl}{l+1} \right\rfloor$ is the maximum number of streams [13].

[4] Since we assume single collision domain networks in this part, we omit the conflict graph input to Extended Greedy.

Input: Stream allocation vector S, link set L, K^t, K^r, A^t, A^r
Output: feasible $\in$ {true, false}, updated stream allocation vector S, A^t, A^r

```
1:  $\mathrm{nfr}^0 = \min(\mathrm{nfr}^t, \mathrm{nfr}^r)$, where $\mathrm{nfr}^t$ and $\mathrm{nfr}^r$ are the number of
    feasible rows in $A^t$ and the number of feasible rows in $A^r$, with
    respect to $S$
2:  repeat
3:      remove the lowest rate stream from $S$
4:  until $\min(\mathrm{nfr}^t, \mathrm{nfr}^r) > \mathrm{nfr}^0$
5:  $\mathrm{nfr}^1 = \min(\mathrm{nfr}^t, \mathrm{nfr}^r)$
6:  for each stream $s_i$ not included in $S$ from highest rate stream to
    lowest rate stream
7:      add $s_i$ to $S$
8:      (feasible, $A^t$, $A^r$) = ExtGr($S, L, K^t, K^r$)
9:      calculate $\mathrm{nfr}^t$ and $\mathrm{nfr}^r$ from $A^t$ and $A^r$
10:     if $\min(\mathrm{nfr}^t, \mathrm{nfr}^r) < \mathrm{nfr}^1$ then remove $s_i$ from $S$
11: end for
```

Figure 5: Update Rule for Algorithm StreamMaxRate

We consider two different primary-interference-free link selection heuristics, based on weighted matching algorithms:

1. The weight of each link is set equal to the inverse of the physical distance between the transmitter and receiver of that link, i.e. $w_i = \frac{1}{d_i}$. Here, we find a *maximum* weighted matching using the algorithm of [11].
2. The weight of each link is set equal to the physical distance between the transmitter and receiver of that link, i.e. $w_i = d_i$. Here, we find a *minimum* weighted matching using a standard algorithm.

7. RESULTS ON STREAM ALLOCATION AND ONE-SHOT SCHEDULING

We have proposed the StreamMaxRate Heuristic for approximately maximizing the throughput over a given set of links in Section 6.1. In this section, we will define an experimental set up and use simulation results to compare the performance of the StreamMaxRate heuristic against the optimal throughput. Because of the relatively small number of links that can be active concurrently within a single collision domain, we are able to calculate the optimal solution for a good portion of the input parameter space considered. Additionally, we have simulated the greedy upward construction approach of finding a stream allocation and use this as a second comparison point. Finally, we show some results on the overall one-shot link scheduling problem by including the two weighted matching methods of selecting a set of primary-interference-free links. Since brute-force searching the space of all possible maximum matchings is infeasible, we only compare our approach to the greedy construction in this case.

7.1 Simulation Set-up

For all simulations, every node is equipped with an antenna array of size $k = 8$. This allows for a maximum of $l = 15$ links to be active concurrently, given the single collision domain assumption. The experimental set up for the stream allocation results is as follows. We distribute 50 nodes (with a uniform distribution) over a field of fixed dimensions. All nodes are within interference range of each other. We select 50 randomly generated matchings (sets of primary-interference-free links) with sizes ranging from two to fifteen. This is averaged by repeating the procedure over a sample space of node distributions. For different matching sizes, we compute (a) the optimal throughput (b) the throughput resulting from application of the StreamMaxRate heuristic and (c) the throughput obtained by applying the greedy upward construction procedure.

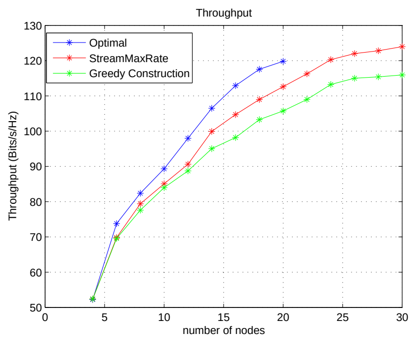

Figure 6: Throughput vs number of nodes for randomly selected matchings

The set-up for the one-shot link scheduling results is as follows. We distribute (uniformly) an even number of nodes, ranging from $N = 2$ to $N = 30$ over a field of fixed dimensions. For each value of N, we select a maximal matching (of size $N/2$, since all nodes are within transmission range of each other). We do this by the two maximum weighted matching procedures described in Section 6.2. We combine these two matching selection procedures with the two stream allocation heuristics (StreamMaxRate and greedy) to produce four different curves. For each data point on a curve, an average was obtained by repeating the process over a large sample space of node distributions.

For all simulations, the channel was modeled as an idealized rich scattering static environment, which corresponds to a quasi-static flat Rayleigh fading channel model. Therefore, the channel has i.i.d. complex, zero mean, unit variance elements as described by [7]. The gain of each channel matrix was calculated using Friis transmission equation and the log-distance path-loss model with a path-loss exponent of 3 [10]. We assume channel state information is available to the transmitters and therefore include optimal power allocation in our rate calculations. The data rate is calculated using Shannon's capacity formula with optimal power allocation [8].

7.2 Simulation Results

Figure 6 shows the results for the stream allocation problem alone. Due to the large computation time of determining the optimal value of the throughput for larger numbers of nodes and links, the optimal result is shown only up to 20 nodes, which corresponds to 10 links. Note that at $n = 20$, the throughput from StreamMaxRate is within 7% of the optimal. The greedy upward construction approach is within 15% of the optimal at this point. Thus, StreamMaxRate cuts the difference between the heuristic and the optimal in half at this point. Note also that the difference between the greedy heuristic and StreamMaxRate increases with network size. Extrapolating the optimal curve in a natural way would indicate that the halving of the difference from optimal produced by StreamMaxRate should continue over the range of net-

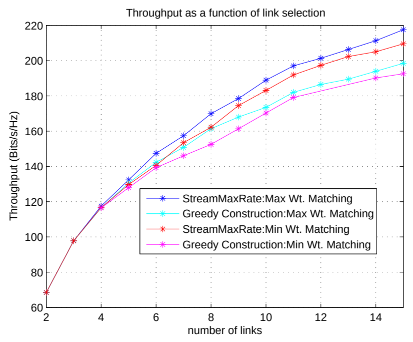

Figure 7: Throughput vs number of active links for two different methods of selecting matchings

work sizes simulated.

We now present results for one-shot link scheduling. In this case, we have the two weighted maximum matching methods for selecting the set of primary-interference-free links and we evaluate those using both Algorithm StreamMaxRate and the greedy construction method for stream allocation. The results are shown in Figure 7. For both methods of finding matchings, StreamMaxRate retains its performance advantage compared to the greedy stream allocation heuristic (about 10-15% higher throughput for the largest network size simulated). We also find that the matching selection approach that finds the maximum weighted matching with links weighted by the inverse of their distances outperforms the one that finds the minimum weighted matching with weights equal to the distances. The difference between the two matching selection approaches is only moderate, however, being about 5% for the largest network size.

Note that, for one-shot link scheduling, we cannot compare against the overall optimal solution, since checking all maximal matchings is not feasible for the network sizes considered. However, given the result of the maximum matching heuristic, we can find the optimal allocation (as we did for the stream allocation problem results). We did this comparison and found that StreamMaxRate was again within 6% of optimal for these specific matchings (the same as its performance on random matchings described earlier).

REFERENCES

[1] M. Alicherry, R. Bathia, L. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks," *Proc. ACM Mobicom,* pp. 58–72, 2005.

[2] L. Badia, A. Erta, L. Lenzini, F. Rossetto, M. Zorzi, "A Physical Model Scheduler for Multi-Hop Wireless Networks Based on Local Information", *Proc. IEEE MASS,* 2008.

[3] R. Bhatia, L. Li, "Throughput Optimization of Wireless Mesh Networks with MIMO Links", *Proc. IEEE Infocom miniconference*, pp. 2326–2330, 2007.

[4] G. Brar, D. Blough, and P. Santi, "Computationally Efficient Scheduling with the Physical Interference Model for Throughput Improvement in Wireless Mesh Networks," *Proc. of ACM Mobicom,* pp. 2–13, 2006.

[5] S. Chu, X. Wang, "Adaptive and Distributed Scheduling in Heterogeneous MIMO-based Ad Hoc Networks", *Proc. IEEE MASS*, 2009.

[6] T. ElBatt, "Towards Scheduling MIMO links in Interference-limited Wireless Ad Hoc Networks", *Proc. IEEE MILCOM*, pp. 1–7, 2007.

[7] G. Foschini and M. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications,* Vol. 6, pp. 311–335, 1998.

[8] D. Gesbert, M. Shafi, D. Shiu, P. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, Vol. 21, pp. 281–302, 2003.

[9] B. Hamdaoui, K.G. Shin, "Characterization and Analysis of Multi-Hop Wireless MIMO Network Throughput", *Proc. ACM MobiHoc*, pp. 120–129, 2007.

[10] A.F. Molisch, *Wireless Communications*, John Wiley and Sons, Chichester, UK, 2005.

[11] R. Preis, "Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs," *Lecture Notes in Computer Science,* Vol. 1563, pp. 259–269, 1999.

[12] S. Ramanathan, E, Lloyd, "Scheduling Algorithms for Multihop Radio Networks", *IEEE Trans. on Networking,* Vol. 1, pp. 166–177, 1993.

[13] R. Srinivasan, D. Blough, P. Santi, "Optimal One-Shot Stream Scheduling for MIMO Links in a Single Collision Domain", *Proc. IEEE Secon*, 2009.

[14] K. Sundaresan, R. Sivakumar, M.A. Ingram, T-Y. Chang, "Medium Access Control in Ad Hoc Networks with MIMO links: Optimization Considerations and Algorithms", *IEEE Trans. on Mobile Computing,* Vol. 3, n. 4, pp. 350–365, 2004.

[15] Y. Wang, D-M. Chiu, J.C.S. Lui, "Characterizing the Capacity Gain of Stream Control Scheduling in MIMO Wireless Mesh Networks", *Wireless Communications and Mobile Computing,* Vol. 9, n. 6, pp. 819–829, 2009.

Appendix

This appendix contains the proof of Theorem 2, which is included for the benefit of reviewers but will be omitted in the final version of the paper.

Proof:

The proof is constructive, i.e., we describe a polynomial time algorithm that, given inputs S and L, returns **True** if and only if stream allocation vector S is feasible for link set L. The algorithm first checks whether L is primary-interference-free in polynomial time (as in the proof of Theorem 1). If L is not primary-interference-free, the algorithm returns **False**, otherwise it continues with the procedure described next.

Let the conflict graph of the MIMO network be $G_c = (L, E_c)$. Every active link l_i carries a number of streams $s_i = \{1,2\}$. Inactive links (with zero streams allocated) are not represented in G_c. Let $L_2 = \{l_i \in L : s_i = 2\}$. Since each link in L_2 utilizes its full multiplexing capacity, no resources for interference suppression are available. The remaining links are contained in the set $L_1 = L \backslash L_2$, composed of links carrying a single stream.

The feasibility algorithm first checks whether all links of L_2 are isolated vertices in G_c. If not, the algorithm returns **False**, otherwise, it considers the subgraph G_1 of G_c induced by node set L_1. Let $G^1, \ldots, G^h$ be the connected components of graph G_1. The algorithm checks whether for each $G^i = (L^i, E^i)$, inequality $|E^i| \leq |L^i|$ is satisfied; if the inequality is not satisfied for any of the G^i, the algorithm returns **False**, otherwise it returns **True** and terminates.

It is immediate to see that the above algorithm has polynomial time complexity. We now prove that, when the algorithm returns **False** on input S, L, stream allocation vector S is infeasible for L. To prove this, we observe that the algorithm returns **False** if only if one of the following conditions hold:

1) set L is not primary-interference-free; in this case, it is clear that any non-zero stream allocation vector S for L is infeasible.
2) L_2 contains at least one link, which is not an isolated vertex in G_c; denote such a link by l_i and suppose it is adjacent to link l_j in the conflict graph. Since l_i carries two streams, it has no DOFs available for suppression. Link l_j carries at least one stream and, therefore, has at most one DOF remaining, which is not enough to suppress the two streams on l_i. Hence, condition (3) for feasibility cannot be satisfied for links l_i, l_j unless conditions (1) and (2) are violated. This implies that stream assignment S is not feasible for link set L.
3) there exists a connected component G^j of graph G_1 such that $|E^j| > |L^j|$. A simple counting argument can be used to prove that S is not feasible for L: for each link $l \in L^j$, two DOFs are available at the link endpoints to suppress interference (one at the transmitter and one at the receiver side). Thus, $2|L^j|$ DOFs in total are available to suppress interference within G^j. On the other hand, suppressing interference between any two adjacent links l_i, l_j in the conflict graph requires using 2 DOFs: one for suppressing interference generated by t_i on r_j, and one for suppressing interference generated by t_j on r_i. Thus, $2|E^j|$ DOFs in total are needed to suppress the interference the $|L^j|$ links in G^j cause to each other receivers. Hence, $|E^j| > |L^j|$ implies that not enough radio resources (DOFs) are available within G^j to completely suppress interference, which proves that stream allocation vector S is infeasible for L.

The next step is to prove that whenever none of conditions $1), 2), 3)$ hold on given input S, L, then stream allocation vector S is feasible for L, which implies correctness of our feasibility checking algorithm (which returns **True** in this situation). We prove this last step by showing a construction (DOF assignment) that makes S feasible for L subject to the fact that none of the conditions $1), 2), 3)$ are satisfied.

If condition $3)$ is not satisfied, we have $|L^j| \leq |E^j|$ for each connected component G^j of G_1. We first observe that DOF assignments for the G^js can be built independently, since links in different G_1 connected components do not interfere with each other. We hence show the construction for a single G^j, making the overall construction the result of the composition of DOF assignments for the single connected components. It is not difficult to see that the topology of G^j can take only one of the four following forms: $a)$ single vertex; $b)$ tree; $c)$ simple cycle; $d)$ connected graph containing a single simple cycle. If G^j is of type $a)$, no DOF has to be allocated for interference suppression. If G^j is a tree (type $b)$), perform the following procedure:

1. Designate some vertex in L^j to be the root.
2. For every edge $(l_i, l_k) \in E^j$, use the two available DOFs of the link deeper in the tree (say, l_k) to suppress mutual interference between l_i and l_k.

It is easy to see that, since every vertex in a tree (except the root) has a single parent, each link in the above construction uses at most 2 DOFs to suppress interference, thus not exceeding the available DOFs. On the other hand, mutual interference between all links in G^j is taken care of at the end of the above procedure, implying that the resulting DOF assignment makes S feasible (when restricted to G^j).

Let us now consider case $c)$. In this case, it is sufficient to give either clock-wise or counterclock-wise orientation to the edges in E^j, and to choose an arbitrary vertex l_i in L^j. Consider any two adjacent vertices l_s, l_t in G^j, and assume w.l.o.g. that l_s precedes l_t in the chosen orientation, starting form l_i. Then, the two DOFs available at l_s are used to suppress mutual interference between l_s and l_t. It is easy to see that, similarly to what happens in case $b)$, this construction results in a feasible DOF assignment for S (when restricted to G^j).

Finally, consider case $d)$. In this case, we start by designating every vertex in L^j that is contained in the simple cycle and is of degree equal to 3 as the root of the tree component it belongs to. DOFs are then assigned by combining the construction for case $b)$ within the trees, with construction for case $c)$ along the single simple cycle contained in G^j. Note that the resulting construction is feasible since root vertices in construction $b)$ do not use their available DOFs to suppress interference with other links in the tree; hence, these available DOFs can be used to suppress interference with the successive vertex (link) in the simple cycle as described in the construction for case $c)$. Thus, the resulting DOF assignment makes S feasible (when restricted to G^j), and the theorem is proved.

∎