Maximum MIMO Flow in Wireless Networks under the SINR Model

Eyjólfur I. Ásgeirsson
School of Science and Engineering
Reykjavik University
Email: eyjo@ru.is

Magnús M. Halldórsson
School of Computer Science
Reykjavik University
Email: mmh@ru.is

Pradipta Mitra
School of Computer Science
Reykjavik University
Email: ppmitra@gmail.com

***Abstract*—We present a framework for the maximum flow problem in wireless networks using the SINR interference model in combination with MIMO nodes. The performance ratio of our algorithm matches the best $\mathcal{O}(\log n)$ approximation factor known for the pure flow problem, but avoids the impractical dependence on the ellipsoid method and features both simpler and more intuitive analysis.**

The objective of a maximum flow in wireless networks is to get as much information from a sender to a receiver using intermediate nodes in a multi-hop environment. The algorithm is based on an LP formulation of the flow problem, and handles gracefully all additional linear constraints. The set of constraints that can be included contains, among other, power limits, fairness between source-sink pairs, capacity limits and bounds, and the use of Multi-Input Multi-Output nodes for the source and the sink nodes, which are often the bottlenecks in a wireless flow.

I. Introduction

The problem of interference is one of the fundamental problem for wireless communications. Concurrent transmissions will interfere with each other, and if the interference is too much, the transmission will fail. There are numerous ways of modeling and dealing with interference, such as simple graph based interference models and spatial, temporal or frequency separation of the transmissions. In recent years, the physical model of interference, or the SINR model, has received much attention, and for interference management, interference alignment is becoming more popular due to the increased availability of Multi-Input Multi-Output (MIMO) wireless transmitters and receivers. However, until now, there have been almost no results where these two approaches, i.e. interference alignment under the SINR model, have been merged successfully.

We treat in this paper a general throughput maximization problem, where we are given a collection of source-destination pairs and seek to find a flow and a schedule for that flow that maximizes the amount of information that can be routed. This is a core problem for many applications, such as video streaming. We present a general framework which we show can be combined with interference alignment for still greater throughput.

Interference alignment is a linear algebraic approach towards interference management. Consider the case where n transmitters intend to transmit to n receivers. If all channel states and intended transmissions are known, the required received signals can be expressed as a linear combination of the variables and solved as a set of n linear equations with n variables. The goal of interference cancellation is to realize this potential in more realistic scenarios.

The SINR model approach and the cancellation or alignment approach have different strengths and weaknesses. Interference alignment can work for *any* channel state, whereas in the SINR model, one typically needs smooth geometric states. Also, cancellation does not require any spatial separation. On the other hand, the SINR model is more robust. Let's assume that we are in a situation where we only know the channel states up to a factor of 2. SINR would handle this gracefully. but interference alignment is impossible under those conditions. The "scalability" of SINR seems to be greater than for interference alignment, if the relevant conditions are met.

Is it possible to combine SINR and alignment? They seem to be a bad fit. Power in alignment is dictated by channel state, whereas in SINR the power is typically length based or based on other geometric considerations. It is unclear that a feasible set in the SINR model would benefit from multiple antennas.

We believe flow problems may be an area where the SINR model and alignment can be combined. Instead of just looking at a simple flow from a source node to a destination node, we use MIMO nodes for the source/destination, thus increasing the bandwidth of those particular nodes, but then harness the power of SINR for the interior paths. We can put both of these gracefully in the same LP.

Our main contributions in this paper are:

- Giving a general framework that holds for any flow problem with linear constraints. These problems include using MIMO nodes for the source and the sink, power limitations, bandwidth of edges or any other linear constraint on the system.
- Show how to add a fairness measure to the flow for multiple source-sink pairs so that instead of maximizing the total flow, we maximize the minimum flow over all the source-sink pairs.
- Matching the best approximation known of $\mathcal{O}(\log n)$ for basic maximum flow in wireless networks, with a simpler and more intuitive analysis.
- Obtaining an efficient framework, resulting in the first practical algorithm with an approximation bound that depends only on n.

II. Related Work

The salient theoretical work on interference alignment is perhaps [1]. Here it is assumed that the global channel state is known by all transmitters, but not the intended symbols. It is shown that each transmitter can successfully transmit half the time. However, to do this, apart from knowing global states, the transmitters need to encode across exponentially many channels.

More practically oriented work includes [2]. In [2] it is shown that in a Multi-Input Multi-Output (MIMO) network where nodes have up to t antennas, t concurrent transmissions can occur. Global state information is needed to this, which is included in RTS-CTS packets and are overheard.

Basically, it seems that the "scalability factor" t of interference cancellation should be considered to be "constant". Considering [2], it is unlikely that channel states of n transmitters can be kept updated for large n, especially if some temporal change in the state is to be expected.

The maximum flow problem has been addressed extensively under graph-based models, but in relatively few works under the SINR model. The first approximation algorithm was given by Chafekar et al. [3], [4] who gave a $\mathcal{O}(\log\Delta\cdot\log\Gamma)$, where Δ is the maximum ratio between link lengths and Γ is the maximum ratio between sender powers. For linear power, they obtained $\mathcal{O}(\log\Delta)$-approximation. Our interest, however, is in performance ratios that do not depend on structural parameters of the instance, such as Δ or Γ. Wan et al. [5] studied the problem of finding a route and maximum throughput for a set of multi-hop requests. They gave a generic reduction to the single-hop throughput problem, known as Capacity [6] or MISL (Maximum Independent Set of Links) [5]. Thus, they obtained a $\mathcal{O}(\log n)$-approximation to the maximum flow problem, both for the case of length-monotone sublinear power assignments [7] and for arbitrary power [8]. For that reduction, however, they used a framework of [9], for which Wan states that "this algorithm is of theoretical interest only, but practically quite infeasible". In [10], Even et al. give a more practical $\mathcal{O}(\log n)$ approximation algorithm for this problem, but their analysis only holds for linear power; for other power assignments, they obtain a $\mathcal{O}(\log n\cdot\log\Lambda)$-approximation, where Λ is the ratio between the maximum received power to the minimum received power of a link. Finally, an exact algorithm is given by Shi et al. [11] via non-linear programming; as the problem is NP-hard, such algorithms necessarily run in exponential time.

A related problem is that of the combined multi-hop routing and scheduling problem. Here we are also given a set of source-destination pairs, but now the objective is to minimize the latency (or schedule length) and must schedule paths between all the pairs. Chafekar et al. [12] gave a $\mathcal{O}(\log^2 n\log\Delta\log^2\Gamma)$-appproximation relative to arbitrary power, which was improved by [13], and further improved by [8] to $\mathcal{O}(\log^2 n)$-factor.

III. Model and Problem Statement

Assume we have a set of wireless nodes V, each a point in a metric space. Also given is a set of links (or directed edges) $E\subseteq V\times V$. Also given are k source-destination pairs $(\hat{s}_i,\hat{t}_i)$ of nodes in V. Each link $\ell_v\in E$ has a sender and a receiver, which we denote by s_v and r_v.

The problem is to compute a multi-commodity flow $f_1,f_2\dots f_k$ and an efficient schedule to support it. Flows have their usual meaning, each flow f_i is a function from edges in E to non-negative numbers respecting flow conservation – the incoming flow to each node (except the source and destination) must be equal to the outgoing flow. The value of the flow f_i is the outgoing flow from the source. We use f_i to indicate both the flow function on edges as well as the value of the flow. A *schedule* is a collection of sets $\{S_1,S_2,\dots S_T\}$ such that each $S_j\subseteq E$. A schedule $\{S_1,S_2,\dots S_T\}$ *supports* the flows $f_1,f_2,\dots f_k$ if $T\cdot f(w)\le|\{S_j\ni\ell_w:1\le j\le T\}|$, for each ℓ_w. In other words, if a link ℓ_w has a flow value $f(w)$ then ℓ_w must appear in at least $T\cdot f(w)$ of the sets $S_1,S_2,\dots S_T$.

Our goal is to find a maximum flow and a schedule that supports it. The constraint on sets S_j is that each of them must be feasible, that is, links in the set can transmit simultaneously without interfering with each other too much. To characterize feasible sets, a model of interference is needed — in this paper, we adopt the SINR or physical model of interference.

From the requirement that a flow must be supported by a schedule, it follows that the total flow on a single link must be smaller than 1. We call such a flow *valid*.

We adopt the *physical model* (or *SINR model*) of interference, in which a node r_v successfully receives a message from a sender s_v if and only if the following condition holds:

$$\frac{P_v/d_v^\alpha}{\sum_{\ell_w\in S\setminus\{\ell_v\}}P_w/d_{wv}^\alpha+N}\ge\beta, \tag{1}$$

where P_v is the transmission power of sender s_v, d_{wv} is the distance from sender s_w to receiver r_v, d_v is the length of link ℓ_v, N is a universal constant denoting the ambient noise, $\beta\ge1$ denotes the minimum SINR (signal-to-interference-noise-ratio) required for a message to be successfully received, and S is the set of links that are transmitting at the same time as link ℓ_v. A set S of links is *SINR-feasible* (or simply *feasible*) if (1) is satisfied for each link in S.

We focus on polynomial powers, where a link ℓ_v uses power $P_v=d_v^{p\alpha}$ for some fixed $p\in[0,1]$. These include the natural and widely used power assignments [14], [15], [16], [7], including uniform ($p=0$), linear ($p=1$) and mean power ($p=\frac{1}{2}$). Mean power is of particular interest as it has been shown to be as much as exponentially more effective than uniform or linear power [17], [18], [6] and never significantly worse [19]. Also it is never more than $\mathcal{O}(\log\log\Delta)$-factor worse than arbitrary power assignment [6]. We also treat the case of *arbitrary power*, when choosing the power assignment is a part of the problem description.

The *affectance* $a_w^P(v)$ [20], [21], [15] of link ℓ_v caused by another link ℓ_w, with a given power assignment P, is the interference of ℓ_w on ℓ_v relative to the power received, or

$$a_w^P(v)=\min\left\{1,c_v\frac{P_w/d_{wv}^\alpha}{P_v/d_v^\alpha}\right\}=\min\left\{1,c_v\frac{P_w}{P_v}\cdot\left(\frac{d_v}{d_{wv}}\right)^\alpha\right\},$$

where $c_v=\beta/(1-\beta Nd_v^\alpha/P_v)$ is a constant depending only on the length and power of the link ℓ_v. We will drop P when it is clear from the context. Let $a_v(v)=0$. For a set S of

links and a link ℓ_v, let $a_v^P(S) = \sum_{w \in S} a_v^P(w)$ and $a_S^P(v) = \sum_{w \in S} a_w^P(v)$. Using this notation, Eqn. 1 can be rewritten as $a_S^P(v) \leq 1$, and this is the form we will use. This transforms the relatively complicated Eqn. 1 into an inequality involving a simple sum that can be manipulated more easily. We use the *length ordered symmetric* version of affectance [6], given by

$$\hat{b}_v(w) = \begin{cases} a_v(w) + a_w(v) & \text{if } d_v \leq d_w \\ 0 & \text{otherwise} , \end{cases} \tag{2}$$

in our linear program formulations.

IV. Algorithm

We propose an LP relaxation and use the fractional flows in combination with a link scheduling algorithm to obtain a good approximation. First, we argue two technical issues that are needed for the LP formulation.

First, we show that assuming that no flows are small incurs only a small cost. For a source destination pair $(\hat{s}_i, \hat{t}_i)$, let $E_i \subseteq E$ be the set of links that are part of some path from $\hat{s}_i$ to $\hat{t}_i$. Now,

Theorem 1: Consider an optimal flow $\{f_1, f_2 \dots f_k\}$. Then there is a valid flow $\{f'_1, f'_2 \dots f'_k\}$ such that for all i, $f'_i(w) \geq \frac{1}{10n^4}$ for all $\ell_w \in E_i$ and $\sum_i f'_i \geq (1 - \frac{1}{5n^2}) \sum_i f_i$.

Proof: Set $g_i = f_i$. Iteratively do the following:

> Consider any link ℓ_w such that $g_i < \frac{1}{5n^4}$ (stop if there are no such links). Find a path from $\hat{s}_i$ to $\hat{t}_i$ including ℓ_w (since $\ell_w \in E_i$, such a path must exist). Augment the flow g_i on this path by $\frac{1}{5n^4} - g_i(w)$.

It is clear that g_i always remains a flow and the process must end. Now, g_i may not be valid since the total flow on a link may be greater than 1. However, the total flow on a link cannot increase by more than $\frac{1}{5n^4} \cdot k \cdot n \leq \frac{1}{5n^2}$. Then if we set $f'_i = g_i(1 - \frac{1}{5n^2})$ we will have a valid flow. Hence, for any ℓ_w, $f'_i(w) \geq \frac{1}{5n^4}(1 - \frac{1}{5n^2}) \geq \frac{1}{10n^4}$.

The construction of f'_i and g_i also gives us that $\sum_i f'_i = \sum_i (1 - \frac{1}{5n^2}) g_i \geq (1 - \frac{1}{5n^2}) \sum_i f_i$. ∎

The second technical issue is to establish a crucial inequality involving affectances.

A link w is said to be *non-weak* if $P_w / d_w^\alpha \geq 2\beta N$, i.e. if the link uses at least slightly more transmission power than is necessary to overcome the ambient noise. We will assume that all links in the flow formulation are non-weak. This assumption is common and reasonable, since it can easily be achieved by scaling the powers.

Theorem 2: [6] Assume ℓ_v is any non-weak link and S is a feasible set of non-weak links transmitting with polynomial powers $0 < p \leq 1$. Then, for a constant γ, $\sum_{w \in S} \hat{b}_v(w) \leq \gamma$.

Note that for uniform power, this theorem does not apply, but a bound of $\sum_{w \in S} \hat{b}_v(w) = O(\log n)$ holds [15]. All results that follow will apply for $p = 0$ with this extra $\log n$ factor.

Now consider any optimal solution $\{f_1, f_2 \dots f_k\}$ and a schedule $S_1, S_2 \dots S_T$ supporting the flow. Define $f(v) = \sum_i f_i(v)$. We know that

$$f(v) \cdot T \leq |\{v \in S_j : 1 \leq j \leq T\}| .$$

Let us define $\sigma_v = |\{v \in S_j : 1 \leq j \leq T\}|$, i.e. σ_v is a counter that tells us how often link ℓ_v appears in the schedule $S_1, S_2 \dots S_T$. In this notation, $f(v) \cdot T \leq \sigma_v$. Using Thm. 2,

$$\begin{aligned} \sum_w (f(w) \cdot T \cdot \hat{b}_v(w)) &\leq \sum_w \sigma_w \hat{b}_v(w) = \sum_{j=1}^{T} \sum_{w \in S_j} \hat{b}_v(w) \\ &\leq \sum_{j=1}^{T} \gamma = T\gamma . \end{aligned} \tag{3}$$

LP relaxation: We can now write the following LP relaxation.

$$\max \quad \sum_i f_i \tag{4}$$

$$\text{s.t.} \quad \sum_{w \in E : r_w = a} f_i(w) = \sum_{v \in E : s_v = a} f_i(v) \quad \forall a \in V \setminus \{\hat{s}_i, \hat{t}_i\}, \forall i \tag{5}$$

$$\sum_w \hat{b}_v(w) f(w) \leq \gamma \ \ \forall v \tag{6}$$

$$f_i(w) \geq \frac{1}{10n^4} \ \ \forall w \in E_i, \ \forall i \tag{7}$$

$$\sum_i f_i(w) \leq 1 \ \ \forall w \in E \tag{8}$$

The objective of the linear problem is to maximize the total flow. Constraint (5) is the conservation of flow. It simply states that for every node in V, with the exception of the source and sink nodes, the number of messages from source $\hat{s}_i$ to sink $\hat{t}_i$ that arrive at the node must be equal to the number of such messages that leave the node. Constraint (6) follows from of Eqn. 3, which holds for feasible integral solutions. Constraint (8) ensures that the flow is valid, i.e. that the total flow on a single link is at most 1. Constraint (7) ensures, for a technical reason, that no flow is too small; recall that it may be assumed to hold by Theorem 1 by paying a small factor. It follows that the LP formulation is a true relaxation of the flow problem.

Algorithm: We will use the following algorithm schema:

> Consider the (fractional) flows f_i^* given by the LP solution. Form a set $\hat{L}$ containing $c(w)$ copies of each link $w \in E$, where $c(w) = \lceil 10n^4 \cdot f^*(w) \rceil$. By (7) and (8), $c(w)$ is an integer between 1 and $10n^4$. Apply a scheduling algorithm to schedule the links of $\hat{L}$.

We show that both of the types of link scheduling algorithms that have been shown to give good approximations can be applied here: centralized algorithms based on finding large feasible subsets [7], and randomized distributed algorithms [15].

The analysis is based on the following structural bound, which results in a good bound for the randomized distributed algorithm of [15]. Define $A(L) = \max_{R \subseteq L} \frac{a_R(R)}{|R|}$ [15].

Lemma 3: Let $\hat{L}$ be the set of links obtained from the fractional flows f^*, as indicated above. Then, $A(\hat{L}) = \mathcal{O}(n^4)$.

Proof: Let R be a subset of $\hat{L}$. First, observe that

$$a_R(R) = \sum_{v \in R} \sum_{w \in R} a_v(w) = \sum_{v \in R} \sum_{w \in R} \hat{b}_v(w) \ .$$

Consider a link w in R and let w' be the link in E from which w is a copy. Observe that $\hat{b}_v(w) = \hat{b}_v(w')$ (for any link v). Since w' has $c(w')$ copies in $\hat{L}$ (and at most that many in R),

$$\sum_{w \in R} \hat{b}_v(w) \leq \sum_{w' \in E} c(w') \hat{b}_v(w) = 10n^4 \sum_{w' \in E} \hat{b}_v(w) \leq \gamma \cdot 10n^4 \ .$$

Hence, $a_R(R) \leq \sum_{v \in R} \gamma \cdot 10n^4 = \gamma \cdot 10n^4 |R|$, as desired. ■

To bound greedy algorithms, the following property is useful.

Lemma 4: Any set L of links contains a feasible subset of size $\Omega(|L|/A(L))$.

Proof: Denote $b_v(L) = a_v(L) + a_L(v)$ and $b_v(w) = a_v(w) + a_w(v)$. Let $L' = \{v \in L : b_v(L) \leq 4 \cdot A(L)\}$ be the subset of links of small affectance to and from other links in L. By Markov's inequality, we can verify that $|L'| \geq |L|/2$.

We can now apply a randomized signal-strengthening argument of [16]. Pick a subset S of cardinality $|L'|/(8 \cdot A(L))$ uniformly at random from L', and let $R = \{v \in S : b_v(S) \leq 1\}$. Let $X_v = a_v(S)$ be the random variable denoting the affectance of link ℓ_v relative to S. Observe that $E[X_v] = \sum_{w \in L'} b_w(v) \cdot \Pr[\ell_v \in S] = b_v(L')/(8A(L)) \leq 1/2$. Thus, the probability that $b_v(S) \leq 1$ is at least $1/2$. It follows that the expected size of R is bounded below by

$$\begin{aligned} E[|R|] &= \sum_{v \in L'} \Pr[\ell_v \in S] \cdot \Pr[X_v \leq 1 | \ell_v \in S] \\ &\geq \sum_{v \in L'} \frac{1}{8A(L)} \cdot \frac{1}{2} = \frac{|L'|}{16A(L)} \geq \frac{|L|}{32A(L)} \ . \end{aligned}$$

Finally, observe that R is feasible, since $a_R(v) \leq b_v(R)$. ■

Theorem 5: The scheduling algorithms of [7] and [15] both lead to poly-time algorithms to compute a polynomial-length schedule that supports $\frac{f_i^*}{c_1 \log n}$.

Proof: We shall argue that either scheduling algorithm produces a schedule $\mathcal{S}$ with at most $T = \mathcal{O}(n^4 \log n)$ sets when applied to $\hat{L}$. Let c_1 be a constant such that $c_1 \geq T/(10n^4 \log n)$. We obtain a new flow vector $\hat{f}$ by scaling the flows vector f^* down by a factor of $c_1 \log n$. Then, for each link ℓ_w in E,

$$T \cdot \hat{f}(w) \leq 10n^4 f^*(w) = c(w) \ ,$$

which is the number of appearances of ℓ_w in $\hat{L}$. Thus, $\mathcal{S}$ supports the flows $\hat{f}$.

To establish the bound on the scheduling algorithms, we recall that Kesselheim and Vöcking [15] gave a randomized scheduling algorithm that uses $T = \mathcal{O}(A(L) \log |L|)$ slots, which in this case is $\mathcal{O}(n^4 \log n)$. Also, an algorithm was given in [7] to the problem of finding a maximum feasible subset of links. By scheduling the linkset by repeatedly applying that algorithm, this also results in a schedule of length $T = \mathcal{O}(n^4 \log n)$ using Lemma 4. ■

A. Arbitrary Power Control

Our schema allows us to handle also the case of arbitrary power control, using the capacity approximation algorithm of Kesselheim [8]. He defined the following interference function between two links ℓ_v and ℓ_w:

$$I(v,w) = \begin{cases} \min\left(1, \frac{d_v}{d(s_v, r_w)}\right) + \min\left(1, \frac{d_v}{d(s_w, r_v)}\right) & \text{if } d_v \leq d_w \\ 0 & \text{otherwise} \ . \end{cases} \quad (9)$$

It plays a role identical to that of $\hat{b}_v(w)$ for fixed power. He gave the following counterpart of Theorem 2:

Theorem 6: [8, Thm. 2.1] Let ℓ_v be a link and S be a set of links that is feasible under some power assignment, located on the plane[1] with $\alpha > 2$. Then, for a constant γ, $\sum_{w \in S} I(v,w) \leq \gamma$.

Our LP formulation is now identical, except that we replace the use of $\hat{b}_v(w)$ by $I(v,w)$. Also, we replace the fixed power algorithm of [7] by Kesselheim's capacity maximization algorithm, or the variant of [22] that allows for power to be limited.

The analysis is also the same with minor changes. We redefine $A(L) = \max_{R \subset L} \sum_{v \in R} \sum_{w \in R} I(v,w)/|R|$. Then, Lemma 3 holds unchanged. We slightly modify Lemma 4 by replacing $b_v(w)$ by $I(v,w)$ and introducing a constant parameter τ, so that it yields a set $S \subseteq L$ of size $\Omega(|L|/|A(L)|)$ such that $\sum_{w \in S} I(v,w) \leq \tau$. Now, by the following result of [8], S is feasible, as desired.

Theorem 7: [8, Thm. 3.1] Let S be a set of links such that $\sum_{w \in S} I(v,w) \leq \tau$, for each link ℓ_v in S. Then there is a power assignment that makes S feasible.

Finally, Theorem 5 carries over without change, giving:

Theorem 8: There is an $\mathcal{O}(\log n)$-approximation algorithm for maximum flow problem under arbitrary power control that uses a polynomial-size LP.

V. Extensions

As we have mentioned, the advantage of our formulation, apart from its simplicity, is that it is easy extend the LP to handle other linear constraints.

For example, consider the problem of finding a flow as before, but with the additional constraint that the average power usage $\sum f(w) P_w$ is bounded by some value P. Since the average power is a linear function, the linear constraint can be added to the LP without any substantial changes to the analysis.

A. MIMO Flow

Let O_i be the outgoing links from $\hat{s}_i$ and I_i be the incoming links in to $\hat{t}_i$. In the MIMO case, the final schedule will contain two collections A and B. A will contain a schedule of links from $\cup_i (O_i \cup I_i)$, and B will contain a schedule of links from the rest of the graph. Interference alignment is used for the transmissions over the links in O_i and I_i.

[1]The result extends to *fading metrics*, which are doubling metrics, where α is greater than the doubling constant.

The modified LP becomes:

$$\max \sum_i f_i \tag{10}$$

$$\text{s.t.} \sum_{w \in E: r_w = a} f_i(w) = \sum_{v \in E: s_v = a} f_i(v) \quad \forall a \in V \setminus \{\hat{s}_i, \hat{t}_i\}, \forall i \tag{11}$$

$$\sum_{w \notin O_i \cup I_i} \hat{b}_v(w) f(w) \leq \gamma \ \ \forall v \notin O_i \cup I_i \tag{12}$$

$$f_i(w) \geq \frac{1}{10n^4} \ \ \forall w \in E_i \setminus (O_i \cup I_i), \forall i \tag{13}$$

$$\sum_i f_i(w) \leq 1 \ \ \forall w \in E \tag{14}$$

$$\sum_{w \in O_i} f_i(w) \leq d/k \ \ \forall i \tag{15}$$

$$\sum_{w \in I_i} f_i(w) \leq d/k \ \ \forall i \tag{16}$$

The changes in Eqn. 12 reflect the fact that links in O_i and I_i do not have to be charged for interference. The MIMO requirements are reflected in (15) and (16).

The set B will be as before, containing the schedule of links $w \notin O_i \cup I_i$ derived by [6], [15].

The set A will be constructed for links in as follows. Set $c(w) = \lceil 10n^4 \cdot f(w) \rceil$ as before. Then each $c(w)$ is an integer between 1 and $10n^4 d/k$, for a total of $\mathcal{O}(n^4 d)$ links. Now MIMO allows d of these links to be scheduled together, making for $\mathcal{O}(n^4)$ slots. The total number of slots including A and B remains asymptotically the same, and thus the final result is same as well.

B. Fairness

One of the problems with maximizing the total flow in a multicommodity flow is that the linear problem will often just maximize the flow for a single commodity and disregard the other. To ensure fairness between the commodities we can add linear constraints to maximize the minimum flow over all the commodities, thus making sure that the flow is fairly distributed. The objective in Eq. 10 becomes:

$$\max \quad z \tag{17}$$

$$\text{s.t.} \quad z \leq f_i \ \ \forall i \tag{18}$$

while Eq. 11 - 16 remain the same.

Constraint (18) enforces a lower limit on the flow for each source-sink pair, while the objective function in (17) tries to maximize this lower limit, i.e. to maximize the value of the smallest flow between any source-sink pair.

The analysis and the theoretical bound of $\log n$ is still valid for the modified objective function. It is easy to see that the new objective function in (17) and Constraint (18) are equivalent to keeping the same objective function as before, $\max \sum_i f_i$, and adding the linear constraints $F_{\min} \leq f_i$ for every $i = 1, \ldots, k$, where $F_{\min}$ is some constant, and then do a binary search for the best value for $F_{\min}$. In this case we only add a linear constraint and we have already shown that we can add any linear constraint to the formulation.

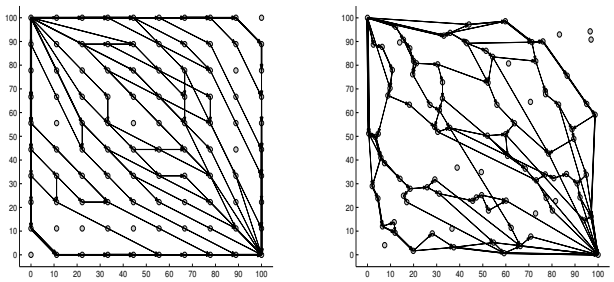

Fig. 1. An illustration of the LP solution for uniform power on grid and random topology of 100 nodes. The source node is in the upper left corner and the sink node in the bottom right corner. The thickness of the edges denotes the value in the LP solution. The throughput values are 0.595 and 0.486.

Instead of having the same flow value for each source-sink pair, we can also add a scaling parameter in Constraint 18 so that we ensure that a source-sink pair will not receive flow that is less than some fraction of the flow for some other source-sink pair. We can also use similar constraints to set any type of proportional or fixed limits (both lower and upper limits) for each source-sink pair in a multicommodity flow or for any given set of flows and edges.

VI. Simulations

To analyze the performance and the throughput of the flow algorithm, we did extensive simulations. The code was written in Matlab R2012b and the LP was solved using the Gurobi 5.0.1 solver. We focus on instances where the nodes are contained in a square with edgelength 100. To determine which links are available to the LP, we look at the links whose length is at most some fraction of the edgelength, we usually used only links that are at most 0.4-0.5 of the edgelength, i.e. links of length at most 40-50. Unless otherwise stated, we set the ambient noise $N = 0$.

To calculate the throughput, we count the number of messages that are successfully received by the sink and divide by the total number of timeslots that the scheduling algorithm required to transmit all the messages. We assume that the length of each timeslot is such that a transmitter can send exactly one message during a single timeslot.

A. Practical modifications

The analysis of the LP requires that every link has a minimum flow of $\frac{1}{10n^4}$ in order to avoid complications that arise if the solution has extremely small positive values. In practice we can remove this requirement and simply ignore extremely small values.

Using a fixed scaling factor of $10n^4$ when scaling the solution has some practical problems. For most instances the number of copies for the links increases quickly as n grows. Even for realtively small values of n, such as $n = 100$, we would frequently end up with over 150 million copies of the same link. In order to make the algorithm in [15] practical, we use a scaling factor of $\max(\frac{1}{f_{\min}}, \frac{1000}{f_{\max}})$ where $f_{\min}$ is the smallest flow value greater than $\frac{1}{10n^4}$ and $f_{\max}$ is the largest flow on any link. The reasoning behind the scaling factor is

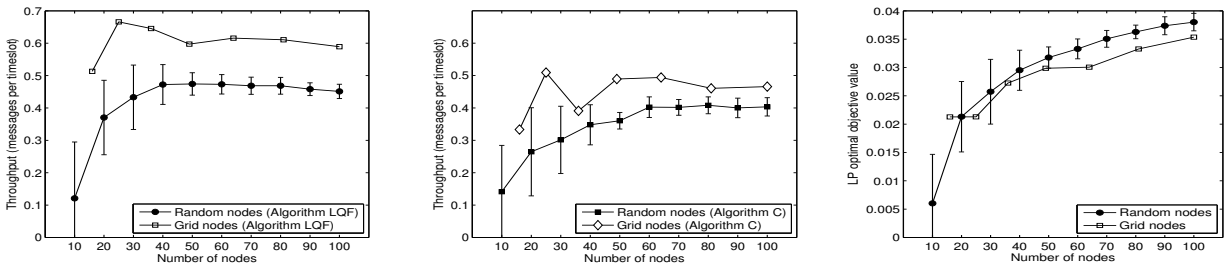

Fig. 2. Throughput and optimal LP objective value for random and grid based instances, as a function of the number of nodes. The first graphs shows the throughput using the LQF algorithm on both random and grid based instances, the second graph shows the throughput using algorithm C on the same instances, and the third graph shows the optimal objective values for both the randomized and grid based instances.

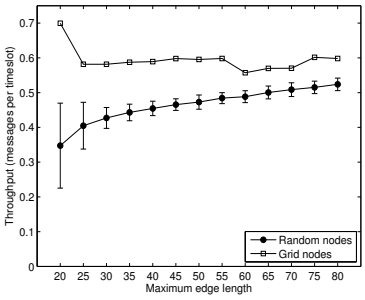

Fig. 3. The average throughput as a function of the maximum edge length. The nodes are either distributed uniformly at random or ordered in a grid.

that we want the smallest flow value to scale up to at least one, so that the corresponding link is included in at least one set. We tried to simply use a scaling factor of $\frac{1}{f_{\min}}$ which worked well in most instances. However, for some instances where the solution values are close, i.e. $|f_{\max} - f_{\min}| \leq \epsilon$ for small ϵ, the resulting requests would only have a few copies of each link, which would lead to inaccurate throughput measurements. To measure the throughput more accurately, we decided that the links should be scaled such that the smallest viable link should have at least one copy, and also that the link with the largest number of messages should have at least 1000 copies.

B. Scheduling algorithm

As mentioned in Section IV, once we have an optimal solution to the flow LP, we use a scheduling algorithm to schedule the links. We looked at three different algorithms to schedule the links. The first algorithm, which is the one we use in our theoretical analysis, is the $\mathcal{O}(\log n)$ algorithm that was introduced in [21]. We use a variation of that algorithm from [7], called algorithm C. The idea behind algorithm C is to repeatedly find good solutions to the capacity problem. The capacity algorithm is a constant factor approximation, and by repeating algorithm C until all links have been scheduled we get a $\mathcal{O}(\log n)$ approximation to the scheduling problem. Another option is to use the Longest-Queue-First (LQF) algorithm. The main difference between algorithm C and LQF is that instead of sorting the links by length, LQF sorts the links by the number of requests on the links. The LQF algorithm does not have the same strong approximation ratio as algorithm C, and there are instances where the approximation ratio of LQF is as high as $\mathcal{O}(n)$ [23]. However, LQF has proven to be efficient in random topologies where each link has multiple transmission requests [24], and in our simulations we noticed that the LQF algorithm usually performed somewhat better than algorithm C. The third option is to use the randomized distributed algorithm of [15] which is also a $\mathcal{O}(\log n)$ approximation. However, even after trying out various performance enhancing modifications, we did not manage to get satisfactory throughputs using the randomized distributed algorithm. We therefore have a choice between algorithm C and LQF. Algorithm C is somewhat faster but LQF usually resulted in a slightly better throughput, as seen in Figure 2. We therefore decided to use the LQF algorithm in our simulations.

C. Simulation results - throughput for a single source/sink pair

Figure 1 shows examples of the LP solution for two examples with 100 nodes and a single source-sink pair. The nodes are ordered on a grid in the first example and scattered randomly in the second. All possible links of length at most 50 are included in the instance. An edge is drawn for all links that have a positive value in the optimal LP solution. Both instances have a single source-sink pair and the source node is in the upper left corner and the sink node in the bottom right corner. The nodes use uniform power for the transmissions with $\alpha = 2.1$, $\beta = 1.0$ and $N = 0$. The thickness of the edges denotes the value in the LP solution, the thicker the link, the higher the solution value for that link. The throughput for the grid based instance is 0.595 and 0.486 for the random instance.

The behavior of the flow algorithm, as shown in Figure 1 corresponds well with our intuition of how the flow should work, the flow quickly spreads out from the source in order to utilize links better, and then combines again at the sink node.

The LP solver managed to solve simple problems almost instantly, and problems with 100 nodes and up to 10000 links were solved in 1-3 minutes on a MacBook Air with a 1.8 GHz Intel Core i7 CPU and 4 GB of memory. However, the running time does increase quite rapidly when the number of links increases, since the LP formulation creates a matrix that is very dense, and therefore somewhat difficult to solve. For example, with a grid-based instance of 100 nodes and 8011 links, the solver started with a matrix of 16122 rows, 8012 columns and 32770881 nonzero entries. After presolve, the matrix had 8109 rows, 8010 columns and 32762747 nonzeros. However, if we make sure that we have relatively few links, we can easily scale up n and solve examples with hundreds or even thousand nodes in only a few minutes.

To solve the LP we need to find a value for γ in Constraint 6. It is not clear exactly what the value of γ should be since Theorem 2 and the original Theorem in [6] only specify that the total symmetric affectance is bounded by a constant, without giving any specific values for said constant. By changing the value of γ, we get changes in the optimal LP solution. If γ is too low, the constraints in (8) that limit the total flow on a single link become useless, and if γ is too large, the affectance restrictions in (6) become unused. After doing multiple experiments with various values of γ, we felt that having γ fairly small would give us much more interesting results, i.e. when the affectance bounds in (6) would have more impact than the more simple maximum bound on the flow on a single link. For low values of γ there was not much difference between the solutions, so we decided to use $\gamma = 1$.

The graphs in Figure 2 show the throughput and the optimal LP objective value for both random topology and grid based topology with increasing number of nodes, where the throughput is calculated using both algorithm C and the LQF algorithm. The parameters are the same as for Figure 1 with $\alpha = 2.1$, $\beta = 1.0$ and $N = 0$. We include all links of length at most 40. For the randomized instances, the results show the average value over 100 instances as well as error bars denoting one standard deviation. Since there is no random element in the grid based instances, we only need a single run for those results. As the number of nodes increases, the optimal LP objective value increases, as was expected. However, the scheduling algorithm does not manage to show the same improvement with increasing number of nodes. One of the reasons for this behavior is that as the number of nodes grows, the optimal LP solutions often include multiple hops instead of fewer hops over longer links, due to the fact that shorter links are stronger than longer links. In our experiments with the distributed randomized scheduling $\log(n)$ algorithm from [15], the throughput would drop significantly when we increased the number of hops between the source and the sink, due to the increased chances of interference from multiple links transmitting at the same time. The greedy algorithms, C and LQF, do not show as much degrade in throughput, but it is still noticeable. Since the grid based instances use square grids, we only have solutions for $16, 25, 36, \ldots, 100$ nodes. For the random topology, the standard deviation for the instances with few number of nodes is much higher than when the number of nodes increases. The reason is that often there was no path from the source to the sink so the throughput would be equal to zero. When the number of nodes increases, the chance of having at least one path from the source to the sink increases and the standard deviation becomes much less.

Figure 2 shows that the LQF algorithm gives a better throughput than algorithm C, even though algorithm C is both faster and has a better worst case approximation ratio. Because of the superior throughput values, we will use LQF to calculate throughput unless noted otherwise.

Figure 3 shows the throughput for both random instances and for grids when we increase the maximum lengths of the links. All parameters are as before with 100 nodes, however we now include all links of length at most $20, 25, 30, \ldots, 80$. As the number of links grow, the average throughput increases when the nodes are distributed at random. However, there is not the same behavior when the links are ordered in a grid. The optimal objective value increases when we increase the number of links, however, the throughput does not have the same monotone behavior. Since we're using a greedy algorithm to calculate the throughput, some instances just fit the algorithms better than others so we can get higher throughput even though the instance contains fewer links. This is especially noticeable when we increase the number of links by increasing the maximum link length from 20 to 25, which leads to a drop in throughput from 0.70 down to 0.58. The reason is that for the instance with maximum link length of 20, the flow is concentrated down three paths, while the grid based instance with link lengths up to 25 focuses the flow on four paths. Due to interference, it's easier to greedily schedule the links when the paths are fewer and further apart.

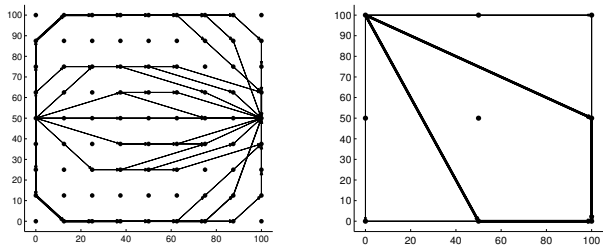

Fig. 4. Two examples for using MIMO nodes for the source and sink nodes. The first graph has 81 nodes and the source and sink nodes are on the sides of the rectangular area, while the second example uses only 9 nodes and the source and the sink are in the top left and bottom right corners.

TABLE I. THROUGHPUT VALUES FOR THE TWO GRAPH EXAMPLES IN FIGURE 4, USING MULTIPLE INPUT AND OUTPUT ANTENNAS.

Antennas	**Example 1**	**Example 2**
1	0.690	0.654
2	0.724	1.000
3	0.777	1.200
4	0.799	1.306
5	0.812	1.306
6	0.819	1.306
7	0.827	1.306

D. Simulation results - MIMO

By using MIMO nodes for the source and sink nodes, we can decrease the bottleneck at those nodes. Figure 4 shows two instances where we look at the effect of increasing the number of antennas at the source and sink nodes. The first instance uses 81 nodes and the source node is in the middle of the left side, while the sink node is on the right side. The second example uses only 9 nodes and the source and sink nodes are in the opposite corners of the rectangular area. Table I shows the throughput values for different number of antennas. We see that once the number of antennas is equal to the number of nodes that are easily reachable from the MIMO nodes, adding more antennas does not make any significant improvements. The number of nodes in the system and the path lengths between the source and the sink have a large impact of how effective it is to add more antennas. When we have few nodes and short paths between the source and the sink, the bottleneck at those nodes becomes more dominant, while for longer paths, the bottlenecks at the source and the sink are less significant. For the smaller example with 9 nodes, we get more than 50%

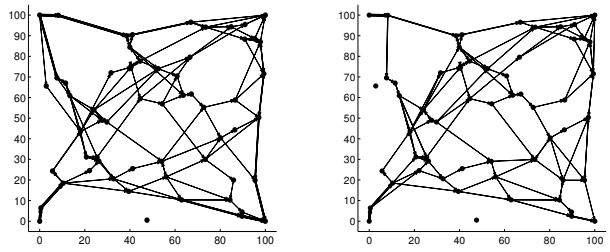

Fig. 5. An illustration of the optimal LP solution for a single instance of multicommodity flow. The solution shown in the figure on top is obtained without using any fairness measure, so the flow from top left to bottom right dominates the flow from bottom left to top right. The second figure contains a fairness criteria so both flows are equal.

increase in throughput by going from a single antenna to two antennas, and more than double the throughput by going to 4 antennas. However, for the larger example with 81 nodes, the improvement is much less, we only get around 5% increase with the first additional antenna and a total increase of 20% by going from a single antenna up to 7 antennas at both the source and the sink nodes.

E. Simulation results - Multiple source/sink pairs

Figure 5 shows an instance where we have a multicommodity flow, i.e. multiple source-sink pairs. The first source-sink pair is transmitting information from the top left corner to the bottom right corner, while the second source-sink pair is transmitting from the bottom left corner to the top right corner. The instances have 60 nodes in an area of size 100×100, with $\alpha = 2.1$, $\beta = 1.0$ and $N = 0$, and we include all links of length at most 30. If we solve the LP solution we find that the first commodity, from top left to bottom right, gets a much higher throughput that the other, the objective value for the first commodity is 0.2344 while the second one only has an optimal objective value of 0.1287. These values translate into throughputs of value 0.32 and 0.17 for the two commodities. If we use the LP with the fairness objective (17) and Constraint (18), we get an equal optimal objective value of 0.1798 for both commodities, and a throughput of 0.24 for each commodity. There is not much visual difference between the solutions, and the total throughput for both cases is almost the same. The difference is the fairness, in the first one, the source-sink pair from top left to bottom right dominates the available resources, while the second instance has a fair distribution of the available resources.

VII. Conclusions

In this paper we have given a general and efficient LP based framework for the maximum flow problem in wireless networks. Our framework holds for any flow problem with linear constraints. Among the constraints that the framework can handle are MIMO nodes for the source and the sink, power limitations and bandwidth restrictions. The framework can also handle fairness measures between multiple source-sink pairs in a multicommodity flow. The framework matches the best known approximation bound of $\mathcal{O}(\log n)$ for the basic maximum flow in wireless networks, but uses a simpler and more intuitive analysis. As far as we know, this is the first practical algorithm with an approximation bound that depends only on the number of nodes for this problem.

References

[1] V. R. Cadambe and S. A. Jafar, "Interference alignment and spatial degrees of freedom for the k user interference channel," in *IEEE ICC*, 2008, pp. 971–975.

[2] K. C.-J. Lin, S. Gollakota, and D. Katabi, "Random access heterogeneous MIMO networks," in *SIGCOMM*, 2011.

[3] D. Chafekar, V. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan, "Approximation algorithms for computing capacity of wireless networks with SINR constraints," in *Infocom*, 2008.

[4] D. Chafekar, V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Capacity of wireless networks under sinr interference constraints," *Wireless Networks*, vol. 17, no. 7, pp. 1605–1624, 2011.

[5] P.-J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang, "Wireless link scheduling under physical interference model," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 838–845.

[6] M. M. Halldórsson, S. Holzer, P. Mitra, and R. Wattenhofer., "The power of non-uniform wireless power," in *SODA*, 2013.

[7] M. M. Halldórsson and P. Mitra, "Wireless Capacity with Oblivious Power in General Metrics," in *SODA*, 2011.

[8] T. Kesselheim, "A Constant-Factor Approximation for Wireless Capacity Maximization with Power Control in the SINR Model," in *SODA*, 2011.

[9] P.-J. Wan, "Multiflows in multihop wireless networks," in *Mobihoc*, 2009, pp. 85–94.

[10] G. Even, Y. Matsri, and M. Medina, "Multi-hop routing and scheduling in wireless networks in the SINR model," in *Algosensors*, 2012, pp. 202–214.

[11] Y. Shi, Y. T. Hou, S. Kompella, and H. D. Sherali, "Maximizing capacity in multihop cognitive radio networks under the sinr model," *IEEE Trans. Mob. Comput.*, vol. 10, no. 7, pp. 954–967, 2011.

[12] D. Chafekar, V. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan, "Cross-layer latency minimization for wireless networks using SINR constraints," in *Mobihoc*, 2007.

[13] A. Fanghänel, T. Kesselheim, and B. Vöcking, "Improved algorithms for latency minimization in wireless networks," in *ICALP*, July 2009, pp. 447–458.

[14] O. Goussevskaia, Y. A. Pignolet, and R. Wattenhofer, "Efficiency of wireless networks: Approximation algorithms for the physical interference model," *Foundations and Trends in Networking*, vol. 4, no. 3, pp. 313–420, 2010.

[15] T. Kesselheim and B. Vöcking, "Distributed contention resolution in wireless networks," in *DISC*, August 2010, pp. 163–178.

[16] A. Fanghänel, T. Kesselheim, H. Räcke, and B. Vöcking, "Oblivious interference scheduling," in *PODC*, August 2009, pp. 220–229.

[17] T. Moscibroda and R. Wattenhofer, "The Complexity of Connectivity in Wireless Networks," in *INFOCOM*, 2006.

[18] M. M. Halldórsson, "Wireless scheduling with power control," *ACM Transactions on Algorithms*, vol. 9, no. 1, p. 7, December 2012.

[19] T. Tonoyan, "On the capacity of oblivious powers," in *Algosensors*, 2011, pp. 225–237.

[20] O. Goussevskaia, M. M. Halldórsson, R. Wattenhofer, and E. Welzl, "Capacity of Arbitrary Wireless Networks," in *INFOCOM*, April 2009, pp. 1872–1880.

[21] M. M. Halldórsson and R. Wattenhofer, "Wireless Communication is in APX," in *ICALP*, July 2009, pp. 525–536.

[22] T. Kesselheim, "Approximation algorithms for wireless link scheduling with flexible data rates," in *ESA*, 2012, pp. 659–670.

[23] L. B. Le, E. Modiano, C. Joo, and N. B. Shroff, "Longest-queue-first scheduling under SINR interference model," in *MobiHoc*, 2010.

[24] E. I. Ásgeirsson, M. M. Halldórsson, and P. Mitra, "A fully distributed algorithm for throughput performance in wireless networks," in *46th Annual Conference on Information Sciences and Systems (CISS)*, 2012.