Maximum Bipartite Flow in Networks with Adaptive Channel Width*

Yossi Azar† Aleksander Mądry‡ Thomas Moscibroda§
Debmalya Panigrahi¶ Aravind Srinivasan‖

Abstract

Traditionally, network optimization problems assume that each link in the network has a fixed capacity. Recent research in wireless networking has shown that it is possible to design networks where the capacity of the links can be changed *adaptively* to suit the needs of specific applications. In particular, one gets a choice of having a *few* high capacity outgoing links or *many* low capacity ones at any node of the network. This motivates us to have a re-look at classical network optimization problems and design algorithms to solve them in this new framework. In particular, we consider the problem of *maximum bipartite flow*, which has been studied extensively in the fixed-capacity network model. One of the motivations for studying this problem arises from the need to maximize the throughput of an infrastructure wireless network comprising base-stations (one set of vertices in the bipartition) and clients (the other set of vertices in the bipartition). We show that this problem has a significantly different combinatorial structure in this new network model from the fixed-capacity one. While there are several polynomial time algorithms for the maximum bipartite flow problem in traditional networks, we show that the problem is NP-hard in the new model. In fact, our proof extends to showing that

*A preliminary version of this paper appeared in the Proceedings of ICALP, 2009.

†Tel Aviv University, Tel Aviv 69978, Israel. Part of this work was done while visiting Microsoft Research, Redmond, WA 98052. Research supported in part by the Israeli Science Foundation (grant No. 1404/10). Email: azar@tau.ac.il.

‡Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. Research supported by NSF contract CCF-0829878 and by ONR grant N00014-05-1-0148. Email: madry@mit.edu.

§Microsoft Research, Redmond, WA 98052. Email: moscitho@microsoft.com.

¶**Corresponding Author.** Contact Address: 32 Vassar Street, G-696, The Stata Center, Cambridge MA 02139. Contact Phone: +1 (617) 2585791. Contact Fax: +1 (617) 2533480. Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. Part of this work was done while the author was an intern at Microsoft Research, Redmond, WA 98052. Research supported in part by NSF contract CCF-0635286. Email: debmalya@mit.edu.

‖Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Part of this work was done while visiting Microsoft Research, Redmond. Research supported in part by NSF ITR Award CNS-0426683, NSF Award CNS-0626636, and NSF Award CNS-1010789. Email: srin@cs.umd.edu.

the problem is APX-hard. We complement our lower bound by giving two algorithms for solving the problem approximately. The first algorithm is deterministic and achieves an approximation factor of $O(\log N)$, where N is the number of nodes in the network, while the second algorithm is randomized and achieves an approximation factor of $\frac{e}{e-1}$.

Keywords: Graph algorithm; maximum flow; linear program rounding; wireless networks; adaptive channel width.

1 Introduction

Wireless networking is in the midst of a major paradigm shift. At the core of this shift is a more flexible interpretation and use of the *spectrum* as the medium of communication. Existing infrastructure-based wireless systems (such as Wi-Fi) partition the available spectrum into fixed channels of equal width (in Wi-Fi, every channel has a channel-width of 20 Mhz), and every node picks one or more (if it is equipped with more than one radio) of these channels to transmit on. In contrast, recent work [5] shows that it is beneficial (and feasible) to vary the width of a wireless communication channel *adaptively*, depending on the needs of particular applications, as well as other factors. A key ramification of adaptively changing channel widths is the trade-off between *throughput* and *range* of communication channels. It has been demonstrated in [5, 21] that throughput increases monotonically with channel width (as predicted by Shannon's capacity formula [23]).[1] However, transmitting on a wider channel reduces the transmission range, thereby disconnecting receivers that are far from the transmitter. This can be alternatively viewed as a link having a threshold channel width, beyond which it ceases to exist. Thus, while choosing channel widths, there is a tension between achievable throughput on the resulting channel and the range of transmissions on the channel. This leads to a new suite of network optimization problems, which are structurally different from their counterparts in fixed-capacity networks.

We focus on a representative problem from this class, that of maximizing throughput in a network comprising base-stations and clients. In this problem, each client is connected to a subset of base-stations via communication links. We are required to choose the channel width of each communication link, with the restriction that all communication links originating at a base-station must have identical channel width. Since throughput is a monotonically increasing function of channel width, selecting a throughput for a base-station uniquely determines the channel width of its outgoing communication links. Further, the threshold channel width of each link (beyond which it does not exist) can be translated into a threshold throughput. Thus, our problem now is to select a throughput for each base-station; all outgoing links which have a higher threshold get channel capacity equal to the selected throughput, while all outgoing links with a lower threshold disappear (i.e. get channel capacity equal to 0).

[1] In fact, it has been shown that throughput is (roughly) linearly related to channel width.

Problem Definition. We are given a set of base-stations $\mathcal{B}$ and a set of clients $\mathcal{C}$ with $|\mathcal{B}| = n$ and $|\mathcal{C}| = m$. Each base-station $B \in \mathcal{B}$ has a *budget* $\beta(B)$, which is the total capacity that the base-station can deliver to its clients. On the other hand, each client $C \in \mathcal{C}$ has a *demand* $\alpha(C)$, which is the total bandwidth it would like to be allocated from all the base-stations together.

For each base-station and client pair (henceforth, called a *base-client pair*) (B, C), there is a *critical capacity* $\eta(B, C)$, which corresponds to the maximum channel width of a link from B to C. To each base station $B \in \mathcal{B}$, the algorithm assigns a *threshold* $\tau(B)$ that determines the *capacity* of a link (B, C) (denoted by $\psi_\tau(B, C)$) as follows

$$\psi_\tau(B, C) := \begin{cases} \tau(B) & , \tau(B) \leq \eta(B, C) \\ 0 & , \text{otherwise} \end{cases}$$

Once the capacities of all links have been fixed, we want to find a *flow* $f(B, C)$ for each link (B, C) such that neither any link capacity is violated (*capacity constraint*), i.e.

$$f(B, C) \leq \psi_\tau(B, C),$$

nor any base-station budget is violated (*budget constraint*), i.e.

$$\sum_{C \in \mathcal{C}} f(B, C) \leq \beta(B).$$

The goal is to find the threshold assignment τ, and corresponding flow f that maximizes the sum of *satisfied demands* of all the clients, where the satisfied demand $\alpha_{\tau,f}(C)$ of a client C is given by

$$\alpha_{\tau,f}(C) = \min\big(\sum_{B \in \mathcal{B}} f(B, C), \alpha(C)\big).$$

Note that given any τ and f, there always exists a flow f' which satisfies the budget and capacity constraints, achieves the same value of total satisfied demand and additionally satisfies the following *demand constraints*,

$$\sum_{B \in \mathcal{B}} f(B, C) \leq \alpha(C)$$

As a result, we will focus on flows that obey demand constraints along with budget and capacity constraints.

The benefit of this assumption is that our problem now corresponds to the maximum bipartite flow problem in networks with adaptive channel width. Recall that in this flow problem, we have two sets of nodes X and Y with edges directed from X to Y, along with a supersource s and a supersink t. To draw the correspondence, let X be the set of base-stations and Y the set of clients. The edge from s to any $x \in X$ (called a *budget arc*) has capacity $\beta(x)$, that from any $x \in X$ to any $y \in Y$ has critical capacity $\eta(x, y)$ and that from any $y \in Y$ to t (called a *demand arc*) has capacity $\alpha(y)$ (refer to Figure 1). We call

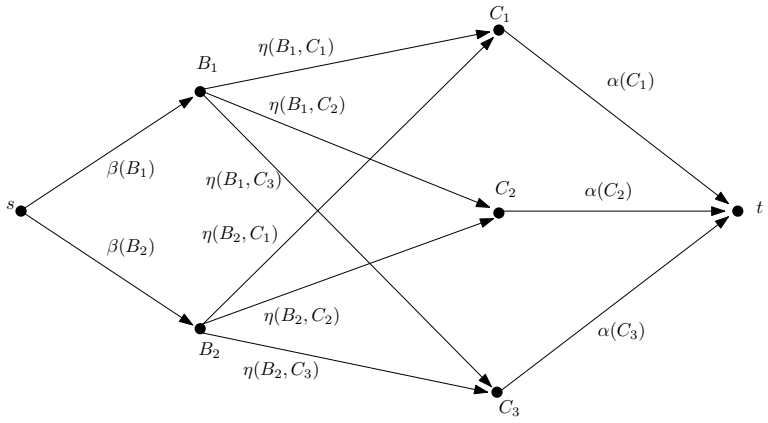

Figure 1: The *augmentation graph* corresponding to an instance of the problem.

this graph the *augmentation graph* of the given problem instance). Our task is to choose threshold values (i.e. the function τ) for vertices in X; this fixes the capacities of all the arcs from X to Y. Our goal is to choose τ so that the maximum flow in the resulting capacitated network is maximized.

Related Work. Classically, the maximum bipartite flow problem has been solved as a special case of the more general maximum flow problem on arbitrary graphs. Suppose the input graph $G = (V, E)$ has maximum flow of c from the source to the sink. Ford and Fulkerson gave the first algorithm for the maximum flow problem in the 1950s, which had a running time of $O(|E|c)$ [10]. Since then, several algorithms have been developed with better time bounds [7, 8, 9, 12] finally culminating in an $\tilde{O}(|E| \min(|E|^{1/2}, |V|^{2/3}) \log c)$ algorithm due to Goldberg and Rao [11], which is currently the fastest known deterministic algorithm for maximum flow. It may be noted here that a substantial amount of work has also been done for developing randomized algorithms for maximum flow [15, 17, 16, 18], but these algorithms apply only to undirected networks. On the other hand, the maximum bipartite flow problem with unit capacities (which is equivalent to *maximum bipartite matching*) can be solved in $O(|E|\sqrt{|V|})$ time [14].

To summarize the above discussion, the maximum bipartite flow problem in directed graphs with capacitated edges is solvable in polynomial time; however, there is no algorithm which solves this problem faster than in general directed graphs. As we will see, this is in sharp contrast to what we observe in networks with adaptive channel width. In such networks, the maximum bipartite flow problem is NP-hard (in fact, it is APX-hard); further, we give a randomized approximation algorithm achieving an approximation factor of $\frac{e}{e-1}$ which does not appear to extend easily to general directed networks.

We also briefly mention a related class of well-studied problems, namely *unsplittable* flow problems. In these problems, typically there are one or more

pairs of source and sink vertices with specific demands, and the goal is to connect the source-sink pairs using paths such that the satisfied demand is maximized while not violating any capacity constraint. These problems are typically NP-hard, and several variants have been studied extensively [19, 24, 13, 3, 4, 6]. Interestingly, though we have a single source and a single sink, and flow is allowed to re-distribute arbitrarily at a node, the techniques we use to give an approximation algorithm for our problem bear similarities with the techniques usually used for solving unsplittable flow problems. Specifically, both problems use a suitable linear programming relaxation which is then rounded ensuring that certain cuts are large in the rounded solution.

Our Results. Our first claim is that the maximum bipartite flow problem in networks with adaptive channel width is APX-hard, i.e., it is unlikely that a polynomial-time algorithm can approximate the problem within a certain constant factor. Specifically, we describe an L-reduction from the APX-hard Maximum Bounded 3-Dimensional Matching problem (Max-3DM) to the channel width assignment problem. As mentioned above, this is in contrast to maximum bipartite flow on fixed-capacity networks, where the problem is solvable in polynomial time.

Theorem 1. *The maximum bipartite flow problem in networks with adaptive channel width is APX-hard.*

Our next contribution is a greedy combinatorial algorithm which achieves an approximation factor of $O(\log N)$, where $N = \max(m, n)$. The algorithm first categorizes links according to their critical capacity in geometrically spaced intervals. Now, observe that for any interval, we can set the assigned capacities at the nodes such that all the links in that interval have capacity equal to their critical capacities (while all other links have potentially no capacity at all). The algorithm needs to decide which interval to choose. For this purpose, a maximum flow algorithm is run on the entire graph, assuming that each link has capacity equal to its critical capacity. This outputs a flow on each link. The algorithm greedily chooses the interval which carries the greatest amount of flow on its links.

Finally, our main result is a randomized algorithm for this problem.

Theorem 2. *There is a randomized polynomial-time algorithm for the maximum bipartite flow problem in networks with adaptive channel width that has an expected approximation factor of* $\frac{e}{e-1}$.

Our algorithm uses a linear programming relaxation of the problem. Recall that the celebrated Menger's theorem implies that maximum flow from s to t equals the minimum $s - t$ cut. So, an algorithm for the problem should aim to choose assigned capacities so as to maximize the minimum $s - t$ cut in the resulting capacitated network. Now, let us consider any linear programming formulation of the problem; such a fractional linear program can be interpreted as a polytope, where its optimal solution is a convex combination of the vertices

of the polytope. Each vertex of the polytope represents a particular choice of assigned capacities and therefore, a particular capacitated graph (call them *vertex graphs*); these correspond to the integral solutions we will round our solution to. The natural linear program that we consider first simply ensures that for each cut, the convex combination of the values of the cut in the vertex graphs is large. However, since each vertex graph may have a different minimum $s-t$ cut, this does not guarantee that sizes of these minimum $s-t$ cuts are large. In fact, this linear program has an integrality gap of $\Omega(\log N/\log\log N)$. To overcome this problem, we design a more sophisticated linear program and a corresponding randomized rounding technique that ensures that the minimal $s-t$ cuts in the vertex graphs are large.

Roadmap. Section 2 describes a reduction from 3-dimensional matching to show APX-hardness of our problem. Section 3 contains both the deterministic and randomized algorithms for our problem. Finally, we conclude in section 4 with some related open problems.

2 APX-hardness

We show that the maximum bipartite flow problem in networks with adaptive channel width is APX-hard, i.e., it is unlikely that a polynomial-time algorithm can approximate the problem within a certain constant factor. Specifically, we describe an L-reduction from the APX-hard Maximum Bounded 3-Dimensional Matching problem (Max-3DM) to the channel width assignment problem. Our construction draws on some ideas from Lenstra, Shmoys & Tardos [20] and from [2]. The Max-3DM problem is defined as follows.

> **Instance:** Three disjoint sets $A=\{a_1,\ldots,a_p\}$, $B=\{b_1,\ldots,b_p\}$, and $C=\{c_1,\ldots,c_p\}$, together with a subset of triples $T\subseteq A\times B\times C$. Any element in A, B, C occurs in one, two, or three triples in T; note that this implies $p\leq|T|\leq 3p$.
>
> **Goal:** Find a subset $T'\subseteq T$ of maximum cardinality such that no two triples of T' agree in any coordinate.
>
> **Measure:** The measure of a feasible solution T' is the cardinality of T'.

Petrank [22] has shown that Max-3DM is APX-hard even if one only allows instances where the optimal solution consists of $p=|A|=|B|=|C|$ triples; in the following we will only consider this additionally restricted version of Max-3DM.

For the L-reduction we specify two functions Γ_1 and Γ_2 as follows. Γ_1 maps each instance I of Max-3DM into an instance of the channel-width assignment problem $R(I)$, and Γ_2 maps a feasible solution of $R(I)$ back to a feasible solution of I. Specifically, any instance I of Max-3DM is mapped by Γ_1 into an instance $R(I)$ that contains $n=|T|$ base-stations and $m=3p+1$ clients as follows:

- For every triple $T_i \in T$, there is a corresponding base-station B_i with a budget $\beta(B_i) = 4$.
- For elements a_j, b_j, and c_j for every $j = 1, \ldots, p$, there are corresponding clients C_j^a, C_j^b, and C_j^c, each with demand 1.
- Additionally, there is one "large client" C_z with demand $\alpha(C_z) = 4(|T| - p) + p = 4|T| - 3p$.
- Each base-station B_i has one link to each of the three clients corresponding to the elements that are contained the triple $T_i = (a_j, b_k, c_l)$: Links (B_i, C_j^a), (B_i, C_k^b), and (B_i, C_l^c) with critical capacity 1, i.e., $\eta(B_i, C_j^a) = \eta(B_i, C_k^b) = \eta(B_i, C_l^c) = 1$. In addition, it has a fourth link (B_i, C_z) with critical capacity $\eta(B_i, C_z) = 4$. No other client is connected to this base station, i.e., $\eta(B_i, C) = 0$ for any other client C.

Note that the total demand is equal to the total budget. This completes the description of the instance $R(I)$. Since we only consider instances of Max-3DM where the optimal solution consists of p triples, we have $\textsc{Opt}(I) = p$. Now consider the following assignment for instance $R(I)$: For each triple $T_i = (a_j, b_k, c_l)$ in the optimal solution to I, we select a capacity threshold $\tau(B_i) = 1$ and send a flow of 1 to the element clients (i.e., $f(B_i, C_j^a) = f(B_i, C_k^b) = f(B_i, C_l^c) = 1$), and $f(B_i, C_z) = 1$ to C_z. All the other $|T| - p$ base-stations choose a capacity threshold of $\tau(B_i) = 4$ and send flow of $f(B_i, C_z) = 4$ to z. These base-stations cannot send any flow to the clients corresponding to their constituent elements. Hence each base-station sends a flow of 4 and each client receives its full demand, i.e., the total flow is $4|T|$. Therefore, $\textsc{Opt}(R(I)) = 4|T| \leq 12p = 12 \cdot \textsc{Opt}(I)$.

We now describe mapping Γ_2. Let τ be an assignment for a channel-width instance $R(I)$; let $c(\tau)$ be the total flow in the network for assignment τ. For every base-station that is assigned a threshold of ≤ 1, we make the threshold 1, and for every base-station that is assigned a threshold of > 1, we make the threshold 4; let B^1 and B^4 be the corresponding sets of base-stations. Observe that all flows in τ remain feasible even after this change in thresholds. We now change the flows in τ so that the total flow does not decrease. The base-stations in B^1 are processed in an arbitrary order and each base-station in B^1 is given a flow of 1 to every client it connects to provided the client's demand has not been satisfied already, and a flow of 1 to C_z. Now, for every base-station $B_i \in B^4$, we give a flow of 4 to the (B_i, C_z) edge. If $|B^1| < p$, then this causes an overflow at C_z and we move $p - |B^1|$ base-stations from B^4 to B^1. Then, C_z's demand is exactly met since $|B^1| = p$. Note that none of the changes to the flow decreases the total flow in the network.

Now, let each base-station in B^1 that has an outflow of 4 be called *good*; let x be the number of good base-stations. Then, (1) the good base-stations form an independent set of triples, and (2) the total flow in the network is at most $4|T| - (|B^1| - x)$. We define $\Gamma_2(\tau) = x$. Then,

$$\textsc{Opt}(I) - \Gamma_2(\tau) = p - x \leq |B^1| - x \leq \textsc{Opt}(R(I)) - c(\tau).$$

Since the functions Γ_1 and Γ_2 are computable in polynomial time, we have proved Theorem 1.

Budget		Critical Capacity				Demand		
B1	B2		C1	C2	C3	C1	C2	C3
20	30	B1	100	20	2	100	10	50
		B2	10	0	40			

Figure 2: The budgets, critical capacities and demands in the running example.

3 Algorithms for Maximum Bipartite Flow

3.1 A Combinatorial Algorithm

Recall that we have n base-stations and m clients. We present a combinatorial greedy algorithm that achieves an approximation ratio of $O(\log N)$, where $N = \max(n, m)$. This algorithm has two steps- the first step pre-processes the given instance of the problem to produce a more structured instance *while modifying the optimal solution by only a constant factor*. The second step provides an algorithm for such structured instances of the problem which has an approximation ratio of $O(\log N)$. The two steps, in combination, yield an $O(\log N)$ approximation algorithm for *all* instances of the problem.

To describe the algorithm, we will use a running example. Let there be 2 base-stations $B1$ and $B2$ and 3 clients $C1, C2$ and $C3$. The budgets, critical capacities and demands are given in the Figure 2.

We will also need the following definitions. A link (B, C) is said to be *maximal* if its critical capacity is the maximum among all the links, i.e. if

$$\eta(B, C) \geq \eta(B', C'),\ \forall B' \in \mathcal{B},\ \forall C' \in \mathcal{C}.$$

In our example, the link $(B1, C1)$ is maximal. A link (B, C) is said to be *infeasible* if its critical capacity is greater than either the budget of base-station B or the demand of client C, i.e. if

$$\eta(B, C) > \min(\beta(B), \alpha(C));$$

otherwise, the link is said to be feasible. In our example, links $(B2, C1)$ and $(B1, C3)$ are feasible (ignoring link $(B2, C2)$ which has critical capacity of 0); all other links are infeasible.

Pre-processing. The pre-processing comprises three steps. In the first step, we decrease the critical capacity of all maximal infeasible links until either some other link becomes maximal or there is at least one maximal feasible link. In the second case, we stop, while in the first case, we again iterate by decreasing the critical capacities of all the (bigger set of) maximal infeasible links. In the example, we first decrease $\eta(B1, C1)$ to 40, at which stage $(B2, C3)$ also becomes maximal (but is also infeasible). We now decrease both $\eta(B1, C1)$ and $\eta(B2, C3)$ to 30, when $(B2, C3)$ becomes feasible.

Lemma 1. *The above modification does not change the optimum value of an instance of the problem.*

	C1	C2	C3
B1	16	16	0
B2	8	0	16

Figure 3: Critical capacities after rounding.

Proof. We prove this inductively, proving that in any particular iteration, an optimal flow before the iteration continues to be achievable after the iteration. Denote the maximum critical threshold among all links at base-station B in the modified instance by $M(B)$. Also denote the maximum flow on a link at B and the threshold at B in the optimal solution by $F^*(B)$ and $\tau^*(B)$ respectively. $F^*(B) \leq M(B)$ since otherwise, $F^*(B)$ would violate either the budget constraint at B or the demand constraint at the corresponding client. We can therefore set the threshold at B to $\max(M(B), \tau^*(B))$ without changing any flow; performing this for all base-stations B produces a solution for the modified instance which is equal in value to an optimal solution for the original instance. □

The important property of this transformation is that we are now guaranteed a solution of value equal to the maximum critical capacity in the modified instance. Specifically, this is achieved by saturating the capacity on the maximal feasible link. This solution is called the *maximal saturation solution.*

In the second pre-processing step, let us consider all links with critical capacity at most $1/N^2$ times that of a maximal link (call these *weak links*). We decrease the critical capacity of all weak links to 0. In our example, $\eta(B1, C3)$ is decreased to 0.

Lemma 2. *The optimum value of the modified instance of the problem is at least half of the optimum value before the modification.*

Proof. Since there are $nm \leq N^2$ links overall, the total flow in all the weak links in an optimal solution before the modification is at most as much as the value of the maximal saturation solution. Thus, either (1) the total flow in the weak links is at most half the original optimum, or (2) the maximal saturation solution (which is also a solution in the modified instance) is at least half in value to the original optimum. □

We now describe the final pre-processing step. We scale down all critical capacities by the minimum non-zero critical capacity; all the (non-zero) critical capacities are in the range 1 to N^2. Then, we round down all critical capacities by a factor of at most 2 to one of the $O(\log N)$ values $\{2^i : 0 \leq i \leq 2 \log N\}$. In our example, the new set of critical capacities is given in Figure 3.

Lemma 3. *The optimum value of the modified instance of the problem is at least half of the optimum value before the modification.*

Proof. All the thresholds at the base-stations and the flows on the links in an optimal solution to the instance before the modification can be halved to obtain a solution to the modified instance. □

Algorithm for Modified Problem Instance. We set the capacity of each link (B, C) to its critical capacity $\eta(B, C)$. Using these link capacities, we construct the augmentation graph of the instance of the problem. We then run a maximum $s - t$ flow sub-routine on this augmentation graph.

Lemma 4. *The maximum $s - t$ flow obtained above is an upper bound on the value of an optimal solution to our problem instance.*

Proof. Irrespective of the threshold at each base-station in the optimal solution, the capacity of each link is at most as much as its critical capacity, which is the capacity in the augmentation graph for which we obtain the maximum $s-t$ flow. Thus, the flows on the links in the optimal solution do not violate any capacity constraint in the augmentation graph. □

We now partition the base-client links into $O(\log N)$ groups according to their critical capacity- the ith group contains all links with capacity 2^i. In our example, $(B1, C1), (B1, C2)$ and $(B2, C3)$ are in the 4th group, while $(B2, C1)$ is in the 3rd group. All other groups are empty. It is important to note that if we set the threshold of all base-stations to 2^i; then all links in the ith group have capacity 2^i. The maximum flow can also be split into $O(\log N)$ parts according to the flow carried by the links in each group. If the ith group carries the largest flow among the groups, then we set the threshold of all base-stations to 2^i and obtain a solution to our problem whose value is at least a $1/\log N$ fraction of the maximum flow. Combining this observation with the above lemma gives the following theorem.

Theorem 3. *The above algorithm has an approximation factor of $O(\log N)$.*

3.2 A Linear Program

Our goal now is to improve upon this combinatorial algorithm. Without loss of generality, we may assume that the assigned capacity chosen at any base-station B in an optimal solution is one among the critical capacities of its outgoing edges, i.e. $\tau(B) \in \{\eta(B, C) : C \in \mathcal{C}\}$. If this is not the case, then the assigned capacity can be increased to the closest value $\geq \tau(B)$ from the set $\{\eta(B, C) : C \in \mathcal{C}\}$ without changing the flow on any link. This allows us introduce the *boolean capacity choice function* $p(B, C)$, which is 1 if $\tau(B) = \eta(B, C)$, and 0 otherwise. For any base-station B, $p(B, C)$ should be 1 for exactly one client C (called the *choice constraint*), breaking ties arbitrarily. We also introduce another new notation, $\mathcal{C}_B(C)$ which represents the set of clients for which the critical capacity of their link to base-station B is less than that for client C, i.e.

$$\mathcal{C}_B(C) = \{C' \in \mathcal{C} : \eta(B, C') \leq \eta(B, C)\}.$$

A natural formulation of the problem is via the following integer linear program (ILP), where constraints (1), (2), (3) and (4) correspond to budget, demand, capacity and choice constraints respectively.

$$\texttt{maximize } \sum_{B\in\mathcal{B}}\sum_{C\in\mathcal{C}} f(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f(B,C) \leq \beta(B),\ \forall B\in\mathcal{B} \quad (1)$$

$$\sum_{B\in\mathcal{B}} f(B,C) \leq \alpha(C),\ \forall C\in\mathcal{C} \quad (2)$$

$$f(B,C) \leq \sum_{C'\in\mathcal{C}_B(C)} p(B,C')\eta(B,C'),\ \forall B\in\mathcal{B},\ \forall C\in\mathcal{C} \quad (3)$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1,\ \forall B\in\mathcal{B} \quad (4)$$

$$p(B,C) \in \{0,1\},\ \forall B\in\mathcal{B},\ \forall C\in\mathcal{C} \quad (5)$$

$$f(B,C) \geq 0,\ \forall B\in\mathcal{B},\ \forall C\in\mathcal{C}. \quad (6)$$

LP Relaxation. To make this ILP tractable, we relax constraint (5) and allow the function p to assume values between 0 and 1 (call this the *fractional program* or FLP). The natural interpretation is that $p(B,C)$ denotes the *goodness* of $\eta(B,C)$ as the assigned capacity of base-station B. Mathematically, it can be thought of as the probability with which $\eta(B,C)$ should be the assigned capacity of base-station B.

Unfortunately, it turns out that this natural linear programming relaxation fails to provide us with an approximation guarantee that is significantly better than the one achieved by the combinatorial algorithm. To understand why this is the case, let us note that for a given choice of values of $p(B,C)$, this linear program is solving the max-flow problem in the augmentation graph where the capacity u of a link (B,C) is the following: if the assigned capacity $\tau(B)$ at base-station B is chosen according to the probability distribution given by $p(B,C)$, then

$$u = \mathbb{E}[\psi_\tau(B,C)].$$

Therefore, by max-flow/min-cut duality, the approach used in the LP boils down to choosing $p(B,C)$ in such a way that the minimal expected capacity among all s-t-cuts in the augmented graph is maximized. Given some final choice of $p(B,C)$ computed by the linear program, it is tempting to round it by choosing assigned capacities according to $p(B,C)$, and then solving the max-flow problem in the resulting graph, hoping that the capacity of minimal cut will be close to the expected one. Clearly, when we focus on one particular cut, say the one that separates base-stations from clients, it will be true, but this does not necessarily mean that for all cuts, such a promise will hold simultaneously. It may happen that for the choice of assigned capacities that we obtain, it will always be the case that for part of the clients the capacity of links leading to them in the resulting graph will be much below the expectation, while for the other part it will excessively large, and this excess will be wasted due to the bottlenecks imposed by not large enough capacity of demand arcs for the respective clients. Thus, even if on expectation each client has reasonable capacity of links leading to it, the rounding procedure might not provide us with a particularly good

solution. Therefore, our analysis of the approximation guarantee given by this LP would need to argue that with good probability all cuts are preserved up to some ratio, and in fact, using Chernoff bounds, we can prove that this is indeed true for the ratio $O(\log(m+n)/\log\log(m+n))$. Unfortunately, we can show, through the following integrality gap example, that this unsatisfactorily large ratio is not only a shortcoming of our particular rounding procedure, but in fact, it is all that we can achieve through *any* rounding algorithm for this LP.

Integrality Gap Example. Let there be a single base-station B and $m = 2r-1$ clients, where r is a parameter in the construction. The base-station has an infinite (or very large) budget. The clients are grouped into $\log r + 1$ groups $\mathcal{C}_i$, $i = 0$ to $\log r$, where $\mathcal{C}_i$ comprises $r/2^i$ clients. For each client in the ith group, i.e. $C \in \mathcal{C}_i$, the critical capacity $\eta(B,C)$ is 2^i and demand $\alpha(C)$ is $2^i/\log r$. A solution to the fractional program is the following:

$$\begin{aligned} p(B,C) &= \frac{2^i}{r\log r},\ \forall B \in \mathcal{B},\ \forall C \in \mathcal{C}_i \\ f(B,C) &= \frac{2^i-1}{\log r},\ \forall B \in \mathcal{B},\ \forall C \in \mathcal{C}_i. \end{aligned}$$

It can be verified that these assignment of values satisfy all the constraints in the linear program. Then, the total satisfied demand is

$$\sum_{i=0}^{\log r}\left(\frac{2^i-1}{\log r}\right)\left(\frac{r}{2^i}\right) = \Theta(r).$$

However, suppose we choose any particular threshold 2^k in the optimal integral solution. Then,

$$f(B,C) \le \begin{cases} 0, & \text{if } j < k, \\ 2^j/\log r, & \text{if } k \le j < k + \log\log r, \\ 2^k, & \text{otherwise.} \end{cases}$$

This gives a total satisfied demand of

$$\sum_{j=k}^{k+\log\log r-1}\left(\frac{2^j}{\log r}\right)\left(\frac{r}{2^j}\right) + \sum_{j=k+\log\log r}^{j=\log r} 2^k\left(\frac{r}{2^j}\right) = O(\frac{r\log\log r}{\log r}) = O(\frac{r\log\log m}{\log m}).$$

We thus obtain an $\Omega(\log m/\log\log m)$ integrality gap for this LP.

3.3 An Alternative Linear Program

In this section, we will describe a more sophisticated ILP which overcomes the shortcomings of the previous ILP. Note that our goal is to choose assigned capacities such that the augmentation graph has a large maximum flow, or equivalently by Menger's theorem, a large minimum cut. The previous FLP

ensures that each cut has large capacity in expectation and therefore the minimum among the expected capacities of the cuts is large; this however does not guarantee that the expected capacity of the minimum cut is large. We need this stronger guarantee from our LP. To achieve this goal, we design an LP which yields a family of flows corresponding to the different choices of assigned capacities, and ensures that the expected value of these flows is large. This will imply that the expected capacity of the minimum cut is large, and therefore provides stronger guarantees than the previous LP. Precisely, we consider the following ILP.

$$\texttt{maximize } \textstyle\sum_{B\in\mathcal{B}}\sum_{C',C\in\mathcal{C}} f_{C'}(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f_{C'}(B,C) \leq p(B,C')\beta(B),\ \forall B\in\mathcal{B},\ \forall C'\in\mathcal{C} \tag{7}$$

$$\sum_{B\in\mathcal{B}}\sum_{C'\in\mathcal{C}} f_{C'}(B,C) \leq \alpha(C),\ \forall C\in\mathcal{C} \tag{8}$$

$$f_{C'}(B,C) \leq \begin{cases} 0, & \text{if } \eta(B,C)<\eta(B,C') \\ p(B,C')\min\{\eta(B,C'),\alpha(C)\}, & \text{otherwise} \end{cases} \quad \forall B\in\mathcal{B},\ \forall C,C'\in\mathcal{C} \tag{9}$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1,\ \forall B\in\mathcal{B} \tag{10}$$

$$p(B,C) \in \{0,1\},\ \forall B\in\mathcal{B},\ \forall C\in\mathcal{C} \tag{11}$$

$$f_{C'}(B,C) \geq 0,\ \forall B\in\mathcal{B},\ \forall C,C'\in\mathcal{C}. \tag{12}$$

Before moving on to the analysis of the algorithm, let us verify that the new ILP does represent the original problem. To do this, let us fix some optimal solution (τ^*, f^*) for the original problem. Consider now a solution to our ILP defined as follows. For each base-station B we set $p(B,C)=1$ if B chooses $\eta(C)$ as its assigned capacity i.e. if $\tau^*(B)=\eta(B,C)$; otherwise $p(B,C)=0$. Next, for each link (B,C), we set $f_{C'}(B,C)=f^*(B,C)$ if $\tau^*(B)=\eta(B,C')$, and $f_{C'}(B,C)=0$ otherwise. Observe that all the constraints are preserved, and the objective value corresponding to this solution has value equal to that for the optimal solution. The converse direction is similar and we omit it for brevity.

The key to understanding this ILP is the rounding technique that we employ in our approximation algorithm; so let us describe our algorithm first. We relax the integrality constraint , i.e. constraint (11) and allow the variables p to take any value between 0 and 1, both inclusive. We solve the resulting FLP, and then round the solution to obtain an integral solution. It is in this rounding procedure that the crux of our algorithm lies. We choose assigned capacities according to $p(B,C)$, noting that for a fixed base-station B, $p(B,C)$ is a valid probability distribution. Now, for any base-station B, if the assigned capacity $\tau(B)=\eta(B,C')$, then for each link (B,C), we add a flow of $g_{C'}(B,C)\equiv f_{C'}(B,C)/p(B,C')$ to the $s-B-C-t$ path in the augmentation graph. Crucially, this does not violate the budget constraint at any base-station B since the total outflow at B is

$\sum_{C\in\mathcal{C}} g_{C'}(B,C)$, which is at most $\beta(B)$ by constraint (7); neither does it violate the capacity constraint on any link (B,C) since constraint (9) ensures that the flow on link (B,C) is at most $\psi_\tau(B,C)$. Hence, we focus on analyzing violations of the demand constraints. The total inflow at C is $\sum_{B\in\mathcal{B}} g_{C'(B)}(B,C)$, where $C'(B)$ is the client chosen by base-station B in the rounding. Unfortunately, assigning these flow values simultaneously for all base-stations might lead to an overflow in a demand arc (i.e., a demand constraint violation). For a client with overflow, we decrease the incoming flows arbitrarily until the flow on the link to t exactly matches its capacity (we call this the *truncation step*). Since such a truncated flow is feasible, our ultimate goal is to prove that the truncation step decreases the initial flow only by a constant fraction in expectation.

Let $F(B,C)$ be the random variable denoting the flow on link (B,C); clearly, $F(B,C) = g_{C'}(B,C)$ with probability $p(B,C')$ and its expectation

$$\mathbb{E}[F(B,C)] = f(B,C) \equiv \sum_{C'\in\mathcal{C}} f_{C'}(B,C) = \sum_{C'\in\mathcal{C}} p(B,C')g_{C'}(B,C).$$

Constraint (8) states that the expected inflow $\sum_{B\in\mathcal{B}} f(B,C)$ at client C is at most its demand $\alpha(C)$. Also, for a given C, the $F(B,C)$ values are independent. Finally, constraint (9) enforces that $F(B,C) \leq \alpha(C)$ irrespective of the choice of the assigned capacity at base-station B, (i.e. inflow due to a single base-station at a client never exceeds the demand of the client). Thus, we ensure that there is some cap on the wasted capacity, i.e. the capacity in the base-client links that is left unused due to truncation; such a cap was absent in the previous formulation and, as we will see, this additional condition will be sufficient for our purpose.

Note. The rounding procedure can be simplified in an actual implementation. Once we obtain the assigned capacities of all the base-stations using randomized rounding as described above, we can run a maximum flow algorithm on the augmentation graph. Note that this achieves at least as much (and potentially more) flow as that achieved by the rounding procedure described above. So an actual implementation of our algorithm will rather employ a maximum flow sub-routine than the above procedure for determining flows. However, we assume that our algorithm uses the above procedure since it would be simpler to analyze—all bounds proved using this assumption hold for an actual implementation using maximum flow as well.

Analysis. If there are n base-stations and m clients, then the algorithm clearly runs in time polynomial in $N = \max(n,m)$. So, we focus on proving guarantees on the approximation factor of the algorithm. By the discussion in the previous section, we know that in our rounding procedure the difference between the objective value of the solution to the FLP and the actual flow that we obtain, consists solely of the amount of initial flow that we have to truncate due to overflows at clients. Thus our main task is to prove upper bounds on the expected overflow. Let $F(C) \equiv \sum_{B\in\mathcal{B}} F(B,C)$ be the random variable denoting total inflow at C before truncation and $T(C) \equiv \min(F(C),\alpha(C))$ be the random

variable representing the inflow at C after truncation. We would like to show that

$$\mathbb{E}[T(C)] \geq (1 - 1/e)\mathbb{E}[F(C)]. \tag{13}$$

Then,

$$\mathbb{E}[\sum_{C \in \mathcal{C}} T(C)] \geq (1 - 1/e)\mathbb{E}[\sum_{C \in \mathcal{C}} F(C)] \geq (1 - 1/e)T^*, \tag{14}$$

where T^* is the total flow in an optimal integral solution. This proves Theorem 2, which was stated in Section 1.

To establish inequality (13), we will need the following theorem (a similar proof appears in [1]).

Theorem 4. *Suppose we have a sequence of independent discrete random variables $X_1, X_2, \ldots, X_n$ such that each X_i has finite support and $0 \leq X_i \leq 1$. Furthermore, suppose $X = \sum_{i=1}^{n} X_i$ and $\mathbb{E}[X] \leq 1$. If $Y = \min(X, 1)$, then*

$$\mathbb{E}[Y] \geq (1 - 1/e)\mathbb{E}[X].$$

We first use this theorem to prove inequality (13), and then give a proof of the theorem itself. If, for client C, we define $X_i = F(B_i, C)/\alpha(C)$ (where $\mathcal{B} = \{B_1, \ldots, B_n\}$) and $Y = T(C)/\alpha(C)$, then such X_is and Y satisfy the assumptions of the theorem. Thus, we can conclude that

$$\mathbb{E}[T(C)] = \alpha(C)\mathbb{E}[Y] \geq (1 - 1/e)\alpha(C)\mathbb{E}[X] = (1 - 1/e)\mathbb{E}[F(C)].$$

Proof of Theorem 4. Our proof has the following outline. We assume for the sake of contradiction that there exists a sequence $\{\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n\}$ of discrete random variables such that

$$\mathbb{E}[\hat{Y}] = \mathbb{E}[\min(\sum_i \hat{X}_i, 1)] < (1 - 1/e)\mathbb{E}[\hat{X}] = (1 - 1/e)\mathbb{E}[\sum_i \hat{X}_i].$$

We call such a sequence $(\hat{X}_i)$ a *nemesis sequence*. First, we prove that we can assume without loss of generality, that $\hat{X}_i$s are 0-1 random variables. Then, we prove our theorem for 0-1 random variables, thus arriving at a contradiction for the general case.

Let $S(\hat{X}_i)$ be the number of distinct values other than 0 and 1 for which $\hat{X}_i$ has non-zero probability. Now, let us consider a nemesis sequence $(\hat{X}_i)$ that minimizes $\sum_i S(\hat{X}_i)$. We will prove that if $\sum_i S(\hat{X}_i) > 0$ then there exists another nemesis sequence $(\widetilde{X}_i)$ with $\sum_i S(\widetilde{X}_i) < \sum_i S(\hat{X}_i)$. The minimality of $(\hat{X}_i)$ implies there exists a nemesis sequence with $\sum_i S(\hat{X}_i) = 0$, i.e. $(\hat{X}_i)$ is a sequence of 0-1 variables.

If $\sum_i S(\hat{X}_i) > 0$, then there exists some k such that $S(\hat{X}_k) > 0$, which in turn means that there exists some $0 < a < 1$ such that $Pr[\hat{X}_k = a] = p > 0$. Suppose that this $\hat{X}_k$ takes value of 0 and 1 with probability $q \geq 0$ and $r \geq 0$ respectively (note that p, q and r do not necessarily sum to 1). Now, consider another random variable $\widetilde{X}_k$ that is distributed identically to $\hat{X}_k$ except that the

probabilities of a, 0 and 1 are changed to 0, $q+(1-a)p$ and $r+ap$ respectively. Note that $\mathbb{E}[\widetilde{X}_k] = \mathbb{E}[\hat{X}_k]$, $0 \leq \widetilde{X}_k \leq 1$ and $S(\widetilde{X}_k) = S(\hat{X}_k) - 1$. So, if we define $\widetilde{X}_i = \hat{X}_i$ for $i \neq k$, then $\mathbb{E}[\hat{X}] = \mathbb{E}[\widetilde{X}] \leq 1$. We would like to compare $\mathbb{E}[\hat{Y}]$ to $\mathbb{E}[\widetilde{Y}] \equiv \mathbb{E}[\min\{\widetilde{X}, 1\}]$. Note that by our definition, for any $\delta \geq 0$,

$$Pr[\widetilde{X} - \widetilde{X}_k = \delta] = Pr[\hat{X} - \hat{X}_k = \delta].$$

Thus, to prove that $\mathbb{E}[\widetilde{Y}] \leq \mathbb{E}[\hat{Y}]$, it is sufficient to prove that

$$\mathbb{E}[\widetilde{Y}|\widetilde{X} - \widetilde{X}_k = \delta] \leq \mathbb{E}[\hat{Y}|\hat{X} - \hat{X}_k = \delta],$$

for all $\delta \geq 0$.

Clearly, if $\delta \geq 1$ then

$$\mathbb{E}[\widetilde{Y}|\widetilde{X} - \widetilde{X}_k = \delta] = 1 = \mathbb{E}[\hat{Y}|\hat{X} - \hat{X}_k = \delta];$$

so the inequality holds. On the other hand, for $\delta < 1$,

$$\begin{aligned} \mathbb{E}[\widetilde{Y}|\widetilde{X} - \widetilde{X}_k = \delta] - \mathbb{E}[\hat{Y}|\hat{X} - \hat{X}_k = \delta] &= \mathbb{E}[\min\{\widetilde{X}_k, 1-\delta\}] - \mathbb{E}[\min\{\hat{X}_k, 1-\delta\}] \\ &= ap(1-\delta) - p\min\{a, 1-\delta\} \leq 0. \end{aligned}$$

Thus, $\mathbb{E}[\widetilde{Y}] \leq \mathbb{E}[\hat{Y}]$, which proves that $\{\hat{X}_i\}$ had to be a zero-one nemesis sequence by minimality of $\sum_i S(\hat{X}_i)$.

Now, when $\{\hat{X}_i\}$ is zero-one,

$$\begin{aligned} \mathbb{E}[\hat{Y}] &= Pr[\hat{X} \geq 1] \\ &= 1 - \prod_i (1 - Pr[\hat{X}_i = 1]) \\ &\geq 1 - (1 - \sum_i \mathbb{E}[\hat{X}_i]/n)^n \\ &\geq 1 - e^{-\mathbb{E}[\hat{X}]} \\ &\geq (1 - 1/e)\mathbb{E}[\hat{X}], \end{aligned}$$

as desired, where in the first inequality we used the fact that

$$\sum_i Pr[\hat{X}_i = 1] = \sum_i \mathbb{E}[\hat{X}_i] = \mathbb{E}[\hat{X}],$$

and the arithmetic/geometric mean inequality; and the last inequality follows from Taylor expansion using the fact that $\mathbb{E}[\hat{X}] \leq 1$. □

Observe that this theorem is tight for n i.i.d. 0-1 random variables X_i with $Pr[X_i = 1] = 1/n$. Further, since it holds any set of discrete random variables, it can be extended to continuous random variables as well using compactness.

4 Conclusion and Open Problems

The ability to adaptively change channel widths in wireless networks introduces interesting algorithmic problems. In this paper, we studied a throughput maximization problem in infrastructure wireless networks that was equivalent to the maximum flow problem in bipartite graphs with adaptive channel width. We gave an LP-rounding based algorithm for this problem that has an approximation ratio of $e/(e-1)$; independent of our work, Chandra Chekuri has suggested that the same result also follows from a submodularity-based analysis. A natural and interesting generalization of this question is maximizing throughput (i.e. flow) in a *general* (i.e. possibly non-bipartite) graph with adaptive channel width. Another important problem in wireless networks is scheduling, which often translates to coloring problems. It would be interesting to explore the ramifications of adaptively changing channel widths on graph coloring problems. As a first step, we might want to understand the implications of adaptively changing capacities on the matching problem, since the matching constraint can be interpreted as a particularly simple edge coloring constraint. Finally, we note that bi-criteria (or multi-criteria) graph optimization problems have not been studied extensively (at least compared to their single criterion variants), often because it turns out that the minimum cost network satisfying multiple criteria is essentially a juxtaposition of individual networks that are optimal from the point of view of one of the criteria. However, the ability to adaptively change channel widths might make it possible to design a single network that is optimal from the point of view of any of the given criteria as long as we choose a corresponding *optimal* threshold settings. Thus, bi-criteria (or multi-criteria) optimization problems in such networks might be substantially different in their combinatorial structure from their counterparts in fixed-capacity networks.

5 Acknowledgements

We are grateful to Chandra Chekuri for suggesting an alternative approach for our problem based on submodular functions. We also thank the anonymous referees for helpful comments.

References

[1] N. Andelman and Y. Mansour. Auctions with budget constraints. In *9th Scandinavian Workshop on Algorithm Theory*, pages 26–38, 2004.

[2] Y. Azar, L. Epstein, Y. Richter and G. J. Woeginger, All-norm approximation algorithms. In *J. Algorithms*, 52(2):120-133, 2004.

[3] Y. Azar and O. Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 44(1):49–66, 2006.

[4] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

[5] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *SIGCOMM*, pages 135–146, 2008.

[6] C. Chekuri, S. Khanna, and F. B. Shepherd. An O(sqrt(n)) approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.

[7] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doklady (Doklady)*, 11:1277–1280, 1970.

[8] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[9] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507—518, 1975.

[10] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[11] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783—797, 1998.

[12] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921—940, 1988.

[13] V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J. Comput. Syst. Sci.*, 67(3):473–496, 2003.

[14] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[15] D. R. Karger. Using random sampling to find maximum flows in uncapacitated undirected graphs. In *STOC*, pages 240—249, 1997.

[16] D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In *SODA*, pages 490–499, 1998.

[17] D. R. Karger and M. S. Levine. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *STOC*, pages 69—78, 1998.

[18] D. R. Karger and M. S. Levine. Random sampling in residual graphs. In *STOC*, pages 63–66, 2002.

[19] J. M. Kleinberg. Single-source unsplittable flow. In *FOCS*, pages 68–77, 1996.

[20] J. K. Lenstra and D. B. Shmoys and É. Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. In *Math. Program.*, 46:256–271, 1990.

[21] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-Aware Spectrum Distribution in Wireless LANs. In *ICNP*, 2008.

[22] E. Petrank. The Hardness of Approximation: Gap Location. In *Computational Complexity*, 4:133-157, 1994.

[23] C. E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):1021, 1949.

[24] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *FOCS*, pages 416–425, 1997.