Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs [*]

Paul Christiano
Mathematics
MIT
paulfchristiano@gmail.com

Jonathan A. Kelner
Mathematics
MIT
kelner@mit.edu

Aleksander Mądry
EECS
MIT
madry@mit.edu

Daniel A. Spielman
Computer Science
Yale University
spielman@cs.yale.edu

Shang-Hua Teng
Computer Science
USC
shanghua@usc.edu

ABSTRACT

We introduce a new approach to computing an approximately maximum s-t flow in a capacitated, undirected graph. This flow is computed by solving a sequence of electrical flow problems. Each electrical flow is given by the solution of a system of linear equations in a Laplacian matrix, and thus may be approximately computed in nearly-linear time.

Using this approach, we develop the fastest known algorithm for computing approximately maximum s-t flows. For a graph having n vertices and m edges, our algorithm computes a $(1-\epsilon)$-approximately maximum s-t flow in time[1] $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$. A dual version of our approach gives the fastest known algorithm for computing a $(1+\epsilon)$-approximately minimum s-t cut. It takes $\widetilde{O}(m+n^{4/3}\epsilon^{-16/3})$ time. Previously, the best dependence on m and n was achieved by the algorithm of Goldberg and Rao (J. ACM 1998), which can be used to compute approximately maximum s-t flows in time $\widetilde{O}(m\sqrt{n}\epsilon^{-1})$, and approximately minimum s-t cuts in time $\widetilde{O}(m+n^{3/2}\epsilon^{-3})$.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms

General Terms

Algorithms, Theory

[*]A full version of this paper is available at `http://arxiv.org/abs/1010.2921`. This research was supported in part by NSF grants 0829878, 0843915, 0915487, 1032367 and by ONR grant N00014-11-1-0053.

[1]We recall that $\widetilde{O}(f(m))$ denotes $O(f(m)\log^c f(m))$ for some constant c.

Keywords

Maximum flows, minimum cuts, Laplacian linear systems, electrical flows, multiplicative weights update method.

1. INTRODUCTION

The maximum s-t flow problem and its dual, the minimum s-t cut problem, are two of the most fundamental and extensively studied problems in Operations Research and Optimization [17, 1]. They have many applications (see [2]) and are often used as subroutines in other algorithms (see e.g. [3, 18]). Many advances have been made in the development of algorithms for this problem (see Goldberg and Rao [12] for an overview). However, for the basic problem of computing or $(1-\epsilon)$-approximating the maximum flow in undirected, unit-capacity graphs with $m = O(n)$ edges, the asymptotically fastest known algorithm is the one developed in 1975 by Even and Tarjan [9], which takes time $O(n^{3/2})$. Despite 35 years of extensive work on the problem, this bound has not been improved.

In this paper, we introduce a new approach to computing approximately maximum s-t flows and minimum s-t cuts in undirected, capacitated graphs. Using it, we present the first algorithms that break the $O(n^{3/2})$ complexity barrier described above. In addition to being the fastest known algorithms for these problems, they are simple to describe and introduce techniques that may be applicable to other tasks. In them, we reduce the problem of computing maximum flows subject to capacity constraints to the problem of computing electrical flows in resistor networks. An approximate solution to each electrical flow problem can be found in time $\widetilde{O}(m)$ using recently developed algorithms for solving systems of linear equations in Laplacian matrices [14, 19].

There is a simple physical intuition that underlies our approach, which we describe here in the case of a graph with unit edge capacities. We begin by thinking of each edge of the input graph as a resistor with resistance one, and we compute the electrical flow that results when we send current from the source s to the sink t. These currents obey the flow conservation constraints, but they are ignorant of capacities of the edges. To remedy this, we increase the resistance of each edge in proportion to the amount of current flowing through it—thereby penalizing edges that violate their capacities—and compute the electrical flow with these new resistances.

After repeating this operation $\widetilde{O}(m^{1/3} \cdot \text{poly}(1/\epsilon))$ times, we will be able to obtain a $(1-\epsilon)$-approximately maximum s-t flow by taking a certain average of the electrical flows that we have computed, and we will be able to extract a $(1+\epsilon)$-approximately minimum s-t cut from the vertex potentials[2]. This will give us algorithms for both problems that run in time $\widetilde{O}(m^{4/3} \cdot \text{poly}(1/\epsilon))$. By combining this with the graph smoothing and sampling techniques of Karger [13], we can get a $(1-\epsilon)$-approximately maximum s-t flow in time $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$. Furthermore, by applying the cut algorithm to a sparsifier [4] of the input graph, we can compute a $(1+\epsilon)$-approximately minimum s-t cut in time $\widetilde{O}(m + n^{4/3}\epsilon^{-16/3})$.

We remark that the results in this paper immediately improve the running time of algorithms that use the computation of an approximately maximum s-t flow on an undirected, capacitated graph as a subroutine. For example, combining our work with that of Sherman [18] allows us to achieve the best currently known asymptotic approximation ratio of $O(\sqrt{\log n})$ for the sparsest cut problem in time $O(m + n^{4/3+\delta})$ for any constant $\delta > 0$.

We are hopeful that our approach can be extended to directed graphs and can also eventually lead to an algorithm that approximately solves the maximum flow problem in nearly-linear time.

1.1 Previous Work on Maximum Flows and Minimum Cuts

The best previously known algorithms for the problems studied here are obtained by combining techniques of Goldberg and Rao [12] and Benczúr and Karger [5]. In a breakthrough paper, Goldberg and Rao [12] developed an algorithm for computing exact maximum s-t flows in directed or undirected capacitated graphs in time

$$O(m \min(n^{2/3}, m^{1/2}) \log(n^2/m) \log U),$$

assuming that the edge capacities are integers between 1 and U. In undirected graphs, one can find faster approximation algortithms for these problems by using the graph sparsification techniques of Benczúr and Karger [4, 5]. Goldberg and Rao [12] observe that by running their exact maximum flow algorithm on the sparsifiers of Benczúr and Karger [4], one can obtain a $(1+\epsilon)$-approximately minimum cut in time $\widetilde{O}(m + n^{3/2}\epsilon^{-3})$. This provides an approximation of the value of the maximum flow. To actually obtain a feasible flow whose value approximates the maximum one can apply the algorithm of Goldberg and Rao in the divide-and-conquer framework of Benczúr and Karger [5]. This provides a $(1-\epsilon)$-approximately maximum flow in time $\widetilde{O}(m\sqrt{n}\epsilon^{-1})$. We refer the reader to the the paper of Goldberg and Rao for a survey of the previous work on algorithms for computing maximum s-t flows.

In more recent work, Daitch and Spielman [8] showed that fast solvers for Laplacian linear systems [19, 14] could be used to make interior-point algorithms for the maximum flow and minimum-cost flow problems run in time $\widetilde{O}(m^{3/2} \log U)$, and Mądry [15] showed that one can approximate a wide range of cut problems, including the minimum s-t cut problem, within a polylogarithmic factor in almost linear time.

1.2 Outline

We begin the technical part of this paper in Section 2 with a review of maximum flows and electrical flows, along with several theorems about them that we will need in the sequel. In Section 3 we give a simplified version of our approximate maximum-flow algorithm that has running time $\widetilde{O}(m^{3/2}\epsilon^{-5/2})$.

In Section 4, we will show how to improve the running time of our algorithm to $\widetilde{O}(m^{4/3}\epsilon^{-3})$; we will then describe how to combine this with existing graph smoothing and sparsification techniques to compute approximately maximum s-t flows in time $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$ and to approximate the value of such flows in time $\widetilde{O}(m+n^{4/3}\epsilon^{-16/3})$. In Section 5, we present a variant of our algorithm that computes approximately minimum s-t cuts in time $\widetilde{O}(m+n^{4/3}\epsilon^{-16/3})$.

2. MAXIMUM FLOWS, ELECTRICAL FLOWS, AND LAPLACIAN SYSTEMS

2.1 Graph Theory Definitions

Throughout the rest of the paper, let $G = (V, E)$ be an undirected graph with n vertices and m edges. We distinguish two vertices, a *source* vertex s and a *sink* vertex t. We assign each edge e a nonzero integral *capacity* $u_e \in \mathbb{Z}^+$, and we let $U := \max_e u_e / \min_e u_e$ be the ratio of the largest to the smallest capacities. We now recall that an *s-t cut* is a partition $(S, V \setminus S)$ of the vertices into two disjoint sets such that $s \in S$ and $t \in V \setminus S$. The *capacity* $u(S)$ of the cut is defined to be the sum $u(S) := \sum_{e \in E(S)} u_e$, where $E(S) \subseteq E$ is the set of edges with one endpoint in S and one endpoint in $V \setminus S$. The *minimum s-t cut problem* is that of finding an s-t cut of minimum capacity.

We also recall that an *s-t flow* is a function $f : E \to \mathbb{R}$ (arbitrarily orienting the edges so that positive flows go one way and negative flows the other) such that for every vertex other than s and t the flow in equals the flow out. The *value* $|f|$ of the flow is defined to be the net flow out of the source vertex. An s-t flow f is *feasible* if $|f(e)| \le u_e$ for each edge e. The *maximum s-t flow problem* is that of finding a feasible s-t flow in G of maximum value. We denote a maximum flow in G by f^*, and we denote its value by $F^* := |f^*|$. We say that f is a $(1-\epsilon)$-approximately maximum flow if it is a feasible s-t flow of value at least $(1-\epsilon)F^*$.

To simplify the exposition, we will take ϵ to be a constant independent of m throughout the paper, and m will be assumed to be larger than some fixed constant. However, our analysis will go through unchanged as long as $\epsilon > \widetilde{\Omega}(m^{-1/3})$. In particular, our analysis will apply to all ϵ for which our given bounds are faster than the $O(m^{3/2})$ time required by existing exact algorithms.

One can easily reduce the problem of finding a $(1-\epsilon)$-approximation to the maximum flow in an arbitrary undirected graph to that of finding a $(1-\epsilon/2)$-approximation in a graph in which the ratio of the largest to smallest capacities is polynomially bounded. To do this, one should first compute a crude approximation of the maximum flow in the original graph. For example, one can compute the s-t path of maximum bottleneck in time $O(m + n \log n)$ [17, Section 8.6e], where we recall that the bottleneck of a path is the

[2]For clarity, we will analyze these two cases separately, and they will use slightly different rules for updating the resistances.

minimum capacity of an edge on that path. If this maximum bottleneck of an s-t path is B, then the maximum flow lies between B and mB. This means that there is a maximum flow in which each edge flows at most mB, so all capacities can be decreased to be at most mB. On the other hand, if one removes all edges with capacities less than $\epsilon B/2m$, the maximum flow can decrease by at most $\epsilon B/2$. So, we can assume that the minimum capacity is at least $\epsilon B/2m$ and the maximum is at most Bm, for a ratio of $2m^2/\epsilon$. Thus, by a simple scaling, we can assume that all edge capacities are integers between 1 and $2m^2/\epsilon$.

2.2 Electrical Flows

We now review some basic facts about electrical flows in networks of resistors and present a theorem that shows they can be computed quickly. For an in-depth treatment of the background material, we suggest [6].

We begin by assigning a *resistance* $r_e > 0$ to each edge $e \in E$, and we collect these resistances into a vector $\boldsymbol{r} \in \mathbb{R}^m$. For a given s-t flow f, we define its *energy* (with respect to $\boldsymbol{r}$) as

$$\mathcal{E}_r(f) := \sum_e r_e f^2(e).$$

The *electrical flow of value F (with respect to $\boldsymbol{r}$, from s to t)* is the flow that minimizes $\mathcal{E}_r(f)$ among all s-t flows f of value F. This flow is easily shown to be unique, and we note that it need not respect the capacity constraints. We recall that an electrical flow is a potential flow, which means that there is a function $\boldsymbol{\phi} : V \to \mathbb{R}$ such that the flow over an edge (u,v) is given by $(\boldsymbol{\phi}(u) - \boldsymbol{\phi}(v))/r_{u,v}$. We define the energy of a potential, written $\mathcal{E}_r(\boldsymbol{\phi})$, to be the energy of the corresponding flow.

Our analysis will make repeated use of the effective s-t resistance, one of the most basic quantities from the theory of electrical networks. Let f be the electrical s-t flow of value 1, and let $\boldsymbol{\phi}$ be its vector of vertex potentials. The *effective s-t resistance of G with respect to the resistances $\boldsymbol{r}$* is given by

$$R_{\text{eff}}(\boldsymbol{r}) = \phi(s) - \phi(t) = \mathcal{E}_r(f),$$

where the last equality is a standard result.

2.3 Electrical Flows and Laplacian Systems

While finding the maximum s-t flow corresponds to solving a linear program, we can compute the electrical flow by solving a system of linear equations. To do so, we introduce the *edge-vertex incidence matrix $\boldsymbol{B}$*, which is an $n \times m$ matrix with rows indexed by vertices and columns indexed by edges, such that

$$\boldsymbol{B}_{v,e} = \begin{cases} 1 & \text{if } e \in E^-(v), \\ -1 & \text{if } e \in E^+(v), \\ 0 & \text{otherwise.} \end{cases}$$

If we treat our flow f as a vector $\boldsymbol{f} \in \mathbb{R}^m$, where we use the orientations of the edges to determine the signs of the coordinates, the v^{th} entry of the vector $\boldsymbol{B}^T\boldsymbol{f}$ will be the difference between the flow out of and the flow into vertex v. As such, the constraints that one unit of flow is sent from s to t and that flow is conserved at all other vertices can be written as

$$B^T\boldsymbol{f} = \boldsymbol{\chi}_{s,t},$$

where $\boldsymbol{\chi}_{s,t}$ is the vector with a 1 in the coordinate corresponding to s, a -1 in the coordinate corresponding to t, and all other coordinates equal to 0.

We define the (weighted) *Laplacian* $\boldsymbol{L}$ of G (with respect to the resistances $\boldsymbol{r}$)) to be the $n \times n$ matrix

$$\boldsymbol{L} := \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^T,$$

where $\boldsymbol{C}$ is the $m \times m$ diagonal matrix with $\boldsymbol{C}_{e,e} = c_e = 1/r_e$. One can easily check that its entries are given by

$$\boldsymbol{L}_{u,v} = \begin{cases} \sum_{e \in E^+(u) \cup E^-(u)} c_e & \text{if } u = v, \\ -c_e & \text{if } e = (u,v) \text{ is an edge of } G, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{R} = \boldsymbol{C}^{-1}$ be the diagonal matrix with $\boldsymbol{R}_{e,e} = r_e$. The energy of a flow $\boldsymbol{f}$ is given by

$$\mathcal{E}_r(f) := \sum_e r_e \boldsymbol{f}(e)^2 = \boldsymbol{f}^T\boldsymbol{R}\boldsymbol{f} = \left\|\boldsymbol{R}^{1/2}\boldsymbol{f}\right\|^2.$$

The electrical flow of value 1 thus corresponds to the vector $\boldsymbol{f}$ that minimizes $\left\|\boldsymbol{R}^{1/2}\boldsymbol{f}\right\|^2$ subject to $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{\chi}_{s,t}$. If f is an electrical flow, it is well known that it is a *potential flow*, which means that there is a vector $\boldsymbol{\phi} \in \mathbb{R}^V$ such that

$$\boldsymbol{f}(u,v) = \frac{\phi_v - \phi_u}{r_{u,v}}.$$

That is,

$$\boldsymbol{f} = \boldsymbol{C}\boldsymbol{B}^T\boldsymbol{\phi} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\phi}.$$

Applying $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{\chi}_{s,t}$, we have $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^T\boldsymbol{\phi} = \boldsymbol{\chi}_{s,t}$, and hence the vertex potentials are given by

$$\boldsymbol{\phi} = \boldsymbol{L}^{\dagger}\boldsymbol{\chi}_{s,t},$$

where $\boldsymbol{L}^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{L}$. Thus, the electrical flow $\boldsymbol{f}$ is given by the expression

$$\boldsymbol{f} = \boldsymbol{C}\boldsymbol{B}^T\boldsymbol{L}^{\dagger}\boldsymbol{\chi}_{s,t}.$$

This lets us rewrite the energy of the electrical flow of value 1 as

$$\mathcal{E}_r(\boldsymbol{f}) = \boldsymbol{f}^T\boldsymbol{R}\boldsymbol{f} = \boldsymbol{\chi}_{s,t}{}^T\boldsymbol{L}^{\dagger}\boldsymbol{\chi}_{s,t} = \boldsymbol{\phi}^T\boldsymbol{L}\boldsymbol{\phi}. \tag{1}$$

2.4 Fast Computation of Electrical Flows

From the algorithmic point of view, the crucial property of the Laplacian $\boldsymbol{L}$ is that it is symmetric and *diagonally dominant*, i.e., for any u, $\sum_{v' \neq u} |\boldsymbol{L}_{u,v}| \leq \boldsymbol{L}_{u,v}$. This allows us to use the result of Koutis, Miller, and Peng [14], which builds on the work of Spielman and Teng [19], to approximately solve our linear system in nearly-linear time. By rounding the approximate solution to a flow, we can prove the following theorem (see the full version [7] for details).

Theorem 2.1 (Fast Approximation of Electrical Flows)**.** *For any $\delta > 0$, any $F > 0$, and any vector $\boldsymbol{r}$ of resistances in which the ratio of the largest to the smallest resistance is at most R, we can compute, in time $\widetilde{O}(m \log R/\delta)$, a vector of vertex potentials $\tilde{\boldsymbol{\phi}}$ and an s-t flow $\widetilde{f}$ of value F such that*

a. $\mathcal{E}_r(\tilde{\boldsymbol{\phi}}) \leq (1+\delta)\mathcal{E}_r(\boldsymbol{f})$, and $\mathcal{E}_r(\widetilde{f}) \leq (1+\delta)\mathcal{E}_r(f)$ where f is the electrical s-t flow of value F,

b. for every edge e, $|r_e f_e^2 - r_e \widetilde{f}_e^2| \leq \frac{\delta}{2mR}\mathcal{E}_r(f)$, where f is the true electrical flow.

c. $\widetilde{\boldsymbol{\phi}}_s - \widetilde{\boldsymbol{\phi}}_t \geq \left(1 - \frac{\delta}{12nmR}\right) F R_{\text{eff}}(\boldsymbol{r})$.

We will refer to a flow meeting the above conditions as a δ-approximate electrical flow*.*

3. A SIMPLE FLOW ALGORITHM

Before describing our $\widetilde{O}(m^{4/3}\epsilon^{-3})$ algorithm, we will describe a simpler algorithm that finds a $(1-\epsilon)$-approximately maximum flow in time $\widetilde{O}(m^{3/2}\epsilon^{-5/2})$. Our final algorithm will be obtained by carefully modifying the one described here.

The algorithm will employ the multiplicative weights update method, a framework established by Arora, Hazan, and Kale [3] to encompass proof techniques exploited in [16, 20, 10, 11]. In our setting, one can understand the multiplicative weights method as a way of taking an algorithm that solves a flow problem very crudely and, by calling it repeatedly, converting it into an algorithm that gives a good approximation for the maximum flow in G. The crude algorithm is called as a black-box, so it can be thought of as an oracle that answers a certain type of query.

3.1 Multiplicative Weights Method: From Electrical Flows to Maximum Flows

For an s-t flow f, we define the *congestion* of an edge e to be the ratio

$$\mathsf{cong}_f(e) := |f(e)|/u_e$$

between the flow on an edge and its capacity.

The multiplicative weights method will use a subroutine that we will refer to as an (ϵ, ρ)-*oracle*. This oracle will take as its input a number F and a vector $\boldsymbol{w}$ of edge weights. For any $F \leq F^*$, we know that there exists a way to route F units of flow in G so that all of the edge capacities are respected. Our oracle will provide a weaker guarantee: When $F \leq F^*$, it will satisfy *all* of the capacity constraints up to a multiplicative factor of ρ,[3] and it will satisfy the *average* of these constraints, weighted by the w_i, up to a (much better) multiplicative factor of $(1+\epsilon)$. When $F > F^*$, the oracle will either output an s-t flow satisfing the conditions above, or it will return **"fail"**.

Formally, we will use the following definition:

Definition 3.1 ((ϵ, ρ) oracle)**.** *For* $\epsilon > 0$ *and* $\rho > 0$*, an* (ϵ, ρ) oracle *is an algorithm that, given a real number* $F > 0$ *and a vector* $\boldsymbol{w}$ *of edge weights with* $w_e \geq 1$ *for all* e*, returns an* s-t *flow* f *such that:*

1. *If* $F \leq F^*$*, then it outputs an* s-t *flow* f *satisfying:*
   (i) $|f| = F$;
   (ii) $\sum_e w_e \mathsf{cong}_f(e) \leq (1+\epsilon)|\boldsymbol{w}|_1$*, where* $|\boldsymbol{w}|_1 := \sum_e w_e$;
   (iii) $\max_e \mathsf{cong}_f(e) \leq \rho$.
2. *If* $F > F^*$*, then it either outputs a flow* f *satisfying conditions (i)-(iii) or outputs* ***"fail"****.*

Our algorithm will be given a flow value F as an input. If $F \leq F^*$, it will return a flow of value at least $(1 - O(\epsilon))F$. If $F > F^*$, it will either return a flow of value at least $(1 - O(\epsilon))F$ (which may occur if F is only slightly greater than F^*) or it will return **"fail"**. This allows us to find a $(1 - O(\epsilon))$-approximation of F^* using binary search. As outlined in Section 2, we can obtain in time $O(m + n \log n)$ a crude bound B such that $B \leq F^* \leq mB$, so the binary search will only call our algorithm a logarithmic number of times.

In Figure 1, we present our simple algorithm, which applies the multiplicative weights update routine to approximate the maximum flow by calling an (ϵ, ρ)-flow oracle.

```
Input  : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, a target
         flow value $F$, and an $(\epsilon, \rho)$-oracle $O$
Output: Either a flow $\overline{f}$, or "fail" indicating that $F > F^*$;
Initialize $w_e^0 \leftarrow 1$ for all edges $e$, and $N \leftarrow \frac{2\rho \ln m}{\epsilon^2}$
for $i := 1, \ldots, N$ do
    Query $O$ with edge weights given by $\boldsymbol{w}^{i-1}$ and target
      flow value $F$
    if $O$ returns "fail" then return "fail"
    else
        Let $f^i$ be the returned flow
        $w_e^i \leftarrow w_e^{i-1}(1 + \frac{\epsilon}{\rho}\mathsf{cong}_{f^i}(e))$ for each $e \in E$
    end
end
return $\overline{f} \leftarrow \frac{(1-\epsilon)^2}{(1+\epsilon)N}(\sum_i f^i)$
```

Figure 1: Main Algorithm: multiplicative-weights-update routine

We analyze this algorithm by using the average congestion constraint on the flows returned by the oracle O to show that the sum of the weights does not grow too quickly. On the other hand, if an edge e consistently suffers large congestion in a sequence of flows returned by O, then its weight increases rapidly. If this were to occur too many times, its weight would exceed the total weight, which obviously cannot occur. In the full version [7], we show that the multiplicative weights algorithm in Figure 1 converges in $2\rho \ln m/\epsilon^2$ iterations, which proves the following theorem:

Theorem 3.2 (Approximating Maximum Flows by Multiplicative Weights)**.** *For any* $0 < \epsilon < 1/2$ *and* $\rho > 0$*, given an* (ϵ, ρ)*-flow oracle with running time* $T(m, 1/\epsilon, U)$*, one can obtain an algorithm that computes a* $(1 - O(\epsilon))$*-approximately maximum flow in a capacitated, undirected graph in* $\widetilde{O}(\rho\epsilon^{-2} \cdot T(m, 1/\epsilon, U))$ *time.*

We remark that this theorem follows directly from the general analysis presented in [3]. However, analyzing our improved algorithm in Section 4 requires several of the lemmas from a slightly modified version of their proof, in addition to the main result.

3.2 Constructing an Oracle of Width $3\sqrt{m/\epsilon}$ Using Electrical Flows

Given Theorem 3.2, our problem is reduced to designing an efficient oracle that has small width. We can prove that the algorithm in Figure 2 is an $\widetilde{O}(m \log \epsilon^{-1})$ time implementation of an $(\epsilon, 3\sqrt{m/\epsilon})$-flow oracle for any $0 < \epsilon < 1/2$. By Theorem 3.2, this will immediately yield an $\widetilde{O}(m^{3/2}\epsilon^{-5/2})$ time algorithm for finding an approximately maximum flow.

The intuition for the oracle is as follows. Consider the (more interesting) case when the target flow value obeys

[3]Up to factors polynomial in $1/\epsilon$, the value of ρ will be $\Theta(\sqrt{m})$ in this section, and $\widetilde{\Theta}(m^{1/3})$ later in the paper.

$F \le F^*$, and let f^* be a maximum flow. The oracle assigns a resistance of $\frac{1}{u_e^2}\left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)$ to each edge e and computes an approximate electrical of value F; for simplicity, we will ignore the approximation errors here and suppose that it computes the *exact* electrical flow $\widehat{f}$.

The electrical flow has the minimum energy among all flows of value at least F, so, in particular $\mathcal{E}_r(\widehat{f}) \le \mathcal{E}_r(f^*)$. Since the maximum flow f^* is feasible, we have $\mathsf{cong}_{f^*}(e) \le 1$ for all e, which gives

$$\mathcal{E}_r(\widehat{f}) \le \mathcal{E}_r(f^*) = \sum_e \left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)\left(\frac{f^*(e)}{u_e}\right)^2$$
$$\le \sum_e \left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right) = \left(1+\frac{\epsilon}{3}\right)|\boldsymbol{w}|_1.$$

This guarantees that the total energy of the electrical flow cannot be too large. In particular, we have

$$\sum_e w_e \left(\frac{\widehat{f}_e}{u_e}\right)^2 \le \mathcal{E}_r(\widehat{f}) \le (1+\epsilon)\,|\boldsymbol{w}|_1,$$

and, if we look at a contribution of any edge $e \in E$ to the energy then

$$\frac{\epsilon|\boldsymbol{w}|_1}{3m}\left(\frac{\widehat{f}_e}{u_e}\right)^2 \le (w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m})\left(\frac{\widehat{f}_e}{u_e}\right)^2 \le \mathcal{E}_r(\widehat{f}) \le (1+\epsilon)\,|\boldsymbol{w}|_1.$$

Rearranging the latter inequality and noting that $\mathsf{cong}_{\widehat{f}}(e) = \widehat{f}_e/u_e$ gives $\mathsf{cong}_{\widehat{f}}(e) \le O(\sqrt{m/\epsilon})$, which provides our desired bound on the maximum congestion. Applying Cauchy-Schwarz to the former inequality gives us

$$\sum_e w_e \mathsf{cong}_{\widehat{f}}(e) \le \sqrt{1+\epsilon}\,|\boldsymbol{w}|_1 < (1+\epsilon)\,|\boldsymbol{w}|_1,$$

which is the required bound on the average congestion.

In the full version [7], we provide a complete proof of the oracle's correctness and running time. Then by Theorem 3.2, we have:

```
Input  : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, a target
         flow value $F$, and edge weights $\{w_e\}_e$
Output: Either a flow $\widetilde{f}$, or "fail" indicating that $F > F^*$
$r_e \leftarrow \frac{1}{u_e^2}\left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)$ for each $e \in E$
Find an $(\epsilon/3)$-approximate electrical flow $\widetilde{f}$ using
    Theorem 2.1 on $G$ with resistances $\boldsymbol{r}$ and target flow
    value $F$
if $\mathcal{E}_r(\widetilde{f}) > (1+\epsilon)|\boldsymbol{w}|_1$ then return "fail"
else return $\widetilde{f}$
```

Figure 2: A simple implementation of an $\left(\epsilon, 3\sqrt{m/\epsilon}\right)$ oracle

Theorem 3.3. *For any $0 < \epsilon < 1/2$, the maximum flow problem can be $(1-\epsilon)$-approximated in $\widetilde{O}(m^{3/2}\epsilon^{-5/2})$ time.*

4. A FASTER FLOW ALGORITHM

In this section, we modify our algorithm to make it run in time $\widetilde{O}(m^{4/3}\epsilon^{-3})$. We then combine it with the smoothing and sampling techniques of Karger [13] to obtain an $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$-time algorithm.

For fixed ϵ, the algorithm in the previous section required us to compute $\widetilde{O}(m^{1/2})$ electrical flows, each of which took time $\widetilde{O}(m)$, which led to a running time of $\widetilde{O}(m^{3/2})$. To reduce this to $\widetilde{O}(m^{4/3})$, we'll show how to find an approximate flow while computing only $\widetilde{O}(m^{1/3})$ electrical flows.

Our analysis of the oracle from Section 3 was fairly simplistic, and one might hope to improve the running time of the algorithm by proving a tighter bound on the width. Unfortunately, the graph in Figure 3 shows that our analysis was essentially tight. The graph consists of k parallel paths of length k connecting s to t, along with a single edge e that directly connects s to t. The max flow in this graph is $k+1$. In the first call made to the oracle by the multiplicative weights routine, all of the edges will have the same resistance. In this case, the electrical flow of value $k+1$ will send $(k+1)/2k$ units of flow along each of the k paths and $(k+1)/2$ units of flow across e. Since the graph has $m = \Theta(k^2)$, the width of the oracle in this case is indeed $\Theta(m^{1/2})$.

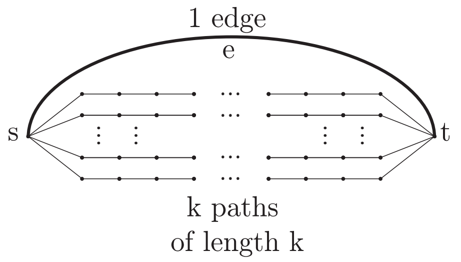

Figure 3: A graph on which the electrical flow sends approximately $\sqrt{m}$ units of flow across an edge when sending the maximum flow F^* from s to t.

4.1 The Improved Algorithm

The above example shows that it is possible for the electrical flow returned by the oracle to exceed the edge capacities by $\Theta(m^{1/2})$. However, the key observation to make here is that if one removes the edge e from the graph in Figure 3, the electrical flow on the resulting graph is much better behaved, but the value of the maximum flow is only very slightly reduced. This demonstrates a phenomenon that will be central to our improved algorithm: while instances in which the electrical flow sends a huge amount of flow over some edges exist, they are somewhat fragile, and they are often drastically improved by removing the bad edges.

This motivates us to modify our algorithm as follows. We decrease ρ to some value that we will fix later. (Up to factors polynomial in $1/\epsilon$, this value will be $\widetilde{O}(m^{1/3})$.) As before, the oracle will begin by computing an electrical flow. However, when this electrical flow exceeds the capacity of some edge e by a factor greater than ρ, we'll remove e from the graph and try again, keeping all of the weights the same. We'll repeat this process until we obtain a flow in which all edges flow at most a factor of ρ times their capacity (or some failure condition is reached), and we'll use this flow in our multiplicative weights routine. When the oracle removes an edge, it is added to a set H of *forbidden edges*. These edges will be permanently removed from the graph, i.e., they will not be included in the graphs supplied to future invocations of the oracle.

In Figures 4 and 5, we present the modified versions of the oracle and overall algorithm. We have highlighted the parts that have changed from the simpler version shown in Figures 1 and 2.

```
Input  : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, a target
         flow value $F$, edge weights $\{w_e\}_e$,
         and a set $H$ of forbidden edges
Output: Either a flow $\widetilde{f}$ and a set $H$ of forbidden edges, or
         "fail" indicating that $F > F^*$

$\rho \leftarrow \frac{8m^{1/3}\ln^{1/3} m}{\epsilon}$
$r_e \leftarrow \frac{1}{u_e^2}\left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)$ for each $e \in E \setminus H$
Find an $(\epsilon/3)$-approximate electrical flow $\widetilde{f}$ using
    Theorem 2.1 on $G_H := (V, E \setminus H)$ with resistances $\boldsymbol{r}$
    and target flow value $F$
if $\mathcal{E}_r(\widetilde{f}) > (1+\epsilon)|\boldsymbol{w}|_1$ or $s$ and $t$ disconnected in $G_H$
then return "fail"
if there exists $e$ with $\mathrm{cong}_{\widetilde{f}}(e) > \rho$
then add $e$ to $H$ and start over
return $\widetilde{f}$
```

Figure 4: The modified oracle O' used by our improved algorithm

```
Input  : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, and a
         target flow value $F$;
Output: Either a flow $\overline{f}$, or "fail" indicating that $F > F^*$;

Initialize $w_e^0 \leftarrow 1$ for all edges $e$, $H \leftarrow \emptyset$,
$\rho \leftarrow \frac{8m^{1/3}\ln^{1/3} m}{\epsilon}$, and $N \leftarrow \frac{2\rho \ln m}{\epsilon^2}$
for $i := 1, \ldots, N$ do
    Query $O'$ with edge weights given by $\boldsymbol{w}^{i-1}$, target
      flow value $F$, and forbidden edge set $H$
    if $O$ returns "fail" then return "fail"
    else
        Let $f^i$ be the returned answer
        Replace $H$ with the returned (augmented) set of
          forbidden edges
        $w_e^i \leftarrow w_e^{i-1}(1 + \frac{\epsilon}{\rho}\mathrm{cong}_{f^i}(e))$ for each $e \in E$
    end
end
return $\overline{f} \leftarrow \frac{(1-\epsilon)^2}{(1+\epsilon)N}(\sum_i f^i)$
```

Figure 5: The faster approximation algorithm for maximum flow

4.2 Analysis of the New Algorithm

It will be helpful to examine what is already guaranteed by the analysis from Section 3, and what we'll need to show to demonstrate the algorithm's correctness and bound its running time.

We first note that, by construction, the congestion of any edge in the flow $\widetilde{f}$ returned by the modified oracle from Figure 4 will be bounded by ρ. Furthermore, it enforces the bound $\mathcal{E}_r(\widetilde{f}) \leq (1+\epsilon)|\boldsymbol{w}|_1$, which guarantees that $\widetilde{f}$ will meet the weighted average congestion bound required for a (ϵ, ρ)-oracle. So, as long as the modified oracle always successfully returns a flow, it will function as an (ϵ, ρ)-oracle, and our analysis from the full paper [7] shows that the multiplicative update scheme employed by our algorithm will yield an approximate maximum flow after $\widetilde{O}(\rho)$ iterations.

Our problem is thus reduced to understanding the behavior of the modified oracle. To prove correctness, we will need to show that whenever the modified oracle is called with $F \leq F^*$, it will return some flow $\widetilde{f}$ (as opposed to returning **"fail"**). To bound the running time, we will need to provide an upper bound on the total number of electrical flows computed by the modified oracle throughout the execution of the algorithm.

To this end, in the full version of the paper, we show the following upper bound on the cardinality $|H|$ and the capacity $u(H)$ of the set of forbidden edges.

Lemma 4.1. *Throughout the execution of the algorithm,* $|H| \leq \frac{30m\ln m}{\epsilon^2\rho^2}$ *and* $u(H) \leq \frac{30mF\ln m}{\epsilon^2\rho^3}$.

If we plug in the value $\rho = (8m^{1/3}\ln^{1/3} m)/\epsilon$ used by the algorithm, Lemma 4.1 gives the bounds $|H| \leq \frac{15}{32}(m\ln m)^{1/3}$ and $u(H) \leq \frac{15}{256}\epsilon F < \epsilon F/12$.

We prove Lemma 4.1 by tracking the effective s-t resistance of the circuits on which we compute electrical flows. The key insight is that we only cut an edge when it flows a lot and thus its flow accounts for a nontrivial fraction of the energy of the electrical flow. We show in full paper [7] that cutting such an edge will cause a substantial increase in the effective resistance. Combining this with an absolute upper bound on how much the effective resistance can increase during the execution of the algorithm will guarantee that we won't cut too many edges.

Given the above lemma, it is now straightforward to show the following theorem, which establishes the correctness and bounds the running time of our algorithm. (See the full version of the paper for details.)

Theorem 4.2. *For any* $0 < \epsilon < 1/2$, *if* $F \leq F^*$ *the algorithm in Figure 5 will return a feasible* s-t *flow* $\overline{f}$ *of value* $|\overline{f}| = (1 - O(\epsilon))F$ *in time* $\widetilde{O}(m^{4/3}\epsilon^{-3})$.

The above theorem allows us to apply the binary search strategy that we used in Section 3.1. This yields our main theorem:

Theorem 4.3. *For any* $0 < \epsilon < 1/2$, *the maximum flow problem can be* $(1-\epsilon)$*-approximated in* $\widetilde{O}(m^{4/3}\epsilon^{-3})$ *time.*

4.3 Further Improvement to $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$

We can now combine our algorithm with existing methods to further improve its running time. In [13] (see also [5]), Karger presented a technique, which he called "graph smoothing", that allows one to use random sampling to speed up an exact or $(1-\epsilon)$-approximate flow algorithm. More precisely, his techniques yield the following theorem, which is implicit in [13] and stated in a more similar form in [5]:

Theorem 4.4 ([13, 5]). *Let* $T(m, n, \epsilon)$ *be the time needed to find a* $(1-\epsilon)$*-approximately maximum flow in an undirected, capacitated graph with* m *edges and* n *vertices. Then one can obtain a* $(1-\epsilon)$*-approximately maximal flow in such a graph in time*

$$\widetilde{O}(\epsilon^2 m/n \cdot T(\widetilde{O}(n\epsilon^{-2}), n, \Omega(\epsilon))).$$

By applying the above theorem to our $\widetilde{O}(m^{4/3}\epsilon^{-3})$ algorithm, we obtain our desired running time bound:

Theorem 4.5. *For any $0 < \epsilon < 1/2$, the maximum flow problem can be $(1-\epsilon)$-approximated in $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$ time.*

5. DUAL MINIMUM-CUT ALGORITHM

In this section, we'll describe a dual perspective that leads to an even simpler algorithm for computing an approximately minimum s-t cut. The algorithm will eschew the oracle abstraction and multiplicative weights machinery. Instead, it will just repeatedly compute an electrical flow, increase the resistances of edges according to the amount flowing over them, and repeat. It will then use the electrical potentials of the last flow computed to find a cut by picking a cutoff and splitting the vertices according to whether their potentials are above or below the cutoff. The algorithm is shown in Figure 6. We will show that it finds a $(1+\epsilon)$-approximately minimum s-t cut in time $\widetilde{O}(m^{4/3}\epsilon^{-8/3})$.

Given any weighted undirected graph $G = (V, E, w)$ with n vertices and m edges, Benczúr and Karger [4] showed that one can construct a graph $G' = (V, E', w')$ (called a *sparsifier of G*) on the same vertex set in time $\widetilde{O}(m)$ such that $|E'| = O(n \log n/\epsilon^2)$ and the capacity of any cut in G' is between 1 and $(1+\epsilon)$ times its capacity in G. Applying our algorithm to a sparsifier will give us:

Theorem 5.1. *For any $0 < \epsilon < 1/7$, we can find a $(1+\epsilon)$-approximate minimum s-t cut in $\widetilde{O}(m + n^{4/3}\epsilon^{-16/3})$ time.*

We note that, in this algorithm, there is no need to deal explicitly with edges flowing more than ρ, maintain a set of forbidden edges, or average the flows from different steps. We will separately study edges with very large flow in our analysis, but the algorithm itself avoids the complexities that appeared in the improved flow algorithm described in Section 4.

```
Input  : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, and a
         target flow value $F$
Output: A cut $(S, V \setminus S)$
Initialize $w_e^0 \leftarrow 1$ for all edges $e$, $\rho \leftarrow 3m^{1/3}\epsilon^{-2/3}$,
$N \leftarrow 5\epsilon^{-8/3}m^{1/3}\ln m$, and $\delta \leftarrow \epsilon^2$.
for $i := 1, \ldots, N$ do
    Find a $\delta$-approximate electrical flow $\widetilde{f}^{i-1}$
        and potentials $\tilde{\phi}$ using Theorem 2.1 on $G$
        with resistances $r_e^{i-1} = \frac{w_e^{i-1}}{u_e^2}$ and target flow
        value $F$
    Update the weights:
        $\mu^{i-1} \leftarrow \sum_e w_e^{i-1}$
        $w_e^i \leftarrow w_e^{i-1} + \frac{\epsilon}{\rho}\mathrm{cong}_{\widetilde{f}^{i-1}}(e) w_e^{i-1} + \frac{\epsilon^2}{m\rho}\mu^{i-1}$
          for each $e \in E$
    Check if potentials give a small cut:
        Scale and translate $\tilde{\phi}$ so that $\tilde{\phi}_s = 1$ and $\tilde{\phi}_t = 0$
        Let $S_x = \{v \in V \mid \phi_v > x\}$
        Let $S$ be the set $S_x$ that minimizes $(S_x, V \setminus S_x)$
        if capacity of $(S, V \setminus S)$ is less than $F/(1-7\epsilon)$
          then return $(S, V \setminus S)$
end
return "fail"'
```

Figure 6: A dual algorithm for finding an s-t cut

5.1 An Overview of the Analysis

To analyze this algorithm, we will track the total weight placed on the edges crossing some minimum cut. The basic observation for our analysis is that the same amount of net flow must be sent across every cut, so edges in small cuts will tend to have higher congestion than edges in large cuts. Since our algorithm increases the weight of an edge according to its congestion, this will cause our algorithm to concentrate a larger and larger fraction of the total weight on the small cuts of the graph. This will continue until almost all of the weight is concentrated on approximately minimum cuts.

We will use the effective resistance between s and t to measure the extent to which weight is concentrated on approximately minimum cuts. In particular, we will show that if we can make the effective resistance large enough then we can find a cut of small capacity.

5.2 Cuts, Electrical Potentials, and Effective Resistance

Given potentials ϕ, the dual algorithm returns a set S_x of the form $\{v : \phi_v > x\}$.

Lemma 5.2. *For $\tilde{\phi}$ such that $\tilde{\phi}_s = 1$, $\tilde{\phi}_t = 0$ and $\tilde{\phi}_x \in [0, 1]$ for all x, there is a cut S_x of capacity at most*

$$\sum_{(u,v)\in E} |\tilde{\phi}_u - \tilde{\phi}_v| u_{(u,v)}. \tag{2}$$

Proof. Consider choosing $x \in [0, 1]$ uniformly at random. The probability that an edge (u, v) is cut is precisely $|\tilde{\phi}(u) - \tilde{\phi}(v)|$. So, the expected capacity of the edges in a random cut is given by (2), and so there is a cut of capacity at most (2). □

The following lemma tell us that a small cut can be found from the electrical potentials when the effective resistance is high.

Lemma 5.3. *Let $\mu = \sum_e u_e^2 r_e$, and let $R_{\mathrm{eff}}(\boldsymbol{r})$ be the effective s-t resistance of G with edge resistances given by $\boldsymbol{r}$. Let $\boldsymbol{\phi}$ be the potentials of the electrical s-t flow, scaled to have potential drop 1 between s and t. Then*

$$\sum_{e\in E} \boldsymbol{\phi}(e) u_e \le \sqrt{\frac{\mu}{R_{\mathrm{eff}}(\boldsymbol{r})}}.$$

If, $\tilde{\boldsymbol{\phi}}$ is an approximate electrical potential returned by the algorithm of Theorem 2.1 when run with parameter $\delta \le 1/3$, re-scaled to have potential difference 1 between s and t, then

$$\sum_{e\in E} \tilde{\boldsymbol{\phi}}(e) u_e \le (1+2\delta)\sqrt{\frac{\mu}{R_{\mathrm{eff}}(\boldsymbol{r})}}.$$

5.3 The Proof that the Dual Algorithm Finds an Approximately Minimum Cut

We'll show that if $F \ge F^*$ then within $N = 5\epsilon^{-8/3}m^{1/3}\ln m$ iterations, the algorithm in Figure 6 will produce a set of resistances $\boldsymbol{r}^i$ such that $R_{\mathrm{eff}}(\boldsymbol{r}^i) \ge (1-7\epsilon)\mu^i/(F)^2$. Once such a set of resistances has been obtained, Lemmas 5.2 and 5.3 tell us that the best potential cut of $\tilde{\boldsymbol{\phi}}$ will have capacity at most $F/(1-7\epsilon)$. The algorithm will then return this cut.

Let C be the set of edges crossing some minimum cut in our graph. Let $u_C = F^*$ denote the capacity of the edges in C. We will keep track of two quantities: the weighted geometric mean of the weights of the edges in C,

$$\nu^i = \Big(\prod_{e \in C} \left(w_e^i\right)^{u_e} \Big)^{1/u_C},$$

and the total weight of edges,

$$\mu^i = \sum_e w_e^i = \sum_e r_e^i u_e^2.$$

Clearly $\nu^i \leq \max_{e \in C} w_e^i$. In particular, $\nu^i \leq \mu^i$ for all i.

Our proof that the effective resistance cannot remain large for too many iterations will be similar to our analysis of the flow algorithm in Section 4. We suppose to the contrary that $\text{Reff}_{st}^i \leq (1-7\epsilon)\frac{\mu^i}{(F)^2}$ for each $1 \leq i \leq N$. We will show that, under this assumption:

1. The total weight μ^i doesn't get too large over the course of the algorithm [**Lemma 5.4**].
2. The quantity ν^i increases significantly in any iteration in which no edge

   as congestion more than ρ [**Lemma 5.5**]. Since μ^i doesn't get too large, and $\nu^i \leq \mu^i$, this will not happen too many times.
3. The effective resistance increases significantly in any iteration in which some edge has congestion more than ρ [**Lemma 5.6**]. Since μ^i does not get too large, and the effective resistance is assumed to be bounded in terms of μ^i, this cannot happen too many times.

The combined bounds from (2) and (3) will be less than N, which will yield a contradiction.

The precise statements of the necessary lemmas are given below. We refer the reader to the full paper [7] for their proofs.

Lemma 5.4. *For each $i \leq N$ such that*

$$R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$$

and

$$\mu^{i+1} \leq \mu^i \exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}\right).$$

Lemma 5.5. *If $\mathsf{cong}_{f^i}(e) \leq \rho$ for all e, then*

$$\nu^{i+1} \geq \exp\left(\frac{\epsilon(1-\epsilon)}{\rho}\right)\nu^i.$$

Lemma 5.6. *If $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$ and there exists some edge e such that $\mathsf{cong}_{\widetilde{f}^i}(e) > \rho$, then*

$$R_{\text{eff}}(\boldsymbol{r}^{i+1}) \geq \exp\left(\frac{\epsilon^2\rho^2}{4m}\right) R_{\text{eff}}(\boldsymbol{r}^i).$$

We combine these lemmas to prove our main bound:

Lemma 5.7. *For $\epsilon \leq 1/7$, after N iterations, the algorithm in Figure 6 will produce a set of resistances such that $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$.*

Our main result follows from combining this with Lemmas 5.2 and 5.3.

Theorem 5.8. *On input $\epsilon < 1/7$, the algorithm in Figure 6 runs in time $\widetilde{O}(m^{4/3}\epsilon^{-8/3})$. If $F \geq F^*$, then it returns a cut of capacity at most $F/(1-7\epsilon)$, where F^* is the minimum capacity of an s-t cut.*

To use this algorithm to find a cut of approximately minimum capacity, one should begin as with the flow algorithm by crudely approximating the minimum cut, and then applying the above algorithm together with binary search. This will incur a multiplicative overhead of $O(\log m/\epsilon)$ in the running time.

Acknowledgments: We would like to thank David Karger for bringing Theorem 4.4 to our attention and Sanjeev Arora for his feedback on a preliminary version of this paper.

6. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Elsevier North-Holland, Inc., New York, NY, USA, 1989.

[2] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. R. Reddy. Applications of network optimization. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 1–75. North-Holland, 1995.

[3] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: A meta-algorithm and applications. Available at http://www.cs.princeton.edu/˜arora/pubs/MWsurvey.pdf.

[4] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 47–55, New York, NY, USA, 1996. ACM.

[5] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *CoRR*, cs.DS/0207078, 2002.

[6] B. Bollobas. *Modern Graph Theory*. Springer, 1998.

[7] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. *CoRR*, abs/1010.2921, 2010.

[8] S. I. Daitch and D. A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 451–460, 2008.

[9] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, Dec. 1975.

[10] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal of Discrete Mathematics*, 13(4):505–520, 2000.

[11] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, 1998.

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

[13] D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In *Proceedings of the 9th Annual Symposium on Discrete Algorithms*, pages

490–499, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[14] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD systems. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, 2010.

[15] A. Mądry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, 2010.

[16] S. A. Plotkin, D. B. Shmoys, and É.. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[17] A. Schrijver. *Combinatorial Optimization, Volume A*. Number 24 in Algorithms and Combinatorics. Springer, 2003.

[18] J. Sherman. Breaking the multicommodity flow barrier for $O\left(\sqrt{\log n}\right)$-approximations to sparsest cut. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, 2009.

[19] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.

[20] N. E. Young. Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 170–178, San Francisco, California, 22–24 Jan. 1995.