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

Paul Christiano
MIT

Jonathan A. Kelner*
MIT

Aleksander Mądry†
EPFL

Daniel Spielman‡
Yale University

Shang-Hua Teng§
University of Southern California

July 29, 2013

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] $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$. A dual version of our approach computes a $(1+\epsilon)$-approximately minimum s-t cut in time $\widetilde{O}\left(m+n^{4/3}\epsilon^{-16/3}\right)$, which is the fastest known algorithm for this problem as well. 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 $\widetilde{O}\left(mn^{1/2}\epsilon^{-1}\right)$, and approximately minimum s-t cuts in time $\widetilde{O}\left(m+n^{3/2}\epsilon^{-3}\right)$.

*Research partially supported by NSF grant CCF-0843915.

†This work was done while the author was a student at MIT. Supported in part by NSF grant CCF-0829878 and by ONR grant N00014-05-1-0148.

‡This material is based upon work supported by the National Science Foundation under Grant Nos. 0634957, 0634904 and 0915487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

§This research is in part supported by NSF grants 1032367, 0964481, and a USC Viterbi School of Engineering startup grant which is in turn supported by a Powell Foundation Award.

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

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 [26, 2]. They have many applications (see [3]) and are often used as subroutines in other algorithms (see [4, 27]). Many advances have been made in the development of algorithms for this problem (see Goldberg and Rao [14] 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 [10], 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 this problem, they are simple to describe and introduce techniques that may be applicable to other problems. Our algorithms 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 [29, 19, 20, 18].

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 from sending current from the source s to the sink t. Such current obeys the flow conservation constraints, but might not respect the 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 $\widetilde{O}\left(m^{1/3} \cdot \mathrm{poly}(1/\epsilon)\right)$ times, we are 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 are 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 $\widetilde{O}\left(m^{4/3} \cdot \mathrm{poly}(1/\epsilon)\right)$. By combining this with the graph smoothing and sampling techniques of Karger [15], we can get a $(1-\epsilon)$-approximately maximum s-t flow in time $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$. Furthermore, by applying the cut algorithm to a sparsifier [5] of the input graph, we can compute a $(1+\epsilon)$-approximately minimum s-t cut in time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-16/3}\right)$.

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 [27] allows us to achieve the best currently known approximation ratio of $O(\sqrt{\log n})$ for the sparsest cut problem in time $\widetilde{O}\left(m + n^{4/3}\right)$.

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 [14] and Benczúr and Karger [6]. In a breakthrough paper, Gold-

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

berg and Rao [14] 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 algorithms for these problems by using the graph sparsification techniques of Benczúr and Karger [5, 6]. Goldberg and Rao [14] observe that by running their exact maximum flow algorithm on the sparsifiers of Benczúr and Karger [5], one can obtain a $(1+\epsilon)$-approximately minimum cut in time $\widetilde{O}\left(m + n^{3/2}\epsilon^{-3}\right)$. This provides an approximation of the value of the maximum flow. To actually obtain a feasible flow whose value approximates the maximum one can combine the algorithm of Goldberg and Rao with graph smoothing technique of Karger [15] (see also [6]). This provides a $(1-\epsilon)$-approximately maximum flow in time $\widetilde{O}\left(m\sqrt{n}\epsilon^{-1}\right)$. We refer the reader to the paper of Goldberg and Rao for a survey of previous breakthroughs in the development of algorithms for computing maximum s-t flows.

In more recent work, Daitch and Spielman [8] showed that fast solvers for Laplacian linear systems [29, 19, 20, 18] could be used to make interior-point algorithms for the maximum flow and minimum-cost flow problems run in time $\widetilde{O}\left(m^{3/2} \log U\right)$, and Mądry [22] showed that one can approximately solve a wide range of cut problems, including the minimum s-t cut problem, within a polylogarithmic factor in almost linear time.

1.2 Subsequent Work

Since the initial publication of this paper, there was a number of developments on the maximum s-t flow and minimum s-t cut problems. First, Kelner et al. [17] showed how to extend our approach to tackle maximum concurrent multicommodity flow problem. They obtain an $(1-\epsilon)$-approximation algorithm for that problem that runs in time $\widetilde{O}\left(m^{4/3} poly(k, 1/\epsilon)\right)$, where k is the number of commodities involved.

Later on, Lee et al. [21] developed a different framework for approximation of the maximum s-t flow and the minimum s-t cut in undirected graphs. Their framework is also based on iterative computation of electrical flows, but employs a purely gradient-descent perspective and makes use of the accelerated gradient-descent method of Nesterov [24] instead of multiplicative-weights-update method. This framework enables them to obtain a $(1-\epsilon)$-approximation algorithm for both the maximum s-t flow and the minimum s-t cut problem in unit-capacity graphs that runs in time $\widetilde{O}\left(mn^{1/3}\epsilon^{-2/3}\right)$, and thus achieves significantly better dependence on ϵ in that case.

Recently, this line of research was culminated with independent results of Sherman [28] and Kelner et al. [16], who showed that one can compute a $(1-\epsilon)$-approximation to undirected maximum s-t flow problem in time $O(m^{1+o(1)}\epsilon^{-2})$. Their approaches have different flavor, but can be viewed as being somewhat dual to each other. Roughly speaking, they are based on developing an understanding of gradient-descent method with respect to non-Euclidean norms and combining it with the ideas behind the ultra-sparsifier construction of Spielman and Teng [29], as well as, the fast poly-logarithmic-approximation algorithms for cut problems of Mądry [22].

Finally, in parallel to the above progress on approximation algorithms for undirected maximum s-t flow and minimum s-t cut, there was also a progress on exact algorithms for these problems in directed graphs. Namely, Mądry [23] presented exact algorithms for the maximum s-t flow and minimum s-t cut problems in directed unit-capacity graphs that run in $\widetilde{O}\left(m^{10/7}\right)$ time. These algorithms employ a primal-dual framework that combines the electrical flow computations with the ideas underlying path-following interior point methods.

1.3 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 $\widetilde{O}\left(m^{3/2}\epsilon^{-5/2}\right)$. In Section 4, we will show how to improve the running time of our algorithm to $\widetilde{O}\left(m^{4/3}\epsilon^{-3}\right)$; we will then describe how to combine this with existing graph smoothing and sparsification techniques to compute approximately maximum s-t flows in time $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$ and to approximate the value of such flows in time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-17/3}\right)$. In Section 5, we present a variant of our algorithm that computes approximately minimum s-t cuts in time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-16/3}\right)$.

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 arbitrarily orient each edge in E; this divides the edges incident to a vertex $v \in V$ into the set $E^+(v)$ of edges oriented towards v and the set $E^-(v)$ of edges oriented away from v. These orientations are merely for notational convenience. We use them to interpret the meaning of a positive flow on an edge. If an edge has positive flow and is in $E^+(v)$, then the flow is towards v. Conversely, if it has negative flow then the flow is away from v. One should keep in mind that our graphs are undirected and that the flow on an edge can go in either direction, regardless of the edge's orientation.

We now define our primary objects of study, s-t cuts and s-t flows.

Definition 2.1 (Cuts)**.** *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 $V \setminus S$.*

Definition 2.2 (Flows)**.** *An s-t* flow *is a function $f : E \to \mathbb{R}$ that obeys the* flow-conservation constraints

$$\sum_{e \in E^-(v)} f_e - \sum_{e \in E^+(v)} f_e = 0 \quad \text{for all } v \in V \setminus \{s, t\}.$$

The value *$|f|$ of the flow is defined to be the net flow out of the source vertex, $|f| := \sum_{e \in E^-(s)} f_e - \sum_{e \in E^+(s)} f_e$.*

It follows easily from the flow conservation constraints that the net flow out of s is equal to the net flow into t, so $|f|$ may be interpreted as the amount of flow that is sent from s to t.

2.2 Maximum Flows and Minimum Cuts

We say that an s-t flow f is *feasible* if $|f_e| \leq u_e$ for each edge e, i.e., if the amount of flow routed through any edge does not exceed its capacity. 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 (with the given

capacities) 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 $O(m + n \log n)$ [26, Section 8.6e], where we recall that the bottleneck of a path is the minimum capacity of an edge on that path. If the 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 the edges with capacities less $\epsilon B/2m$, 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 reduce our considerations to the case in which all edge capacities lie between 1 and $2m^2/\epsilon$.

The *minimum s-t cut problem* is that of finding the s-t cut of minimum capacity. The *Max Flow-Min Cut Theorem* ([12, 9]) states that the capacity of the minimum s-t cut is equal to F^*, the value of the maximum s-t flow.

In particular, the Max Flow-Min Cut Theorem implies that one can use the capacity of any s-t cut as an upper bound on the value of any feasible s-t flow, and that the task of finding the *value* of the maximum flow is equivalent to the task of finding the *capacity* of a minimum s-t cut.

One should note, however, that the above equivalence applies only to the values of the flow and the capacity and that although one can easily obtain a minimum s-t cut of a graph given its maximum flow, there is no known procedure that obtains a maximum flow from minimum s-t cut more efficiently than just by computing the maximum flow from scratch.

2.3 Electrical Flows and the Nearly Linear Time Laplacian Solver

In this section, we review some basic facts about electrical flows in networks of resistors and present a theorem that allows us to quickly approximate these flows. For an in-depth treatment of the background material, we refer the reader to [7].

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 $\boldsymbol{r}$) to be

$$\mathcal{E}_{\boldsymbol{r}}(f) := \sum_e r_e f_e^2.$$

The *electrical flow of value F (with respect to $\boldsymbol{r}$, from s to t)* is the flow that minimizes $\mathcal{E}_{\boldsymbol{r}}(f)$ 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.

From a physical point of view, the electrical flow of value one corresponds to the current that is induced in G if we view it as an electrical circuit in which each edge e has resistance of r_e, and we send one unit of current from s to t, say by attaching s to a current source and t to ground.

2.3.1 Electrical Flows and Linear Systems

While finding the maximum s-t flow corresponds to solving an appropriate 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

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

where $\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}$ 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

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

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

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

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

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

2.3.2 Effective s-t Resistance and Effective s-t Conductance

Our analysis will make repeated use of two basic quantities from the theory of electrical networks, the effective s-t resistance and effective s-t conductance.

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

$$R_{\text{eff}}(\boldsymbol{r}) = \phi(s) - \phi(t).$$

Using our linear algebraic description of the electrical flow and Equation (1), we have

$$R_{\text{eff}}(\boldsymbol{r}) = \phi(s) - \phi(t) = \boldsymbol{\chi}_{s,t}{}^T\boldsymbol{\phi} = \boldsymbol{\chi}_{s,t}{}^T\boldsymbol{L}^\dagger\boldsymbol{\chi}_{s,t} = \mathcal{E}_r(\boldsymbol{f}).$$

This gives us an alternative description of the effective s-t resistance as the energy of the electrical flow of value 1.

It will sometimes be convenient to use the related notion of the *effective s-t conductance of G with respect to the resistances $\boldsymbol{r}$*, which we define by

$$C_{\text{eff}}(\boldsymbol{r}) = 1/R_{\text{eff}}(\boldsymbol{r}).$$

We note that this is equal to the value of the electrical s-t flow in which $\phi(s) - \phi(t) = 1$.

2.3.3 Approximately Computing 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, $\sum_{v \neq u} |\boldsymbol{L}_{u,v}| \leq \boldsymbol{L}_{u,u}$. This allows us to use the result of Koutis, Miller, and Peng [20], which builds on the work of Spielman and Teng [29] (see also [19, 18]), 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 Appendix A for a proof).

Theorem 2.3 (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 $\widetilde{O}\left(m \log R/\delta\right)$, a vector of vertex potentials $\tilde{\boldsymbol{\phi}}$ and an s-t flow $\tilde{f}$ of value F such that*

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

b. for every edge e,

$$\left|r_e f_e^2 - r_e \tilde{f}_e^2\right| \leq \frac{\delta}{2mR}\mathcal{E}_{\boldsymbol{r}}(f),$$

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*.*

2.4 How the Resistance of an Edge Influences the Effective Resistance

In this section, we will study how changing the resistance of an edge affects the effective resistance of the graph. This will be a key component of the analysis of our $\widetilde{O}\left(m^{4/3} \cdot \text{poly}(1/\epsilon)\right)$ algorithms. We will make use of the following standard fact about effective conductance; we refer the reader to Bollobas [7, Chapter IX.2, Corollary 5] for a proof.

Fact 2.4. *For any* $G = (V, E)$ *and any vector of resistances* $\boldsymbol{r}$*,*

$$C_{\text{eff}}(\boldsymbol{r}) = \min_{\substack{\boldsymbol{\phi} \mid \phi(s)=1, \\ \phi(t)=0}} \sum_{(u,v)\in E} \frac{(\phi(u) - \phi(v))^2}{r_{(u,v)}}.$$

Furthermore, the minimum is attained when $\boldsymbol{\phi}$ *is the vector of vertex potentials corresponding to the electrical* s*-*t *flow in* G *(with respect to* $\boldsymbol{r}$*) of value* $1/R_{\text{eff}}(\boldsymbol{r})$*.*

Corollary 2.5 (Rayleigh Monotonicity)**.** *If* $r'_e \geq r_e$ *for all* $e \in E$*, then* $R_{\text{eff}}(\boldsymbol{r}') \geq R_{\text{eff}}(\boldsymbol{r})$*.*

Proof. For any $\boldsymbol{\phi}$,

$$\sum_{(u,v)\in E} \frac{(\phi(u) - \phi(v))^2}{r'_{(u,v)}} \leq \sum_{(u,v)\in E} \frac{(\phi(u) - \phi(v))^2}{r_{(u,v)}},$$

so the minima of these expressions over possible values of $\boldsymbol{\phi}$ obey the same relation, and thus $C_{\text{eff}}(\boldsymbol{r}') \leq C_{\text{eff}}(\boldsymbol{r})$. Inverting both sides of this inequality yields the desired result. □

Our analysis of the $\widetilde{O}\left(m^{4/3}\right)$ algorithm will require the following lemma, which gives a lower bound on the change in effective resistance when the resistance of an edge increases.

Lemma 2.6. *Let* f *be an electrical* s*-*t *flow on a graph* G *with resistances* $\boldsymbol{r}$*. Suppose that some edge* $h = (i, j)$ *accounts for a* β *fraction of the total energy of* f*, i.e.,*

$$f_h^2 r_h = \beta \mathcal{E}_{\boldsymbol{r}}(f).$$

For some $\gamma > 0$*, define new resistances* $\boldsymbol{r}'$ *such that* $r'_h = \gamma r_h$*, and* $r'_e = r_e$ *for all* $e \neq h$*. Then*

$$R_{\text{eff}}(\boldsymbol{r}') \geq \frac{\gamma}{\beta + \gamma(1-\beta)} R_{\text{eff}}(\boldsymbol{r}).$$

In particular:

- *If we "cut" the edge* h *by setting* $\gamma = \infty$*, then*

$$R_{\text{eff}}(\boldsymbol{r}') \geq \frac{R_{\text{eff}}(\boldsymbol{r})}{1-\beta}.$$

- *If we slightly increase the effective resistance of the edge h by setting $\gamma = (1+\epsilon)$ with $\epsilon \leq 1$, then*

$$R_{\text{eff}}(\boldsymbol{r}') \geq \frac{1+\epsilon}{\beta + (1+\epsilon)(1-\beta)} R_{\text{eff}}(\boldsymbol{r}) \geq \left(1 + \frac{\epsilon\beta}{2}\right) R_{\text{eff}}(\boldsymbol{r}).$$

Proof. The assumptions of the theorem are unchanged if we multiply f by a constant, so we may assume without loss of generality that f is the electrical s-t flow of value $1/R_{\text{eff}}(\boldsymbol{r})$. If $\boldsymbol{\phi}$ is the vector of vertex potentials corresponding to f, this gives $\phi(s) - \phi(t) = 1$. Since adding a constant to the potentials doesn't change the flow, we may assume that $\phi(s) = 1$ and $\phi(t) = 0$. By Fact 2.4,

$$C_{\text{eff}}(\boldsymbol{r}) = \sum_{(u,v)\in E} \frac{\big(\phi(u) - \phi(v)\big)^2}{r_{(u,v)}} \quad = \quad \frac{\big(\phi(i) - \phi(j)\big)^2}{r_h} + \sum_{(u,v)\in E\setminus\{h\}} \frac{\big(\phi(u) - \phi(v)\big)^2}{r_{(u,v)}}.$$

The assumption that h contributes a β fraction of the total energy implies that, in the above expression,

$$\frac{(\phi(i) - \phi(j))^2}{r_h} = \beta C_{\text{eff}}(\boldsymbol{r}),$$

and thus

$$\sum_{(u,v)\in E\setminus\{h\}} \frac{(\phi(u) - \phi(v))^2}{r_{(u,v)}} = (1-\beta) C_{\text{eff}}(\boldsymbol{r}).$$

We will obtain our bound on $C_{\text{eff}}(\boldsymbol{r}')$ by plugging the original vector of potentials ϕ into the expression in Fact 2.4:

$$\begin{aligned} C_{\text{eff}}(\boldsymbol{r}') &= \min_{\boldsymbol{\phi} \mid \substack{\phi(s)=1,\\ \phi(t)=0}} \sum_{(u,v)\in E} \frac{\big(\phi(u) - \phi(v)\big)^2}{r'_{(u,v)}} \quad \leq \quad \sum_{(u,v)\in E} \frac{\big(\phi(u) - \phi(v)\big)^2}{r'_{(u,v)}} \\ &= \frac{\big(\phi(i) - \phi(j)\big)^2}{r'_h} + \sum_{(u,v)\in E\setminus\{h\}} \frac{\big(\phi(u) - \phi(v)\big)^2}{r'_{(u,v)}} \quad = \quad \frac{\big(\phi(i) - \phi(j)\big)^2}{\gamma r_h} + \sum_{(u,v)\in E\setminus\{h\}} \frac{\big(\phi(u) - \phi(v)\big)^2}{r_{(u,v)}} \\ &= \frac{\beta}{\gamma} C_{\text{eff}}(\boldsymbol{r}) + (1-\beta) C_{\text{eff}}(\boldsymbol{r}) \quad = \quad C_{\text{eff}}(\boldsymbol{r}) \left(\frac{\beta + \gamma(1-\beta)}{\gamma}\right). \end{aligned}$$

Since $R_{\text{eff}}(\boldsymbol{r}) = 1/C_{\text{eff}}(\boldsymbol{r})$ and $R_{\text{eff}}(\boldsymbol{r}') = 1/C_{\text{eff}}(\boldsymbol{r}')$, the desired result follows. □

3 A Simple $\widetilde{O}\left(m^{3/2}\epsilon^{-5/2}\right)$-Time Flow Algorithm

Before describing our $\widetilde{O}\left(m^{4/3}\epsilon^{-3}\right)$ algorithm, we will describe a simpler algorithm that finds a $(1-\epsilon)$-approximately maximum flow in time $\widetilde{O}\left(m^{3/2}\epsilon^{-5/2}\right)$. 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 [4] to encompass proof techniques exploited in [25, 30, 11, 13]. 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, converts it into an algorithm that gives a good approximation to 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.

In this section, we provide a self-contained description of the multiplicative weights method when it is specialized to our setting. In Section 3.1, we will describe the requirements on the oracle, give an algorithm that iteratively uses it to obtain a $(1-\epsilon)$-approximately maximum flow, and state how the number of iterations required by the algorithm depends on the properties of the oracle. In Section 3.2, we will describe how to implement the oracle using electrical flows. Finally, in Section 3.3 we will provide a simple proof of the convergence bound set forth in Section 3.1.

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

$$\mathsf{cong}_f(e) := \frac{|f_e|}{u_e}$$

of the flow on an edge to its capacity. In particular, an s-t flow is feasible if and only if $\mathsf{cong}_f(e) \leq 1$ for all $e \in E$.

The multiplicative weights method will use a subroutine that we will refer to as an (ϵ, ρ)-*oracle*. This oracle will take as 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 takes as input an undirected, capacitated graph, a real number $F > 0$ and a vector $\boldsymbol{w}$ of edge weights with $w_e \geq 1$ for all e, and whose output satisfies:*

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), (ii), (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 2.2, we can obtain a crude bound B in time $O(m + n\log n)$ such that $B \leq F^* \leq mB$, so the binary search will only call our algorithm $O(\log(m/\epsilon))$ 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. The algorithm initializes

[3] Up to polynomial factors 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.

all of the weights to 1 and calls the oracle with these weights. It then multiplies the weight of each edge e by $(1 + \frac{\epsilon}{\rho}\mathsf{cong}_{f^i}(e))$. Note that if the congestion of an edge is high, say close to ρ, then its weight will increase by a factor close to $(1+\epsilon)$. On the other hand, if the flow on an edge is no more than its capacity, then the new weight of the edge is essentially unchanged. This will put a larger fraction of the weight on the violated constraints, so the oracle will be forced to return a solution that comes closer to satisfying them (possibly at the expense of other edges). In the end, the algorithm returns the average of all of the flows.

Input : A graph $G = (V, E)$ with capacities $\{u_e\}_e$, a target flow value F, and an (ϵ, ρ)-oracle O
Output: Either a flow $\bar{f}$, 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 $\bar{f} \leftarrow \frac{(1-\epsilon)^2}{(1+\epsilon)N}(\sum_i f^i)$

Figure 1: Multiplicative-weights-update routine

The key point in the analysis of this algorithm is that the total weight on G does not grow too quickly, due to the average congestion constraint on the flows returned by the oracle O. However, if an edge e consistently suffers large congestion in a sequence of flows returned by O, then its weight increases rapidly relative to the total weight. This will significantly penalize any further congestion of that edge in subsequent flows. If this were to occur too many times, its weight would exceed the total weight, which obviously cannot occur. In Section 3.3, we will prove the following theorem by showing that our algorithm converges in $2\rho \ln m/\epsilon^2$ iterations.

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 $T(m, 1/\epsilon, U)$, the algorithm in Figure 1 computes a $(1 - O(\epsilon))$-approximately maximum flow in a capacitated, undirected graph in time $\widetilde{O}\left(\rho\epsilon^{-2} \cdot T(m, 1/\epsilon, U)\right)$.*

Note that the number of iterations of the algorithm above grows linearly with the value of ρ, which we call the *width* of the oracle. Intuitively, this should be necessary because the final flow across an edge e is equal to the average of the flows sent over it by all f^i. If we send $\rho \cdot u_e$ units of flow across the edge e in some step, then we will need at least $\Omega(\rho)$ iterations to drop the average to 1.[4]

[4] Strictly speaking, it is possible that we could do better than this by sending a large amount of flow across the edge in the opposite direction. However, nothing in our algorithm aims to obtain this kind of cancellation, so we shouldn't expect to be able to systematically exploit it.

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

Given Theorem 3.2, our problem is thus reduced to designing an efficient oracle that has a small width. In this subsection, we give a simple $\widetilde{O}\left(m \log(Um\epsilon^{-1})\right)$ time implementation of an $(\epsilon, 3\sqrt{m/\epsilon})$ oracle for any $0 < \epsilon < 1/2$. By Theorem 3.2, this will immediately yield an $\widetilde{O}\left(m^{3/2}\epsilon^{-5/2}\right)$ time algorithm for finding an approximately maximum flow.

To build such an oracle, we set

$$r_e := \frac{1}{u_e^2}\left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right) \tag{2}$$

for each edge e, and we use the procedure from Theorem 2.3 to approximate the electrical flow that sends F units of flow from s to t in a network whose resistances are given by the r_e. The pseudocode for this oracle is shown in Figure 2.

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 $\tilde{f}$, 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 $\tilde{f}$ using Theorem 2.3 on G with resistances $\boldsymbol{r}$ and target flow value F

if $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) > (1+\epsilon)|\boldsymbol{w}|_1$ **then return** ***"fail"***
else return $\tilde{f}$

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

We now show that the resulting flow $\tilde{f}$ has the properties required by Definition 3.1. Since $|\tilde{f}| = F$ by construction, we only need to demonstrate the bounds on the average congestion (weighted by the w_e) and the maximum congestion. We will use the basic fact that electrical flows minimize the energy of the flow. Our analysis will then compare the energy of $\tilde{f}$ with that of a maximum flow. Intuitively, the w_e term in Equation (2) guarantees the bound on the average congestion, while the $\epsilon|\boldsymbol{w}|_1/(3m)$ term guarantees the bound on the maximum congestion.

Suppose f^* is a maximum flow. By its feasibility, $\mathsf{cong}_{f^*}(e) \leq 1$ for all e, so

$$\begin{aligned}
\mathcal{E}_{\boldsymbol{r}}(f^*) &= \sum_e \left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)\left(\frac{f^*(e)}{u_e}\right)^2 \\
&= \sum_e \left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right)\left(\mathsf{cong}_{f^*}(e)\right)^2 \\
&\leq \sum_e \left(w_e + \frac{\epsilon|\boldsymbol{w}|_1}{3m}\right) \\
&= \left(1 + \frac{\epsilon}{3}\right)|\boldsymbol{w}|_1.
\end{aligned}$$

Since the electrical flow minimizes the energy, $\mathcal{E}_{\boldsymbol{r}}(f^*)$ is an upper bound on the energy of the electrical flow of value F whenever $F \leq F^*$. In this case, Theorem 2.3 implies that the $(\epsilon/3)$-

approximate electrical flow $\tilde{f}$ satisfies

$$\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) \leq \left(1+\frac{\epsilon}{3}\right)\mathcal{E}_{\boldsymbol{r}}(f^*) \leq \left(1+\frac{\epsilon}{3}\right)^2 |\boldsymbol{w}|_1 \leq (1+\epsilon)\,|\boldsymbol{w}|_1. \tag{3}$$

This shows that the oracle will never output **"fail"** when $F \leq F^*$. It thus suffices to show that the energy bound $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) \leq (1+\epsilon)|\boldsymbol{w}|_1$, which holds whenever the algorithm does not return **"fail"**, implies the required bounds on the average and worst-case congestion. To see this, we note that the energy bound implies that

$$\sum_e w_e \left(\mathsf{cong}_{\tilde{f}}(e)\right)^2 \leq (1+\epsilon)\,|\boldsymbol{w}|_1, \tag{4}$$

and, for all $e \in E$,

$$\frac{\epsilon|\boldsymbol{w}|_1}{3m}\left(\mathsf{cong}_{\tilde{f}}(e)\right)^2 \leq (1+\epsilon)\,|\boldsymbol{w}|_1. \tag{5}$$

By the Cauchy-Schwarz inequality,

$$\left(\sum_e w_e \mathsf{cong}_{\tilde{f}}(e)\right)^2 \leq |\boldsymbol{w}|_1 \left(\sum_e w_e \left(\mathsf{cong}_{\tilde{f}}(e)\right)^2\right), \tag{6}$$

so inequality (4) implies

$$\sum_e w_e \mathsf{cong}_{\tilde{f}}(e) \leq \sqrt{1+\epsilon}\,|\boldsymbol{w}|_1 < (1+\epsilon)\,|\boldsymbol{w}|_1, \tag{7}$$

which is the required bound on the average congestion. Furthermore, inequality (5) and the fact that $\epsilon < 1/2$ implies that

$$\mathsf{cong}_{\tilde{f}}(e) \leq \sqrt{\frac{3m(1+\epsilon)}{\epsilon}} \leq 3\sqrt{m/\epsilon}$$

for all e, which establishes the required bound on the maximum congestion. So our algorithm implements an $(\epsilon, 3\sqrt{m/\epsilon})$-oracle, as desired.

To bound the running time of this oracle, recall that all edge capacities lie between 1 and U, and compute

$$R = \max_{e,e'} \frac{r_e}{r_{e'}} \leq U^2 \frac{(1+\epsilon/3m)\,|\boldsymbol{w}|_1}{(\epsilon/3m)\,|\boldsymbol{w}|_1} = U^2 \frac{3m+\epsilon}{\epsilon} \leq 4U^2 m/\epsilon. \tag{8}$$

This establishes an upper bound on the ratio of the largest resistance to the smallest resistance. Thus, by Theorem 2.3, the running time of this implementation is $\widetilde{O}\left(m \log R/\epsilon\right) = \widetilde{O}\left(m \log(Um/\epsilon)\right)$. Combining this with Theorem 3.2 and the fact that we can assume that U is polynomially bounded in m and $\frac{1}{\epsilon}$ (see our discussion in Section 2.2), we have shown

Theorem 3.3. *For any $0 < \epsilon < 1/2$, the algorithm in Figures 1 and 2 computes a $(1-\epsilon)$-approximately maximum flow in $\widetilde{O}\left(m^{3/2}\epsilon^{-5/2}\right)$ time.*

3.3 The Convergence of Multiplicative Weights

In this section, we prove Theorem 3.2 by analyzing the multiplicative weights update algorithm shown in Figure 1.

Our analysis will be based on the potential function $\mu_i := |\boldsymbol{w}^i|_1$. Clearly, $\mu_0 = m$ and this potential only increases during the course of the algorithm. It follows from condition (ii) of Definition 3.1 that if we run the (ϵ, ρ)-oracle O with $F \leq F^*$, then

$$\sum_e w_e^i \mathsf{cong}_{f^{i+1}}(e) \leq (1+\epsilon)|\boldsymbol{w}^i|_1, \quad \text{for all } i \geq 1. \tag{9}$$

By condition (iii) of Definition 3.1,

$$\mathsf{cong}_{f^i}(e) \leq \rho, \quad \text{for all } i \geq 1 \text{ and every edge } e. \tag{10}$$

We start by upper bounding the total growth of μ_i thoughout the algorithm.

Lemma 3.4. *For any $i \geq 0$,*

$$\mu_{i+1} \leq \mu_i \exp\left(\frac{(1+\epsilon)\epsilon}{\rho}\right).$$

In particular, $\|\boldsymbol{w}^N\|_1 = \mu_N \leq m \exp\left(\frac{(1+\epsilon)\epsilon}{\rho} N\right) = n^{O(1/\epsilon)}$.

Proof. For any $i \geq 0$, we have

$$\mu_{i+1} = \sum_e w_e^{i+1} = \sum_e w_e^i \left(1 + \frac{\epsilon}{\rho}\mathsf{cong}_{f^{i+1}}(e)\right) = \sum_e w_e^i + \frac{\epsilon}{\rho}\sum_e w_e^i \mathsf{cong}_{f^{i+1}}(e) \leq \mu_i + \frac{(1+\epsilon)\epsilon}{\rho}|\boldsymbol{w}^i|_1,$$

where the last inequality follows from (9). Thus, we can conclude that

$$\mu_{i+1} \leq \mu_i + \frac{(1+\epsilon)\epsilon}{\rho}|\boldsymbol{w}^i|_1 = \mu_i\left(1 + \frac{(1+\epsilon)\epsilon}{\rho}\right) \leq \mu_i \exp\left(\frac{(1+\epsilon)\epsilon}{\rho}\right),$$

as desired. The lemma follows. □

One of the consequences of the above lemma is that whenever we make a call to the oracle, the total weight $\|\boldsymbol{w}^i\|_1$ is at most $n^{O(1/\epsilon)}$.

Next, we bound the final weight w_e^N of a particular edge e with the congestion $\mathsf{cong}_{\bar{f}}(e)$ that this edge suffers in our final flow $\bar{f}$.

Lemma 3.5. *For every edge e and $i \geq 0$,*

$$w_e^i \geq \exp\left(\frac{(1-\epsilon)\epsilon}{\rho}\sum_{j\geq 1}^{i}\mathsf{cong}_{f^j}(e)\right).$$

In particular, $w_e^N \geq \exp\left(\frac{(1+\epsilon)\epsilon N}{(1-\epsilon)\rho}\mathsf{cong}_{\bar{f}}(e)\right)$.

Proof. For any $i \geq 0$, we have

$$w_e^i = \prod_{j \geq 1}^{i} \left(1 + \frac{\epsilon}{\rho} \mathsf{cong}_{f^j}(e)\right) \geq \prod_{j \geq 1}^{i} \exp\left(\frac{(1-\epsilon)\epsilon}{\rho} \mathsf{cong}_{f^j}(e)\right),$$

where we used (10) and that for all ϵ and x in $[0,1]$, $1 + \epsilon x \geq \exp((1-\epsilon)\epsilon x)$.

The lemma now follows since

$$w_e^i \geq \prod_{j \geq 1}^{i} \exp\left(\frac{(1-\epsilon)\epsilon}{\rho} \mathsf{cong}_{f^j}(e)\right) = \exp\left(\frac{(1-\epsilon)\epsilon}{\rho} \sum_{j \geq 1}^{i} \mathsf{cong}_{f^j}(e)\right),$$

and for $i = N$

$$w_e^N \geq \exp\left(\frac{(1-\epsilon)\epsilon}{\rho} \sum_{j \geq 1}^{N} \mathsf{cong}_{f^j}(e)\right) = \exp\left(\frac{(1+\epsilon)\epsilon N}{(1-\epsilon)\rho} \mathsf{cong}_{\bar{f}}(e)\right).$$

□

Lemmas 3.4 and 3.5 imply that for every edge e,

$$m \exp\left(\frac{(1+\epsilon)\epsilon N}{\rho}\right) \geq \mu_N = |\boldsymbol{w}^N|_1 \geq w_e^N \geq \exp\left(\frac{(1+\epsilon)\epsilon N}{(1-\epsilon)\rho} \mathsf{cong}_{\bar{f}}(e)\right).$$

This implies that

$$\mathsf{cong}_{\bar{f}}(e) \leq 1 - \epsilon + \frac{(1-\epsilon)\rho \ln m}{(1+\epsilon)\epsilon N} = 1 - \epsilon + \frac{\epsilon(1-\epsilon)}{2(1+\epsilon)} \leq 1$$

for every edge e. Thus, we see that $\bar{f}$ is a feasible s-t flow and, since each flow f^i has value F, the value $|\bar{f}|$ of $\bar{f}$ is $\frac{(1-\epsilon)^2}{(1+\epsilon)} F \geq (1 - O(\epsilon))F$ for $1/2 > \epsilon > 0$, as desired.

4 An $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$ Algorithm for Approximate Maximum Flow

In this section, we modify our algorithm to run in time $\widetilde{O}\left(m^{4/3}\epsilon^{-3}\right)$. Then, in Section 4.4, we combine this with the graph smoothing technique of Karger [15] to obtain an $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$-time algorithm.

For fixed ϵ, the algorithm in the previous section requires us to compute $\widetilde{O}\left(m^{1/2}\right)$ electrical flows, each of which takes time $\widetilde{O}\left(m\right)$, this leads to a running time of $\widetilde{O}\left(m^{3/2}\right)$. To reduce this time bound to $\widetilde{O}\left(m^{4/3}\right)$, we will show how to find an approximately maximum flow while computing only $\widetilde{O}\left(m^{1/3}\right)$ electrical flows.

Our analysis of the oracle from Section 3.2 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 $\Theta(m^{1/2})$.

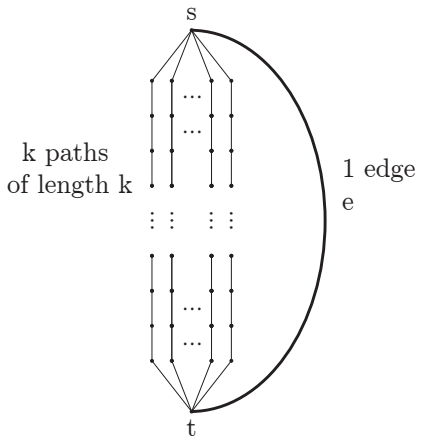

Figure 3: A graph on which the electrical flow sends approximately $\sqrt{m}$ units of flow across an edge when sending the maximum flow F^* from s to t.

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, we note 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 dramatically improved by removing the bad edges.

This motivates us to modify our algorithm as follows. We'll set ρ to be some value smaller than the actual worst-case bound of $\widetilde{O}\left(m^{1/2}\right)$. (It will end up being $\widetilde{O}\left(m^{1/3}\right)$.) The oracle will begin by computing an electrical flow as before. 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 other 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.

4.2 Analysis of the New Algorithm

Before proceeding to a formal 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

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 $\bar{f}$ 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 approximate electrical flow $\tilde{f}$ using Theorem 2.3 on $G_H := (V, E \setminus H)$ with resistances $\boldsymbol{r}$, target flow value F, and parameter $\delta = \epsilon/3$.

if $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) > (1+\epsilon)|\boldsymbol{w}|_1$ or s and t are disconnected in G_H **then return** ***"fail"***

if there exists e with $\text{cong}_{\tilde{f}}(e) > \rho$ **then** add one such e to H and start over

return $\tilde{f}$

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 $\bar{f}$, 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
    $w_e^i \leftarrow w_e^{i-1}(1 + \frac{\epsilon}{\rho}\text{cong}_{f^i}(e))$ for each $e \in E$
  end
end
return $\bar{f} \leftarrow \frac{(1-\epsilon)^2}{(1+\epsilon)N}(\sum_i f^i)$

Figure 5: An improved $(1 - O(\epsilon))$-approximation algorithm for the maximum flow problem

the algorithm's correctness and bound its running time.

We first note that, by construction, the congestion of any edge in the flow $\tilde{f}$ returned by the modified oracle from Figure 4 will be bounded by ρ. Furthermore, it enforces the bound $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) \leq (1+\epsilon)|\boldsymbol{w}|_1$; by Equations (4), (6), and (7) in Section 3.2, this guarantees that $\tilde{f}$ 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 Section 3 will show that the multiplicative update scheme employed by our algorithm will yield an approximately 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 $\tilde{f}$ (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, we will show the following upper bound on the cardinality $|H|$ and the capacity $u(H)$ of the set of forbidden edges, whose proof we postpone until the next section:

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 $u(H) \leq \frac{15}{256}\epsilon F < \epsilon F/12$.

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.

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 $\bar{f}$ of value $|\bar{f}| = (1 - O(\epsilon))F$ in time $\widetilde{O}\left(m^{4/3}\epsilon^{-3}\right)$.*

Proof. To bound the running time, we note that, whenever we invoke the algorithm from Theorem 2.3 , we either advance the number of iterations or we increase the cardinality of H, so the number of linear systems we solve is at most $N + |H| \leq N + \frac{15}{32}(m \ln m)^{1/3}$.

Equation (8) implies that the value of R from Theorem 2.3 is $O((m/\epsilon)^{O(1)})$, so solving each linear system takes time at most $\widetilde{O}(m \log 1/\epsilon)$. This gives an overall running time of

$$\widetilde{O}\left(\left(N + \tfrac{15}{32}(m \ln m)^{1/3}\right) m\right) = \widetilde{O}\left(m^{4/3}\epsilon^{-3}\right),$$

as claimed.

It thus remains to prove correctness. For this, we need to show that if $F \leq F^*$, then the oracle does not return **"fail"**, which would occur if we disconnect s from t or if $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) > (1+\epsilon)|\boldsymbol{w}|_1$. By Lemma 4.1 and the comment following it, we know that throughout the whole algorithm G_H has maximum flow value of at least $F^* - \epsilon F/12 \geq (1 - \epsilon/12)\, F$ and thus, in particular, we will never disconnect s from t.

Furthermore, this implies that there exists a feasible flow in our graph of value $(1 - \epsilon/12)\, F$, even after we have removed the edges in H. There is thus a flow of value F in which every edge has

congestion at most $1/(1-\epsilon/12)$. We can therefore use the argument from Section 3.2 (Equation (3) and the lines directly preceding it) to show that we always have

$$\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) \leq (1-\epsilon/12)^{-2}(1+\epsilon/3)^2|\boldsymbol{w}|_1 \leq (1+\epsilon)|\boldsymbol{w}|_1,$$

as required. □

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

Theorem 4.3. *For any $0 < \epsilon \leq 1/2$, the algorithm in Figures 5 and 4 computes a $(1-\epsilon)$-approximately maximum flow in $\widetilde{O}\left(m^{4/3}\epsilon^{-3}\right)$ time.*

4.3 The Proof of Lemma 4.1

All that remains is to prove the bounds given by Lemma 4.1 on the cardinality and capacity of H. To do so, we will use the effective resistance of the circuits on which we compute electrical flows as a potential function. The key insight is that we only cut an edge when its flow accounts for a nontrivial fraction of the energy of the electrical flow, and that cutting such an edge will cause a substantial change in the effective resistance. Combining this with a bound on how much the effective resistance can change during the execution of the algorithm will guarantee that we won't cut too many edges.

Let $\boldsymbol{r}^j$ be the resistances used in the j^{th} electrical flow computed during the execution of the algorithm[5]. If an edge e is in H when this electrical flow is computed, set $r_e^j = \infty$. We define the potential function

$$\Phi(j) = R_{\text{eff}}(\boldsymbol{r}^j) = \mathcal{E}_{\boldsymbol{r}^j}(f_{\boldsymbol{r}^j}),$$

where $f_{\boldsymbol{r}^j}$ is the (exact) electrical flow of value 1 arising from $\boldsymbol{r}^j$. Lemma 4.1 will follow easily from:

Lemma 4.4. *Suppose that $F \leq F^* \leq mF$. Then:*

1. *$\Phi(j)$ never decreases during the execution of the algorithm.*
2. *$\Phi(1) \geq m^{-4}F^{-2}$.*
3. *If we add an edge to H after the computation of the $(j-1)-st$ electrical flow, then $(1-\frac{\epsilon\rho^2}{5m})\Phi(j) > \Phi(j-1)$.*

Proof. **Proof of (1)**

The only way that the resistance r_e^j of an edge e can change is if the weight w_e is increased by the multiplicative weights routine, or if e is added to H so that r_e^j is set to ∞. As such, the resistances are nondecreasing during the execution of the algorithm. By Rayleigh Monotonicity (Corollary 2.5), this implies that the effective resistance is nondecreasing as well.

[5] Note that the oracle can compute many electrical flows each time that it is called: it computes a new electrical flow for each edge that it adds to H. So, $\boldsymbol{r}^j$ is typically not the set of resistances arising from $\boldsymbol{w}^j$.

Proof of (2)

In the first linear system, $H = \emptyset$ and $r_e^1 = \frac{1+\epsilon/3}{u_e^2}$ for all $e \in E$. Let $(S, V \setminus S)$ be the minimum s-t cut of G. By the Max Flow-Min Cut Theorem ([12, 9]), we know that the capacity $u(S) = \sum_{e \in E(S)} u_e$ of this cut is equal to F^*. In particular, for all $e \in E(S)$

$$r_e^1 = \frac{1+\epsilon/3}{u_e^2} \geq \frac{1+\epsilon/3}{F^{*2}} > \frac{1}{F^{*2}}.$$

As $f_{\boldsymbol{r}^1}$ is an electrical s-t flow of value 1, it sends 1 unit of net flow across $(S, V \setminus S)$; so, some edge $e' \in E(S)$ must have $f_{\boldsymbol{r}^1}(e') \geq 1/m$. This gives

$$\Phi(1) = \mathcal{E}_{\boldsymbol{r}^1}(f_{\boldsymbol{r}^1}) = \sum_{e \in E} f_{\boldsymbol{r}^1}(e)^2 r_e^1 \geq f_{\boldsymbol{r}^1}(e')^2 r_{e'}^1 > \frac{1}{m^2 F^{*2}}. \tag{11}$$

Since $F^* \leq mF$ by assumption, the desired inequality follows.

Proof of (3)

Suppose we add the edge h to H after the $j-1$st computation of an electrical flow. We will show that h accounts for a substantial fraction of the total energy of the electrical flow with respect to the resistances $\boldsymbol{r}^{j-1}$, and our result will then follow from Lemma 2.6.

Let $\boldsymbol{w}$ be the weights submitted to the oracle when the $j-1$st electrical flow, $\tilde{f}$, is computed. Because we added h to H, we know that $\mathsf{cong}_{\tilde{f}}(h) > \rho$. Since our algorithm did not return **"fail"** after computing this $\tilde{f}$, we must have that

$$\mathcal{E}_{\boldsymbol{r}^{j-1}}(\tilde{f}) \leq (1+\epsilon)|\boldsymbol{w}|_1. \tag{12}$$

Using the definition of r_h^{j-1}, the fact that $\mathsf{cong}_{\tilde{f}}(h) > \rho$, and inequality (12), we obtain:

$$\begin{aligned}
\tilde{f}_h^2 r_h^{j-1} &= \tilde{f}_h^2 \frac{w_e + \epsilon \frac{|\boldsymbol{w}|_1}{3m}}{u_e^2} \\
&\geq \tilde{f}_h^2 \frac{\epsilon |\boldsymbol{w}|_1}{3m u_e^2} \\
&= \frac{\epsilon}{3m} \left(\frac{\tilde{f}_h}{u_e} \right)^2 |\boldsymbol{w}|_1 \\
&= \frac{\epsilon}{3(1+\epsilon)m} \mathsf{cong}_{\tilde{f}}(h)^2 \left((1+\epsilon)|\boldsymbol{w}|_1 \right) \\
&> \frac{\epsilon \rho^2}{3(1+\epsilon)m} \mathcal{E}_{\boldsymbol{r}^{j-1}}(\tilde{f}).
\end{aligned}$$

The above inequalities establish that edge h accounts for more than a $\frac{\epsilon\rho^2}{3(1+\epsilon)m}$ fraction of the total energy $\mathcal{E}_{\boldsymbol{r}^i}(\tilde{f})$ of the flow $\tilde{f}$.

The flow $\tilde{f}$ is the approximate electrical flow computed by our algorithm, but our argument will require that an inequality like this holds in the exact electrical flow $f_{\boldsymbol{r}^{j-1}}$. This is guaranteed

by part b of Theorem 2.3, which, along with the facts that $\mathcal{E}_{\boldsymbol{r}}(\tilde{f}) \geq \mathcal{E}_{\boldsymbol{r}}(f_{\boldsymbol{r}^{j-1}})$, $\rho \leq 1$, and $\epsilon < 1/2$, gives us that

$$f_{\boldsymbol{r}^{j-1}}(h)^2 r_h^{j-1} > \tilde{f}_h^2 r_h^{j-1} - \frac{\epsilon/3}{2mR}\mathcal{E}_{\boldsymbol{r}}(f_{\boldsymbol{r}^{j-1}}) > \left(\frac{\epsilon\rho^2}{3(1+\epsilon)m} - \frac{\epsilon/3}{2mR}\right)\mathcal{E}_{\boldsymbol{r}}(f_{\boldsymbol{r}^{j-1}}) > \frac{\epsilon\rho^2}{5m}\mathcal{E}_{\boldsymbol{r}}(f_{\boldsymbol{r}^{j-1}}).$$

The result now follows from Lemma 2.6. □

We are now ready to prove Lemma 4.1.

Proof of Lemma 4.1. Let k be the cardinality of the set H at the end of the algorithm. Let j be the number of approximate electrical flows computed by the algorithm, let $\boldsymbol{r}^j$ be the resistances used to compute the last electrical flow, let $\tilde{f}$ be the approximate electrical flow computed, and let $\boldsymbol{w}$ be the weights submitted to the oracle when this flow was computed. Note that no edges are added to H after the flow $\tilde{f}$ is computed.

As the energy required by an s-t flow scales with the square of the value of the flow,

$$\Phi(j) = \frac{\mathcal{E}_{\boldsymbol{r}^j}(f_{\boldsymbol{r}^j})}{F^2} \leq \frac{\mathcal{E}_{\boldsymbol{r}^j}(\tilde{f})}{F^2}. \tag{13}$$

By the construction of our algorithm, it must have been the case that $\mathcal{E}_{\boldsymbol{r}^j}(\tilde{f}) \leq (1+\epsilon)|\boldsymbol{w}|_1$. This inequality together with inequality (13) and part 2 of Lemma 4.4 implies that

$$\Phi(j) = \mathcal{E}_{\boldsymbol{r}^j}(f_{\boldsymbol{r}^j}) \leq \frac{\mathcal{E}_{\boldsymbol{r}^j}(\tilde{f})}{F^2} \leq (1+\epsilon)|\boldsymbol{w}|_1 m^4 \Phi(1).$$

Now, since k edges were added to H, parts 1 and 3 of Lemma 4.4, and Lemma 3.4 imply that

$$\left(1 - \frac{\epsilon\rho^2}{5m}\right)^{-k} \leq \frac{\Phi(j)}{\Phi(1)} \leq (1+\epsilon)|\boldsymbol{w}|_1 m^4 \leq (1+\epsilon)m^4\left(m\exp\left(\tfrac{(1+\epsilon)\epsilon}{\rho}N\right)\right) \leq 2m^5\exp(3\epsilon^{-1}\ln m),$$

where in the last inequality we used the fact that $\epsilon < 1/2$. Rearranging the terms in the above inequality gives us that

$$k \leq -\frac{\ln 2 + 5\ln m + 3\epsilon^{-1}\ln m}{\ln\left(1 - \frac{\epsilon\rho^2}{5m}\right)} < -\frac{6\epsilon^{-1}\ln m}{\ln\left(1 - \frac{\epsilon\rho^2}{5m}\right)} < \frac{30m\ln m}{\epsilon^2\rho^2},$$

where we used the inequalities $\epsilon < 1/2$ and $\log(1-c) < -c$ for all $c \in (0,1)$. This establishes our bound on cardinality of the set H.

To bound the value of $u(H)$, let us note that we add an edge e to H only when we send at least ρu_e units of flow across it. But since we never flow more than F units across any single edge, we have that $u_e \leq F/\rho$. Therefore, we may conclude that

$$u(H) \leq |H|\frac{F}{\rho} \leq \frac{30mF\ln m}{\epsilon^2\rho^3},$$

as desired. □

4.4 Improving the Running Time to $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$

We can now combine our algorithm with existing methods to further improve its running time. In [15] (see also [6]), 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 [15] and stated in a similar form in [6]:

Theorem 4.5 ([15, 6])**.** *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 $\widetilde{O}\left(\epsilon^2 m/n \cdot T(\widetilde{O}\left(n\epsilon^{-2}\right), n, \Omega(\epsilon))\right)$.*

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

Theorem 4.6. *For any $0 < \epsilon < 1/2$, a $(1-\epsilon)$-approximately maximum flow can be computed in $\widetilde{O}\left(mn^{1/3}\epsilon^{-11/3}\right)$ time.*

4.5 Approximating the Value of the Maximum s-t Flow in Time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-17/3}\right)$

Given any weighted undirected graph $G = (V, E, w)$ with n vertices and m edges, Benczúr and Karger [5] 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 from Section 4 to a sparsifier of G gives us an algorithm for $(1-\epsilon)$-approximating the value of the maximum s-t flow on G in time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-17/3}\right)$.

We note that this only allows us to approximate the *value* of the maximum s-t flow on G. It gives us a flow on G', not one on G. We do not know how to use an approximately maximum s-t flow on G' to obtain one on G in less time than would be required to compute a maximum flow in G from scratch using the algorithm from Section 4.

For this reason, there is a gap between the time we require to find a maximum flow and the time we require to compute its value. We note, however, that this gap will not exist for the minimum s-t cut problem, since an approximately minimum s-t cut on G' will also be an approximately minimum s-t cut on G. We will present an algorithm for finding such a cut in the next section. By the Max Flow-Min Cut Theorem, this will provide us with an alternate algorithm for approximating the value of the maximum s-t flow. It will have a slightly better dependence on ϵ, which will allow us to approximate the value of the maximum s-t flow in time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-16/3}\right)$.

5 A Dual Algorithm for Finding an Approximately Minimum s-t Cut in Time $\widetilde{O}\left(m + n^{4/3}\epsilon^{-16/3}\right)$

In this section, we will describe a dual perspective that yields an even simpler algorithm for computing an approximately minimum s-t cut. Rather than using electrical flows to obtain a flow, it will use the electrical potentials to obtain a 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. It finds a $(1+\epsilon)$-approximately minimum s-t cut in time $\widetilde{O}\left(m^{4/3}\epsilon^{-8/3}\right)$; applying it to a sparsifier will give us:

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

We further note that the update rule is slightly modified from the one that appeared in the flow algorithm. It guarantees that the effective resistance increases substantially when some edge flows more than ρ, without having to explicitly cut it. Our previous rule allowed the weight (but not resistance) of an edge to constitute a very small fraction of the total weight; in this case, a significant multiplicative increase in the weight of an edge may not produce a substantial change in the effective resistance of the graph.

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 an approximate electrical flow $\tilde{f}^{i-1}$ and potentials $\tilde{\phi}$ using Theorem 2.3 on G with resistances $r_e^{i-1} = \frac{w_e^{i-1}}{u_e^2}$, target flow value F, and parameter δ.
  $\mu^{i-1} \leftarrow \sum_e w_e^{i-1}$
  $w_e^i \leftarrow w_e^{i-1} + \frac{\epsilon}{\rho}\mathsf{cong}_{\tilde{f}^{i-1}}(e)w_e^{i-1} + \frac{\epsilon^2}{m\rho}\mu^{i-1}$ for each $e \in E$
  Scale and translate $\tilde{\phi}$ so that $\widetilde{\phi}(s) = 1$ and $\widetilde{\phi}(t) = 0$
  Let $S_x = \{v \in V \mid \widetilde{\phi}(v) > x\}$
  Set S to be the set S_x that minimizes the capacity of the cut $(S_x, V \setminus S_x)$
  If the capacity of $(S, V \setminus S)$ is less than $F/(1-7\epsilon)$, **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.

Of course, the edges crossing a minimum cut will also cross many other (likely much larger) cuts, so we certainly can't hope to obtain a graph in which every edge crossing a large cut has negligible weight. In order to formalize the above argument, we will thus need some good way to measure the extent to which the weight is "concentrated on approximately minimum cuts".

In Section 5.2, we will show how to use effective resistance to formulate such a notion. In particular, we will show that if we can make the effective resistance large enough then we can find a cut of small capacity. In Section 5.3, we will use an argument like the one described above to show that the resistances produced by the algorithm in Figure 6 converge after $N = \widetilde{O}\left(m^{1/3}\epsilon^{-8/3}\right)$ steps to one that meets such a bound.

5.2 Cuts, Electrical Potentials, and Effective Resistance

During the algorithm, we scale and translate the potentials of the approximate electrical flow so that $\widetilde{\boldsymbol{\phi}}(s) = 1$ and $\widetilde{\boldsymbol{\phi}}(t) = 0$. We then produce a cut by choosing $x \in [0, 1]$ and dividing the graph into the sets $S = \{v \in V \mid \widetilde{\boldsymbol{\phi}}(v) > x\}$ and $V \setminus S = \{v \in V \mid \widetilde{\boldsymbol{\phi}}(v) \leq x\}$. The following lemma upper bounds the capacity of the resulting cut in terms of the electrical potentials and edge capacities.

Lemma 5.2. *Let $\tilde{\boldsymbol{\phi}}$ be as above. Then there is a cut S_x of capacity at most*

$$\sum_{(u,v)\in E} |\widetilde{\boldsymbol{\phi}}(u) - \widetilde{\boldsymbol{\phi}}(v)| u_{(u,v)}. \tag{14}$$

Proof. Consider choosing $x \in [0, 1]$ uniformly at random. The probability that an edge (u, v) is cut is precisely $|\widetilde{\boldsymbol{\phi}}(u) - \widetilde{\boldsymbol{\phi}}(v)|$. So, the expected capacity of the edges in a random cut is given by (14), and so there is a cut of capacity at most (14). □

Now, suppose that one has a fixed total amount of resistance μ to distribute over the edges of a cut of size F, and that every other edge is assigned resistance zero. It is not difficult to see that the maximum possible effective resistance between s and t in such a case is $\frac{\mu}{F^2}$, and that this is achieved when one puts a resistance of $\frac{\mu}{F}$ on each of the edges. This suggests the following lemma, which bounds the quantity in Lemma 5.2 in terms of the effective resistance and the total resistance (appropriately weighted when the edges have non-unit capacities):

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

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

If, $\tilde{\boldsymbol{\phi}}$ is an approximate electrical potential returned by the algorithm of Theorem 2.3 when run with parameter $\delta \leq 1/3$, re-scaled to have potential difference 1 between s and t, then

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

Proof. By Fact 2.4, the rescaled true electrical potentials correspond to a flow of value $1/R_{\text{eff}}(\boldsymbol{r})$ and

$$\sum_e \frac{\boldsymbol{\phi}(e)^2}{r_e} = \frac{1}{R_{\text{eff}}(\boldsymbol{r})}.$$

So, we can apply the Cauchy-Schwarz inequality to prove

$$\sum_e \phi(e) u_e \leq \sqrt{\sum_e \frac{\phi(e)^2}{r_e} \sum_e u_e^2 r_e}$$
$$= \sqrt{\frac{\mu}{R_{\text{eff}}(\boldsymbol{r})}}.$$

When we call the algorithm implicit in Theorem 2.3, we should specify the value of the flow, even though it will become immaterial after we normalize the potential drop. Let's assume the value is $F = 1$. By part a of Theorem 2.3, the energy of the potential returned is at most $(1 + \delta)$ times the energy of the electrical flow of value 1. By part c of that theorem, the potential drop between s and t of the potential returned at least $(1 - \delta/12nmR)$ times the potential drop of the electrical flow of value 1. So, if $\tilde{\boldsymbol{\phi}}$ is the result of rescaling the potential returned to have potential drop 1 between s and t, then it will have energy

$$\sum_e \frac{\tilde{\boldsymbol{\phi}}(e)^2}{r_e} \leq \frac{1+\delta}{1-\delta/12nmR} \frac{1}{R_{\text{eff}}(\boldsymbol{r})} \leq (1+2\delta) \frac{1}{R_{\text{eff}}(\boldsymbol{r})},$$

as $\delta \leq 1/3$. The rest of the analysis follows from another application of Cauchy-Schwarz. □

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

We'll show that if $F \geq F^*$ and $\epsilon < 1/7$, 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

$$R_{\text{eff}}(\boldsymbol{r}^i) \geq (1 - 7\epsilon) \frac{\mu^i}{F^2}. \tag{15}$$

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

$$\frac{1+2\delta}{\sqrt{1-7\epsilon}} F \leq \frac{1}{1-7\epsilon} F, \quad (\text{as } \epsilon < 1/7).$$

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,

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

and the total weight

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

of the edges of the entire graph. 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 small for too many iterations will be similar to our analysis of the flow algorithm in Section 4. To obtain a contradiction, we will examine what happens if $\text{Reff}^i_{st} \leq (1-7\epsilon)\frac{\mu^i}{F^2}$ 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 has 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 we are assuming that the effective resistance is at most $(1-7\epsilon)\frac{\mu^i}{F^2}$, this cannot happen too many times.

The combined bounds on the number of iterations of the algorithm from 2 and 3 will be less than N, which will yield a contradiction.

Lemma 5.4. *For each $i \leq N$ such that $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$,*

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

So, if for all $i \leq N$ we have $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$, then

$$\mu^N \leq \mu^0 \exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}N\right). \tag{16}$$

Proof. If $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$ then the electrical flow f of value F has energy

$$F^2 R_{\text{eff}}(\boldsymbol{r}^i) = \sum_e (f^i_e)^2 r_e^2 = \sum_e \mathsf{cong}_{f^i}(e)^2 w^i_e \leq (1-7\epsilon)\mu^i.$$

By Theorem 2.3, the approximate electrical flow $\tilde{f}$ has energy at most $(1+\delta)$ times the energy of f^i. So,

$$\sum_e \mathsf{cong}_{\tilde{f}^i}(e)^2 w^i_e \leq (1+\delta)(1-7\epsilon)\mu^i \leq (1-6\epsilon)\mu^i \quad (\text{as } \delta = \epsilon^2 \text{ and } \epsilon < 1/7).$$

Applying the Cauchy-Schwarz inequality, we find

$$\sum_e \mathsf{cong}_{\tilde{f}^i}(e) w^i_e \leq \sqrt{\sum_e w^i_e}\sqrt{\sum_e \mathsf{cong}_{\tilde{f}^i}(e)^2 w^i_e} \leq \sqrt{1-6\epsilon}\mu^i \leq (1-3\epsilon)\mu^i.$$

Now we can compute

$$\begin{aligned}
\mu^{i+1} &= \sum_e w^{i+1}_e \\
&= \sum_e w^i_e + \frac{\epsilon}{\rho}\sum_e \mathsf{cong}_{f^i}(e) w^i_e + \frac{\epsilon^2}{\rho}\mu^i \\
&\leq \left(1 + \frac{\epsilon(1-2\epsilon)}{\rho}\right)\mu^i \\
&\leq \mu^i \exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}\right)
\end{aligned}$$

as desired. □

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

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

Proof. To bound the increase of ν^{i+1} over ν^i, we use the inequality

$$(1+\epsilon x) \ge \exp\left(\epsilon(1-\epsilon)x\right), \tag{17}$$

which holds for ϵ and x between 0 and 1. We apply this inequality with $x = \mathsf{cong}_{\tilde{f}^i}(e)/\rho$. As $\tilde{f}^i$ is a flow of value F and C is a cut, $\sum_{e\in C}\left|\tilde{f}^i_e\right| \ge F$. We now compute

$$\begin{aligned}
\nu^{i+1} &= \left(\prod_{e\in C}\left(w_e^{t+1}\right)^{u_e}\right)^{1/u_C} \\
&\ge \left(\prod_{e\in C}\left(w_e^i\left(1+\frac{\epsilon}{\rho}\mathsf{cong}_{\tilde{f}^i}(e)\right)\right)^{u_e}\right)^{1/u_C} \\
&= \nu^i\left(\prod_{e\in C}\left(1+\frac{\epsilon}{\rho}\mathsf{cong}_{\tilde{f}^i}(e)\right)^{u_e}\right)^{1/u_C} \\
&\ge \nu^i \exp\left(\frac{1}{u_C}\sum_{e\in C} u_e\frac{\epsilon(1-\epsilon)}{\rho}\mathsf{cong}_{\tilde{f}^i}(e)\right) \\
&= \nu^i \exp\left(\frac{1}{u_C}\sum_{e\in C}\frac{\epsilon(1-\epsilon)}{\rho}\left|\tilde{f}^i_e\right|\right) \\
&\ge \nu^i \exp\left(\frac{1}{u_C}\frac{\epsilon(1-\epsilon)}{\rho}F\right) \\
&\ge \nu^i \exp\left(\frac{\epsilon(1-\epsilon)}{\rho}\right),
\end{aligned}$$

as $F \ge F^* = u_C$. □

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

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

Proof. We first show by induction that

$$w_e^i \ge \frac{\epsilon}{m}\mu^i.$$

If $i = 0$ we have $1 \geq \epsilon$. For $i > 0$, we have

$$\begin{aligned}
w_e^{i+1} &\geq w_e^i + \frac{\epsilon^2}{m\rho}\mu^i \\
&\geq \left(\frac{\epsilon}{m} + \frac{\epsilon^2}{m\rho}\right)\mu^i \\
&= \frac{\epsilon}{m}\left(1 + \frac{\epsilon}{\rho}\right)\mu^i \\
&\geq \frac{\epsilon}{m}\exp\left(\frac{\epsilon(1-\epsilon)}{\rho}\right) \quad \text{(by applying (17) with } x = 1/\rho) \\
&\geq \frac{\epsilon}{m}\exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}\right)\mu^i \\
&\geq \frac{\epsilon}{m}\mu^{i+1},
\end{aligned}$$

by Lemma 5.4.

We now show that an edge e for which $\mathsf{cong}_{\tilde{f}^i}(e) \geq \rho$ contributes a large fraction of the energy to the true electrical flow of value F. By the assumptions of the lemma, the energy of the true electrical flow of value F is

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

On the other hand, the energy of the edge e in the approximate electrical flow is

$$(\tilde{f}_e^i)^2 r_e^i = \mathsf{cong}_{\tilde{f}^i}(e)^2 w_e \geq \rho^2 \frac{\epsilon}{m}\mu^i.$$

As a fraction of the energy of the true electrical flow, this is at least

$$\frac{1}{1-7\epsilon}\frac{\rho^2\epsilon}{m} = \frac{9}{(1-7\epsilon)\epsilon^{1/3}m^{1/3}}.$$

By part b of Theorem 2.3, the fraction of the energy that e accounts for in the true flow is at least

$$\frac{9}{(1-7\epsilon)\epsilon^{1/3}m^{1/3}} - \frac{\epsilon^2}{2mR} \geq \frac{9}{\epsilon^{1/3}m^{1/3}} = \frac{\rho^2\epsilon}{m} \quad (\text{as } \epsilon < 1/7 \text{ and } R \geq 1).$$

As

$$w_e^{i+1} \geq w_e^i\left(1 + \frac{\epsilon}{\rho}\mathsf{cong}_{\tilde{f}^i}(e)\right) \geq (1+\epsilon)w_e^i,$$

we have increased the weight of edge e, and therefore its resistance, by a factor of at least $(1+\epsilon)$. So, by Lemma 2.6,

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

as $\rho^2\epsilon^2/2m \leq 1$ and $1 + x \geq \exp(x/2)$ for $x \leq 1$. □

We now combine these lemmas to obtain our main bound:

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

As observed at the beginning of this section, our main result follows by combining this lemma with with Lemmas 5.2 and 5.3.

Theorem 5.8. *On input $\epsilon < 1/7$, the algorithm in Figure 6 runs in time $\widetilde{O}\left(m^{4/3}\epsilon^{-8/3}\right)$. 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 described in Section 2.2 by crudely approximating the minimum cut, and then applying the above algorithm in a binary search. Crudely analyzed, this incurs a multiplicative overhead of $O(\log m/\epsilon)$.

Proof of Lemma 5.7. Suppose as before that $R_{\text{eff}}(\boldsymbol{r}^i) \leq (1-7\epsilon)\frac{\mu^i}{F^2}$ for all $1 \leq i \leq N$. By Lemma 5.4 and the fact that $\mu^0 = m$, the total weight at the end of the algorithm is bounded by

$$\mu^N \leq \mu^0 \exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}N\right) = m\exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}N\right). \tag{18}$$

Let $A \subseteq \{1,\ldots,N\}$ be the set of i for which $\mathsf{cong}_{\tilde{f}^i}(e) \leq \rho$ for all e, and let $B \subseteq \{1,\ldots,N\}$ be the set of i for which there exists an edge e with $\mathsf{cong}_{\tilde{f}^i}(e) > \rho$. Let $a = |A|$ and $b = |B|$, and note that $a + b = N$. We will obtain a contradiction by proving bounds on a and b that add up to something less than N.

We begin by bounding a. By applying Lemma 5.5 to all of the steps in A and noting that ν^i never decreases during the other steps, we obtain

$$\nu^N \geq \exp\left(\frac{\epsilon(1-\epsilon)}{\rho}a\right)\nu^0 = \exp\left(\frac{\epsilon(1-\epsilon)}{\rho}a\right) \tag{19}$$

Since $\nu^N \leq \mu^N$, we can combine inequalities (18) and (19) to obtain

$$\exp\left(\frac{\epsilon(1-\epsilon)}{\rho}a\right) \leq m\exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}N\right).$$

Taking logs of both sides and solving for a yields

$$\frac{\epsilon(1-\epsilon)}{\rho}a \leq \frac{\epsilon(1-2\epsilon)}{\rho}N + \ln m,$$

from which we obtain

$$a \leq \frac{1-2\epsilon}{1-\epsilon}N + \frac{\rho}{\epsilon(1-\epsilon)}\ln m \leq (1-\epsilon)N + \frac{\rho}{\epsilon(1-\epsilon)}\ln m < (1-\epsilon)N + \frac{7\rho}{6\epsilon}\ln m. \tag{20}$$

We now use a similar argument to bound b. By applying Lemma 5.6 to all of the steps in B and noting that $R_{\text{eff}}(\boldsymbol{r}^i)$ never decreases during the other steps, we obtain

$$R_{\text{eff}}(\boldsymbol{r}^N) \geq \exp\left(\frac{\epsilon^2\rho^2}{4m}b\right)R_{\text{eff}}(\boldsymbol{r}^0) \geq \exp\left(\frac{\epsilon^2\rho^2}{4m}b\right)\frac{1}{m^2F^{*2}} \geq \exp\left(\frac{\epsilon^2\rho^2}{4m}b\right)\frac{1}{m^2F^2},$$

where the second inequality applies the bound $R_{\text{eff}}(\boldsymbol{r}^0) \geq 1/(m^2F^{*2})$, which follows from an argument like the one used to justify inequality (11). Using our assumption that $R_{\text{eff}}(\boldsymbol{r}^N) \leq (1-7\epsilon)\frac{\mu^N}{F^2}$ and the bound on μ^N from inequality (18), this gives us

$$\exp\left(\frac{\epsilon^2\rho^2}{4m}b\right)\frac{1}{m^2F^2} \leq \frac{1-7\epsilon}{F^2}m\exp\left(\frac{\epsilon(1-2\epsilon)}{\rho}N\right).$$

Taking logs and rearranging terms gives us

$$b \leq \frac{4m(1-2\epsilon)}{\epsilon\rho^3}N + \frac{4m}{\epsilon^2\rho^2}\ln\left((1-7\epsilon)m^3\right) < \frac{4m}{\epsilon\rho^3}N + \frac{12m}{\epsilon^2\rho^2}\ln m. \tag{21}$$

Adding the inequalities in (20) and (21), grouping terms, and plugging in the values of ρ and N, yields

$$\begin{aligned}
N = a + b &< \left((1-\epsilon) + \frac{4m}{\epsilon\rho^3}\right)N + \left(\frac{7\rho}{6\epsilon} + \frac{12m}{\epsilon^2\rho^2}\right)\ln m \\
&= \left((1-\epsilon) + \frac{4m}{\epsilon(27m\epsilon^{-2})}\right)(5\epsilon^{-8/3}m^{1/3}\ln m) + \left(\frac{7m^{1/3}\epsilon^{-2/3}}{2\epsilon} + \frac{12m}{\epsilon^2(9m^{2/3}\epsilon^{-4/3})}\right)\ln m \\
&= \left((1-\epsilon) + \frac{4\epsilon}{27}\right)(5\epsilon^{-8/3}m^{1/3}\ln m) + \left(\frac{7m^{1/3}}{2\epsilon^{5/3}} + \frac{12m^{1/3}}{9\epsilon^{2/3}}\right)\ln m \\
&= \frac{5m^{1/3}}{\epsilon^{8/3}}\ln m - \left(\frac{41}{54} - \frac{12\epsilon}{9}\right)\frac{m^{1/3}}{\epsilon^{5/3}}\ln m \\
&< \frac{5m^{1/3}\ln m}{\epsilon^{8/3}} = N.
\end{aligned}$$

This is our desired contradiction, which completes the proof. □

References

[1] I. Abraham and O. Neiman. Using petal-decompositions to build a low stretch spanning tree. In *STOC'12: Proceedings of the 44th Annual ACM Symposium on the Theory of Computing*, pages 395–406, 2012.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

[3] 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.

[4] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[5] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *STOC'96: Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 47–55, 1996.

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

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

[8] S. I. Daitch and D. A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 451–460, 2008.

[9] P. Elias, A. Feinstein, and C. E. Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, 2, 1956.

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

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

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

[13] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS'98: Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

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

[15] D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In *SODA '98: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 490–499, 1998.

[16] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. 2013.

[17] J. A. Kelner, G. L. Miller, and R. Peng. Faster approximate multicommodity flow using quadratically coupled flows. In *STOC'12: Proceedings of the 44th Annual ACM Symposium on the Theory of Computing*, pages 1–18, 2012.

[18] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *STOC'13: Proceedings of the 45th Annual ACM Symposium on the Theory of Computing*, pages 911–920, 2013.

[19] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD systems. In *FOCS'10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 235–244, 2010.

[20] I. Koutis, G. L. Miller, and R. Peng. A nearly $m \log n$-time solver for SDD linear systems. In *FOCS'11: Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 590–598, 2011.

[21] Y. T. Lee, S. Rao, and N. Srivastava. A new approach to computing maximum flows using electrical flows. In *STOC'13: Proceedings of the 45th Annual ACM Symposium on the Theory of Computing*, pages 755–764, 2013.

[22] A. Mądry. Fast approximation algorithms for cut-based problems in undirected graphs. In *FOCS'10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 245–254, 2010.

[23] A. Mądry. Navigating central path with electrical flows: from flows to matchings, and back. In *FOCS'13: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, 2013.

[24] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

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

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

[27] J. Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cuts. In *FOCS'09: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 363–372, 2009.

[28] J. Sherman. Nearly maximum flows in nearly linear time. In *FOCS'13: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, 2013.

[29] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC'04: Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 81–90, 2004.

[30] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS'01: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.

A Computing Electrical Flows

Proof of Theorem 2.3. Without loss of generality, we may scale the resistances so that they lie between 1 and R. This gives the following relation between F and the energy of the electrical s-t flow of value F:

$$\frac{F^2}{m} \leq \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) \leq F^2 R m. \tag{22}$$

Let $\boldsymbol{L}$ be the Laplacian matrix for this electrical flow problem. Note that all off-diagonal entries of $\boldsymbol{L}$ lie between -1 and $-1/R$. Recall that the vector of optimal vertex potentials $\boldsymbol{\phi}$ of the electrical flow is the solution to the following linear system:

$$\boldsymbol{L}\boldsymbol{\phi} = F\chi_{s,t}$$

For any $\epsilon > 0$, the algorithm of Koutis, Miller and Peng [20], using the low-stretch trees of Abraham and Neiman [1], provides in time $O\left(m \log m \log\log m \log \epsilon^{-1}\right)$ a set of potentials $\widehat{\boldsymbol{\phi}}$ such that

$$\left\| \widehat{\boldsymbol{\phi}} - \boldsymbol{\phi} \right\|_{\boldsymbol{L}} \leq \epsilon \left\| \boldsymbol{\phi} \right\|_{\boldsymbol{L}},$$

where

$$\left\| \boldsymbol{\phi} \right\|_{\boldsymbol{L}} \stackrel{\text{def}}{=} \sqrt{\boldsymbol{\phi}^T \boldsymbol{L} \boldsymbol{\phi}}.$$

Note also that

$$\|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2 = \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}),$$

where $\boldsymbol{f}$ is the potential flow corresponding to $\boldsymbol{\phi}$. Now, let $\widehat{\boldsymbol{f}}$ be the potential flow corresponding to $\widehat{\boldsymbol{\phi}}$,

$$\widehat{\boldsymbol{f}} = \boldsymbol{C}\boldsymbol{B}^T\widehat{\boldsymbol{\phi}}.$$

The flow $\widehat{\boldsymbol{f}}$ is not necessarily an s-t flow because $\widehat{\boldsymbol{\phi}}$ only approximately satisfies the linear system. The amount of flow entering and leaving s and t might not exactly equal F, and a small amount of flow can enter or leave every other vertex. In other words, $\widehat{\boldsymbol{f}}$ may not satisfy $\boldsymbol{B}\widehat{\boldsymbol{f}} = F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}$. Below, we will compute an approximation $\tilde{\boldsymbol{f}}$ of $\widehat{\boldsymbol{f}}$ that satisfies $\boldsymbol{B}\tilde{\boldsymbol{f}} = F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}$.

But first, we first observe that the energy of $\widehat{\boldsymbol{f}}$ is not much more than the energy of $\boldsymbol{f}$. We have

$$\left\|\widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 = \mathcal{E}_{\boldsymbol{r}}(\widehat{\boldsymbol{f}}),$$

and by the triangle inequality

$$\left\|\widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}} \leq \|\boldsymbol{\phi}\|_{\boldsymbol{L}} + \left\|\widehat{\boldsymbol{\phi}} - \boldsymbol{\phi}\right\|_{\boldsymbol{L}} \leq (1+\epsilon)\|\boldsymbol{\phi}\|_{\boldsymbol{L}}.$$

So

$$\mathcal{E}_{\boldsymbol{r}}(\widehat{\boldsymbol{f}}) \leq (1+\epsilon)^2\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}).$$

We now set $\tilde{\boldsymbol{\phi}} = \widehat{\boldsymbol{\phi}}$ to be the potentials returned by the algorithm. We will set $\epsilon < \delta/3$, so that these potentials satisfy the requirements of part a.

To see that the flow $\widehat{\boldsymbol{f}}$ does not flow too much into or out of any vertex other than s and t, note that the flows into and out of vertices are given by the vector

$$\boldsymbol{i}_{ext} \stackrel{\text{def}}{=} \boldsymbol{B}\widehat{\boldsymbol{f}}.$$

Note that the sum of the entries in $\boldsymbol{i}_{ext}$ is zero. We will produce an s-t flow of value F by adding a flow to $\widehat{\boldsymbol{f}}$ to obtain a flow $\tilde{\boldsymbol{f}}$ for which

$$\boldsymbol{B}\tilde{\boldsymbol{f}} = F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}.$$

Let η be the maximum discrepancy between $\boldsymbol{i}_{ext}$ and $F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}$:

$$\eta \stackrel{\text{def}}{=} \|\boldsymbol{i}_{ext} - F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}\|_\infty.$$

We have

$$\eta \leq \|\boldsymbol{i}_{ext} - F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}\|_2 = \left\|\boldsymbol{L}\widehat{\boldsymbol{\phi}} - \boldsymbol{L}\boldsymbol{\phi}\right\|_2 \leq \|\boldsymbol{L}\|_2\left\|\widehat{\boldsymbol{\phi}} - \boldsymbol{\phi}\right\|_{\boldsymbol{L}} \leq 2n\left\|\widehat{\boldsymbol{\phi}} - \boldsymbol{\phi}\right\|_{\boldsymbol{L}} \leq 2n\epsilon\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})}.$$

It is an easy matter to produce an s-t flow $\tilde{\boldsymbol{f}}$ that differs from $\widehat{\boldsymbol{f}}$ by at most $n\eta$ on each edge. For example, one can do this by solving a flow problem in a spanning tree of the graph. Let T be any spanning tree of the graph G. We now construct a flow in T with the demands $F\boldsymbol{\chi}_{\boldsymbol{s},\boldsymbol{t}}(u) - \boldsymbol{i}_{ext}(u)$ at each vertex u. As the sum of the positive demands in this flow is at most $n\eta$, one can find such a flow in which every edge flows at most $n\eta$. Moreover, since the edges of T are a tree, one can

find the flow in linear time. We obtain the flow $\tilde{\boldsymbol{f}}$ by adding this flow to $\widehat{\boldsymbol{f}}$. The resulting flow is an s-t flow of value F. Moreover,

$$\left\|\widehat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\right\|_\infty \leq n\eta.$$

To ensure that we get a good approximation of the electrical flow, we will set

$$\epsilon = \frac{\delta}{14n^2 m R^{3/2}}.$$

To check that $\tilde{\boldsymbol{f}}$ satisfies the requirements of part a, observe that

$$\begin{aligned}
\mathcal{E}_{\boldsymbol{r}}(\tilde{\boldsymbol{f}}) &= \sum_e r_e \tilde{f}_e^2 \\
&\leq \sum_e r_e (\widehat{\boldsymbol{f}}_e + n\eta)^2 \\
&\leq \sum_e r_e \left((1+\delta/2)\widehat{\boldsymbol{f}}_e^2 + (1+2/\delta)(n\eta)^2\right) \\
&\leq (1+\delta/2)(1+\epsilon)^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) + (1+2/\delta)4n^4\epsilon^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) \\
&\leq (1+\delta/2)(1+\epsilon)^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) + (\delta/3)\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) && \text{(by our choice of } \epsilon) \\
&\leq (1+\delta)\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}),
\end{aligned}$$

as desired.

To establish part b, note that

$$\|\boldsymbol{\phi}\|_{\boldsymbol{L}} = \sum_e r_e f_e^2$$

and so

$$\sum_e r_e (f_e - \widehat{f}_e)^2 = \left\|\boldsymbol{\phi} - \widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 \leq \epsilon^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}).$$

We also have

$$\sum_e r_e (f_e + \widehat{f}_e)^2 = \left\|\boldsymbol{\phi} + \widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 \leq \left(2\|\boldsymbol{\phi}\|_{\boldsymbol{L}} + \left\|\boldsymbol{\phi} - \widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}\right)^2 \leq (2+\epsilon)^2 \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2.$$

We may now bound $\left|r_e f_e^2 - r_e \widehat{f}_e^2\right|$ by

$$\begin{aligned}
\left|r_e f_e^2 - r_e \widehat{f}_e^2\right|^2 &\leq \left\|\boldsymbol{\phi} - \widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 \left\|\boldsymbol{\phi} + \widehat{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 \\
&= r_e \left(f_e - \widehat{f}_e\right)^2 r_e \left(f_e + \widehat{f}_e\right)^2 \\
&\leq \epsilon^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})(2+\epsilon)^2 \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}),
\end{aligned}$$

which implies

$$\left|r_e f_e^2 - r_e \widehat{f}_e^2\right| \leq \epsilon(2+\epsilon)\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}).$$

It remains to bound $\left|r_e\widehat{f}_e^2 - r_e\tilde{f}_e^2\right|$. We begin by recalling that we can assume all resistances lie between 1 and R. So,

$$\begin{aligned}\left|r_e\widehat{f}_e^2 - r_e\tilde{f}_e^2\right| &= \sqrt{r_e}\left|\widehat{f}_e - \tilde{f}_e\right|\sqrt{r_e}\left|\widehat{f}_e + \tilde{f}_e\right| \\ &\leq \sqrt{R}\left|\widehat{f}_e - \tilde{f}_e\right|\sqrt{r_e}\left|\widehat{f}_e + \tilde{f}_e\right| \\ &\leq \sqrt{R}n\eta\sqrt{r_e}\left|\widehat{f}_e + \tilde{f}_e\right|.\end{aligned}$$

We bound the right-hand part of the right-hand expression by

$$\begin{aligned}\sqrt{r_e}\left|\widehat{f}_e + \tilde{f}_e\right| &\leq 2\sqrt{r_e}\left|\widehat{f}_e\right| + \sqrt{r_e}\left|\widehat{f}_e - \tilde{f}_e\right| \\ &\leq 2\sqrt{\mathcal{E}_{\boldsymbol{r}}(\widehat{f})} + \sqrt{r_e}n\eta \\ &\leq 2(1+\epsilon)\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})} + 2\sqrt{r_e}n^2\epsilon\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})} \\ &\leq 3\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})},\end{aligned}$$

by our choice of ϵ.

So,

$$\begin{aligned}\left|r_e\widehat{f}_e^2 - r_e\tilde{f}_e^2\right| &\leq \sqrt{R}n\eta\sqrt{r_e}\left|\widehat{f}_e + \tilde{f}_e\right| \\ &\leq 3\sqrt{R}n\eta\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})} \\ &\leq 6\epsilon\sqrt{R}n^2\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}).\end{aligned}$$

Putting these inequalities together we find

$$\left|r_e f_e^2 - r_e\tilde{f}_e^2\right| \leq \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})\left(\epsilon(2+\epsilon) + 6\epsilon\sqrt{R}n^2\right) \leq \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})\frac{\delta}{2mR},$$

by our choice of ϵ.

To prove part c, first recall that

$$\boldsymbol{L}\boldsymbol{\phi} = F\chi_{s,t}$$

and that

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) = F^2 R_{\text{eff}}(\boldsymbol{r}).$$

So,

$$\begin{aligned}\epsilon^2\left\|\boldsymbol{\phi}\right\|_{\boldsymbol{L}}^2 &\geq \left\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 \\ &= \left(\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\right)^T \boldsymbol{L}\left(\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\right) \\ &= \left\|\tilde{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 + \left\|\boldsymbol{\phi}\right\|_{\boldsymbol{L}}^2 - 2\boldsymbol{\phi}^T\boldsymbol{L}\tilde{\boldsymbol{\phi}} \\ &= \left\|\boldsymbol{\phi}\right\|_{\boldsymbol{L}}^2 + \left\|\tilde{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 - 2F\chi_{s,t}^T\tilde{\boldsymbol{\phi}}.\end{aligned}$$

Thus,

$$\begin{aligned} 2F\chi_{s,t}^T\tilde{\boldsymbol{\phi}} &\geq \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2 + \left\|\tilde{\boldsymbol{\phi}}\right\|_{\boldsymbol{L}}^2 - \epsilon^2 \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2 \\ &\geq \left(1 + (1-\epsilon)^2 - \epsilon^2\right) \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2 \\ &= (2-2\epsilon) \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2 \\ &= (2-2\epsilon) F^2 R_{\text{eff}}(\boldsymbol{r}). \end{aligned}$$

Part c now follows from $\tilde{\boldsymbol{\phi}}(s) - \tilde{\boldsymbol{\phi}}(t) = \chi_{s,t}^T\tilde{\boldsymbol{\phi}}$. □