2016 IEEE 57th Annual Symposium on Foundations of Computer Science

Computing Maximum Flow with Augmenting Electrical Flows (Extended Abstract)

Aleksander Mądry*
MIT
Cambridge, MA
madry@mit.edu

***Abstract*—We present an $\widetilde{O}\left(m^{\frac{10}{7}}U^{\frac{1}{7}}\right)$-time algorithm for the maximum s-t flow problem (and the minimum s-t cut problem) in directed graphs with m arcs and largest integer capacity U. This matches the running time of the $\widetilde{O}\left((mU)^{\frac{10}{7}}\right)$-time algorithm of Mądry [30] in the unit-capacity case, and improves over it, as well as over the $\widetilde{O}\left(m\sqrt{n}\log U\right)$-time algorithm of Lee and Sidford [25], whenever U is moderately large and the graph is sufficiently sparse. By well-known reductions, this also implies similar running time improvements for the maximum-cardinality bipartite b-matching problem.**

One of the advantages of our algorithm is that it is significantly simpler than the ones presented in [30] and [25]. In particular, these algorithms employ a sophisticated interior-point method framework, while our algorithm is cast directly in the classic augmenting path setting that almost all the combinatorial maximum flow algorithms use. At a high level, the presented algorithm takes a primal dual approach in which each iteration uses electrical flows computations both to find an augmenting s-t flow in the current residual graph and to update the dual solution. We show that by maintain certain careful coupling of these primal and dual solutions we are always guaranteed to make significant progress.

***Keywords*-maximum flow problem; augmenting paths; minimum s-t cut problem; bipartite matchings; electrical flows; Laplacian linear systems;**

I. 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 graph problems in combinatorial optimization. They have a wide range of applications, are often used as subroutines in other algorithms (see, e.g., [1], [36]), and a number of other important problems – e.g., bipartite matching problem – can be reduced to them. Furthermore, these two problems were often a testbed for development of fundamental algorithmic tools and concepts. Most prominently, the Max-Flow Min-Cut theorem constitutes the prototypical primal-dual relation.

Several decades of extensive work resulted in a number of developments on these problems (see Goldberg and Rao [12] for an overview) and many of their generalizations and special cases. Still, despite all this effort, the basic problem of computing maximum s-t flow and minimum s-t cut in general graphs resisted progress for a long time. In particular, for a number of years, the best running time bound for the problem was an $O(m\min\{m^{\frac{1}{2}}, n^{\frac{2}{3}}\}\log(n^2/m)\log U)$ (with U denoting the largest integer arc capacity) bound established in a a breakthrough paper by Goldberg and Rao [12] and this bound, in turn, matched the $O(m\min\{m^{\frac{1}{2}}, n^{\frac{2}{3}}\})$ bound for unit-capacity graphs due to Even and Tarjan [9] – and, independently, Karzanov [15] – that were put forth more than 40 years ago.

The above bounds were improved only fairly recently. Specifically, in 2013, Mądry [30] presented an interior-point method based framework for flow computations that gave an $\widetilde{O}\left(m^{\frac{10}{7}}\right)$-time[1] algorithm for the unit-capacity case of the maximum s-t flow and minimum s-t cut problems. This finally broke the long-standing $\widetilde{O}\left(n^{\frac{3}{2}}\right)$ running time barrier for sparse graphs, i.e., for $m = O(n)$. Later on, Lee and Sidford [25] developed a variant of interior-point method that delivers an improvement for the regime of dense graphs. In particular, their algorithm is able to compute the (general) maximum s-t flow and minimum s-t cut in $\widetilde{O}\left(m\sqrt{n}\log U\right)$ time and thus improve over the Goldberg-Rao bound whenever the input graph is sufficiently dense.

It is also worth mentioning that, as a precursor to the above developments, substantial progress was made in the context of $(1-\varepsilon)$-approximate variant of the maximum s-t flow problem in undirected graphs. In 2011, Christiano et al. [3] developed an algorithm that allows one to compute a $(1+\varepsilon)$-approximation to the undirected maximum s-t flow (and the minimum s-t cut) problem in $\widetilde{O}\left(mn^{\frac{1}{3}}\varepsilon^{-11/3}\right)$ time. Their result relies on devising a new approach to the problem that combines electrical flow computations with multiplicative weights update method (see [1]). Later, Lee et al. [24] presented a quite different – but still electrical-flow-based – algorithm that employs purely gradient-descent-type view to obtain an $\widetilde{O}\left(mn^{1/3}\varepsilon^{-2/3}\right)$-time $(1+\varepsilon)$-approximation for the case of unit capacities. Very recently, this line of work was culminated by Sherman [37] and Kelner et al. [16] who independently showed how to in-

*Supported by NSF grant CCF-1553428, and Sloan Research Fellowship.

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

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.70

tegrate non-Euclidean gradient-descent methods with fast poly-logarithmic-approximation algorithms for cut problems of Mądry [28] to get an $O(m^{1+o(1)}\varepsilon^{-2})$-time $(1+\varepsilon)$-approximation to the undirected maximum flow problem. Then, Peng [34] built on these works to obtain a truly nearly-linear, i.e., $\widetilde{O}\left(m\varepsilon^{-2}\right)$, running time.

Finally, we note that, in parallel to the above work that is focused on designing weakly-polynomial algorithms for the maximum s-t flow and minimum s-t cut problems, there is also a considerable interest in obtaining running time bounds that are strongly-polynomial, i.e., that do not depend on the values of arc capacities. The current best such bound is $O(mn)$ and it follows by combining the algorithms of King et al. [18] and Orlin [33].

Bipartite Matching Problem.: Another problem that is related to the maximum s-t problem – and, in fact, can be reduced to it – is the (maximum-cardinality) bipartite matching problem. This problem is a fundamental assignment task with numerous applications and long history. Already in 1931, König [19] and Egerváry [8] provided first constructive characterization of maximum matchings in bipartite graphs. This characterization can be turned into a polynomial-time algorithm. Then, in 1973, Hopcroft and Karp [14] and, independently, Karzanov [15], devised the celebrated $O(m\sqrt{n})$-time algorithm. For 40 years this bound remained the best one known in the regime of relatively sparse graphs. Only recently Mąadry [30] obtained an improved $\widetilde{O}\left(m^{10/7}\right)$ running time. It turns out, however, that whenever the input graph is dense, i.e., when m is close to n^2 even better bounds can be obtain. In this regime, one can combine the algebraic approach of Rabin and Vazirani [35] – that builds on the work of Tutte [39] and Lovász [26] – with matrix-inversion techniques of Bunch and Hopcroft [2] to get an algorithm that runs in $O(n^{\omega})$ time, where $\omega \leq 2.3727$ is the exponent of matrix multiplication [5], [40].

Finally, a lot of developments has been done in the context of the (maximum-cardinality) matching problem in general, i.e., not necessarily bipartite, graphs. Starting with the pioneering work of Edmonds [6], these developments led to bounds that essentially match the running time guarantees that were previously known only for bipartite case. More specifically, the running time bound of $O(m\sqrt{n})$ for the general-graph case was obtained by Micali and Vazirani [31], [41] (see also [10] and [11]). Then, Mucha and Sankowski [32] gave an $O(n^{\omega})$-time algorithm. This algorithm was later simplified by Harvey [13].

A. Our Contribution

In this paper, we put forth a new algorithm for solving the maximum s-t flow and the minimum s-t cut problems in directed graphs. More precisely, we develop an algorithm that computes the maximum s-t flow of an input graph in time $\widetilde{O}\left(m^{\frac{10}{7}}U^{\frac{1}{7}}\right)$, where m denotes the number of arcs of that graph and U its largest integer capacity. Known reductions imply similar running time bounds for the minimum s-t cut problem as well as for the maximum-cardinality bipartite $\boldsymbol{b}$-matching problem, a natural generalization of the maximum bipartite matching problem in which each vertex v has a degree demand b_v. For that problem, our algorithm yields an $\widetilde{O}\left(m^{\frac{10}{7}}B^{\frac{1}{7}}\right)$-time algorithm, with B being the largest (integer) vertex demand.

In the light of the above, for the unit-capacity/demand cases, the resulting algorithms match the performance of the algorithm of Mądry [30]. The latter algorithm, however, runs in $\widetilde{O}\left(mU^{\frac{10}{7}}\right)$ time (which translates into an $\widetilde{O}\left((mB)^{\frac{10}{7}}\right)$ running time for the bipartite $\boldsymbol{b}$-matching problem) in the case of arbitrary capacities/demands. Consequently, the significantly better dependence of the running time of our algorithm on the largest capacity U/ largest demand B makes it much more favorable in that setting. In fact, even though that dependence on U/B is still polynomial it enables our algorithm to remain competitive, for a non-trivial range of parameters, with the best existing algorithms that run in time that is logarithmic in U/B, such as the $\widetilde{O}\left(m\sqrt{n}\log U\right)$-time algorithm of Lee and Sidford [25].

Even more crucially, the key advantage of our algorithm is that it is significantly simpler than both the algorithm of Mądry [30] and that of Lee and Sidford [25]). Both these algorithms rely heavily on the interior-point method framework. Specifically, [30] designed a certain new variant of path-following interior-point method algorithm for the near-perfect bipartite $\boldsymbol{b}$-matching problem that encoded the input maximum s-t flow instance. It then used electrical flow computations to converge to the near-optimal solution for that problem. In order to break the bottlenecking $\widetilde{O}\left(m^{\frac{1}{2}}\right)$ iteration bound, however, Mądry [30] needed to, first, develop an extensive toolkit for perturbing and preconditioning the underlying electrical flow computation and, then, to combine this machinery with a very careful and delicate analysis of the resulting dynamics.

Our algorithm also relies on electrical flow computations but it abandons the above methodology and works instead fully within the classic augmenting path framework that almost all the previous combinatorial maximum s-t flow algorithms used. In this framework, the flow is built in stages. Each stage corresponds to finding a so-called augmenting flow in the current residual graph, which is a directed graph that encodes the solution found so far. The algorithm terminates when the residual graph admits no more augmenting flows, i.e., there is no path from s to t in it, since in this case the solution found so far has to be already optimal.

The chief bottleneck in the running time analysis of augmenting path based algorithms is ensuring that each flow augmentation stage makes sufficient progress. Specifically, one wants to obtain a good trade off between the amount of flow pushed in each augmentation step and the time needed to implement each such flow push. One simple approach is to just use here s-t path computations. This is a nearly-linear time procedure but it only guarantees pushing one unit of flow each time. A much more sophisticated primitive developed in this context are blocking flow computations. Combining this primitive with a simple duality argument enabled Goldberg and Rao [12], who built on the work of Even and Tarjan [9], to obtain an $O(m \min\{m^{\frac{1}{2}}, n^{\frac{2}{3}}\} \log U)$-time maximum flow that remained the best known algorithm for nearly two decades. Unfortunately, trying to improve such blocking flow-based approaches turned out to be extremely difficult and no progress was made here so far, even in the unit-capacity case for which the best known bounds were established over 40 years ago.

One of the key contributions of this paper is bringing a new type of primitive: electrical flows to the augmenting path framework; and showing how to successfully use it to outperform the blocking flow-based methods. Specifically, our algorithm finds augmenting flows by computing electrical flows in certain symmetrization of the current residual graph – see Section III-C for more details. (Note that performing such a symmetrization is necessary as residual graphs are inherently directed while electrical flows are inherently undirected.) The key difficulty that arises here though is that this symmetrized residual graph might not support anymore a significant fraction of the s-t capacity of the original residual graph. It is not hard to see that, in general, this could be the case. To address this problem we introduce a certain careful coupling of the primal and dual solutions, which is inspired by the so-called centrality condition arising in interior-point method based maximum flow algorithms (see [30]). We then show that maintaining this coupling and applying a simple preconditioning technique let us guarantee that looking only for the flows in the symmetrized version of the residual graph still provides sufficient progress in each iteration and, in particular, immediately delivers a $\widetilde{O}\left(m^{\frac{3}{2}} \log U\right)$-time algorithm.

We then build on that basic algorithm and develop an ℓ_p-geometric understanding of its running time analysis. This understanding guides us towards a simple electrical flow perturbation technique – akin to the perturbation techniques used in [3] and [30] – that enables us to break the $\Omega(\sqrt{m})$ iterations bottleneck that all the blocking flow-based algorithms were suffering from, and thus get the final, improved result.

We believe that further study of this new augmenting flow based framework will deliver even faster and simpler algorithms.

II. Preliminaries

Throughout this paper, we will be viewing graphs as having both lower and upper capacities. Specifically, we will denote by $G = (V, E, \boldsymbol{u})$ a directed graph with a vertex set V, an arc set E (we allow parallel arcs), and two (non-negative) integer capacities u_e^- and u_e^+, for each arc $e \in E$. (We will explain the role of these capacities below.) Usually, m will denote the number $|E|$ of arcs of the graph in question and $n = |V|$ will be the number of its vertices. We view each arc e of G as an ordered pair (u, v), where u is its *tail* and v is its *head*.

Observe that this perspective enables us to view undirected graphs as directed ones in which the ordered pair $(u, v) \in E$ is an (undirected) *edge* (u, v) and the order just specifies the *orientation* of that edge (from u to v).

Maximum Flow Problem.: The basic notion of this paper is the notion of a *flow*. Given a graph G, we view a flow in G as a vector $\boldsymbol{f} \in \mathbb{R}^m$ that assigns a value f_e to each arc e of G. If this value is negative we interpret it as having a flow of $|f_e|$ flowing in the direction opposite to the arc orientation.

We say that a flow $\boldsymbol{f}$ is an *$\boldsymbol{\sigma}$-flow*, for some *demands* $\boldsymbol{\sigma} \in \mathbb{R}^n$ iff it satisfies *flow conservation constraints* with respect to that demands. That is, we have that

$$\sum_{e \in E^+(v)} f_e - \sum_{e \in E^-(v)} f_e = \sigma_v, \quad \text{for each vertex } v \in V. \tag{1}$$

Here, $E^+(v)$ (resp. $E^-(v)$) is the set of arcs of G that are entering (resp. leaving) vertex v. Intuitively, these constraints enforce that the net balance of the total in-flow into vertex v and the total out-flow out of that vertex is equal to σ_v, for every $v \in V$. (Observe that this implies, in particular, that $\sum_v \sigma_v = 0$.)

Furthermore, we say that a $\boldsymbol{\sigma}$-flow $\boldsymbol{f}$ is *feasible* in G iff $\boldsymbol{f}$ obeys the *the capacity constraints*:

$$-u_e^- \leq f_e \leq u_e^+, \quad \text{for each arc } e \in E. \tag{2}$$

In other words, we want each arc e to have a flow that is at most u_e^+ if it flows in the direction of e's orientation (i.e., $f_e \geq 0$), and at most u_e^-, if it flows in the opposite direction (i.e., $f_e < 0$). Note that setting all u_e^-s be equal to zero recovers the standard notion of flow feasibility in directed graphs.

One type of flows that will be of special interest to us are s-t flows, where s (the *source*) and t (the *sink*) are two distinguish vertices of G. Formally, an *s-t flow* is a $\boldsymbol{\sigma}$-flow whose demand vector $\boldsymbol{\sigma}$ is equal to $F \cdot \boldsymbol{\chi}_{s,t}$, where $F \geq 0$ is called the *value* of $\boldsymbol{f}$ and $\boldsymbol{\chi}_{s,t}$ is a demand vector that has -1 (resp. 1) at the coordinate corresponding to s (resp. t) and zeros everywhere else.

Now, the *maximum flow problem* corresponds to a task in which we are given a (directed) graph $G = (V, E, \boldsymbol{u})$ with integer capacities as well as a source vertex s and a sink vertex t and want to find a *feasible* (in the sense of (2)) s-t flow of maximum value. We will denote this maximum value as F^*.

Residual Graphs.: A fundamental object in many maximum flow algorithms (including ours) is the notion of a residual graph. Given a graph $G = (V, E, \boldsymbol{u})$ and a feasible $\boldsymbol{\sigma}$-flow $\boldsymbol{f}$ in that graph (it is useful to think $\boldsymbol{\sigma} = F \cdot \boldsymbol{\chi}_{s,t}$), we define the *residual graph* $G_{\boldsymbol{f}}$ (of G with respect to $\boldsymbol{f}$) as a graph $G_{\boldsymbol{f}} = (V, E, \widehat{\boldsymbol{u}}(\boldsymbol{f}))$ over the same vertex and arc set as G and such that, for each arc $e = (u, v)$ of G, its lower and upper capacities are defined as

$$\widehat{u}_e^+(\boldsymbol{f}) := u_e^+ - f_e \quad \text{and} \quad \widehat{u}_e^-(\boldsymbol{f}) := u_e^- + f_e. \qquad (3)$$

We will refer to $\widehat{u}_e^+(\boldsymbol{f})$ (resp. $\widehat{u}_e^-(\boldsymbol{f})$) as *forward residual capacity* (resp. *backward residual capacity*) of e and also define the *residual capacity* $\widehat{u}_e(\boldsymbol{f})$ of e as the minimum of these two, i.e., $\widehat{u}_e(\boldsymbol{f}) := \min\{\widehat{u}_e^-(\boldsymbol{f}), \widehat{u}_e^+(\boldsymbol{f})\}$. Note that the value of residual capacity depends on the flow $\boldsymbol{f}$ but we will ensure that it is always clear from the context with respect to which flow the residual capacity is measured. Also, observe that feasibility of $\boldsymbol{f}$ implies that all residual capacities are always non-negative (cf. (2)).

The main reason why residual graphs are useful in computing maximum flows is that they constitute a very convenient representation of the progress made so far. Specifically, we have the following important fact. (Again, it is useful to think here of the maximum s-t flow setting, in which $\boldsymbol{\sigma} = F^* \boldsymbol{\chi}_{s,t}$.)

Fact II.1. *Let $\boldsymbol{\sigma}$ be some demand and $G = (V, E, \boldsymbol{u})$ be a graph in which a demand of $\boldsymbol{\sigma}$ can be routed, i.e., there exists a $\boldsymbol{\sigma}$-flow $\boldsymbol{f}^*$ that is feasible in G. Also, for any $0 \leq \alpha \leq 1$, let $\boldsymbol{f}$ be a feasible $\alpha\boldsymbol{\sigma}$-flow in G, and $G_{\boldsymbol{f}} = (V, E, \widehat{\boldsymbol{u}}(\boldsymbol{f}))$ be the residual graph of G with respect to $\boldsymbol{f}$. We have that (a)*

1) *one can route a demand of $(1 - \alpha)\boldsymbol{\sigma}$ in $G_{\boldsymbol{f}}$;*
2) *if $\boldsymbol{f}'$ is a feasible $\alpha'\boldsymbol{\sigma}$-flow in $G_{\boldsymbol{f}}$, for some α', then $\boldsymbol{f} + \boldsymbol{f}'$ is a feasible $(\alpha + \alpha')\boldsymbol{\sigma}$-flow in G.*

Intuitively, the above fact enables us to reduce the task of routing a demand $\boldsymbol{\sigma}$ in G to a sequence of computations of augmenting $\alpha'\boldsymbol{\sigma}$-flows in the residual graph $G_{\boldsymbol{f}}$. We know that as long as we have not yet computed a feasible $\boldsymbol{\sigma}$-flow in G, $G_{\boldsymbol{f}}$ can route a demand of $(1-\alpha)\boldsymbol{\sigma}$-flow, where $(1 - \alpha) > 0$ is the fraction of routed demand that we are still "missing", and each new augmenting $\alpha'\boldsymbol{\sigma}$-flow found in $G_{\boldsymbol{f}}$ brings us closer to routing $\boldsymbol{\sigma}$ in full in G. (Note that initially $G_{\boldsymbol{f}}$ is equal to G and $G_{\boldsymbol{f}}$ is changing after each new augmenting $\alpha'\boldsymbol{\sigma}$-flow is found.)

Electrical Flows and Vertex Potentials.: Another notion that will play a fundamental role in this paper is the notion of electrical flows. We briefly review some of the key properties that we will need later.

Consider a graph G and a vector of resistances $\boldsymbol{r} \in \mathbb{R}^m$ that assigns to each edge e its *resistance* $r_e > 0$. For a given $\boldsymbol{\sigma}$-flow $\boldsymbol{f}$ in G, let us define its *energy* (with respect to resistances $\boldsymbol{r}$) $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})$ to be

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) := \sum_e r_e f_e^2. \qquad (4)$$

For a given demand vector $\boldsymbol{\sigma}$ and a vector of resistances $\boldsymbol{r}$, we define the *electrical $\boldsymbol{\sigma}$-flow* in G (that is *determined* by resistances $\boldsymbol{r}$) to be the flow that minimizes the energy $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})$ among all flows with demand $\boldsymbol{\sigma}$ in G. As energy is a strictly convex function, one can easily see that such a flow is unique. (It is important to keep in mind that such flow is *not* required to be feasible with respect to capacities of G, in the sense of (2).)

A very useful property of electrical flows is that they can be characterized in terms of vertex potentials inducing them. Namely, one can show that a flow $\boldsymbol{f}$ with demands $\boldsymbol{\sigma}$ in G is an electrical $\boldsymbol{\sigma}$-flow determined by resistances $\boldsymbol{r}$ iff there exist *vertex potentials* ϕ_v (that we collect into a vector $\boldsymbol{\phi} \in \mathbb{R}^n$) such that, for any edge $e = (u, v)$ in G,

$$f_e = \frac{\phi_v - \phi_u}{r_e}. \qquad (5)$$

In other words, a $\boldsymbol{f}$ with demands $\boldsymbol{\sigma}$ is an electrical $\boldsymbol{\sigma}$-flow iff it is *induced* via (5) by some vertex potential $\boldsymbol{\phi}$. (Note that the orientation of edges matters in this definition.) The above equation corresponds to the Ohm's law known from physics.

Note that we are able to express the energy $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f})$ (see (4)) of an electrical $\boldsymbol{\sigma}$-flow $\boldsymbol{f}$ in terms of the potentials $\boldsymbol{\phi}$ inducing it as

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}) = \sum_{e=(u,v)} \frac{(\phi_v - \phi_u)^2}{r_e}. \qquad (6)$$

One of the consequences of the above is that one can develop a dual characterization of the energy of an electrical $\boldsymbol{\sigma}$-flow in terms of optimization over vertex potentials. Namely, we have the following lemma whose proof can be found, e.g., in [30] Lemma 2.1.

Lemma II.2. *For any graph $G = (V, E)$, any vector of resistances $\boldsymbol{r}$, and any demand vector $\boldsymbol{\sigma}$,*

$$\frac{1}{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}^*)} = \min_{\boldsymbol{\phi} | \boldsymbol{\sigma}^T \boldsymbol{\phi} = 1} \sum_{e=(u,v) \in E} \frac{(\phi_v - \phi_u)^2}{r_e},$$

where $\boldsymbol{f}^*$ is the electrical $\boldsymbol{\sigma}$-flow determined by $\boldsymbol{r}$ in G. Furthermore, if $\boldsymbol{\phi}^*$ are the vertex potentials corresponding to $\boldsymbol{f}^*$ then the minimum is attained by taking $\boldsymbol{\phi}$ to be equal to $\widehat{\boldsymbol{\phi}} := \boldsymbol{\phi}^* / \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{f}^*)$.

Note that the above lemma provides a convenient way of lowerbounding the energy of an electrical $\boldsymbol{\sigma}$-flow. One just needs to expose any vertex potentials $\boldsymbol{\phi}$ such that $\boldsymbol{\sigma}^T \boldsymbol{\phi} = 1$ and this will immediately constitute an energy lowerbound.

Laplacian Solvers.: The fact that the electrical $\boldsymbol{\sigma}$-flow determined by resistances $\boldsymbol{r}$ is the only flow with demands $\boldsymbol{\sigma}$ that can be induced by vertex potentials (cf. (5)) has an important consequence. It enables us to reduce electrical $\boldsymbol{\sigma}$-flow computations to solving a linear system. In fact, the task of finding vertex potentials that induce that flow can be cast as a *Laplacian* linear system. That is, a linear system in which the constraint matrix corresponds to a Laplacian of the underlying graph with weights given by the (inverses of) the resistances $\boldsymbol{r}$.

Now, from the algorithmic point of view, the crucial property of Laplacian systems is that we can solve them, up to a very good approximation, very efficiently. Namely, there is a long line of work [38], [20], [21], [17], [4], [22], [23] that gives us a number of Laplacian system solvers that run in only nearly-linear time and, in case of more recent variants, are conceptually fairly simple. In particular, this line of work establishes the following theorem.

Theorem II.3. *For any $\varepsilon > 0$, any graph G with n vertices and m edges, any demand vector $\boldsymbol{\sigma}$, and any resistances $\boldsymbol{r}$, one can compute in $\widetilde{O}\left(m \log \varepsilon^{-1}\right)$ time vertex potentials $\tilde{\boldsymbol{\phi}}$ such that $\|\tilde{\boldsymbol{\phi}} - \boldsymbol{\phi}^*\|_{\boldsymbol{L}} \leq \varepsilon \|\boldsymbol{\phi}^*\|_{\boldsymbol{L}}$, where $\boldsymbol{L}$ is the Laplacian of G, $\boldsymbol{\phi}^*$ are potentials inducing the electrical $\boldsymbol{\sigma}$-flow determined by resistances $\boldsymbol{r}$, and $\|\boldsymbol{\phi}\|_{\boldsymbol{L}} := \sqrt{\boldsymbol{\phi}^T \boldsymbol{L} \boldsymbol{\phi}}$.*

Even though the solutions delivered by the above Laplacian solvers are only approximate, the quality of approximation that it delivers is more than sufficient for our purposes. Therefore, in the rest of this paper we assume that these solutions are exact. (See, e.g., [30] for discussion how to deal with inexactness of the electrical flows computed.)

III. Augmenting Residual Graphs with Electrical Flows

In this section, we put forth the general framework we will use to solve the maximum s-t problem. In particular, we demonstrate how this framework enables us to solve the maximum s-t flow problem in $\widetilde{O}\left(m^{\frac{3}{2}} \log U\right)$ time, where $m = |E|$ is the number of arcs in of the input graph and U is its largest (integer) capacity.

More precisely, for any maximum s-t flow instance $G = (V, E, \boldsymbol{u})$ and any value $F \geq 0$, our algorithm will work in $\widetilde{O}\left(m^{\frac{3}{2}} \log U\right)$ time and either: compute a feasible s-t flow of value F in G; or conclude that the maximum s-t flow value F^* of G is strictly smaller than F.

Note that such procedure can be turned into a "classic" maximum s-t flow by applying binary search over values of F and incurring a multiplicative running time overhead of only $O(\log Un)$. (In fact, a standard use of capacity scaling technique [7] enables one to keep the overall running time of the resulting algorithm be only linear, instead of quadratic, in $\log U$.)

Our algorithm follows the primal dual augmenting paths based framework. At a high level, in each iteration (see Section III-B), we use electrical flow computations to compute an augmenting flow as well as an update to the dual solution. To ensure that each augmenting iteration makes enough progress, we maintain a careful coupling of the primal and dual solution. We describe it below.

A. Primal Dual Coupling

Let us fix our target flow value F and, for notational convenience, for any $0 \leq \alpha \leq 1$, let us denote by $\boldsymbol{\chi}_\alpha$ the demand vector $\alpha F \boldsymbol{\chi}_{s,t}$, i.e., the demand corresponding to routing α-fraction of the target flow value F of the s-t flow. Also, let us define $\boldsymbol{\chi}$ to be the demand equal to $\boldsymbol{\chi}_1$.

Again, our algorithm will be inherently primal dual in nature. That is, in addition to maintaining a primal solution: a $\boldsymbol{\chi}_\alpha$-flow $\boldsymbol{f}$, for some $0 \leq \alpha \leq 1$, that is feasible in G, it will also maintain a dual solution $\boldsymbol{y} \in \mathbb{R}^n$, which should be viewed as an embedding of all the vertices of the graph G into a line.

Consequently, our goal will be to either to make $\boldsymbol{f}$ be a feasible flow with demands $\boldsymbol{\chi}_\alpha$ and $\alpha = 1$, which corresponds to routing the target s-t flow value F in full, or to make the dual solution $\boldsymbol{y}$ certify that the target demand $\boldsymbol{\chi}_1 = \boldsymbol{\chi}$ cannot be fully routed in G and thus $F > F^*$.

Well-coupled Solutions.: Our primal dual scheme will be enforcing a very specific coupling of the primal solution $\boldsymbol{f}$ and the dual solution $\boldsymbol{y}$. More precisely, let us define for each arc $e = (u, v)$

$$\Delta_e(\boldsymbol{y}) := y_v - y_u, \tag{7}$$

to be the "stretch" of the arc e in the embedding given by $\boldsymbol{y}$. Also, let $G_{\boldsymbol{f}}$ be the residual graph of G with respect to $\boldsymbol{f}$ and let us define, for a given arc e, a potential function

$$\Phi_e(\boldsymbol{f}) := \frac{1}{\widehat{u}_e^+(\boldsymbol{f})} - \frac{1}{\widehat{u}_e^-(\boldsymbol{f})}, \tag{8}$$

where we recall that $\widehat{u}_e^+(\boldsymbol{f}) := u_e^+ - f_e$ (resp. $\widehat{u}_e^-(\boldsymbol{f}) := u_e^- + f_e$) are forward (resp. backward) residual capacities of the arc e. (See preliminaries, i.e., Section II, for details.)

Then, our intention is to maintain the following relation between $\boldsymbol{f}$ and $\boldsymbol{y}$:

$$\Delta_e(\boldsymbol{y}) = \Phi_e(\boldsymbol{f}) \qquad \text{for each arc } e. \tag{9}$$

Intuitively, this condition ensures that the line embedding $\boldsymbol{y}$ stretches each arc e in the direction of the smaller of the residual capacities, and that this stretch is inversely proportional to the value of that capacity. (Note that if G was undirected and thus the initial capacities u_e^+ and u_e^- were equal, the direction of smaller residual capacity is also the direction in which the flow $\boldsymbol{f}$ flows through the arc e.) It is worth pointing out that this coupling condition is directly inspired by (and, in fact, can be directly derived from) a certain variant of centrality condition used by interior-point method based maximum flow algorithms (see [30]).

Even though condition (9) expresses our intended coupling, it will be more convenient to work with a slightly relaxed condition that allows us to have small violations of that ideal coupling. Specifically, we say that a primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$ is γ-*coupled* iff

$$|\Delta_e(\boldsymbol{y}) - \Phi_e(\boldsymbol{f})| \leq \frac{\gamma_e}{\widehat{u}_e(\boldsymbol{f})} \qquad \text{for all arcs } e = (u, v). \tag{10}$$

Here, $\widehat{u}_e(\boldsymbol{f}) = \min\{\widehat{u}_e^+(\boldsymbol{f}), \widehat{u}_e^-(\boldsymbol{f})\}$ is the (symmetrized) residual capacity of the arc e, and $\boldsymbol{\gamma} \in \mathbb{R}^m$ is the *violation vector* that we intend to keep very small. In particular, we say that a primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$ is *well-coupled* iff its violation vector $\boldsymbol{\gamma}$ has its ℓ_2-norm be at most $\frac{1}{100}$, i.e., $\|\boldsymbol{\gamma}\|_2 \leq \frac{1}{100}$.

One of the key consequences of maintaining a well-coupled primal dual pair of solutions $(\boldsymbol{f}, \boldsymbol{y})$ is that it enables us to use $\boldsymbol{y}$ as a dual certificate for inability to route certain demands in G. The following lemma gives us a concrete criterion for doing that. Its proof is deferred to the full version of the paper [27].

Lemma III.1. *Let $(\boldsymbol{f}, \boldsymbol{y})$ be a well-coupled primal dual solution with $\boldsymbol{f}$ being a $\boldsymbol{\chi}_\alpha$-flow, for some $0 \leq \alpha < 1$. If*

$$\boldsymbol{\chi}^T \boldsymbol{y} > \frac{2m}{(1-\alpha)}$$

then the demand $\boldsymbol{\chi}$ cannot be routed in G, i.e., $F > F^*$.

Note that the choice of the constant 2 in the above lemma is fairly arbitrary. In principle, any constant strictly larger than 1 would suffice.

In the light of the above discussion, for a given well-coupled primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$, we should view the value of α as a measure of our primal progress, while the value of the inner product $\boldsymbol{\sigma}^T \boldsymbol{y}$ can be seen as a measure of our dual progress.

Initialization.: The coupling condition (10) ties the primal and dual solutions $\boldsymbol{f}$ and $\boldsymbol{y}$ fairly tightly. In fact, coming up with some primal dual solutions that are well-coupled, which we need to initialize our framework, might be difficult.

Fortunately, finding such a pair of initial well-coupled solutions turns out to be easy, if our input graph G is undirected. In that case, just taking $\boldsymbol{f}$ to be a trivial zero flow and $\boldsymbol{y}$ to be a trivial all-zeros embedding makes condition (10) satisfied (with γ_es being all zero). After all, the residual graph $G_{\boldsymbol{f}}$ with respect to such zero flow $\boldsymbol{f}$ is just the graph G itself and thus $\widehat{u}_e^+(\boldsymbol{f}) = u_e^+ = u_e^- = \widehat{u}_e^-(\boldsymbol{f})$.

Furthermore, even though we are interested in solving directed instances too, every such instance can be reduced to an undirected one. Specifically, we have the following lemma.

Lemma III.2. *Let G be an instance of the maximum s-t flow problem with m arcs and the maximum capacity U, and let F be the corresponding target flow value. In $\widetilde{O}(m)$ time, one can construct an instance G' of undirected maximum s-t flow problem that has $O(m)$ arcs and the maximum capacity U, as well as target flow value F' such that: (a)*

1) *if there exists a feasible s-t flow of value F in G then a feasible s-t flow of value F' exists in G';*
2) *given a feasible s-t flow of value F' in G' one can construct in $\widetilde{O}(m)$ time a feasible s-t flow of value F in G.*

The proof of the above lemma boils down to a known reduction of the directed maximum flow problem to its undirected version – see, e.g., Theorem 3.6.1 in [29] for details. Consequently, from now on we can assume without loss of generality that we always have a well-coupled primal dual pair to initialize our framework.

B. Progress Steps

Once we described our basic framework and how to initialize it, we are ready to put forth its main ingredient: progress steps that enable us to gradually improve the primal dual solutions that we maintain. To this end, let us fix a well-coupled primal dual solutions $(\boldsymbol{f}, \boldsymbol{y})$ with $\boldsymbol{f}$ being a $\boldsymbol{\chi}_\alpha$-flow, for some $0 \leq \alpha < 1$, that is feasible in G. Our goal in this section will be to use $(\boldsymbol{f}, \boldsymbol{y})$ to compute, in nearly-linear time, another pair of well-coupled primal dual solutions $(\boldsymbol{f}^+, \boldsymbol{y}^+)$ that bring us closer to the optimal solutions. The flow $\boldsymbol{f}^+$ we obtain will be a $\boldsymbol{\chi}_{\alpha'}$-flow feasible in G, for $\alpha' > \alpha$. So, the resulting flow update $\boldsymbol{f}^+ - \boldsymbol{f}$ is an augmenting flow that is feasible in our current residual graph $G_{\boldsymbol{f}}$ and pushes $(\alpha' - \alpha)$-fraction of the target s-t flow.

We will compute $(\boldsymbol{f}^+, \boldsymbol{y}^+)$ in two stages. First, in the *augmentation step*, we obtain a pair of solutions $(\widehat{\boldsymbol{f}}, \widehat{\boldsymbol{y}})$, with $\widehat{\boldsymbol{f}}$ being a $\boldsymbol{\chi}_{\alpha'}$-flow, for $\alpha' > \alpha$, that is feasible in G. These solutions make progress toward the optimal solutions but might end up being not well-coupled. Then, in the *fixing step*, we correct $(\widehat{\boldsymbol{f}}, \widehat{\boldsymbol{y}})$ slightly by adding a carefully chosen flow circulation, i.e., a flow with all-zeros demands, to $\widehat{\boldsymbol{f}}$ and an dual update to $\widehat{\boldsymbol{y}}$ so as to make the resulting solutions $(\boldsymbol{f}^+, \boldsymbol{y}^+)$ be well-coupled, as desired.

The key primitive in both these steps will be electrical flow computation. As we will see, the crucial property of electrical flows we will rely on here is their"self-duality". That is, the fact that each electrical flow computation gives us both the flow and the corresponding vertex potentials that are coupled to it via Ohm's law (5). This enables us not only to update our primal and dual solutions with that flow and vertex potentials, respectively, but also, much more crucially, this coupling introduced by Ohm's law will be exactly what will allow us to (approximately) maintain our desired primal dual coupling property (10).

Augmentation Step.: To perform an augmentation step we compute first an electrical $\boldsymbol{\chi}$-flow $\widetilde{\boldsymbol{f}}$ in G with the resistances $\boldsymbol{r}$ defined as

$$r_e := \frac{1}{(\widehat{u}_e^+(\boldsymbol{f}))^2} + \frac{1}{(\widehat{u}_e^-(\boldsymbol{f}))^2}, \tag{11}$$

for each arc e. Note that the resistance r_e is proportional, roughly, to the inverse of the square of the residual capacity $\widehat{u}_e(\boldsymbol{f})$ of that arc. So, in particular, it becomes very large whenever residual capacity of the arc e is small, and vice versa. As we will see shortly, this correspondence will allow us to control the amount of flow that $\widetilde{\boldsymbol{f}}$ sends over each edge and thus ensure that the respective residual capacities are not violated.

Let $\widetilde{\boldsymbol{\phi}}$ be the vertex potentials inducing $\widetilde{\boldsymbol{f}}$ (via the Ohm's law (6)). Then, we obtain the new primal and dual solution $(\widehat{\boldsymbol{f}}, \widehat{\boldsymbol{y}})$ as follows:

$$\begin{aligned} \widehat{f}_e &:= f_e + \delta \widetilde{f}_e \quad \text{for each arc } e \\ \widehat{y}_v &:= y_v + \delta \widetilde{\phi}_v \quad \text{for each vertex } v, \end{aligned} \tag{12}$$

where δ is the desired *step size*. Observe that this update is exactly an augmentation of our current flow $\boldsymbol{f}$ with the (scaled) electrical flow $\delta \widetilde{\boldsymbol{f}}$ and adding to our dual solution the (scaled) vertex potentials $\delta \widetilde{\boldsymbol{\phi}}$. This, in particular, means that the new flow $\widehat{\boldsymbol{f}}$ we obtain here is a $\boldsymbol{\chi}_{\alpha'}$-flow with

$$\alpha' = \alpha + \delta. \tag{13}$$

The step size δ, however, will have to be carefully chosen. On one hand, as we see in (13), the larger it is the more progress we make. On the other hand, though, it has to be small enough so as to keep the flow $\delta \widetilde{\boldsymbol{f}}$ feasible in $G_{\boldsymbol{f}}$ (and thus, by Fact II.1, to make the flow $\widehat{\boldsymbol{f}} + \boldsymbol{f}$ feasible in G).

Note that, a priori, we have no direct control over neither the directions in which the electrical $\boldsymbol{\chi}$-flow $\widetilde{\boldsymbol{f}}$ is flowing thorough each arc nor the amount of that flow. So, in order to establish a grasp on what is the right setting of δ, it is useful to define a *congestion vector* $\boldsymbol{\rho}$ given by

$$\rho_e := \frac{\widetilde{f}_e}{\widehat{u}_e(\boldsymbol{f})}, \tag{14}$$

for each arc e. One can view ρ_e as a normalized measure of how much the electrical flow $\widetilde{\boldsymbol{f}}$ overflows the residual capacity $\widehat{u}_e(\boldsymbol{f})$ of the arc e and in what direction. In other words, the sign of ρ_e encodes the direction of the flow $\widetilde{f}_e$.

It is now not hard to see that to ensure that $\delta \widetilde{\boldsymbol{f}}$ is feasible in $G_{\boldsymbol{f}}$, i.e., that no residual capacity is violated by the update (12), it suffices that $\delta |\rho_e| \leq \frac{1}{4}$, for all arcs e, or, equivalently, that

$$\delta \leq \frac{1}{4 \|\boldsymbol{\rho}\|_\infty}, \tag{15}$$

where $\|\cdot\|_\infty$ is the standard ℓ_∞-norm.

It is also worth pointing out that the congestion vector $\boldsymbol{\rho}$ turns out to capture (up to a small multiplicative factor) the contribution of each arc e to the energy $\mathcal{E}_{\boldsymbol{r}}(\widetilde{\boldsymbol{f}})$ of $\widetilde{\boldsymbol{f}}$. In particular, we have the following simple but important observation.

Lemma III.3. *For any arc e, $\rho_e^2 \leq r_e \widetilde{f}_e^2 \leq 2\rho_e^2$ and $\|\boldsymbol{\rho}\|_2^2 \leq \mathcal{E}_{\boldsymbol{r}}(\widetilde{\boldsymbol{f}}) \leq 2\|\boldsymbol{\rho}\|_2^2$, where $\|\cdot\|_2$ is the standard ℓ_2-norm.*

This link between the energy-minimizing nature of the electrical $\boldsymbol{\sigma}$-flow $\widetilde{\boldsymbol{f}}$ and the ℓ_2-norm of the congestion vector $\boldsymbol{\rho}$ will end up being very important. One reason for that is the fact that ℓ_∞-norm is always bounded by the ℓ_2-norm. Consequently, we can use this connection to control the ℓ_∞-norm of the vector $\boldsymbol{\rho}$ and thus the value of δ needed to satisfy the feasibility condition (15).

It turns out, however, that just ensuring that our augmenting flow is feasible is not enough for our purposes. Specifically, we also need to control the coupling of our primal dual solutions, and the feasibility bound (15) might be not sufficiently strong for this purpose. We thus have to develop analyze the impact of the update (12) on the coupling condition (10) more closely.

To this end, let us first notice the following fact that stems from a standard application of the Taylor's theorem. Its proof is deferred to the full version of the paper [27].

Fact III.4. *For any $u_1, u_2 > 0$ and x such as $|x| \leq \frac{u}{4}$, where $u = \min\{u_1, u_2\}$, we have that*

$$\left(\frac{1}{u_1 - x} - \frac{1}{u_2 + x}\right) = \frac{1}{u_1} - \frac{1}{u_2} + \left(\frac{1}{u_1^2} + \frac{1}{u_2^2}\right) x + x^2 \zeta,$$

where $|\zeta| \leq \frac{5}{u^3}$.

Now, the above approximation bound enables us to get an accurate estimate of how the coupling condition evolves during the augmentation step (12). Specifically, for any arc e, the first order approximation of the change in the primal contribution of the arc e to the coupling condition (10) caused by the update (12) is exactly

$$\Phi_e(\widehat{\boldsymbol{f}}) - \Phi_e(\boldsymbol{f}) \approx \left(\frac{1}{(\widehat{u}_e^+(\boldsymbol{f}))^2} + \frac{1}{(\widehat{u}_e^-(\boldsymbol{f}))^2}\right) \delta \widetilde{f}_e = r_e \delta \widetilde{f}_e,$$

where we also used (11). (In fact, the choice of the resistances $\boldsymbol{r}$ was made exactly to make the above statement true.)

Furthermore, by Ohm's law (5) and the definition of our augmentation step (12), we have that

$$r_e \delta \widetilde{f}_e = \delta \left(\widetilde{\phi}_v - \widetilde{\phi}_u\right) = \Delta_e(\widehat{\boldsymbol{y}}) - \Delta_e(\boldsymbol{y}),$$

which is exactly the change in the dual contribution of the arc $e = (u, v)$ to the coupling condition (10) caused by the augmentation step update (12).

So, up to first order approximation, these two contributions cancel out, leaving the coupling (10) intact. Consequently, any increase in the violation of the coupling condition must come from the second-order terms that we suppressed. The following lemma, whose proof is deferred to the full version of the paper [27], makes this precise.

Lemma III.5. *Let $0 < \delta \leq (4\|\boldsymbol{\rho}\|_\infty)^{-1}$ and the primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$ be $\boldsymbol{\gamma}$-coupled. Then, we have that, for any arc $e = (u, v)$,*

$$\left|\Delta_e(\widehat{\boldsymbol{y}}) - \Phi_e(\widehat{\boldsymbol{f}})\right| \leq \frac{\frac{4}{3}\gamma_e + 7(\delta \rho_e)^2}{\widehat{u}_e(\widehat{\boldsymbol{f}})}.$$

Fixing Step.: Although Lemma III.5 enables us to bound the deterioration of the primal dual coupling during the augmentation step, we cannot prevent this effect altogether. Therefore, we need to introduce a *fixing step* that deals with this problem. More precisely, we develop a procedure that uses a single electrical flow computation to significantly reduce that violation, provided it was not too large to begin with. This is formalized by the following lemma, whose proof is deferred to the full version of the paper [27].

Lemma III.6. *Let $(\boldsymbol{g}, \boldsymbol{z})$ be a ς-coupled primal dual solution, with $\boldsymbol{g}$ being a feasible $\boldsymbol{\chi}_{\alpha'}$-flow and $\|\varsigma\|_2 \leq \frac{1}{50}$. In $\widetilde{O}(m)$ time, we can compute a primal dual solution $(\bar{\boldsymbol{g}}, \bar{\boldsymbol{z}})$ that is well-coupled and in which $\bar{\boldsymbol{g}}$ is still a $\boldsymbol{\chi}_{\alpha'}$-flow.*

Now, after putting Lemmas III.5 and III.6 together, we are finally able to state the condition that δ in the update (12) has to satisfy in order to ensure that the solutions $(\boldsymbol{f}^+, \boldsymbol{y}^+)$ we obtain after performing the augmentation and fixing step is still well-coupled.

Lemma III.7. *$(\boldsymbol{f}^+, \boldsymbol{y}^+)$ is a well-coupled primal dual solution with $\boldsymbol{f}^+$ being a $\boldsymbol{\chi}_{\alpha'}$-flow that is feasible in G whenever*

$$\delta \leq (33\|\boldsymbol{\rho}\|_4)^{-1},$$

The above lemma (whose proof is deferred to the full version of the paper [27]) tells us that the step size δ of our augmentation step (12) should be governed by the ℓ_4-norm of the congestion vector (14). Observe that the ℓ_4-norm of a vector is always upper bounding its ℓ_∞-norm. So, the condition (15) is subsumed by this ℓ_4-norm bound.

C. Analysis of the Algorithm

We want now to analyze the overall running time of our algorithm. Recall that given our target demand $\boldsymbol{\chi}$ that corresponds to sending F units of flow from the source s to the sink t, our overarching goal is to either route this demand fully in G or provide a dual certificate that it is impossible to route $\boldsymbol{\chi}$ in G.

We aim to achieve this goal by maintaining and gradually improving a primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$. In this solution, $\boldsymbol{f}$ is a $\boldsymbol{\chi}_\alpha$-flow (which corresponds to routing an α fraction of the desired demand $\boldsymbol{\chi}$) that is feasible in G and $\boldsymbol{f}$ and $\boldsymbol{y}$ are well-coupled, i.e., tied to each other via condition (10) with the violation vector $\widehat{\gamma}$ having sufficiently small ℓ_2-norm. As described in Section III-B, each iteration runs in nearly-linear time and boils down to employing electrical flow computations to find an augmenting flow in the current residual graph $G_{\boldsymbol{f}}$ (as well as to update the dual solution to maintain well-coupling).

Consequently, all we need to do now is to lower bound the amount of progress that each of these iteration makes. Ideally, we would like to prove that in each iteration in which $\boldsymbol{f}$ already routed α-fraction of the desired flow, i.e., $\boldsymbol{f}$ is a feasible $\boldsymbol{\chi}_\alpha$-flow, the step size δ (see (12)) can be taken to be at least

$$\delta \geq (1-\alpha)\hat{\delta}, \tag{16}$$

for some fixed $\hat{\delta} > 0$. Observe that if such a lower bound was established then, by (13), it would imply that each iteration finds an augmenting flow that routes at least $\hat{\delta}$-fraction of the amount of flow still to be routed. As a result, after executing at most $O(\hat{\delta}^{-1} \log mU)$ iterations, the remaining value of flow to be routed would be at most 1 and thus a simple flow rounding and augmenting path finding would yield the final answer (see, e.g., [30]), making the overall running time be at most $\widetilde{O}\left(\hat{\delta}^{-1} m \log U\right)$.

Unfortunately, a priori, it is difficult to provide any such non-trivial unconditional lower bound on the amount of primal progress we make in each iteration. After all, it could be the case that the target flow cannot be even routed in G. More importantly though, even if the target flow could be routed in G, and thus the residual graph always admitted augmenting flows of sufficiently large value, it is still not clear that our flow augmenting procedure would be able to find them. (It is worth noting that this problem is by no means specific to our algorithm. In fact, in *all* the maximum flow algorithms that rely on the augmenting paths framework ensuring that each iteration makes a sufficient primal progress is a chief bottleneck in the analysis.)

The root of the problem here is that our flow augmenting procedure is based on electrical flows and these are undirected in nature. Consequently, the augmenting flows that it finds have to come from a fairly restricted class: s-t flows that are feasible in a certain "symmetrized" version of the residual graph.

To make it precise, given a residual graph $G_{\boldsymbol{f}}$, let us define its *symmetrization* $\widehat{G}_{\boldsymbol{f}}$ to be an undirected graph in which each arc e has its forward and backward capacity equal to $\widehat{u}_e(\boldsymbol{f})$, i.e., to the minimum of the forward $\widehat{u}_e^+(\boldsymbol{f})$ and backward $\widehat{u}_e^-(\boldsymbol{f})$ residual capacities in $G_{\boldsymbol{f}}$. Observe now that each (electrical) augmenting flow $\delta\boldsymbol{f}$ found in the augmentation step (cf. (12)) is not only feasible in the residual graph $G_{\boldsymbol{f}}$ but also in its symmetrization $\widehat{G}_{\boldsymbol{f}}$ – this is exactly what the condition (15) enforces.

However, not all augmenting s-t flows that are feasible in $G_{\boldsymbol{f}}$ have to be feasible in $\widehat{G}_{\boldsymbol{f}}$ too. In fact, it can happen that a large maximum s-t flow value that the residual graph $G_{\boldsymbol{f}}$ supports mostly vanishes in its symmetrization $\widehat{G}_{\boldsymbol{f}}$, and thus prevents our algorithm from making a sufficient good primal progress. (Again, a difficulty of a exactly the same nature arises in the analysis of the classic flow augmenting algorithms such as [9], [15], [12].)

Preconditioning Arcs.: It turns out, however, that there is a fairly simple way to circumvent the above difficulty and ensure that the kind of direct, "primal-only" analysis we hoped for above can indeed be performed. Namely, we just need to "precondition" our input graph by adding to it a large number of s-t arcs of sufficiently large capacities.

More precisely, we modify our input graph G by adding to it m undirected arcs between the source s and sink t with a forward and backward capacities equal to $2U$ and their orientation being from s to t. We will call these arcs *preconditioning arcs*. Observe that after this modification the number of arcs of our graph as well as its maximum capacity at most doubled, and the maximum s-t flow value changed additively by exactly $2mU$. In particular, the preconditioning arcs constitute exactly half of all the arcs and the amount of s-t flow that they alone can support is at least twice the s-t throughput of the rest of the graph. (Also, as these arcs are undirected they do not interfere with our initialization procedure – cf. Lemma III.2.) Consequently, we can just focus on analyzing the running time of our algorithm on this preconditioned instance and the bounds we establish will immediately translate over to the original instance.[2]

As already mentioned, somewhat surprisingly, once such preconditioning arcs are in place and our primal dual solution $(\boldsymbol{f}, \boldsymbol{y})$ is well-coupled, it is always the case that the symmetrization $\widehat{G}_{\boldsymbol{f}}$ of our current residual graph $G_{\boldsymbol{f}}$ retains a constant fraction of the s-t throughput. Intuitively speaking, well-coupling prevents the "shortcutting" preconditioning arc from getting "clogged" too quickly. Instead, their residual capacity is consumed at the same rate as that of the rest of the graph. Consequently, these arcs alone are always able to provide enough of s-t throughput in the symmetrization $\widehat{G}_{\boldsymbol{f}}$ of the residual graph $G_{\boldsymbol{f}}$. This is made precise in the following lemma.

Lemma III.8. *Let $(\boldsymbol{f}, \boldsymbol{y})$ be a well-coupled primal dual solution in the (preconditioned) graph G and let $\boldsymbol{f}$ be a $\boldsymbol{\chi}_\alpha$-flow, for some $0 \leq \alpha < 1$, that is feasible in G. We have either that: (a)*

1) *there exists a $\boldsymbol{\chi}_{\frac{(1-\alpha)}{10}}$-flow $\boldsymbol{f}'$ that is feasible in the symmetrization $\widehat{G}_{\boldsymbol{f}}$ of the residual graph $G_{\boldsymbol{f}}$;*
2) *or $\boldsymbol{\chi}^T \boldsymbol{y} > \frac{2m}{(1-\alpha)}$ implying that our target demand $\boldsymbol{\chi}$ cannot be routed in G (cf. Lemma III.1).*

Note that if our target demand $\boldsymbol{\chi}$ is exactly the demand $F^* \boldsymbol{\chi}_{s,t}$ of the maximum s-t flow, the second condition cannot ever trigger and thus indeed it is the case that the symmetrization of the (preconditioned) residual graph retains a constant fraction of the original s-t throughput.

Lower Bounding $\hat{\delta}$.: Once we proved that the symmetrization $\widehat{G}_{\boldsymbol{f}}$ of the residual graph $G_{\boldsymbol{f}}$ retains most of its s-t flow throughput (see Lemma III.8), we are finally able to provide an absolute lower bound $\hat{\delta}$ (cf. (16)) on the amount of primal progress each iteration of our algorithm makes. To this end, we upper bound first the energy, or, (almost) equivalently, the ℓ_2-norm of the congestion vector (see Lemma III.3) of the electrical flow that we use in our augmentation step (see (12)). Its proof is deferred to the full version of the paper [27].

Lemma III.9. *Let $(\boldsymbol{f}, \boldsymbol{y})$ be a well-coupled primal dual solution, with $\boldsymbol{f}$ being a $\boldsymbol{\chi}_\alpha$-flow that is feasible in $G_{\boldsymbol{f}}$, for some $0 \leq \alpha < 1$. Let $\widetilde{\boldsymbol{f}}$ be an electrical $\boldsymbol{\chi}$-flow determined by the resistances $\boldsymbol{r}$ given by* (11)*. We have that either: (a)*

1) *$\|\boldsymbol{\rho}\|_2^2 \leq \mathcal{E}_{\boldsymbol{r}}(\widetilde{\boldsymbol{f}}) \leq \frac{C_{\mathcal{E}} m}{(1-\alpha)^2}$, where $\boldsymbol{\rho}$ is the congestion vector defined in* (14)*, and $C_{\mathcal{E}} > 0$ is an explicit constant;*
2) *or, $\boldsymbol{\chi}^T \boldsymbol{y} > \frac{2m}{(1-\alpha)}$, i.e., our target demand $\boldsymbol{\chi}$ cannot be routed in G.*

Now, we should notice that by Lemma III.7 it suffices that we always have that

$$\delta \leq \frac{1}{33\|\boldsymbol{\rho}\|_4} \leq \frac{1}{33\|\boldsymbol{\rho}\|_2} \leq \frac{(1-\alpha)}{33\sqrt{C_{\mathcal{E}} m}}, \tag{17}$$

where we used Lemma III.9 and the fact that $\|\boldsymbol{\rho}\|_4 \leq \|\boldsymbol{\rho}\|_2$. By (16), we see that we can take $\hat{\delta} := (33\sqrt{C_{\mathcal{E}} m})^{-1}$, which gives us the desired $\widetilde{O}\left(m^{\frac{3}{2}} \log U\right)$ time algorithm.

Finally, we want to emphasize again that even though our above analysis was based solely on analyzing our primal progress[3], maintaining the dual solution and the primal dual coupling (10) was absolutely critical for its success.

The description and analysis of the improved, $\widetilde{O}\left(m^{\frac{10}{7}} U^{\frac{1}{7}}\right)$-time algorithm is deferred to the full version of the paper [27].

Acknowledgments. We are grateful to Michael Cohen, Slobodan Mitrović, Dimitris Tsipras, and Adrian Vladu for a number of helpful discussions on this topic.

References

[1] 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.

[2] J. R. Bunch and J. E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

[3] P. Christiano, J. Kelner, A. Mądry, D. Spielman, and S.-H. Teng. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *STOC'11*, pages 273–281, 2011.

[4] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving sdd linear systems in nearly m $\log^{1/2} n$ time. In *STOC'14*, pages 343–352, 2014.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[6] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

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

[2]Note that the preconditioning arcs have to be fully saturated in any maximum s-t flow. So, simply dropping these arcs and the flow on them will yield the maximum s-t flow in the original graph.

[3]In fact, one could perform it even without resorting explicitly to the dual infeasibility certificates $\boldsymbol{\chi}^T \boldsymbol{y}$.

[8] J. Egerváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38:16–28, 1931.

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

[10] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[11] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows and matchings. *Mathematical Programming*, 100(3):537–568, 2004.

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

[13] N. J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM Journal on Computing*, 39(2):679–702, 2009.

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

[15] A. V. Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh. *Matematicheskie Voprosy Upravleniya Proizvodstvom*, 5:81–94, 1973.

[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. In *SODA'14*, pages 217–226, 2014.

[17] 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*, pages 911–920, 2013.

[18] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17(3):447–474, 1994.

[19] D. König. Graphok és matrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[20] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD systems. In *FOCS'10*, pages 235–244, 2010.

[21] I. Koutis, G. L. Miller, and R. Peng. A nearly $m \log n$-time solver for SDD linear systems. In *FOCS'11*, pages 590–598, 2011.

[22] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman. Sparsified cholesky and multigrid solvers for connection Laplacians. In *STOC'16*, 2016.

[23] R. Kyng and S. Sachdeva. Approximate Gaussian elimination for Laplacians: Fast, sparse, and simple. In *FOCS'16*, 2016.

[24] Y. T. Lee, S. Rao, and N. Srivastava. A new approach to computing maximum flows using electrical flows. In *STOC'13*, pages 755–764, 2013.

[25] Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flows. In *FOCS'14*, pages 424–433, 2014.

[26] L. Lovász. On determinants, matchings and random algorithms. *Fundamentals of Computation Theory*, 565–574, 1979.

[27] A. Mądry. Computing Maximum Flow with Augmenting Electrical Flows. Available at http://arxiv.org/abs/1608.06016.

[28] A. Mądry. Fast approximation algorithms for cut-based problems in undirected graphs. In *FOCS'10*, pages 245–254, 2010.

[29] A. Mądry. From Graphs to Matrices, and Back: New Techniques for Graph Algorithms. PhD thesis, MIT, 2011.

[30] A. Mądry. Navigating central path with electrical flows: from flows to matchings, and back. In *FOCS'13*, pages 253–262, 2013.

[31] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algoithm for finding maximum matching in general graphs. In *FOCS'80*, pages 17–27, 1980.

[32] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *FOCS'04*, pages 248–255, 2004.

[33] J. B. Orlin. Max flows in O(nm) time, or better. In *STOC'13*, pages 765–774, 2013.

[34] R. Peng. Approximate undirected maximum flows in $O(mpolylog(n))$ time. In *SODA'16*, pages 1862–1867, 2016.

[35] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, Dec. 1989.

[36] J. Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cuts. In *FOCS'09*, pages 363–372, 2009.

[37] J. Sherman. Nearly maximum flows in nearly linear time. In *FOCS'13*, pages 263–269, 2013.

[38] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC'04*, pages 81–90, 2004.

[39] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22:107–111, 1947.

[40] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *STOC'12*, pages 887–898, 2012.

[41] V. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph matching algorithms. *Combinatorica*, 14 (1):71–109, 1994.