Negative-Weight Shortest Paths and Unit Capacity Minimum Cost Flow in $\tilde{O}\left(m^{10/7}\log W\right)$ Time[*]
(Extended Abstract)

Michael B. Cohen [†] MIT
Aleksander Mądry [‡] MIT
Piotr Sankowski [§] University of Warsaw
Adrian Vladu [¶] MIT

Abstract

In this paper, we study a set of combinatorial optimization problems on weighted graphs: the shortest path problem with negative weights, the weighted perfect bipartite matching problem, the unit-capacity minimum-cost maximum flow problem, and the weighted perfect bipartite b-matching problem under the assumption that $\|b\|_1 = O(m)$. We show that each of these four problems can be solved in $\tilde{O}(m^{10/7}\log W)$ time, where W is the absolute maximum weight of an edge in the graph, providing the first polynomial improvement in their sparse-graph time complexity in over 25 years.

At a high level, our algorithms build on the interior-point method-based framework developed by Mądry (FOCS 2013) for solving unit-capacity maximum flow problem. We develop a refined way to analyze this framework, as well as provide new variants of the underlying preconditioning and perturbation techniques. Consequently, we are able to extend the whole interior-point method-based approach to make it applicable in the weighted graph regime.

1 Introduction.

In 2013, Mądry [24] put forth an algorithm for the maximum flow and maximum-cardinality bipartite matching problems that improved over a long standing $O(n^{3/2})$ running time barrier for sparse graphs. Specifically, he presented an $\tilde{O}(m^{10/7})$ time algorithm for computing maximum flow in a unit capacity network – which implies an $\tilde{O}(m^{10/7})$ time algorithm for the maximum-cardinality bipartite matching problem as well. The core of his approach is a new path-following interior-point type algorithm for solving a certain "bipartite b-matching" problem. At a high level, this algorithm uses electrical flow computations to improve the maintained primal dual solution and move it along the so-called central path, i.e., a set of primal dual solutions in which every edge contributes approximately the same amount to the duality gap. As Mądry has shown, one can use this framework to establish an $O(m^{3/7})$ bound on the number of such electrical flow computations needed to compute a (near-) optimal solution to the considered problem and thus to improve upon the generic worst-case $O(\sqrt{m})$ bound that all the previous interior-point-method-based algorithms provided. The key ingredient needed to obtaining this improved bound was a technique for perturbing and preconditioning the intermediate solutions that emerge during the computations. Unfortunately, the technique [24] used in was inherently unable to cope with large capacity or *any* weights on edges. In fact, it did not provide any meaningful result for the unit-capacity minimum cost maximum flow problem even when all the edge weights were equal to 1. Consequently, it remains an open question whether a similar running time improvement can be achieved for either: (a) non-unit capacity networks; or (b) weighted variants of the unit-capacity graph problems.

1.1 Our Contribution. In this paper, we answer the second question above affirmatively by providing an $\tilde{O}(m^{10/7}\log W)$ time algorithm for the minimum cost unit-capacity maximum flow problem. In addition to the improvement for this fundamental

[*]A full version of this paper is available as [4].
[†]Supported by NSF grant CCF-1111109. Email: `micohen@mit.edu`.
[‡]Supported by NSF grant CCF-1553428, and Sloan Research Fellowship. Email: `madry@mit.edu`.
[§]Supported by NCN grant NN206 567940. Email: `sank@mimuw.edu.pl`.
[¶]Supported by a Teaching Assistantship provided by the MIT Department of Mathematics. Email: `avladu@mit.edu`.

graph problem, this result also improves several other standard problems as immediate corollaries. Namely, by well-known reductions, it implies $\tilde{O}(m^{10/7} \log W)$ time algorithms for the minimum-weight bipartite perfect matching problem, the minimum-weight bipartite b-matching problem, and, the shortest path problem for graphs with negative weights. This constitutes the first polynomial improvement of sparse graph time complexity for each one of these problems in more than 25 years.

To obtain these results we simplify and extend the framework from [24] by developing new preconditioning and perturbation techniques. These techniques provide us with much better control of the intermediate solutions that we precondition/perturb. In particular, in stark contrast to [24], the preconditioning and perturbation steps do not lead to any changes in edge costs. Also, the resulting changes in vertex demands are very minimal. Thus, there is no more need for repeated fixing of these vertex demand changes throughout the execution of the algorithm – a single demand correction step is performed only at the very end. Finally, our analysis of the resulting algorithm is much more streamlined and principled compared to the analysis performed in [24], providing us with much tighter grip of the trade-offs underlying the whole framework.

1.2 Previous Work.

The minimum-cost flow, min-weight bipartite perfect matching as well as the shortest path with negative weights problems are core combinatorial optimization tasks that now have been studied for over 85 years, starting with the work of Egerváry [9] from 1931. Due to immense number of works on these topics we will not review them. Instead we will concentrate only on the ones that are relevant to the sparse graph case, as that is the regime where our results are of main importance.

Shortest Paths with Negative Weights. A list of the complexity results on single source shortest paths with negative weights is included in Table 1. Observe that the sparse case can either be solved in $O(mn)$ time [1, 25] or $\tilde{O}(\sqrt{n}m \log W)$ time [12, 14]. The only progress that we had since these papers was the reduction of the problem to fast matrix multiplication [27, 34] that is relevant only for dense graphs with small integral weights.

	Complexity	Author
	$O(n^4)$	Shimbel (1955) [30]
	$O(Wn^2m)$	Ford (1956) [16]
*	$O(nm)$	Bellman (1958) [1], Moore (1959) [25]
	$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983) [11]
	$O(\sqrt{n}m \log(nW))$	Gabow & Tarjan (1989) [12]
*	$O(\sqrt{n}m \log(W))$	Goldberg (1993) [14]
*	$\tilde{O}(Wn^{\omega})$	Sankowski (2005) [27], Yuster & Zwick (2005) [34]
*	$\tilde{O}(m^{10/7} \log W)$	this paper

Table 1: The complexity results for the SSSP problem with negative weights (* indicates asymptotically the best bound for some range of parameters).

	Complexity	Author
	$O(Wn^2m)$	Egerváry (1931) [9]
	$O(n^4)$	Khun (1955) [20], Munkers (1957) [26]
	$O(n^2m)$	Iri (1960) [15]
	$O(n^3)$	Dinic & Kronrod (1969) [6]
*	$O(nm + n^2 \log n)$	Edmonds & Karp (1970) [8]
	$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983) [11]
*	$O(\sqrt{n}m \log(nW))$	Gabow & Tarjan (1989) [12]
*	$O(\sqrt{n}m \log W)$	Duan and Su (2012) [7]
*	$O(Wn^{\omega})$	Sankowski (2006) [28]
*	$\tilde{O}(m^{10/7} \log W)$	this paper

Table 2: The complexity results for the minimum weight bipartite perfect matching problem (* indicates asymptotically the best bound for some range of parameters).

Min-cost Perfect Bipartite Matching. The complexity survey of for the minimum weight bipartite perfect matching problem is given in Table 2. Here, the situation is very similar to the case of shortest paths. We have two results that are relevant for the sparse case considered here: $O(nm + n^2 \log n)$ time [8] and $O(\sqrt{n}m \log(nW))$ time [12]. Again the only polynomial improvement that was achieved during the last 25 years is relevant to the dense case only [28].

Minimum-cost Unit-capacity Maximum Flow Problem. Due to the vast number of results for this topic we restricted ourselves to present in Table 3 only algorithms for the unit-capacity case that yielded significant improvement. We note, however, that handling general capacities for this problem is a major challenge and our restriction to unit-capacity networks oversimplifies the history of this problem. Nevertheless, for there sparse case there are two relevant complexities: a $\tilde{O}(\sqrt{n}m)$ time bound of [23] that, for the sparse graph case, matches the previously best known bound $O(m^{3/2}\log(nW))$ due to [12]. We note that there was also a limited progress [10] on fast matrix multiplication based algorithms for the small vertex capacity variant of this problem.

Complexity	Author
$O(m(m+n\log n)$	Edmonds & Karp (1972) [8]
$O(n^{5/3}m^{2/3}\log(nW))$	Goldberg & Tarjan (1987) [13]
$O(\sqrt{m}m\log(nW))$ $O(n^{2/3}m\log(nW))$	Gabow & Tarjan (1989) [12]
$\tilde{O}(m^{3/2})$	Daitch & Spielman (2008) [5]
$\tilde{O}(\sqrt{n}m)$	Lee & Sidford (2014) [23]
$\tilde{O}(m^{10/7}\log W)$	this paper

Table 3: The summary of the results for the min-cost unit-capacity max-flow problem. For simplicity we only list exact algorithms that yielded polynomial improvement or new strongly polynomial bounds. For the full list of complexities we refer to Chapter 12 in [29].

Complexity	Author
$O(m(m+n\log n)$	Edmonds & Karp (1972) [8]
$O(m^{7/4}\log W)$	Gabow (1985) [11]
$O(m^{3/2}\log(nW))$	Gabow & Tarjan (1989) [12]
$\tilde{O}(m^{10/7}\log W)$	this paper

Table 4: The summary of the results for the min-cost perfect bipartite b-matching problem under assumption that $b(V) = O(m)$. For simplicity we only list exact algorithms that yielded polynomial improvement or new strongly polynomial bounds. For the full list of complexities we refer to Chapter 21 in [29].

Minimum-cost Perfect Bipartite b-Matching. Minimum-cost perfect bipartite b-matching problem bears many similarities to the minimum-cost maximum flow problem, e.g., see our reduction of min-cost flow to b-matchings in Section 3. Hence some of the complexities in Table 4 are the same as in Table 3. However, not all results seem to carry over as exemplified in the tables. b-matchings seems slightly harder than max-flow as one needs to route exact amount of flow between many sources and sinks. The results relevant for sparse case are: weakly polynomial $O(m^{3/2}\log(nW))$ time algorithm [12] or strongly polynomial $O(m(m+n\log n))$ time algorithm [8].

1.3 The Outline of Our Algorithm. As mentioned above, our focus is on development of a faster, $\tilde{O}(m^{10/7}\log W)$-time algorithm for the minimum-cost unit-capacity maximum flow problem, since well-known reductions immediately yield $\tilde{O}(m^{10/7}\log W)$-time algorithms for the remaining problems (see also Section 8). In broad outline, our approach to solving that flow problem follows the framework introduced by Mądry [24] and comprises three major components.

First, in Section 3, we reduce our input minimum-cost flow instance to an instance of the bipartite minimum-cost b-matching problem. The latter instance will have a special structure. In particular, it can be viewed as a minimum-cost flow problem instance that has general flow demands but no capacities.

Then, in Section 4, we put forth a basic interior-point method framework that builds on the framework of Mądry [24] for solving this kind of uncapacitated minimum-cost flow instances. This basic framework alone will not be capable of breaking the $O(\sqrt{m})$ iteration bound. Therefore, in Sections 5 and 6, we develop a careful perturbation and preconditioning technique to help us control the convergence behavior of our interior-point method framework. An important feature of this technique is that, in contrast to the technique used by [24], our perturbations do no alter the arc costs. Thus they are suitable for dealing with weighted problems. We then prove that the resulting algorithms indeed obtains the improved running time bound of $\tilde{O}(m^{10/7}\log W)$ but the (nearly-) optimal flow solution it outputs might have some of the flow demands changed.

Finally, in Section 7, we address this problem by developing a fast procedure that relies on classic combinatorial techniques and recovers from this perturbed (near-)optimal solution an optimal solution to our original minimum-cost flow instance.

2 Preliminaries.

In this section, we introduce some basic notation and definitions that we will need later.

2.1 Minimum Cost σ-Flow.

We denote by $G = (V, E, c)$ a directed graph with vertex set V, arc set E and cost function c. We denote by $m = |E|$ the number of arcs in G, and by $n = |V|$ its number of vertices. An arc e of G connects an ordered pair (w, v), where w is called *tail* and v is called *head*. We will be mostly working with *σ-flows* in G, where $\sigma \in \mathbb{R}^n$, satisfying $\sum_v \sigma_v = 0$, is the *demand vector*. A σ-flow in G is defined to be a vector $f \in \mathbb{R}^m$ (assigning values to arcs of G) that satisfies the following *flow conservation constraint*s:

$$\text{(2.1)} \qquad \sum_{e\in E^+(v)} f_e - \sum_{e\in E^-(v)} f_e = \sigma_v, \quad \text{for each } v \in V.$$

We denote by $E^+(v)$ (resp. $E^-(v)$) the set of arcs of G that are leaving (resp. entering) vertex v. The above constraints enforce that, for every $v \in V$, the total out-flow from v minus the total in-flow (i.e. the net flow) out of v is equal to σ_v.

Furthermore, we say that a σ-flow f is *feasible* in G iff it satisfies *non-negativity and capacity constraints* (in this paper we are only concerned with unit capacities):

$$0 \le f_e \le 1, \quad \text{for each arc } e \in E.$$

For our interior point method the basic object is the *minimum cost σ-flow problem* consists of finding feasible σ-flow f that minimizes *the cost of the flow* $c(f) = \sum_e c_e f_e$.

2.2 Electrical Flows and Potentials.

A notion that will play a fundamental role in this paper is the notion of electrical flows. Here, we just briefly review some of the key properties that we will need later. For an in-depth treatment we refer the reader to [2].

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

$$\text{(2.2)} \qquad \mathcal{E}_r(f) := \sum_e r_e f_e^2 = f^\top R f,$$

where $R = \mathrm{diag}(r)$ is an $m \times m$ diagonal matrix with $R_{e,e} = r_e$, for each edge e. In order to simplify notation, we drop the subscript or the parameter whenever it is clear from the context.

For a given undirected graph G, a demand vector σ, and a vector of resistances r, we define an *electrical σ-flow* in G (that is *determined* by the resistances r) to be the σ-flow that minimizes the energy $\mathcal{E}_r(f)$ among all σ-flows in G. As energy is a strictly convex function, one can easily see that such a flow is unique. Also, we emphasize that we do *not* require here that this flow is feasible with respect to the (unit) capacities of G (cf. (2.1)). Furthermore, whenever we consider electrical flows in the context of a directed graph G, we will mean an electrical flow – as defined above – in the (undirected) projection $\bar{G}$ of G.

One of very useful properties of electrical flows is that it can be characterized in terms of vertex potentials inducing it. Namely, one can show that a σ-flow f in G is an electrical σ-flow determined by resistances r iff there exist *vertex potentials* ϕ_v (that we collect into a vector $\phi \in \mathbb{R}^n$) such that, for any edge $e = (u, v)$ in G that is oriented from u to v,

$$\text{(2.3)} \qquad f_e = \frac{\phi_v - \phi_u}{r_e}.$$

In other words, a σ-flow f is an electrical σ-flow iff it is *induced* via (2.3) by some vertex potentials ϕ. (Note that orientation of edges matters in this definition.)

Using vertex potentials, we are able to express the energy $\mathcal{E}_r(f)$ (see (2.2)) of an electrical σ-flow f in terms of the potentials ϕ inducing it as

$$\text{(2.4)} \qquad \mathcal{E}_r(f) = \sum_{e=(u,v)} \frac{(\phi_v - \phi_u)^2}{r_e}.$$

2.3 Laplacian Solvers.

A very important algorithmic property of electrical flows is that one can compute very good approximations of them in nearly-linear time. Below, we briefly describe the tools enabling that.

To this end, let us recall that electrical σ-flow is the (unique) σ-flow induced by vertex potentials via (2.3). So, finding such a flow boils down to computing the corresponding vertex potentials ϕ. It turns out that computing these potentials can be cast as a task of solving certain type of linear system called *Laplacian* systems. To see that, let us define the *edge-vertex incidence matrix* B being an $n \times m$ matrix with rows indexed by vertices and columns indexed by edges such that

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

Now, we can compactly express the flow conservation constraints (2.1) of a σ-flow f (that we view as a vector in $\mathbb{R}^m$) as

$$Bf = \sigma.$$

On the other hand, if ϕ are some vertex potentials, the corresponding flow f induced by ϕ via (2.3) (with respect to resistances r) can be written as

$$f = R^{-1}B^\top \phi,$$

where again R is a diagonal $m \times m$ matrix with $R_{e,e} := r_e$, for each edge e.

Putting the two above equations together, we get that the vertex potentials ϕ that induce the electrical σ-flow determined by resistances r are given by a solution to the following linear system

$$BR^{-1}B^\top \phi = L\phi = \sigma, \tag{2.5}$$

where $L := BR^{-1}B^T$ is the (weighted) *Laplacian* L of G (with respect to the resistances r). One can easily check that L is a symmetric $n \times n$ matrix indexed by vertices of G with entries given by

$$L_{u,v} = \begin{cases} \sum_{e\in E(v)} 1/r_e & \text{if } u = v, \\ -1/r_e & \text{if } e = (u,v) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

One can see that the Laplacian L is not invertible, but – as long as, the underlying graph is connected – its null-space is one-dimensional and spanned by the all-ones vector. As we require our demand vectors σ to have its entries sum up to zero (otherwise, no σ-flow can exist), this means that they are always orthogonal to that null-space. Therefore, the linear system (2.5) has always a solution ϕ and one of these solutions[1] is given by

$$\phi = L^+\sigma,$$

where L^+ is the Moore-Penrose pseudo-inverse of L.

Now, from the algorithmic point of view, the crucial property of the Laplacian L is that it is symmetric and *diagonally dominant*, i.e., for any $v \in V$, $\sum_{u\neq v} |L_{u,v}| \leq L_{v,v}$. This enables us to use fast approximate solvers for symmetric and diagonally dominant linear systems to compute an approximate electrical σ-flow. Namely, there is a long line of work [32, 18, 19, 17, 3, 21, 22] that builds on an earlier work of Vaidya [33] and Spielman and Teng [31], that designed an SDD linear system solver that implies the following theorem.

Theorem 1. *For any $\epsilon > 0$, any graph G with n vertices and m edges, any demand vector σ, and any resistances r, one can compute in $\tilde{O}(m \log m \log \epsilon^{-1})$ time vertex potentials $\tilde{\phi}$ such that $\|\tilde{\phi} - \phi^*\|_L \leq \epsilon \|\phi^*\|_L$, where L is the Laplacian of G, ϕ^* are potentials inducing the electrical σ-flow determined by resistances r, and $\|\phi\|_L := \sqrt{\phi^\top L\phi}$.*

To understand the type of approximation offered by the above theorem, observe that $\|\phi\|_L^2 = \phi^\top L\phi$ is just the energy of the flow induced by vertex potentials ϕ. Therefore, $\|\tilde{\phi} - \phi^*\|_L$ is the energy of the electrical flow $\bar{f}$ that "corrects" the vertex demands of the electrical $\tilde{\sigma}$-flow induced by potentials $\tilde{\phi}$, to the ones that are dictated by σ. So, in other words, the above theorem tells us that we can quickly find an electrical $\tilde{\sigma}$-flow $\tilde{f}$ in G such that $\tilde{\sigma}$ is a slightly perturbed version of σ and $\tilde{f}$ can be corrected to the electrical σ-flow f^* that we are seeking, by adding to it some electrical flow $\bar{f}$ whose energy is at most ϵ fraction of the energy of the flow f^*. (Note that electrical flows are linear, so we indeed have that $f^* = \tilde{f} + \bar{f}$.) As we will see, this kind of approximation is completely sufficient for our purposes.

2.4 Bipartite b-Matchings.

For a given weighted bipartite graph $G = (V, E)$ with $V = P \cup Q$ – where P and Q are the two sets of bipartition – as well as, a *demand vector* $b \in \mathbb{R}_+^V$, a *perfect b-matching* is a vector $x \in \mathbb{R}_+^V$ such that $\sum_{e\in E(v)} x_e = b_v$ for all $v \in V$. A perfect b-matching is a generalization of perfect matching; in the particular case where all b's equal 1, integer b-matchings (or 1-matchings) are exactly perfect matchings. x is called *b-matching* if the equality in the above equation is satisfied with inequality, i.e., we have just $\sum_{e\in E(v)} x_e \leq b_v$ for all $v \in V$. For perfect b-matchings we usually require that $b(P) = b(Q)$ as otherwise they trivially do not exist.

A *weighted perfect bipartite b-matching problem* is a problem in which, given a weighted bipartite graph $G = (V, E, w)$ our goal is to either return a perfect b-matching in G that has minimum weight, or conclude that there is no perfect b-matching in G. We say that a b-matching x in a graph G is *h-near* if the size of x is within an additive factor of h of the size of a perfect b-

[1]Note that the linear system (2.5) will have many solutions, but any two of them are equivalent up to a translation. So, as the formula (2.3) is translation-invariant, each of these solutions will yield the same unique electrical σ-flow.

matching. The dual problem to the weighted perfect bipartite b-matching is a *b-vertex packing problem* where we want to find a vector $y \in \mathbb{R}^V$ satisfying the following LP

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v b_v, \\ & y_u + y_v \le w_{uv} \quad \forall uv \in E. \end{aligned}$$

Also, we define the *size* of a b-matching x to be $\|x\|_1/2$. A *h-near b-matching* is a b-matching with size at least $\|x\|_1/2 - h$. Finally, we observe that a bipartite$b$-matching instance can be reinterpreted as a unit capacitated σ-flow instance just by setting

$$\sigma_v = \begin{cases} b_v & \text{if } v \in P, \\ -b_v & \text{otherwise.} \end{cases}$$

and leaving costs unchanged. We require this alternative view, since our interior-point algorithm will work with σ-flows, but the rounding that will need to be performed at the end is done in the b-matching view.

3 Reducing Minimum-Cost Flow to Bipartite b-Matching.

In this section we show how to convert an instance of unit capacity min-cost flow into an instance of min-cost b-matching. This is done via a simple combinatorial reduction similar to the one in [24]. As noted in the preliminaries, bipartite b-matchings can be reinterpreted as σ-flows. This alternative view will be useful for us, as our interior point method will work with σ-flows, whereas it is easier to repair a near-optimal solution in the b-matching view.

We first show a straightforward reduction from min-cost flow to b-matching. One desirable feature of this reduction is that the obtained b-matching instance does not contain upper capacities on arcs, since these are going to be implicitly encoded by properly setting the demands. The part of lemma that refers to half-integral flow and half-integral matching will be essential for the initialization step.

Lemma 2. *Given a directed graph $G = (V, E, c)$ and a demand vector σ, one can construct in linear time a bipartite graph $G' = (V', E', c')$, $V' = P \cup Q$ along with a demand vector b' such that given a minimum cost b'-matching in G', one can reconstruct a flow f that routes demand σ in G with minimum cost. Moreover, if flow $f = \frac{1}{2} \cdot \vec{1}$ is feasible in G then $x = \frac{1}{2} \cdot \vec{1}$ is a feasible fractional b-matching in G'.*

Proof. Let $P = V$ and $Q = E$. For each arc $(u, v) \in E$, let e_{uv} be the corresponding "edge" vertex in Q. Create arcs (u, e_{uv}) with cost c_{uv}, and (v, e_{uv}) with cost 0. For each $e_{uv} \in Q$ set demand $b'(e_{uv}) = 1$. For each $v \in P$, set demand $b'(v) = \sigma(v) + \deg_{in}^G(v)$. Let us now argue that the solution to b'-matching encodes a valid flow in G. Observe that in the b'-matching instance each vertex e_{uv} can be in two states: it is either matched to u or to v. When e_{uv} is matched to u we "read" that there is one unit of flow on arc (u, v), whereas when e_{uv} is matched to v we "read" that there is no flow on arc (u, v). With this interpretation flow conservation constraints (2.1) are satisfied.

Now assume that the flow $f = \frac{1}{2} \cdot \vec{1}$ is feasible in G and consider the b-matching $x = \frac{1}{2} \cdot \vec{1}$. We observe that feasibility of f implies that x is feasible for each vertex in P, as each such vertex has the same number of incident edges and the same demand as the corresponding vertex in G. On the other hand, by construction vertices in Q have demand -1 and two incoming edges, what settles feasibility of x for them.

The effect of this reduction on a single arc is presented in the figure below. Note that vertices in P correspond to *vertices* from the original graph, whereas vertices in Q correspond to *arcs* form the original graph. Essentially, every arc (u, v) in G adds a demand pair of $(1, 1)$ on the pair of vertices (v, e_{uv}) in the b-matching instance. The amount of flow routed on (v, e_{uv}) corresponds to the residual capacity of the arc (u, v) in the original graph.

While this reduction rephrases the problem into a form that is amenable to our interior-point framework, the remaining issue is that we have to be able to start the algorithm with a feasible solution where all the flows on arcs are similar in magnitude (this enables us to enforce the centrality property defined in 4.1). Ideally, we should be able to obtain a feasible instance simply by placing half a unit of flow on every arc of the b-matching instance. While doing so clearly satisfies the demands of vertices in Q, demand violations might occur on vertices in P.

We can easily fix this problem by adding one extra vertex in the original graph, along with a set of arcs

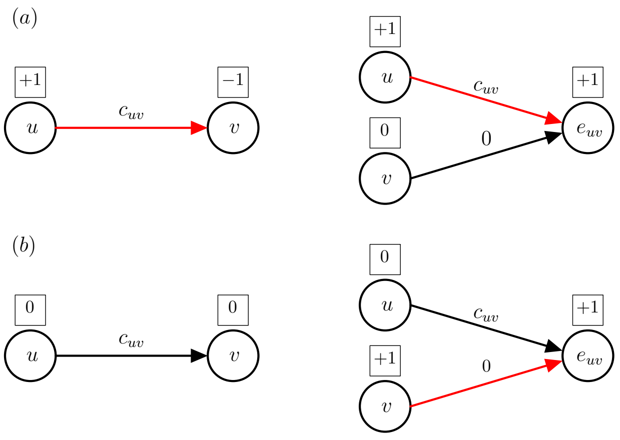

Figure 1: The left side contains two σ-flow instances, the right side contains their corresponding reductions to b-matching. In (a) we can see that the unit flow from u to v gets routed from u to e_{uv} in the reduction. In (b), since there is no flow from u to v, the new demand added by the reduction gets routed directly from v to e_{uv}.

with costs chosen sufficiently high, that the optimal solution will never consider them. The goal is to add the extra arcs such that flowing $\frac{1}{2}$ on every arc satisfies the demand. This instance can then be converted to a b-matching instance with the same property. The reduction is described by the following lemma:

LEMMA 3. *Given a directed graph $G = (V, E, c)$ and an integral demand vector σ, one can construct in linear time a graph $G' = (V', E', c')$ along with an integral demand vector σ' such that the demand is satisfied by placing $\frac{1}{2}$ units of flow on every arc. Furthermore, given a solution to the min-cost σ'-flow satisfying demand σ' in G', one can reconstruct in linear time a solution f that routes demand σ in G with minimum cost.*

Proof. Create one extra vertex v_{aux} with demand 0. Then, for all $v \in V$, let $t(v) = \sigma(v) + \frac{1}{2} \cdot \deg_{in}^{G}(v)/2 - \frac{1}{2} \cdot \deg_{out}^{G}(v)$ be the residual demand corresponding to the flow that has value $\frac{1}{2}$ everywhere.

Fix this residual by creating $|2t(v)|$ parallel arcs (v_{aux}, v) with costs $\|c\|_1$, for each vertex with $t(v) < 0$, respectively $|2t(v)|$ parallel arcs (v, v_{aux}) with costs $\|c\|_1$ for each vertex v with $t(v) > 0$. This enforces our condition to be satisfied for all vertices in V.

Also note that v_{aux} has an equal number of arcs entering and leaving it, since the sum of residuals at vertices in V equals the sum of degree imbalances, plus the sum of demands, both of which are equal to 0. More precisely, $\deg_{in}(v_{\text{aux}}) = \sum_{v:t(v)>0} 2t(v)$, and $\deg_{out}(v_{\text{aux}}) = \sum_{v:t(v)<0} -2t(v)$; since $0 = \sum_v t(v) = \sum_{v:t(v)>0} t(v) - \sum_{v:t(v)<0} (-t(v)) = \deg_{in}^{G'}(v_{\text{aux}})/2 - \deg_{out}^{G'}(v_{\text{aux}})/2$, the vertex v_{aux} is balanced; hence $t(v_{\text{aux}}) = 0$, and the residual demand is 0 at all vertices in the graph.

Finally, observe that a flow in G satisfying σ is a valid flow in G' for σ' that does not use any edge incident to v_{aux}. This flow has cost smaller than $\|c\|_1$. Hence, a min-cost σ'-flow in G', if it has cost smaller than $\|c\|_1$, gives a min-cost σ-flow in G.

A pictorial description of the reduction is presented in Figure 2.

We can use these two reductions together, first making the flow $f = \frac{1}{2} \cdot \vec{1}$ feasible, then converting the instance to a b-matching instance that can be directly changed back to σ-flow instance. As a result we obtain an instance where constructing a feasible starting solution that is required for our interior point method is straightforward.

4 Our Interior-Point Method Framework.

In this section we describe our interior point method framework for solving our instance of the uncapacitated minimum-cost σ-flow problem that results from casting the bipartite $b-$matching problem instance we produced in Section 3 as an instance of the minimum-cost $\sigma-$flow problem. Our basic setup is largely following the framework used by Mądry [24]. The crucial differences emerge only later on, in Sections 5 and 6, when we refine it to obtain our desired running time improvements.

4.1 Primal–Dual Solutions and Centrality.

Our framework will be inherently primal–dual. That is, we will always maintain a feasible primal dual solution (f, y). Here, the primal solution f is simply a σ-flow, i.e., a flow with demands σ and its feasibility condition is that $f_e \geq 0$ for each arc e, i.e., the flow never flows against the orientation of the arc. The dual solution y, on the other hand, corresponds to embedding of the vertices of our graph into the line, i.e., y assigns a real value y_v to each vertex v. The dual feasibility condition is that, for each arc $e = (u, v)$, we have that its *slack* $s_e := c_e + y_u - y_v$ is

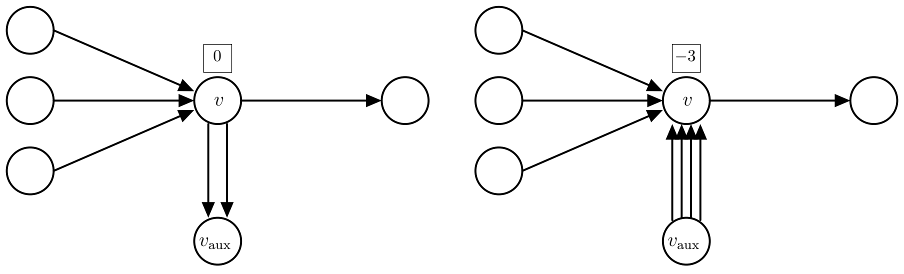

Figure 2: Two examples of balancing a vertex using extra arcs connected to v_{aux}. In the first case, adding two arcs to v_{aux} makes the residual at v equal to 0, when routing 1/2 on every arc, since the in-degree of v is equal to its out-degree, and there is no demand on v. In the second case, we add four arcs from v_{aux} to v in order for the net flow at v (when routing 1/2 on every arc) to match the demand -3.

non-negative, i.e., $s_e \geq 0$, for each arc e. Intuitively, this means that no arc e is stretched in the embedding defined by y more than its actual cost c_e. (Note that here we measure stretch only in the direction of the arc orientation.)

Observe that the dual solution y is define uniquely (up to a translation) by the slack vector s and the arc costs c. So, we will sometime find it more convenient to represent our dual solution in terms of the slack vector s instead of the embedding vector y. From this perspective, the dual feasibility condition is simply non-negativity of all the slacks s. (In fact, we will use y and s representation interchangeably, depending on which one of them is more convenient in given context.)

Duality Gap. A very convenient feature of working with primal dual solutions is that they provide a very natural way of measuring optimality: the duality gap, i.e., the difference between the upper bound on the cost of the optimal solution provided by current primal feasible solution f and the lower bound on the optimal cost provided by our current dual solution s. It turns out that the duality gap of a primal dual solution (f, s) is exactly

$$f^\top s = \sum_e f_e s_e = \sum_e \mu_e,$$

where $\mu_e := f_e s_e$ is the contribution of the arc e to that duality gap. Observe that by our primal dual feasibility condition we have always that $\mu_e \geq 0$, for each arc e.

Centrality. In the light of the above, the duality gap $\|\mu\|_1 = \sum_e \mu_e$ provides a natural measure of progress for our algorithm. In particular, our goal is simply to find a primal dual feasible solution (f, s) with $\|\mu\|_1$ sufficiently close to 0, and, from this perspective, any way of updating that primal dual solution that leads to reduction of the duality gap should be desirable.

However, for reasons that will become clear later, we will insist on reducing the duality gap $\|\mu\|_1$ in a more restricted manner. To define this, let us first associate with each arc e a value ν_e that we will refer to as the *measure* of e. We will always make sure that $\nu_e \geq 1$, for each arc e, and that the total sum of all the measures is not too large. Specifically, we want to maintain the following invariant that ensures that the average measure of an arc is at most 3.

INVARIANT 1. *We always have that*

$$\|\nu\|_1 = \sum_e \nu_e \leq 3m.$$

As it turns out, once we introduced the notion of arc measure, it will be much more convenient to introduce analogues of the standard ℓ_p-norms that are reweighted by the measures of arcs. To this end, for a given vector $x \in \mathbb{R}^{\mathrm{m}}$, and measure vector ν, let us define

$$\|x\|_{\nu,p} := \left(\sum_e \nu_e |x_e|^p\right)^{\frac{1}{p}}.$$

We will sometimes extend this notation to refer to the weighted norm of a subset of indices in the support. More specifically, given $S \subseteq \{1, \ldots, m\}$, we will call

$$\|x_S\|_{\nu,p} := \left(\sum_{e \in S} \nu_e |x_e|^p\right)^{\frac{1}{p}}.$$

Also, we will extend our notation and use (f, s, ν) to denote a primal dual solution (f, s) with its corresponding measure vector ν.

Now, we can make precise the additional constraint on our primal dual solutions (f, s, ν) that we want to maintain. Namely, we want each solution (f, s, ν) to be $\widehat{\mu}-$*centered* (or, simply, *centered*), i.e., we want that, for each arc e,

$$\mu_e = \nu_e \widehat{\mu},$$

where $\widehat{\mu}$ is a normalizing value we will refer to as *average duality gap.* Intuitively, centrality means that each arc's contribution to the duality gap is exactly proportional to its measure.[2]

Note that the above notions enable us to express the duality gap of a solution (f, y, ν) as exactly $\sum_e \nu_e \widehat{\mu}$, which by Invariant 1 is at most $3m\widehat{\mu}$. That is, we have that

$$\sum_e \mu_e = \sum_e \nu_e \widehat{\mu} \leq 3m\widehat{\mu}.$$

Consequently, we can view $\widehat{\mu}$ as a measure of our progress - driving it to be smaller translates directly into making the duality gap smaller too.

4.2 Making Progress with Electrical Flow Computations.

Once we setup basic definitions, we are ready to describe how we initialize our framework and then how we can use electrical flow computations to gradually improve the quality, i.e., the average duality gap $\widehat{\mu}$ of our $\widehat{\mu}$-centered primal dual solution (f, y, ν).

Initialization. As we want our solutions (f, y, ν) to be always centered, initialization of our framework, i.e., finding the initial centered primal dual solution, might be difficult. Fortunately, one of the important properties of the reduction we performed in Section 3 is that a centered primal dual feasible solution of the resulting uncapacitated minimum-cost σ-flow instance can be specified explicitly.

LEMMA 4. *Given a bipartite graph $G = (V, E, c)$ along with an integral demand vector σ and a subset of vertices P such that that for all $v \in P$ we have $\sigma_v = \deg(v)/2$, whereas for all $v \notin P$ we have $\deg(v) = 2$, one can construct in linear time a feasible primal-dual set of variables (f, s) that satisfy the centrality bound for $\widehat{\mu} = \|c\|_\infty$ and $\|\nu\|_1 \leq m$.*

Proof. Since $\sigma_v = \deg(v)/2$ for all $v \in P$, while all $w \notin P$ have degree precisely 2, we can set $f = \frac{1}{2} \cdot \vec{1}$ and have all the demands satisfied exactly. Moreover, we set the dual variables $y_v = \|c\|_\infty$ for all $v \in P$, and $y_w = 0$ for all $w \notin P$. This way the slacks $s_{vw} = c_{vw} - y_w + y_v$ are all within the range $[\|c\|_\infty, 2\|c\|_\infty]$. We set $\nu_e = \frac{s_e}{2\|c\|_\infty}$ and $\widehat{\mu} = \|c\|_\infty$ so that

$$\mu_e = f_e s_e = \frac{1}{2} s_e = \frac{s_e}{2\|c\|_\infty} \|c\|_\infty = \nu_e \widehat{\mu},$$

and

$$\|\nu\|_1 = \sum_e \nu_e = \sum_e \frac{s_e}{2\|c\|_\infty} \leq \sum_e 1 = m.$$

Taking an Improvement Step. Let us fix some $\widehat{\mu}$-centered primal dual solution (f, y, ν) and let us define resistances r to be equal to

$$r_e := \frac{1}{\hat{\mu}} \cdot \frac{s_e}{f_e} = \frac{1}{\hat{\mu}} \cdot \frac{\mu_e}{f_e^2} = \frac{\nu_e}{f_e^2}, \tag{4.7}$$

for each arc e. (Note that f has to always be positive due to centrality condition, and thus these resistances are well-defined.)

The fundamental object that will drive our improvements of the quality of our current primal dual solution will be the electrical $\sigma-$flow $\hat{f}$ determined by the above resistances r. For future reference, we will call the electrical flow $\hat{f}$ *associated with* (f, s, ν). The key property of that electrical flow is that it will enable us to update our primal and dual solutions *simultaneously.* That is, we can use the flow itself to update the primal solution f, and we can use the vertex potentials $\hat{\phi}$ that induced $\hat{f}$ to update our dual solution s. Specifically, our main improvement update step, for each arc $e = (u, v)$ is:

$$\begin{aligned} f'_e &:= (1-\delta) f_e + \delta \hat{f}_e, \\ s'_e &:= s_e - \frac{\delta}{(1-\delta)} \left(\widehat{\phi}_v - \widehat{\phi}_u\right), \end{aligned}$$

where δ is a *step size* that we will choose later.

REMARK 5. *The step derived from the standard primal-dual interior-point method computes an electrical flow along with potentials determined by resistances $\frac{s_e}{f_e}$, which are off by precisely a factor of $\hat{\mu}$ from the resistances we consider in this paper. However, scaling all resistances by the same factor has no effect on the electrical flow or the potentials produced. Setting resistances the way we do in (4.7) has*

[2]The framework in [24] works with a slightly relaxed notion of centrality. However, we deviate from that here.

the benefit that it will enable us to relate the electrical energy with another quantity of interest without having to carry along the extra $\frac{1}{\hat{\mu}}$ factor, as we will see in Lemma 7.

Intuitively, this update step mixes the electrical flow $\hat{f}$ with the current solution f by taking a convex combination of them. (Note that the resulting flow is guaranteed to be a σ- flow in this way.) On the other hand, the dual update corresponds to updating the line embedding of each vertex by adding an appropriately scaled vertex potential to it.

It is worth pointing out that the electrical flow $\hat{f}$ is inherently undirected. So, it is not a priori clear if the flow f' resulting from the above update is even feasible. As a result, we will need to ensure, in particular, that the step size δ is chosen to be small enough so as to ensure that f' is still feasible. (In fact, as we will see shortly, there are some even stronger restrictions on the value of δ. So, the feasibility will be enforced implicitly.)

Congestion Vector. A notion that will be extremely useful in analyzing our improvement step and the performance of our algorithm in general is the notion of congestion vectors. Specifically, given the electrical σ-flow $\widehat{f}$ associated with our solution (f, y, ν), let us define *congestion* ρ_e of an arc e as

$$\rho_e := \frac{\left|\hat{f}_e\right|}{f_e}. \tag{4.8}$$

Now, observe that we can express the duality contribution μ'_e of an arc e in the new solution (f', s') as

$$\begin{aligned}
\mu'_e &= f'_e s'_e \\
&= \left((1-\delta)f_e + \delta\hat{f}_e\right)\left(s_e - \frac{\delta}{(1-\delta)}\left(\widehat{\phi}_v - \widehat{\phi}_u\right)\right) \\
&= (1-\delta)f_e s_e - \delta f_e\left(\widehat{\phi}_v - \widehat{\phi}_u\right) + \delta\hat{f}_e s_e \\
&\quad - \frac{\delta^2}{1-\delta}\hat{f}_e\left(\widehat{\phi}_v - \widehat{\phi}_u\right) \\
&= (1-\delta)f_e s_e - \delta f_e \cdot \hat{f}_e\frac{s_e}{f_e} + \delta\hat{f}_e s_e \\
&\quad - \frac{\delta^2}{1-\delta}\hat{f}_e \cdot \hat{f}_e\frac{s_e}{f_e} \\
&= (1-\delta)\mu_e - \frac{\delta^2}{1-\delta}\mu_e\rho_e^2.
\end{aligned}$$

So, if we ignore the second-order term in δ, the duality gap contribution of each arc e goes down at the same rate. In this way, not only the duality gap gets reduced by a factor of $(1-\delta)$ but also the centrality of the solution would be perfectly preserved.

However, we cannot really ignore the second-order term and this term will make our solution lose centrality. Fortunately, one can show that as long as the total degradation of centrality condition is not too large one can easily correct it with a small number of electrical flow computations. Specifically, for the correction to be possible, we need to have that the total ℓ_2^2-norm of the degradations (measured with respect to measure ν and normalized by the duality gap contributions μ) has to be a small constant. That is, we need that

$$\begin{aligned}
&\sum_e \nu_e\left(\frac{\mu'_e}{(1-\delta)\mu_e} - 1\right)^2 \\
&= \sum_e \nu_e\left(\frac{\delta^2\mu_e\rho_e^2}{(1-\delta)^2\mu_e}\right)^2 \frac{\delta^4}{(1-\delta)^4}\sum_e \nu_e\rho_e^4 \\
&= \frac{\delta^4}{(1-\delta)^4}\|\rho\|_{\nu,4}^4 \leq \frac{1}{256},
\end{aligned}$$

which implies that it is sufficient to have

$$\delta \leq \frac{1}{8\cdot\|\rho\|_{\nu,4}},$$

i.e., that the step size δ should be bounded by the ℓ_4 norm of the congestion vector ρ. The following theorem makes these requirements, as well as the result of the full improvement step, precise. Its complete proof can be found in the full version of the paper.

Theorem 6. *Let (f, s, ν) be a $\widehat{\mu}-$centered solution and let ρ the congestion vector of the electrical flow $\widehat{f}$ associated with that solution. For any $\delta > 0$ such that*

$$\delta \leq \min\left\{\frac{1}{8\cdot\|\rho\|_{\nu,4}}, \frac{1}{8}\right\},$$

we can compute in $\tilde{O}(m)$ time a $\widehat{\mu}'-$centered solution (f', s', ν'), such that $\nu' = \nu$, $\widehat{\mu'} \leq (1-\delta)\widehat{\mu}$, and, for each arc e,

$$r'_e = \frac{1}{\hat{\mu}}\cdot\frac{s'_e}{f'_e} \geq (1 + 4\cdot\delta\rho_e + \kappa_e)^{-1} r_e,$$

where κ is a vector with $\|\kappa\|_{\nu,2} \leq 1$.

4.3 A Simple $O(\sqrt{m}\log W)$-iteration Bound.

Once the ℓ_4 norm bound provided in Theorem 6 is established we are already able to prove in a simple way that our algorithm needs at most $O(\sqrt{m}\log W)$ iterations to compute the optimal solution, making its total running time be at most $O\left(m^{3/2}\log W\right)$. To achieve that, we just need to argue that we always have that

$$\|\rho\|_{\nu,4} \le O\left(\sqrt{m}\right). \tag{4.9}$$

Once this is established, by Theorem 6, we know that we can always take $\delta = \Omega(m^{-1/2})$ and thus make $\widehat{\mu}$ decrease by a factor of $(1-\delta)$ in each iteration. So, after $O(\sqrt{m}\log W)$ iterations, $\widehat{\mu}$ and thus the duality gap becomes small enough that a simple rounding (see Section 7) will recover the optimal solution.

Now, to argue that 4.9 indeed holds, we first notice that we can always upper bound ℓ_4 norm $\|\rho\|_{\nu,4}$ by the ℓ_2 norm $\|\rho\|_{\nu,2}$ and then bound the latter norm instead. Next, as it turns out, we can tie the energy $\mathcal{E}(\hat{f})$ of the electrical flow $\hat{f}$ associated with a given solution (f,s,ν) to the corresponding ℓ_2 norm $\|\rho\|_{\nu,2}$. Specifically, we have the following lemma.

Lemma 7. *For any centered solution (f,s,ν), we have that*

$$\|\rho\|_{\nu,2}^2 = \mathcal{E}(\hat{f}),$$

where $\hat{f}$ is the electrical flow associated with that solution and ρ is its congestion vector.

Proof. Observe that by 4.8 and 4.7, we have that

$$\|\rho\|_{\nu,2}^2 = \sum_e \nu_e\rho_e^2 = \sum_e \nu_e\left(\frac{\widehat{f}_e}{f_e}\right)^2 = \sum_e r_e\hat{f}_e^2 = \mathcal{E}(\hat{f}).$$

Note that we used (4.7) to write $r_e = \frac{\nu_e}{f_e^2}$, which assumes centrality.

Due to this identity, we can view the ℓ_2 norm $\|\rho\|_{\nu,2}^2$ as energy. Finally, we can use the bound from Invariant 1 to show that the energy $\mathcal{E}(\hat{f})$ and thus the norm $\|\rho\|_{\nu,2}$ can be appropriately bounded as well.

Lemma 8. *For a centered solution, $\|\rho\|_{\nu,2}^2 \le \sum_e \nu_e = \|\nu\|_1$.*

Proof. By Lemma 7 and (4.7), we have that

$$\|\rho\|_{\nu,2}^2 = \mathcal{E}(\hat{f}) \le \mathcal{E}(f) = \sum_e r_e f_e^2 = \sum_e \nu_e\left(\frac{f_e}{f_e}\right)^2$$
$$= \sum_e \nu_e = \|\nu\|_1,$$

where the inequality follows as f is a σ-flow and, by the virtue of being an electrical σ-flow, $\widehat{f}$ has to have minimum among all the σ-flows.

Now, since by Invariant 1, $\|\nu\|_1 \le 3m$, we can conclude that

$$\|\rho\|_{\nu,4}^2 \le \|\rho\|_{\nu,2}^2 \le \|\nu\|_1 \le 3m,$$

and the bound 4.9 follows.

With this upper bound on the $\|\rho\|_{\nu,4}$ we can immediately derive a bound on the running time required to obtain an exact solution. This is summarized in the following theorem.

Theorem 9. *We can produce an exact solution to the unit-capacitated minimum cost σ-flow problem in $\tilde{O}\left(m^{3/2}\log W\right)$ time.*

Proof. Given an instance of the unit-capacitated minimum cost σ-flow, we can apply the reduction from Section 3 in linear time. Then, Lemma 4 establishes the initial centering with $\hat{\mu} = \|c\|_\infty \le W$. We previously saw that $\|\rho\|_{\nu,4} = O(\sqrt{m})$. Therefore, according to Theorem 6, we can set $\delta = 1/O(\sqrt{m})$, and reduce $\hat{\mu}$ by a factor of $1 - \frac{1}{O(\sqrt{m})}$ with every interior-point iteration. Therefore, in $O\left(m^{1/2}(\log m + \log W)\right)$ iterations we reduce $\hat{\mu}$ to $O(m^{-3})$. Using the fact that $\|\nu\|_1 \le 3m$, the duality gap of this solution will be $\sum_e \nu_e\hat{\mu} \le m^{-2}$. Note that each iteration requires $\tilde{O}(1)$ electrical flow computations, and each of them can be implemented in near-linear time, according to Theorem 1.

Therefore in $\tilde{O}\left(m^{3/2}\log W\right)$ time, we obtain a feasible primal-dual solution with duality gap less than m^{-2}. This can easily be converted to an integral solution in nearly-linear time using the method described in Section 7. Hence the total running time of the algorithm is $\tilde{O}(m^{3/2}\log W)$.

5 Taking Longer Steps.

In the previous section, we established that the amount of progress we can make in one iteration of our framework is limited by the ℓ_4 norm of the congestion, $\|\rho\|_{\nu,4}$ - see Theorem 6. We then showed (cf. (4.9)) that this norm is always upper bounded by $O(\sqrt{m})$, which resulted in our overall $\widetilde{O}\left(m^{3/2}\log W\right)$ time bound.

Unfortunately, a priori, this upper bound is tight, i.e., it indeed can happen that $\|\rho\|_{\nu,4} = \|\rho\|_{\nu,2} = \Omega(\sqrt{m})$. In fact, this is exactly the reason why all classic analyses of interior-point algorithms are able to obtain only an $O(\sqrt{m})$ iteration bound.

To circumvent this problem, [24] introduced a perturbation technique into the framework. These perturbations target arcs that contribute a significant fraction of the norm $\|\rho\|_{\nu,4}$, by increasing their resistance (and thus the corresponding energy), in order to discourage the emergence of such high contributing arcs in the future. A careful analysis done in [24] shows that such perturbations indeed ensure that there are sufficiently many iterations with relatively small $\|\rho\|_{\nu,4}$ norm to guarantee ability to take longer steps, and thus converge faster. Unfortunately the changes to the underlying optimization problem that these perturbations introduced, although sufficiently mild to enable obtaining a result for unit-capacity maximum flow, were too severe to enable solving any of the weighted variants that we aim to tackle here.

Our approach will also follow the same general outline. The crucial difference though is that we use a different perturbation technique, along with a somewhat simpler analysis. This technique still achieves the desired goal of increasing the resistance of the perturbed arcs. However, in big contrast to the technique used by [24], our technique does not affect the costs of those arcs – it affects only their measure. Also, as an added benefit, our perturbation treatment simplifies the analysis significantly.

We describe our preconditioning technique in Section 5.1. Also, in the table below we present a general outline of our algorithm. (This algorithm will be later modified further to include a preconditioning step.) Observe that this algorithm uses a stronger, ℓ_3 norm criterion for whether to make a perturbation or a progress step, instead of the ℓ_4 norm criterion that Theorem 6 suggests. As we will see, the reason for that is that maintaining such an ℓ_3 norm condition will enable us to have a sufficiently tight control over the change after each progress step of our potential function: the energy of electrical flows $\hat{f}$ associated with our primal dual solutions.

Our goal is to obtain an $\tilde{O}(m^{1/2-\eta})$ bound on the overall number of iterations, where we fix η to be $\eta = \frac{1}{14}$.

Algorithm 1 Perturbed interior-point method (parameters: $c_\rho = 400\sqrt{3} \cdot \log^{1/3} W$, $c_T = 3c_\rho \log W$)

```
1. Initialize primal and dual variables $(f, y)$ (as in Section 4).
2. Repeat $c_T \cdot m^{1/2-\eta}$ times:
3.     While $\|\rho\|_{\nu,3} > c_\rho \cdot m^{1/2-\eta}$
4.         Perform perturbation (as in Section 5.1).
5.     Perform progress steps (as in Section 4.2).
```

5.1 Our Perturbation Technique.

Let us start by describing our perturbation technique, which heavily uses the structure of the b-matching instance obtained after applying the reduction from Section 3. We first show how a perturbation is applied to an arc, then we define the set of arcs that get perturbed in each iteration. As we will see, whenever we perturb a particular arc (to increase its resistance) we always make sure to perturb its "partner" arc, i.e., the unique arc sharing a common vertex from the set Q of the bipartition, as well.

Definition 10. *Given an arc $e = (u, v)$, with $u \in P$, $v \in Q$, the partner arc of e is the unique arc $\bar{e} = (\bar{u}, v)$, $\bar{u} \in P$ sharing vertex v with e.*

5.1.1 Perturbing an Arc.

Let $e = (u, v)$ be an arc with cost c_{uv} and vertex potentials y_u, respectively y_v, and slack $s_{uv} = c_{uv} + y_u - y_v$. Note that due to the structure of our b-matching instance (see Section 3), vertex v is of degree 2. Let $\overline{e} = (\overline{u}, v)$ be the partner arc that shares with e this vertex v. We first modify our dual solution by setting $y_v \leftarrow y_v - s_{uv}$. This effectively doubles the resistance of e, defined as in (4.7), which is our desired effect.

Unfortunately, this update breaks centrality of *both* arc e and its partner arc $\bar{e}$. To counteract that, we first double the measure ν_e of e - this immediately restores the centrality of that arc. Now, it remains to fix the centrality of the partner arc $\overline{e} = (\overline{u}, v)$. Specifically, we need to deal with the fact that the slack $s_{\overline{e}}$ of that partner arc gets increased by s_e. To fix this problem, recall that the centrality condition for $\overline{e}$ guaranteed that $s_{\overline{e}} f_{\overline{e}} = \nu_{\overline{e}} \hat{\mu}$. So, we need to set the new measure $\nu'_{\overline{e}}$ such that $(s_{\overline{e}} + s_e) f_{\overline{e}} = \nu'_{\overline{e}} \hat{\mu}$.

Therefore we just set the new measure to be

$$\nu'_{\bar{e}} = \frac{(s_{\bar{e}} + s_e) f_{\bar{e}}}{\hat{\mu}} = \nu_{\bar{e}} + \frac{s_e f_{\bar{e}}}{\hat{\mu}} = \nu_{\bar{e}} + \nu_e \cdot \frac{f_{\bar{e}}}{f_e}.$$

Consequently, the total change in measure of that arc is

$$(5.10) \qquad \nu_e \left(1 + \frac{f_{\bar{e}}}{f_e}\right) \le \nu_e \left(1 + \frac{1}{f_e}\right),$$

as in our instance we have that $f_{\bar{e}} \le 1$, since the arcs are unit capacitated, and f is always feasible.

We remark that we may want to perturb both an arc $e = (u, v)$ and its partner $\bar{e} = (\bar{u}, v)$. In this case, we can perturb the arcs simultaneously by setting $y_v \leftarrow y_v - s_{uv} - s_{\bar{u}v}$, and updating the measures: $\nu_{\bar{e}} \leftarrow 2\nu_{\bar{e}} + \nu_e \cdot \frac{f_{\bar{e}}}{f_e}$, $\nu_e \leftarrow 2\nu_e + \nu_{\bar{e}} \cdot \frac{f_e}{f_{\bar{e}}}$. This maintains centrality, and the bound from (5.10) still holds.

So, to summarize, one effect of the above operation is that it made the resistance of the perturbed arc e double. As we will see, similarly as it was the case in [24], this will enable us to ensure that the total number of perturbation steps is not too large. Also, note that the above operation does not change any vertex demands or costs. It only affects the dual solution and the arcs' measure. Therefore, the only undesirable long term effect of it is the measure increase, since it might lead to violation of Invariant 1. [3]

5.1.2 Which Arcs to Perturb? As we have just seen, while perturbing an arc doubles its resistance, this operation has the side effect of increasing the total measure. To control the latter, undesirable effect, we show that every time we need to perturb the problem, we can actually select a subset of arcs with the property that perturbing all of them increases the energy by a lot while keeping the measure increase bounded. Ultimately, the entire goal of the analysis will be to show that:

1. we do not need to perturb the problem more than $\tilde{O}(m^{1/2-\eta})$ times;
2. the total increase in measure caused by the perturbations is at most $2m$, thus maintaining Invariant 1.

Below we define the subset of arcs that will get perturbed. Intuitively, we only want to perturb arcs e with high congestion ρ_e. Furthermore, we choose to perturb them only if the amount of flow they carry is not very small. This extra restriction enforces an upper bound on the amount by which the measure of the perturbed arc increases, as per equation (5.10).

As we will see in Corollary 16, perturbing edge e will increase total energy by at least a quantity that is proportional to the amount of contribution to total energy of that edge; therefore the total measure increase will be upper bounded by a quantity proportional the total energy increase due to perturbations.

We will soon see that the total energy increase across iterations is, as a matter of fact, bounded by $O\left(c_T c_\rho^2 \cdot m^{3/2-3\eta}\right)$, where c_T and c_ρ are some appropriately chosen constants, which immediately yields the desired bound on the total measure increase.

DEFINITION 11. *An arc e is perturbable if $\frac{1}{f_e} \le m^{1/2-3\eta}$ and $\rho_e \ge \sqrt{40 c_T c_\rho^2 \cdot m^{1/2-3\eta}}$. An arc that is not perturbable is called unperturbable. Denote by $\mathcal{S}$ the set of all perturbable arcs.*

A useful property of perturbable arcs the way they are chosen enforces a small increase in measure compared to that in energy during each perturbation. We will make this property precise below, and it is what we will be using for the remainder of the section.

COROLLARY 12. *A perturbable arc satisfies $1 + \frac{1}{f_e} \le C\rho_e^2$, where $C = \frac{1}{20 c_T c_\rho^2} \cdot m^{-1/2+3\eta}$.*

5.2 Runtime Analysis. The analysis of our algorithm is based on two major parts.

The first part shows that throughout the execution of the algorithm, the total energy increase caused by perturbations can not be too large. This will automatically imply that total measure increase will be bounded by $2m$, and therefore Invariant 1 is preserved. The key idea is that, since they are applied only when the ℓ_3 norm of the congestion is "small" (i.e. $c_\rho \cdot m^{1/2-\eta}$), progress steps do not decrease the energy by a lot (i.e. $O(c_\rho^2 \cdot m^{1-2\eta})$, as we will soon see). However, since total measure, and hence energy,

[3]In fact, if we were to formulate our problem as a primal interior-point method, one could think of these perturbations on arcs and their partners as acting on both the lower and the upper barriers. In that formulation, the barrier would be of the form $-\sum_e \nu_e \log f_e + \nu_{\bar{e}} \log(1 - f_e)$. The reduction from Section 3 essentially eliminates the upper barrier, in order to be make our problem amenable to a primal-dual approach, which we preferred to use here.

was $O(m)$ to begin with, perturbations could not have increased energy by more than progress steps have decreased it overall. Over the $O\left(c_T \cdot m^{1/2-\eta}\right)$ iterations, progress steps decrease energy by at most $O(c_T c_\rho^2 \cdot m^{3/2-3\eta})$; therefore this is also a bound on the total increase in energy.

The second part use the invariant that perturbable edges consume most of the energy in the graph, in order to argue that the number of perturbations is small. While a priori we only had a bound on the time required for progress steps, with no guarantee on how many iterations the algorithm spends performing perturbations, this argument provides a bound on the number of perturbations, and hence on the running time of the algorithm. Showing that, every time we perform a perturbation, energy increases by at least $\Omega\left(c_\rho \cdot m^{1-2\eta}\right)$ implies, together with the bound proven in the first part, that throughout the execution of the algorithm we perform only $O\left(c_T c_\rho \cdot m^{1/2-\eta}\right)$ perturbations. This bounds the running time by $\tilde{O}\left(c_T c_\rho \cdot m^{3/2-\eta}\right)$, since perturbing the problem takes only $\tilde{O}(m)$ time.

The invariant that the second part relies on is motivated by the fact that, whenever we have to perturb the problem, the ℓ_3 norm of the congestion vector is large, so the energy of the system is also large (at least $c_\rho^2 \cdot m^{1-2\eta}$). Since perturbable arcs are highly congested, we expect them to contribute most of the energy; so perturbing those should increase the energy of the system by a quantity proportional to their current contribution to energy. Maintaining this invariant requires a finer control over how the electrical flows behave, and will be guaranteed via a modification of the algorithm, which will be carefully analyzed in Section 6. However, the future modification will not affect any of the analysis described in this section. Therefore this section will be concerned only with proving the runtime guarantee, assuming validity of the invariant.

5.2.1 Bounding the Total Increase in Measure and Energy. We formalize the intuition described at the beginning of the section. First, we show that Theorem 6 provides a bound on how much energy can decrease during one progress step. This relies on the following lemma, which allows us to lower bound the energy of an electrical flow.

LEMMA 13. *Let $\mathcal{E}_r$ be the energy of the electrical flow in a graph with demands σ and resistances r. Then*

$$\mathcal{E}_r = \max_\phi \left(2\sigma^\top \phi - \sum_{e=(u,v)} \frac{(\phi_u - \phi_v)^2}{r_e} \right).$$

Proof. The result can be derived by letting L be the Laplacian corresponding to the graph with resistances r, and rewriting the above maximization problem as $\max_\phi 2\sigma^\top\phi - \phi^\top L\phi$. By first order optimality conditions we get that the maximizer satisfies $L\phi = \sigma$, hence $\phi = L^+\sigma$. Plugging in makes the expression equal to $\sigma^T L^+ \sigma$, which is precisely the energy $\mathcal{E}_r$.

In our context, this lemma enables us to provide a more convenient formula for lower bounding the new value of energy after resistances change.

LEMMA 14. *Let $\mathcal{E}_r$ be the energy of the electrical flow corresponding to a centered instance with resistances r, and let $\mathcal{E}_{r'}$ be the energy of the electrical flow corresponding to the new centered instance with resistances r', obtained after applying one progress step or one perturbation. Then the change in energy can be lower bounded by:*

$$\mathcal{E}_{r'} - \mathcal{E}_r \geq \sum_{e=(u,v)} \nu_e \rho_e^2 (1 - r_e / r_{e'}).$$

A first application of this lemma is that it enables us to lower bound the increase in energy when perturbing arcs.

LEMMA 15. *After perturbing arcs in $\mathcal{S}$, energy increases by at least $\frac{1}{2}\left\|\rho_\mathcal{S}\right\|_{\nu,2}^2$.*

Proof. According to the effects of the perturbation described in Section 5.1.1, all resistances of arcs in $\mathcal{S}$ get doubled, while the others can only increase. Therefore, applying Lemma 14, we obtain a lower bound on the energy increase:

$$\begin{aligned}\mathcal{E}_{r'} - \mathcal{E}_r &\geq \sum_{e\in\mathcal{S}} \nu_e \rho_e^2 \left(1 - \frac{r_e}{2r_e}\right) = \frac{1}{2}\sum_{e\in\mathcal{S}} \nu_e \rho_e^2 \\ &= \frac{1}{2}\left\|\rho_\mathcal{S}\right\|_{\nu,2}^2 .\end{aligned}$$

An immediate corollary is that the increase in energy during a perturbation upper bounds the increase in measure.

Corollary 16. *If a perturbation increases energy by Δ, then the total measure increases by at most $2C \cdot \Delta$.*

Proof. By definition, the arcs we perturb satisfy $1 + \frac{1}{f_e} \leq C\rho_e^2$. According to (5.10), the measure increase cause by perturbing an arc e is at most $\nu_e(1+1/f_e)$. Therefore, perturbing all the arcs in $\hat{\mathcal{S}}$, increases measure by at most $\sum_{e\in\hat{\mathcal{S}}} \nu_e \cdot C\rho_e^2 = C \cdot \|\rho_{\mathcal{S}}\|_{\nu,2}^2$. But Lemma 15 shows that $\Delta \geq \frac{1}{2}\|\rho_{\mathcal{S}}\|_{\nu,2}^2$. Combining these two bounds yields the result.

While Lemma 15 tells us that perturbations increase energy, we can show that progress steps do not decrease it by too much, using another application of Lemma 14.

Lemma 17. *Let $\mathcal{E}_r$ be the energy of an electrical flow corresponding to a centered solution with congestion vector ρ, and let $\mathcal{E}_{r'}$ be the new energy after applying one progress step. Then $\mathcal{E}_{r'} \geq \mathcal{E}_r - 5 \cdot \|\rho\|_{\nu,3}^2$.*

With this tool in hand we can now upper bound the total energy increase caused by perturbations.

Lemma 18. *The total energy increase due to perturbations is at most $16c_Tc_\rho^2 \cdot m^{3/2-3\eta}$. Furthermore, the total measure always satisfies $\|\nu\|_1 \leq 3m$, i.e. Invariant 1 is preserved.*[4]

Proof. We start by introducing some notation. Let $\mathcal{E}^t$ and ν^t be the energy, respectively the vector of measures at the end of the t^{th} iteration. Also, let Δ^t be the total amount of energy increases during that iteration.[5]

Note that, since we only perform progress steps when $\|\rho\|_{\nu,3} \leq c_\rho m^{1/2-\eta}$, one progress step decreases energy by at most $5 \cdot c_\rho^2 m^{1-2\eta}$, according to Lemma 17. Therefore the amount by which energy increases during an iteration can be bounded by

$$\Delta^t \leq \mathcal{E}^t - \mathcal{E}^{t-1} + 5c_\rho^2 \cdot m^{1-2\eta}. \tag{5.11}$$

At any point, the energy is capped by the total measure (Lemma 8). Therefore

$$\mathcal{E}^t \leq \left\|\nu^t\right\|_1.$$

Also, using Corollary 16 we get that every increase in energy by Δ^t increases the total measure by at most $2C \cdot \Delta^t$. Hence

$$\left\|\nu^t\right\|_1 \leq \left\|\nu^{t-1}\right\|_1 + 2C \cdot \Delta^t.$$

Using (5.11) and summing over all $T = c_T \cdot m^{1/2-\eta}$ iterations of the algorithm we obtain:

$$\sum_{t=1}^{T} \Delta^t \leq 10c_Tc_\rho^2 \cdot m^{3/2-3\eta},$$

and the measure increase is upper bounded by

$$2C \cdot \left(\sum_{t=1}^{T} \Delta^t\right) = 2C \cdot 10c_Tc_\rho^2 \cdot m^{3/2-3\eta} = m.$$

So Invariant 1 is satisfied.

5.2.2 Bounding the Number of Perturbations.

We have just seen that the energy increase suffered due to perturbations is $O(c_Tc_\rho^2 \cdot m^{3/2-3\eta})$, which should intuitively enable us to bound the number of perturbations, and thus wrap up the analysis of the algorithm. The reason is that whenever we have to perturb the problem, the ℓ_3 norm of the congestion vector is large (i.e. $\|\rho\|_{\nu,3} \geq c_\rho \cdot m^{1/2-\eta}$), so the energy of the system is large: $\mathcal{E} = \|\rho\|_{\nu,2}^2 \geq \|\rho\|_{\nu,3}^2 \geq c_\rho^2 \cdot m^{1-2\eta}$. Since perturbable arcs are highly congested (see Definition 11), we expect them to contribute most of the energy. This feature of perturbable arcs is highlighted by the following invariant:

Invariant 2. *Whenever we perform a perturbation,*

$$\|\rho_{\mathcal{S}}\|_{\nu,2}^2 \geq \frac{1}{2}c_\rho \cdot m^{1-2\eta}.$$

This guarantees that every perturbation increases energy by at least $\Omega\left(m^{1-2\eta}\right)$, which automatically implies that the number of perturbations is bounded by $O\left(c_Tc_\rho^2 \cdot m^{1/2-\eta}\right)$. Indeed, as seen in Lemma 15, with every perturbation energy increases by $\frac{1}{2}\|\rho_{\mathcal{S}}\|_{\nu,2}^2$. Therefore, assuming Invariant 2, we get

[4]While the constants provided here are worse than those seen in the proof, we will use these loose bounds in order to accommodate some future changes in the algorithm.

[5]Remember that energy can decrease during progress steps, as per Lemma 17; Δ^t measures the total amount of all increases, without accounting for the lost energy due to progress steps.

that each perturbation increases energy by at least $\frac{1}{4}c_\rho \cdot m^{1-2\eta}$. Since we know from Lemma 18 that total energy increase is bounded by $16c_T c_\rho^2 \cdot m^{3/2-3\eta}$, we get that the number of perturbations performed during the execution of the algorithm is at most $64c_T c_\rho \cdot m^{1/2-\eta}$.

Enforcing the validity of this invariant will be done in Section 6, where we introduce a preconditioning technique which enables us to gain more control over the behavior of electrical flows.

Hence we have proved the following Lemma:

Lemma 19. *Assuming Invariant 2 is valid, the number of perturbations is at most* $64c_T c_\rho \cdot m^{1/2-\eta}$.

This immediately concludes the running time analysis. Indeed, both progress steps and perturbations can be implemented in $\tilde{O}(m)$ time by computing electrical flows using a fast Laplacian solver (see Theorem 1). The number of progress steps is precisely $c_T \cdot m^{1/2-\eta}$, since this is hard coded in the description of the algorithm. Also, the number of perturbations is $O(c_T c_\rho \cdot m^{1/2-\eta})$, according to Lemma 19. Therefore the total running time is $\tilde{O}\left(c_T c_\rho \cdot m^{3/2-\eta}\right)$.

Theorem 20. *Assuming Invariant 2 is valid, we can produce an exact solution to the unit-capacitated minimum cost σ-flow problem in* $\tilde{O}(m^{10/7}\log^{4/3} W)$ *time.*

Proof. The proof is similar to the one for Theorem 9. The algorithm performs a progress step only when $\|\rho\|_{\nu,4} \leq \|\rho\|_{\nu,3} \leq m^{1/2-\eta}$, therefore $\hat{\mu}$ decreases by a factor of $1 - \frac{1}{c_\rho \cdot m^{1/2-\eta}}$ with every iteration. Therefore, in $c_\rho m^{1/2-\eta}\left(2\log m + \log W\right) \leq c_T \cdot m^{1/2-\eta}$ iterations we reduce $\hat{\mu}$ to $O(m^{-3})$, and by Invariant 1 the duality gap of this solution will be $\sum_e \nu_e \hat{\mu} \leq m^{-2}$. Each of the $c_T \cdot m^{1/2-\eta}$ progress steps requires $\tilde{O}(1)$ electrical flow computations, and each of them can be implemented in near-linear time, according to Theorem 1. Furthermore, assuming Invariant 2, we have that the number of perturbations is at most $64c_T c_\rho \cdot m^{1/2-\eta}$, by Lemma 19. Similarly, each perturbation can be implemented in nearly-linear time. Therefore the total running time required for obtaining a duality gap of m^{-2} is $\tilde{O}(c_T m^{3/2-\eta} + c_T c_\rho \cdot m^{3/2-\eta}) = \tilde{O}(m^{10/7}\log^{4/3} W)$. Then, using the repairing algorithm from Section 7, we can round the solution to to an optimal one in nearly-linear time. So the total time is $\tilde{O}(m^{10/7}\log^{4/3} W)$.

One should note that the $\tilde{O}(m^{10/7}\log^{4/3} W)$ running time can be reduced to $\tilde{O}(m^{10/7}\log W)$ by employing the scaling technique of [11]. This technique is also described in the full version of the paper, Section 7. Thus, we can reduce our problem to solving $O(\log W)$ instances of our problem where the costs are polynomially bounded. This enables us to change the $\operatorname{poly}\log W$ factors from the running time to $\operatorname{poly}\log n$ (and thus have them absorbed by the $\tilde{O}$ notation) at the cost of paying only an extra factor of $\log W$.

However, ensuring that Invariant 2 always holds is a bit more subtle. Obtaining a provable guarantee will actually be done by adding a preconditioner, which is carefully analyzed in Section 6.

6 Preconditioning the Graph.

Our analysis from the previous section was crucially relying on the assumption that perturbable arcs contribute most of the energy. Unfortunately, this assumption is not always valid. To cope with this problem, we develop a modification of our algorithm that ensures that this assumption holds after all. Roughly speaking, we achieve that by an appropriate preconditioning of our graph. This preconditioning is based on augmenting the graph with additional, auxiliary edges which make the computed electrical flows better behaved. These edges *should not* be thought of as being part of the graph we are computing our σ-flows on. Their sole effect is to guide the electrical flow computation in order to obtain a better electrical flow in the original graph at the cost of slightly changing the demand we are routing.

These edges achieve the optimal trade-off between providing good connectivity in the augmented graph (which lowers the congestion of arcs with low residual capacity) and preventing too much flow from going through them (because of their sufficiently high resistance).

One difficulty posed by this scheme is that we need to control how much the routed demand gets modified. This is easily controlled by setting the resistances of the auxiliary edges to be sufficiently high; in contrast, the magnitude of these resistances needs to be traded against the effect they have on the computed electrical flow. At the end, we fix the demand using a combinatorial procedure (see Section 7) whose running time is proportional to the ℓ_1 difference between the initial demand and the one routed by the algorithm. Therefore we need to simultaneously ensure that preconditioner edges have sufficiently high resistance such that the change in demand is not significant, and guarantee that the graph has good enough connectivity for Invariant 2 to hold. This trade-off will ultimately determine the choice of the parameter $\eta = 1/14$.

6.1 Using Auxiliary Edges for Electrical Flow Computations. In order to properly describe preconditioning, we need to partition the iterations of the algorithm into *phases* (each of them consisting of a sequence of $m^{2\eta}$ iterations), and show that a finer version of Lemma 18 holds, for each of these phases. The reason is that the resistances on the auxiliary edges need to depend on the set of measures in the graph. But measures increase over time, so the resistances need to be updated accordingly. One should be careful about this aspect, since changing the resistances of auxiliary edges during every iteration of the algorithm would destroy the potential based argument we described in Section 5. Therefore, instead of adjusting the resistances every iteration, we do this only at the beginning of a phase. Over the course of a phase, measures can not change too much, so the preconditioning argument will still be valid.

Definition 21. *Partition the $c_T \cdot m^{1/2-\eta}$ iterations of the algorithm into consecutive blocks of size $m^{2\eta}$. Such a block is called a phase. Hence the algorithm consists of $c_T \cdot m^{1/2-3\eta}$ phases.*

Preconditioning consists of adding an extra vertex v_0 along with undirected edges (v_0, v) for each vertex $v \in P$ (recall that vertices in P correspond to vertices from only one side of the bipartition in the b-matching instance). We will call these newly added edges *auxiliary edges*. Each of these auxiliary edges will have resistance set to

$$r_{v_0 v} = \frac{m^{1+2\eta}}{a(v)}, \tag{6.12}$$

where we define

$$a(v) = \sum_{u \in Q : e=(v,u) \in E} \nu_e + \nu_{\bar{e}}.$$

Recall that by $\bar{e}$ we denote the partner arc of e (introduced in Definition 10), and that the quantities above are defined with respect to the measures existing at the beginning of the phase.

Also, remember that these auxiliary edges exist only in order to provide a mildly different demand for which electrical flows are better behaved. Once we are done perturbing, we perform a progress step on the graph without auxiliary edges, but with the modified demand (i.e. the one that gets routed on the original graph, after removing auxiliary edges).

The inclusion of auxiliary edges in the electrical flow computations requires the contribution of these edges to the energy of the system to be included in the potential based argument from Theorem 18, when analyzing the preconditioned steps.[6]

Remark 22. *Even though we include additional edges for electrical flow computations, the energy bound from Lemma 8 still holds (since including additional edges can only decrease energy).*

This motivates partitioning the iterations into phases, since changing the resistances of auxiliary edges too often could potentially make the energy vary wildly.

The following lemma shows that we can individually bound the energy and measure increase over any single phase. What is crucial about the new proof is that it does not require any control over how energy changes between iterations belonging to different phases. Therefore, resetting the resistances of auxiliary edges at the beginning of a phase will have no effect on the result described in Lemma 18.

The precise statement concerning energy and measure increase during a phase is summarized in the following lemma, whose proof we defer to the full version of the paper.

Lemma 23. *During a single phase, the total energy increase due to perturbations is at most $16c_\rho^2 \cdot m$. Furthermore, the measure increase during a single phase is at most $\frac{2}{c_T} \cdot m^{1/2+3\eta}$. Also, the total measure always satisfies $\|\nu\|_1 \leq 3m$, i.e. Invariant 1 is preserved.*

It immediately follows that this is simply a refinement of Lemma 18:

Corollary 24. *The total energy increase due to perturbations is at most $16c_T c_\rho^2 \cdot m^{3/2-3\eta}$, and the result described in Lemma 18 is still valid.*

The new version of the algorithm which includes the effect of the auxiliary edges is described below.

[6]The ℓ_3 norm of the congestion vector will still be measured only with respect to the arcs in the original graph.

Algorithm 2 Perturbed interior-point method with preconditioning edges (parameters: $c_\rho = 400\sqrt{3} \cdot \log^{1/3} \tilde{W}$, $c_T = 3c_\rho \log \tilde{W}$)

```
1. Initialize primal and dual variables $(f, y)$ (as in Section 4).
2. Repeat for $c_T \cdot m^{1/2-3\eta}$ phases:
3.   Reset auxiliary edge resistances (as in (6.12)).
4.   Repeat $m^{2\eta}$ times:
5.     While $\|\rho\|_{\nu,3} > c_\rho \cdot m^{1/2-\eta}$,
6.       Perform perturbation (as in Section 5.1).
7.     Perform progress steps on the original graph (as in Section 4.2).
```

Before proving that this version of the algorithm forces Invariant 2 to stay valid, we first bound the change in demand caused by the auxiliary edges. We first show that, due to Invariant 1, the total amount of flow that gets routed electrically through auxiliary edges is small.

Proposition 25. *Let $\mathcal{P}$ be the set of auxiliary edges. Then the total amount of electrical flow on these edges during any progress step satisfies $\left\|\hat{f}_{\mathcal{P}}\right\|_1 \leq 5 \cdot m^{1/2-\eta}$.*

This proposition shows that the demand routed within a progress step is off by at most $5m^{1/2-\eta}$ from the demand routed by the iterate f at that point. Using this fact, we can show that the flow obtained in the end routes a demand that is off by at most $c_T \cdot m^{1/2-\eta}$ from the original demand.

Lemma 26. *Consider the last flow iterate f^T, and let σ^T be the demand routed by this flow. Then the difference between σ^T and the original demand σ satisfies $\left\|\sigma^T - \sigma\right\|_1 \leq c_T \cdot m^{1/2-\eta}$.*

6.2 Proving Invariant 2.

We can finally proceed with proving that, for this version of the algorithm, Invariant 2 holds. We do so by upper bounding the ℓ_3 norm of the congestions of unperturbable arcs $\|\rho_{\bar{\mathcal{S}}}\|_{\nu,3}$. Showing that this quantity is significantly smaller than $\tilde{O}((c_T c_\rho^2)^{1/6} \cdot m^{1/2-\eta})$ whenever we perform a perturbation automatically implies our result; this is because this lower bounds the ℓ_3 norm of congestions of perturbable arcs, and therefore also their energy. We defer the proof to the full version of the paper.

7 Repairing the Matching.

In this section we assume the b-matching view on σ-flows. Hence, summarizing Theorem 20, Lemma 26 and the proof of Invariant 2 in Section 6 we obtain the following result.

Theorem 27. *Consider a ternary instance $G = (V, E, c)$ of the weighted perfect bipartite b-matching problem when $\|b\|_1 = O(m)$. In $\tilde{O}(m^{10/7} \log^{4/3} W)$ time we can either conclude that G does not have a perfect b-matching or return a primal-dual pair with duality gap at most m^{-2} to the perfect b^+-matching problem, where $\|b^+ - b\|_1 \leq c_T m^{3/7}$.*

In this section "repair" the feasible primal-dual solution given by the above theorem. Our repair procedure will consists out of four steps. In the first step, we reduce the duality gap to 0. Next, we round the solution to be integral. In the third step, we repair the perturbations done to demands during our rounding procedure. Finally, we will use scaling to reduce the dependence on $\log W$.

All of these steps are described and analyzed in the full version of the paper, so we omit them here.

This will ultimately yield the following result:

Corollary 28. *Given a graph $G = (V, E, c)$ and a demand vector σ, we can produce an exact solution to the unit-capacity minimum cost σ-flow problem in $\tilde{O}(m^{10/7} \log W)$ time.*

8 Shortest Paths with Negative Weights.

We are given a directed graph $G(V, E, c)$ together with the edge weight function $c : E \rightarrow \{-W, \ldots, 0, \ldots, W\}$ and a source vertex s. Our goal is to compute shortest paths from s to all vertices in V. We start by reducing this shortest paths problem to the weighted perfect 1-matching problem using the reduction that was given by Gabow [11]. The main step of this reduction is a construction of a bipartite graph $G_{12} = (V_1 \cup V_2, E_{12}, c_{12})$ such that the vertex packing problem in G_{12} induces a valid potential function on G. Using this potential function we can reweigh w to remove negative weights.

The description of this reduction can be found in the full version of the paper. By applying the analysis from Section 7 to this reduction we obtain a reweighed non-negative instance that can be solved using Dijkstra's algorithm, and hence the following result:

Corollary 29. *Single source shortest paths in a graph with negative weights can be computed in* $\tilde{O}(m^{10/7} \log W)$ *time.*

References

[1] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[2] Bela Bollobas. *Modern Graph Theory*. Springer, 1998.

[3] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving sdd linear systems in nearly m $\log^{1/2} n$ time. In *STOC'14: Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 343–352, 2014.

[4] Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time. *CoRR*, abs/1605.01717, 2016.

[5] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 451–460, New York, NY, USA, 2008. ACM.

[6] E.A. Dinic and M. A. Kronrod. An Algorithm for the Solution of the Assignment Problem. *Soviet Math. Dokl.*, 10:1324–1326, 1969.

[7] Ran Duan and Hsin-Hao Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1413–1424, 2012.

[8] J. Edmonds and R.M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM*, 19(2):248–264, 1972.

[9] E. Egerváry. Matrixok kombinatorius tulajdonságairól (hungarian) on combinatorial properties of matrices. *Matematikai és Fizikai Lapok*, 38:16–28, 1931.

[10] H. N. Gabow and P. Sankowski. Algebraic algorithms for b-matching, shortest undirected paths, and f-factors. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 137–146, Oct 2013.

[11] H.N. Gabow. Scaling Algorithms for Network Problems. *J. Comput. Syst. Sci.*, 31(2):148–168, 1985.

[12] H.N. Gabow and R.E. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989.

[13] A. Goldberg and R. Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 7–18, New York, NY, USA, 1987. ACM.

[14] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 222–231. Society for Industrial and Applied Mathematics, 1993.

[15] M. Iri. A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3:27–87, 1960.

[16] L.R. Ford Jr. Network Flow Theory. Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[17] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen 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.

[18] Ioannis Koutis, Gary L. Miller, and Richard 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.

[19] Ioannis Koutis, Gary L. Miller, and Richard 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.

[20] H.W Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[21] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *STOC'16: Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, 2016.

[22] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians: Fast, sparse, and simple. In *FOCS'16: Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*, 2016.

[23] Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in $\sqrt{rank}$ iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433, Oct 2014.

[24] Aleksander 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*, pages 253–262, 2013.

[25] E. F. Moore. The Shortest Path Through a Maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.

[26] J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of SIAM*, 5(1):32–38, 1957.

[27] Piotr Sankowski. *Algorithms – ESA 2005: 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005. Proceedings*, chapter Shortest Paths in Matrix Multiplication Time, pages 770–778. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[28] Piotr Sankowski. *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, chapter Weighted Bipartite Matching in Matrix Multiplication Time, pages 274–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[29] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Springer, 2003.

[30] A. Shimbel. Structure in Communication Nets. In *In Proceedings of the Symposium on Information Networks*, pages 199–203. Polytechnic Press of the Polytechnic Institute of Brooklyn, Brooklyn, 1955.

[31] Daniel A. Spielman and Shang-Hua Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time $O(m^{1.31})$. In *FOCS'03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 416–427, 2003.

[32] Daniel A. Spielman and Shang-Hua 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.

[33] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript, UIUC 1990. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Mineapolis.

[34] R. Yuster and U. Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 389–396, Oct 2005.