Strongly polynomial algorithm for generalized flow maximization
(Extended Abstract)

László A. Végh

Department of Management
London School of Economics and Political Science
L.Vegh@lse.ac.uk

November 11, 2013

Abstract

A strongly polynomial algorithm is given for the generalized flow maximization problem. It uses a new variant of the scaling technique, called continuous scaling. The main measure of progress is that within a strongly polynomial number of steps, an arc can be identified that must be tight in every dual optimal solution, and thus can be contracted.

The full version is available on arXiv:1307.6809.

The generalized flow model is a classical extension of network flows. Besides the capacity constraints, for every arc e there is a gain factor $\gamma_e > 0$, such that flow amount gets multiplied by γ_e while traversing the arc e. We study the flow maximization problem, where the objective is to send the maximum amount of flow to a sink node t. The model was already formulated by Kantorovich [17], as one of the first examples of linear programming; it has several applications in operations research [2, Chapter 15]. Gain factors can be used to model physical changes such as leakage or theft. Other common applications use the nodes to represent different types of entities, e.g. different currencies, and the gain factors correspond to the exchange rates.

The existence of a strongly polynomial algorithm for linear programming is a major open question from a theoretical perspective. This refers to an algorithm with the number of arithmetic operations polynomially bounded in the number of variables and constraints, and the size of the numbers during the computations polynomially bounded in the input size. The landmark result by Tardos [27] gives an algorithm with the running time dependent only on the size of numbers in the constraint matrix, but independent from the right-hand side and the objective vector. This gives strongly polynomial algorithms for several combinatorial problems such as minimum cost flows (see also Tardos [26]) and multicommodity flows.

Instead of the sizes of numbers, one might impose restrictions on the structure of the constraint matrix. Hence a natural question arises whether there exists a strongly polynomial algorithm for linear programs (LPs) with at most two nonzero entries per column (that can be arbitrary numbers). This question is still open; as shown by Hochbaum [15], all such LPs can be polynomially transformed to instances of the minimum cost generalized flow problem. (Note also that every LP can be polynomially transformed to an equivalent one with at most three nonzero entries per column.)

Generalized flow maximization is an important special case of minimum cost generalized flows; it is probably the simplest natural class of LPs where no strongly polynomial algorithm has been known. The existence of such an algorithm has been a well-studied and longstanding open problem

(see e.g. [8, 3, 31, 23, 25]) A strongly polynomial algorithm for the corresponding dual feasibility problem was given by Megiddo [19], but this is not applicable to flow maximization. A strongly polynomial algorithm for some restricted classes of generalized flow problems was given by Adler and Cosares [1].

In this paper, we exhibit a strongly polynomial algorithm for generalized flow maximization. Let n denote the number of nodes and m the number of arcs in the network, and let B denote the largest integer used in the description of the input (see Section 1 for the precise problem setting). A strongly polynomial algorithm for the problem entails the following (see [14]): *(i)* it uses only elementary arithmetic operations (addition, subtraction, multiplication, division), and comparisons; *(ii)* the number of these operations is bounded by a polynomial of n and m; *(iii)* if all numbers in the input are rational, then all numbers occurring in the computations are rational numbers of encoding size polynomially bounded in n, m and B. In this extended abstract we outline a simpler version of our algorithm that satisfies requirements (i) and (ii) only, using a model with real number computations, ignoring the encoding sizes. In order to satisfy (iii) as well, additional rounding steps have to be introduced; this is done in Section 7 of the full version.

Combinatorial approaches have been applied to generalized flows already in the sixties by Dantzig [4] and Jewell [16]. However, the first polynomial-time combinatorial algorithm was only given in 1991 by Goldberg, Plotkin and Tardos [8]. This was followed by a multitude of further combinatorial algorithms e.g. [3, 10, 12, 28, 6, 11, 13, 31, 23, 24, 30]; a central motivation of this line of research was to develop a strongly polynomial algorithm. The algorithms of Cohen and Megiddo [3], Wayne [31], and Restrepo and Williamson [24] present fully polynomial time approximation schemes, that is, for every $\varepsilon > 0$, they can find a solution within ε from the optimum value in running time polynomial in n, m and $\log(1/\varepsilon)$. This can be transformed to an optimal solution for a sufficiently small ε; however, this value does depend on B and hence the overall running time will also depend on $\log B$. The current most efficient weakly polynomial algorithms are the interior point approach of Kapoor an Vaidya [18] with running time $O(m^{1.5}n^2 \log B)$, and the combinatorial algorithm by Radzik [23] with running time $\tilde{O}(m^2 n \log B)$.[1] For a survey on combinatorial generalized flow algorithms, see Shigeno [25].

The generalized flow maximization problem exhibits deep structural similarities to the minimum cost circulation problem, as first pointed out by Truemper [29]. Most combinatorial algorithms for generalized flows, including both algorithms by Goldberg et al. [8], exploit this analogy and adapt existing efficient techniques from minimum cost circulations. For the latter problem, several strongly polynomial algorithms are known, the first given by Tardos [26]; others relevant to our discussion are those by Goldberg and Tarjan [9], and by Orlin [21]; see also [2, Chapters 9-11]. Whereas these algorithms serve as starting points for most generalized flow algorithms, the applicability of the techniques is by no means obvious, and different methods have to be combined. As a consequence, the strongly polynomial analysis cannot be carried over when adapting minimum cost circulation approaches to generalized flows, although weakly polynomial bounds can be shown. To achieve a strongly polynomial guarantee, further new algorithmic ideas are required that are specific to the structure of generalized flows. The new ingredients of our algorithm are highlighted in Section 1.3.

Let us now outline the scaling method for minimum cost circulations, a motivation of our generalized flow algorithm. The first (weakly) polynomial time algorithm for minimum cost circulations was given by Edmonds and Karp [5], introducing the simple yet powerful idea of scaling (see also [2, Chapter 9.7]). The algorithm consists of Δ-phases, with the value of $\Delta > 0$ decreasing by a factor of at least two between every two phases, yielding an optimal solution for sufficiently small Δ. In the Δ-phase, the flow is transported in units of Δ from nodes with excess to nodes with deficiency using shortest paths in the graph of arcs with residual capacity at least Δ. Orlin [21], (see also

[1] The $\tilde{O}()$ notation hides a polylogarithmic factor.

[2, Chapters 10.6-7]) devised a strongly polynomial version of this algorithm. The key notion is that of "abundant arcs". In the Δ-phase of the scaling algorithm [5], the arc e is called *abundant* if it carries $> 4n\Delta$ units of flow. For such an arc e, it can be shown that $x^*_e > 0$ must hold for some optimal solution x^*. By primal-dual slackness, the corresponding constraint must be tight in every dual optimal solution. Based on this observation, Orlin [21] shows that such an arc can be contracted; the scaling algorithm is then restarted on the smaller graph. This enables to obtain a dual optimal solution in strongly polynomial time; that provided, a primal optimal solution can be found via a single maximum flow computation. Orlin [21] also presents a more sophisticated and efficient implementation of this idea.

Let us now turn to generalized flows. The analogue of the scaling method was an important component of the Fat-Path algorithm of [8]; the algorithm of Goldfarb, Jin and Orlin [12] and the one in [30] also use this technique. The notion of "abundant arcs" can be easily extended to these frameworks: if an arc e carries a "large" amount of flow as compared to Δ, then it must be tight in every dual optimal solution, and hence can be contracted. This idea was already used by Radzik [23], to boost the running time of [12]. Nevertheless, it is not known whether an "abundant arc" would always appear in any of the above algorithms within a strongly polynomial number of steps.

Our contribution is a new type of scaling algorithm that suits better the dual structure of the generalized flow problem, and thereby the quick appearance of an "abundant arc" will be guaranteed. Whereas in all previous methods, the scaling factor Δ remains constant for a linear number of path augmentations, our *continuous scaling method* keeps it decreasing in every elementary iteration of the algorithm, even in those that lead to finding the next augmenting path.

The rest of this extended abstract is structured as follows. Section 1 first defines the problem setting, introduces relabelings, gives the characterization of optimality, and defines the notion of Δ-feasibility. Section 1.3 then gives a more detailed account of the main algorithmic ideas. In Section 2 we first describe a simpler weakly polynomial version of our algorithm, whereas a sketch of the strongly polynomial algorithm is given in Section 3.

1 Preliminaries

Let $G = (V, E)$ be a directed graph with a designated sink node $t \in V$. Let $n = |V|$, $m = |E|$, and for each node $i \in V$, let d_i denote total number of arcs incident to i (both entering and leaving). We will always assume $n \leq m$. We do not allow parallel arcs and hence we may use ij to denote the arc from i to j. This is for notational convenience only, and all result straightforwardly extend to the setting with parallel arcs. For an arc set $F \subseteq E$ and node set $S \subseteq V$, let $F[S] := \{ij \in F : i, j \in S\}$ denote the set of arcs in F spanned in S. All paths and cycles in the paper will refer to directed paths and directed cycles.

The following is the *standard formulation* of the problem. Let us be given arc capacities $u : E \to \mathbb{Q}_{>0}$ and gain factors $\gamma : E \to \mathbb{Q}_{>0}$.

$$\begin{aligned} \max \sum_{j:jt\in E} \gamma_{jt} f_{jt} - \sum_{j:tj\in E} f_{tj} & \\ \sum_{j:ji\in E} \gamma_{ji} f_{ji} - \sum_{j:ij\in E} f_{ij} \geq 0 \quad & \forall i \in V - t \qquad (P_u) \\ 0 \leq f \leq u & \end{aligned}$$

It is common in the literature to define the problem with equalities in the node constraints. The two forms are essentially equivalent, see e.g. [25]; moreover, the form with equality is often solved via a reduction to (P_u). In this paper, we prefer to use yet another equivalent formulation, where all arc

capacities are unbounded, but there are node demands instead. A problem given in the standard formulation can be easily transformed to an equivalent instance in this form; the transformation is described in Section 8.1 of the full version. Given a node demand vector $b : V - t \to \mathbb{Q}$ and gain factors $\gamma : E \to \mathbb{Q}_{>0}$, the *uncapacitated formulation* is defined as

$$\begin{aligned} \max \sum_{j:jt\in E} \gamma_{jt} f_{jt} - \sum_{j:tj\in E} f_{tj} & \\ \sum_{j:ji\in E} \gamma_{ji} f_{ji} - \sum_{j:ij\in E} f_{ij} \geq b_i \quad & \forall i \in V - t \qquad (P) \\ 0 \leq f & \end{aligned}$$

For a vector $f \in \mathbb{R}_+^{|E|}$, let us define the *excess* of a node $i \in V$ by

$$e_i(f) := \sum_{j:ji\in E} \gamma_{ji} f_{ji} - \sum_{j:ij\in E} f_{ij} - b_i.$$

The node constraints in (P) can be written as $e_i(f) \geq 0$, and the objective is equivalent to maximizing $e_t(f)$. When f is clear from the context, we will denote the excess simply by $e_i := e_i(f)$. By a *generalized flow* we mean a feasible solution to (P), that is, a nonnegative vector $f \in \mathbb{R}_+^{|E|}$ with $e_i(f) \geq 0$ for all $i \in V - t$. For convenience, we define $b_t = -\infty$, or some very small value, such that $e_t(f) < 0$ must hold for every feasible f. Let us define the *surplus* of f as

$$Ex(f) := \sum_{i\in V-t} e_i(f).$$

It will be convenient to make the following assumptions; in Section 8.1 of the full version it is shown that any problem in the standard form can be transformed to an equivalent one in the uncapacitated form that also satisfies these assumptions.

$$\text{There is an arc } it \in E \text{ for every } i \in V - t; \qquad (\star)$$

$$\text{We are given an initial feasible solution } \bar{f} \text{ to } (P); \qquad (\star\star)$$

Note that for (P_u), $f \equiv 0$ is a feasible solution; $\bar{f}$ in $(\star\star)$ will be the image of 0 under the transformation. Our main result is the following.

Theorem 1.1. *There exists a strongly polynomial algorithm for the uncapacitated formulation* (P) *with running time* $O(n^3m^2)$.

This in turn gives an $O(m^5)$ time strongly polynomial algorithm for the standard formulation (P_u).

1.1 Labelings and optimality conditions

Dual solutions to (P) play a crucial role in the entire generalized flow literature. Let $\lambda : V \to \mathbb{R}_+$ be a solution to the dual of (P). Following Glover and Klingman [7], the literature standard is not to consider the λ values but their inverses instead. With $\mu_i := 1/\lambda_i$, we can write the dual of (P) in

the following form.

$$\begin{aligned} \max & \sum_{i \in V} \frac{b_i}{\mu_i} \\ \gamma_{ij}\mu_i &\leq \mu_j \quad \forall ij \in E \\ \mu_i &> 0 \quad \forall i \in V - t \\ \mu_t &= 1 \\ \mu &\in (\mathbb{R}_+ \cup \{\infty\})^{|V|} \end{aligned} \tag{D}$$

A feasible solution μ to this program will be called a *relabeling* or *labeling*. An *optimal labeling* is an optimal solution to (D). Under assumption $(\star)$, all μ_i values must be finite. A useful and well-known property is the following.

Proposition 1.2. *Given an optimal solution to (D), an optimal solution to (P) can be obtained in strongly polynomial time, and conversely, given an optimal solution to (P), an optimal solution to (D) can be obtained in strongly polynomial time.*

In fact, our strongly polynomial algorithm will proceed via finding an optimal solution to (D), and computing the primal optimal solution via a single maximum flow computation. Relabelings will be used in all parts of the algorithm and proofs. For a generalized flow f and a labeling μ, we define the relabeled flow f^μ by $f^\mu_{ij} := \frac{f_{ij}}{\mu_i}$ for all $ij \in E$. This can be interpreted as changing the base unit of measure at the nodes (i.e. in the example of the currency exchange network, it corresponds to changing the unit from pounds to pennies). To get a problem setting equivalent to the original one, we have to relabel all other quantities accordingly. That is, we define relabeled gains, demands, excesses and surplus by

$$\gamma^\mu_{ij} := \gamma_{ij}\frac{\mu_i}{\mu_j}, \quad b^\mu_i := \frac{b_i}{\mu_i}, \quad e^\mu_i := \frac{e_i}{\mu_i}, \text{ and } \quad Ex^\mu(f) := \sum_{i \in V-t} e^\mu_i,$$

respectively. Another standard notion is the *residual network* $G_f = (V, E_f)$ of a generalized flow f, defined as

$$E_f := E \cup \{ij : ji \in E, f_{ji} > 0\}.$$

Arcs in E are called *forward arcs*, while arcs in the second set are *reverse arcs*. For a forward arc ij, let γ_{ij} be the same as in the original graph. For a reverse arc ji, let $\gamma_{ji} := 1/\gamma_{ij}$. Also, we define $f_{ji} := -\gamma_{ij} f_{ij}$ for every reverse arc $ji \in E_f$. By increasing (decreasing) f_{ji} by α on a reverse arc $ji \in E_f$, we mean decreasing (increasing) f_{ij} by α/γ_{ij}.

The crucial notion of conservative labelings is motivated by primal-dual slackness. Let f be a generalized flow (that is, a feasible solution to (P)), and let $\mu : V \to \mathbb{R}_{>0}$. We say that μ is a *conservative labeling* for f, if μ is a feasible solution to (D) with the further requirement that $\gamma^\mu_{ij} = 1$ whenever $f_{ij} > 0$ for $ij \in E$. The following characterization of optimality is a straightforward consequence of primal-dual slackness in linear programming.

Theorem 1.3. *Assume $(\star)$ holds. A generalized flow f is an optimal solution to (P) if and only if there exists a conservative labeling μ that such that $e_i = 0$ for all $i \in V - t$.*

Given a labeling μ, we say that an arc $ij \in E_f$ is *tight* if $\gamma^\mu_{ij} = 1$. A directed path in E_f is called *tight* if it consists of tight arcs.

1.2 Δ-feasible labels

Let us now introduce a relaxation of conservativity crucial in the algorithm. This is new notion, although similar concepts have been used in previous scaling algorithms [10, 30]. Section 1.3 explains the background and motivation of this notion. Given a labeling μ, let us call arcs in E with $\gamma_{ij}^{\mu} < 1$ *non-tight*, and denote their sets by

$$F^{\mu} := \{ij \in E : \gamma_{ij}^{\mu} < 1\}.$$

For every $i \in V$, let

$$R_i := \sum_{j:ji \in F^{\mu}} \gamma_{ji} f_{ji}$$

denote the total flow incoming on non-tight arcs; let $R_i^{\mu} := \frac{R_i}{\mu_i} = \sum_{j:ji \in F^{\mu}} \gamma_{ji}^{\mu} f_{ji}^{\mu}$. For some $\Delta \geq 0$, let us define the Δ-*fat graph* as

$$E_f^{\mu}(\Delta) = E \cup \{ij : ji \in E, f_{ji}^{\mu} > \Delta\}.$$

We say that μ is a Δ-*conservative labeling* for f, or that (f, μ) is a Δ-*feasible pair*, if

- $\gamma_{ij}^{\mu} \leq 1$ holds for all $ij \in E_f^{\mu}(\Delta)$, and
- $\mu_t = 1$, and $\mu_i > 0$, $e_i \geq R_i$ for every $i \in V - t$.

Note that in particular, μ must be a feasible solution to (D). The first condition is equivalent to requiring $f_{ij}^{\mu} \leq \Delta$ for every non-tight arc. Note that 0-conservativeness is identical to conservativeness: $E_f^{\mu}(\Delta) = E_f^{\mu}$, and therefore every arc carrying positive flow must be tight; the second condition simply gives $e_i \geq 0$ whenever $\mu_i > 0$. The next lemma can be seen as the converse of this observation.

Lemma 1.4. *Let (f, μ) be a Δ-feasible pair for some $\Delta > 0$. Let us define the generalized flow $\tilde{f}$ with $\tilde{f}_{ij} = 0$ if $ij \in F^{\mu}$ and $\tilde{f}_{ij} = f_{ij}$ otherwise. Then μ is a conservative labeling for $\tilde{f}$, and $Ex^{\mu}(\tilde{f}) \leq Ex^{\mu}(f) + |F^{\mu}|\Delta$.*

Claim 1.5. *In a Δ-conservative labeling, $R_i^{\mu} < d_i \Delta$ holds for every $i \in V$.*

1.3 Overview of the algorithms

We now informally describe some fundamental ideas of our algorithms Continuous Scaling and Enhanced Continuous Scaling, and explain their relations to previous generalized flow algorithms.

Basic features of the algorithms

By sending α units of flow on a walk $P = w_1 w_2 \dots w_t \subseteq E_f$, we mean increasing the flow on w_1 by $p_1 = \alpha$, and increasing the flow on w_i by $p_i = \gamma_{w_{i-1}} p_{i-1}$ for all $1 < i \leq t$ (assuming that the residual capacities do not get violated). Note that if P is a cycle then this will change w_i on precisely one node i.

Given a generalized flow f, a cycle C in the residual graph E_f is called *flow generating*, if $\gamma(C) = \prod_{e \in C} \gamma_e > 1$. If there exists a flow generating cycle, then some positive amount of flow can be sent around it to create positive excess in an arbitrary node i incident to C.

The notion of conservative labellings is closely related to flow generating cycles. Notice that for an arbitrary labeling μ, $\gamma(C) = \gamma^{\mu}(C)$. Therefore, if μ is a conservative labeling, then there

cannot be any flow generating cycle in E_f. It is also easy to verify the converse: if there are no flow generating cycles, then there exists a conservative labeling.

The Maximum-mean-gain cycle-canceling procedure, introduced in [8], can be used to eliminate all flow generating cycles efficiently. The subroutine proceeds by choosing a cycle $C \subseteq E_f$ maximizing $\gamma(C)^{1/|C|}$, and from an arbitrary node i incident to C, sending the maximum possible amount of flow around C admitted by the capacity constraints, thereby increasing the excess e_i. It terminates once there are no more flow generating cycles left in E_f. This is a natural analogue of the minimum mean cycle cancellation algorithm of Goldberg and Tarjan [9] for minimum cost circulations. Radzik [22] (see also [25]) gave a strongly polynomial running time bound $O(m^2 n \log^2 n)$ for the Maximum-mean-gain cycle-canceling algorithm.

Our algorithm also starts with performing this algorithm, with the input being the initial solution $\bar{f}$ provided by $(\star\star)$. Hence one can obtain a feasible solution f along with a conservative labeling μ in strongly polynomial time.

Such an f can be transformed to an optimal solution using Onaga's algorithm [20]: while there exists a node $i \in V - t$ with $e_i > 0$, find a *highest gain augmenting path* from i to t, that is, a path P in the residual graph E_f with the product of the gains maximum. Send the maximum amount of flow on this augmenting path enabled by the capacity constraints. A conservative labeling can be used to identify such paths: we can transform a conservative labeling to a *canonical labeling* (see [8]), where every node i is connected to the sink via a tight path. Such a canonical labeling can be found via a Dijkstra-type algorithm, increasing the labels of certain nodes. The correctness of Onaga's algorithm follows by the observation that sending flow on a tight path maintains the conservativeness of the labeling, hence no new flow generating cycles may appear.

Unfortunately, Onaga's algorithm may run in exponentially many steps, and might not even terminate if the input is irrational. The Fat-Path algorithm [8] introduces a scaling technique to overcome this difficulty. The algorithm maintains a scaling factor Δ that decreases geometrically. In the Δ-phase, flow is sent on a highest gain "Δ-fat" augmenting path, that is, a highest gain path among those that have sufficient capacity to send Δ units of flow to the sink. However, this might create new flow generating cycles, that have to be cancelled by calling the cycle-canceling subroutine at the beginning of every phase.

Our notion of Δ-feasible pairs in Section 1.2 is motivated by the idea of Δ-fat paths: note that every arc in the Δ-fat graph $E_f(\Delta)$ has sufficient capacity to send Δ units of relabeled flow. A main step in our algorithm will be sending Δ units of relabeled flow on a tight path in $E_f(\Delta)$ from a node with "high" excess to the sink t or another node with "low" excess. This is in contrast to Fat-Path and most other algorithms, where these augmenting paths always terminate in the sink t. We allow other nodes as well in order to guarantee that the conditions $e_i \geq R_i$ are maintained during the algorithm. The purpose of these conditions is to make sure that we always stay "close" to a conservative labeling: recall Lemma 1.4 asserting that if (f, μ) is a Δ-feasible pair, then if we set the flow values to 0 on every non-tight arc, the resulting $\tilde{f}$ is a feasible solution to (P) not containing any flow generating cycles. That is the reason why we need to call the cycle-canceling algorithm only once, at the initialization, in contrast to Fat-Path.

Similar ideas have been already used previously. The algorithm of Goldfarb, Jin and Orlin [10] also uses a single initial cycle-canceling and then performs highest-gain augmentations in a scaling framework, combined with a clever bookkeeping on the arcs. The algorithm in [30] does not perform any cycle cancellations and uses a homonymous notion of Δ-conservativeness that is closely related to ours; however, it uses a different problem setup (called "symmetric formulation"), and includes a condition stronger than $e_i \geq R_i$.

The way to the strongly polynomial bound

The basic principle of our strongly polynomial algorithm is motivated by Orlin's strongly polynomial algorithm for minimum cost circulations ([21], see also [2, Chapters 10.6-7]). The true purpose of the algorithm will be to compute a dual optimal solution to (D). Provided a dual optimal solution, we can compute a primal optimal solution to (P) by a single maximum flow computation on the network of tight arcs.

The main measure of progress will be identifying an arc $ij \in E$ that must be tight in every dual optimal solution. Such an arc can be contracted, and an optimal dual solution to the contracted instance can be easily extended to an optimal dual solution on the original instance. The algorithm can be simply restarted from scratch in the contracted instance. Our algorithm Enhanced Continuous Scaling is somewhat more complicated and keeps the previous primal solution to achieve better running time bounds by a global analysis of all contraction phases.

We use a scaling-type algorithm to identify such arcs tight in every dual optimal solution. Our algorithm always maintains a scaling parameter Δ, and a Δ-feasible pair (f, μ) such that $Ex^{\mu}(f) \leq 16m\Delta$. Using standard flow decomposition techniques, it can be shown that an arc ij with $f^{\mu}_{ij} \geq 17m\Delta$ must be positive in some optimal solution f^* to (P). Then by primal-dual slackness it follows that this arc is tight in every dual optimal solution. Arcs with $f^{\mu}_{ij} \geq 17m\Delta$ will be called *abundant*.

A simple calculation shows that once $|b^{\mu}_i| \geq 20mn\Delta$ for a node $i \in V - t$, there must be an abundant arc leaving or entering i. Hence our goal is to design an algorithm where such a node appears within a strongly polynomial number of iterations.

A basic step in the scaling approaches (e.g. [8, 10, 30]) is sending Δ units of relabeled flow on a tight path; we shall call this a path augmentation. In all previous approaches, the scaling factor Δ remained fixed for a number of path augmentations, and reduces by a substantial amount (by at least a factor of two) for the next Δ-phase. Our main idea is what we call *continuous scaling*: the boundaries between Δ-phases are dissolved, and the scaling factor decreases continuously, even during the iterations that lead to finding the next path for augmentation. We now give a high-level overview.

We shall have a set T_0 with nodes of "high" relabelled excess; another set N will be the set of nodes with "low" relabelled excess, always including the sink t. We will look for tight paths connecting a node in T_0 to one in N; we will send Δ units of relabeled flow along such a path. In an intermediate elementary step, we let T to denote the set of nodes reachable from T_0 on a tight path; if it does not intersect N, then we increase the labels μ_i for all $i \in T$ by the same factor α hoping that a new tight arc appears between T and $V \setminus T$, and thus T can be extended. We simultaneously decrease the value of Δ by the same factor α. Thus the relabeled excess of nodes in $V \setminus T$ increases relative to Δ. This might lead to changes in the sets T_0 and N; hence an elementary step does not necessarily terminate when a new tight arc appears, and therefore the value of α has to be carefully chosen.

This framework is undoubtedly more complicated than the traditional scaling algorithms. The main reason for this approach is the phenomenon one might call "inflation" in the previous scaling-type algorithms. There it might happen that the relabeling steps used for identifying the next augmenting paths increase some labels by very high amounts, and thus the relabeled flow remains small compared to Δ on every arc of the network - therefore a new abundant arc can never be identified. It could even be the case that most Δ-scaling phases do not perform any path augmentations at all, but only label updates: the relabeled excess at every node becomes smaller than Δ during the relabeling steps.[2]

[2]However, to the extent of the author's knowledge, no actual examples are known for these phenomena in any of the algorithms.

The advantage of changing Δ continuously in our algorithm is that the ratios $\frac{|b_i^\mu|}{\Delta}$ are nondecreasing for every $i \in V - t$ during the entire algorithm. In the above described situation, these ratios are unchanged for $i \in T$ and increase for $i \in V \setminus T$. As remarked above, there must be an abundant arc incident to i once $\frac{|b_i^\mu|}{\Delta} \geq 20mn$.

We first present a simpler version of this algorithm, CONTINUOUS SCALING, proving only a weakly polynomial running time bound. Whereas the ratios $\frac{|b_i^\mu|}{\Delta}$ are nondecreasing, we are not able to prove that one of them eventually reaches the level $20mn$ in a strongly polynomial number of steps. This is since the set $V \setminus T$ where the ratio increases might always consist only of nodes where $\frac{|b_i^\mu|}{\Delta}$ is very small. The algorithm ENHANCED CONTINUOUS SCALING therefore introduces one additional subroutine, called FILTRATION. In case $|b_i^\mu| < \Delta/n$ for every $i \in (V \setminus T) - t$, we "tidy-up" the flow inside $V \setminus T$, by performing a maximum flow computation here. This drastically reduces all relabeled excesses in $V \setminus T$, and thereby guarantees that most iterations of the algorithm will have to increase certain $\frac{|b_i^\mu|}{\Delta}$ values that are already at least $1/n$.

In summary, the strongly polynomiality of our algorithm is based on the following three main new ideas.

- The definition of Δ-feasible pairs, in particular, the condition on maintaining a security reserve R_i. It is a cleaner and more efficient framework than similar ones in [10] and [30]; we believe this is the "real" condition a scaling type algorithm has to maintain.
- Continuous scaling, that guarantees that the ratios $\frac{|b_i^\mu|}{\Delta}$ are nondecreasing during the algorithm. This is achieved by doing the exact opposite of [8, 10, 30] that use the natural analogue of the scaling technique for minimum cost circulations.
- The FILTRATION subroutine that intervenes in the algorithm whenever the nodes on a certain, relatively isolated part of the network have "unreasonably high" excesses as compared to the small node demands in this part.

2 The Continuous Scaling algorithm

The algorithm CONTINUOUS SCALING is shown on Figure 1. It starts with the subroutine INITIALIZE, that returns an initial flow f, along with a $\Delta = \bar{\Delta}$-conservative labeling μ such that $e_i^\mu < (d_i + 2)\Delta$ holds for every $i \in V$. This is based on the MAXIMUM-MEAN-GAIN CYCLE-CANCELING algorithm as in [8, 22], a standard method also used e.g. in [12] the same way. The termination condition depends on the parameter $\bar{B}$ bounding the input description as described in the full version. After the while loops, the exact optimum solution can be obtained via a single maximum flow computation on the residual network of tight arcs; this is also the standard method used in all previous generalized flow algorithms. The main part of the algorithm (the while loop) consists of iterations. The value of the scaling parameter Δ is monotone decreasing and all μ_i values are monotone increasing during the algorithm. In every iteration, a Δ-feasible pair (f, μ) is maintained.

The set N always denotes the set of nodes with $e_i^\mu < (d_i + 1)\Delta$, and T_0 will consist of a certain set of nodes (but not all) with $e_i^\mu \geq (d_i + 2)\Delta$. The set T will denote a set of nodes that can be reached from T_0 on a tight path in the Δ-fat graph $E_f^\mu(\Delta)$. Both T_0 and T are initialized empty. Note that $t \in N$ as we chose b_t such that $e_t < 0$ always holds.

Every iteration first checks whether $N \cap T \neq \emptyset$. If yes, then nodes $p \in T_0$ and $q \in N$ are picked connected by a tight path P in the Δ-fat graph. Δ units of relabeled flow is sent from p to q on P: that is, f_{ij} is increased by $\Delta\mu_i$ for every $ij \in P$ (if ij was a reverse arc, this means decreasing f_{ji} by $\Delta\mu_j$). The only e_i values that change are e_p and e_q. If the new value is $e_p^\mu < (d_p + 2)\Delta$, then p

Algorithm CONTINUOUS SCALING
INITIALIZE;
$T_0 \leftarrow \emptyset$; $T \leftarrow \emptyset$;
WHILE $\Delta \geq 1/(17m\bar{B}^3)$ DO
  $N \leftarrow \{i \in V : e_i^\mu < (d_i+1)\Delta\}$;
  if $N \cap T \neq \emptyset$ **then**
    pick $p \in T_0$, $q \in N$ connected by a tight path P in $E_f^\mu(\Delta)$;
    send Δ units of relabeled flow from p to q along P;
    if $e_p^\mu < (d_p+2)\Delta$ **then** $T_0 \leftarrow T_0 \setminus \{e_p\}$;
    $T \leftarrow T_0$;
  else
    if $\exists ij \in E_f^\mu(\Delta)$, $\gamma_{ij}^\mu = 1$, $i \in T$, $j \in V \setminus T$ **then** $T \leftarrow T \cup \{j\}$;
    else ELEMENTARY STEP(T);
TIGHT-FLOW(V, μ) ;

Figure 1: Description of the weakly polynomial algorithm

is removed from T_0. The iteration finishes in this case by resetting $T = T_0$ (irrespective to whether p was removed or not).

Let us now turn to the case $N \cap T = \emptyset$. If there is a node $j \in V \setminus T$ connected by a tight arc in $E_f^\mu(\Delta)$ to T, then we extend T by j, and the iteration terminates. Otherwise, the subroutine ELEMENTARY STEP(T) is called. The precise description is given in Section 3.2 of the full version; we give an outline below.

For a carefully chosen $\alpha > 1$, all μ_i values are multiplied by α for $i \in T$, and μ_i is left unchanged for $i \in V \setminus T$. At the same time, Δ is divided by α (this is the only step in the main part of the algorithm modifying the μ_i's and the value of Δ). The flow is divided by α on all non-tight arcs in $F^\mu[V \setminus T]$, and on every arc entering T. The value of α is chosen to be the largest such that the labeling remains Δ-feasible with the above changes, and further $e_i^\mu \leq 4(d_i+2)\Delta$ holds for all $i \in V \setminus T$. All nodes i for which equality holds are added both to T_0 and to T. On the other hand, the e_i^μ values might also decrease both for $i \in T$ and $i \in V \setminus T$. If for some $i \in T_0$, the value of e_i^μ drops below $(d_i+2)\Delta$, then i is removed from T_0, and T is reset to $T = T_0$. In every step when T_0 is not extended, a tight arc in $E_f^\mu(\Delta)$ leaving T must appear. Hence T will be extended in the next iteration.

We shall prove the following running time bound:

Theorem 2.1. *The algorithm* CONTINUOUS SCALING *can be implemented to find an optimal solution for the uncapacitated formulation (P) in running time* $\max\{O(m(m+n\log n)\log \bar{B}), O(m^2 n \log^2 n)\}$.

The high level idea of the analysis is the following. The e_i^μ values for nodes $i \in T_0$ are non increasing, and a path augmentation starting from i reduces e_i^μ by Δ. The node i leaves T_0 once e_i^μ drops below $(d_i+2)\Delta$, and may enter again once it increases to $4(d_i+2)\Delta$. We show that the value of Δ must decrease by at least a factor 2 between two such events. Also, it is easy to verify that within every $2n$ ELEMENTARY STEP operations, either a path augmentation must be carried out, or a node i must leave T_0 due to decrease in e_i^μ caused by label changes. These facts together lead to a polynomial bound on the running time. We can adapt the algorithm to work directly

for the standard formulation (P_u); one can obtain a running time bound $O(m^2(m + n \log n) \log B)$, matching the one by Goldfarb et al. [12].

3 The strongly polynomial algorithm

The while loop of the algorithm Enhanced Continuous Scaling (see Section 5.3 of the full version) proceeds very similarly to Continuous Scaling, with the addition of one more subroutine. However, the termination criterion is quite different. As discussed in the Introduction, the goal is to find a node $i \in V - t$ with $\frac{|b_i^\mu|}{\Delta} \geq 20mn$. There must be an abundant arc incident to such a node that we can contract and restart the algorithm in the smaller graph. (Instead of restarting from scratch, the actual implementation of Enhanced Continuous Scaling in the full version uses the flow f obtained before the contraction to achieve better running time bounds.) We shall define the set

$$D := \left\{ i \in V - t : \frac{|b_i^\mu|}{\Delta} \geq \frac{1}{n} \right\},$$

and modify the algorithm in order to guarantee that most iterations when Δ is multiplied by α will multiply $\frac{|b_i^\mu|}{\Delta}$ by α for some $i \in D$. This will ensure that $\frac{|b_i^\mu|}{\Delta} \geq 20mn$ happens within $O(nm \log n)$ number of steps. Note that in the subroutine Elementary step(T), the $\frac{|b_i^\mu|}{\Delta}$ ratio is multiplied by α for all nodes $i \in V \setminus T$ and remains unchanged for $i \in T$.

Therefore we modify the while loop of Continuous Scaling as follows. If $(V \setminus T) \cap D \neq \emptyset$, Elementary step$(T)$ is performed identically. If $(V \setminus T) \cap D = \emptyset$, then before Elementary step(T), the special subroutine Filtration$(V \setminus T)$ is executed, as outlined below(see Section 5.2 of the full version).

The value of f is set to 0 for every arc entering T, and f_{ij} is left unchanged for $i \in T$. The flow value on arcs inside $E[V \setminus T]$ is replaced by an entirely new flow f' computed by a maximum flow subroutine as follows.

Add a new source node s to $V \setminus T$, and add an arc si from s to every $i \in (V \setminus T) - t$; let E' denote the union of these new arcs and the tight arcs in $E[V \setminus T]$ with respect to μ. Set lower and upper arc capacities $\ell_{si} := -\infty$ and $u_{si} := -b_i^\mu$ for all $i \in (V \setminus T) - t$, and $\ell_{ij} := 0$, $u_{ij} := \infty$ on all other arcs. Compute a maximum flow x from s to t on the network $((V \setminus T) \cup \{s\}, E')$ with capacities ℓ and u. Define $f' : E \to \mathbb{R}_+$ by $f'_{ij} := x_{ij}\mu_i$ if $ij \in E'$ and $f'_{ij} := 0$ otherwise.

The next lemma plays an important role in the analysis.

Lemma 3.1. *The above flow problem is feasible, and μ is a conservative labeling for f' on $V \setminus T$. Further,*

$$e_i^\mu(f') \leq n \max_{j \in (V \setminus T) - t} |b_j^\mu| \quad \forall i \in V \setminus T.$$

Based on this lemma, we can show the following. Either the set D must be extended in the iteration following Filtration$(V \setminus T)$, or a path augmentation must be performed within the next two iterations. Note that once a node enters D, it stays there forever. Together with the argument on the running time bound of Continuous Scaling, this enables us to show that an abundant arc can be found in $O(nm \log n)$ number of steps.

4 Conclusion

We have given a strongly polynomial algorithm for the generalized flow maximization problem. A natural next question is to address the minimum cost generalized flows. As noted in the Introduction, this problem is equivalent to solving LPs with two nonzero entries per column (see Hochbaum [15]).

In contrast to the vast literature on the flow maximization problem, there is only one weakly polynomial combinatorial algorithm known for this setting, the one by Wayne [31]. This setting is more challenging since the dual structure cannot be characterized via the convenient relabelling framework, and thereby most tools for minimum cost circulations, including the scaling approach also used in this paper, become difficult if not impossible to apply.

Acknowledgment

The author is grateful to Joseph Cheriyan and Ian Post for several suggestions that helped to improve the presentation, and for pointing out a bug in the previous version.

References

[1] I. Adler and S. Cosares. A strongly polynomial algorithm for a special class of linear programs. *Operations Research*, 39(6):955–960, 1991.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., feb 1993.

[3] E. Cohen and N. Megiddo. New algorithms for generalized network flows. *Mathematical Programming*, 64(1):325–336, 1994.

[4] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, 1963.

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

[6] L. K. Fleischer and K. D. Wayne. Fast and simple approximation schemes for generalized flow. *Mathematical Programming*, 91(2):215–238, 2002.

[7] F. Glover and D. Klingman. On the equivalence of some generalized network problems to pure network problems. *Mathematical Programming*, 4(1):269–278, 1973.

[8] A. V. Goldberg, S. A. Plotkin, and É. Tardos. Combinatorial algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 16(2):351, 1991.

[9] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)*, 36(4):873–886, 1989.

[10] D. Goldfarb and Z. Jin. A faster combinatorial algorithm for the generalized circulation problem. *Mathematics of Operations Research*, 21(3):529–539, 1996.

[11] D. Goldfarb, Z. Jin, and Y. Lin. A polynomial dual simplex algorithm for the generalized circulation problem. *Mathematical Programming*, 91(2):271–288, 2002.

[12] D. Goldfarb, Z. Jin, and J. B. Orlin. Polynomial-time highest-gain augmenting path algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 22(4):793–802, 1997.

[13] D. Goldfarb and Y. Lin. Combinatorial interior point methods for generalized network flow problems. *Mathematical Programming*, 93(2):227–246, 2002.

[14] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimizations*. Springer-Verlag, 1993.

[15] D. S. Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Operations Research Letters*, 32(1):49–58, 2004.

[16] W. S. Jewell. Optimal flow through networks. *Operations Research*, 10:476–499, 1962.

[17] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Publication House of the Leningrad State University*, page 68, 1939. English translation in *Management Science* 6(4):366-422, 1960.

[18] S. Kapoor and P. M. Vaidya. Speeding up Karmarkar's algorithm for multicommodity flows. *Mathematical Programming*, 73(1):111–127, 1996.

[19] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, 12(2):347–353, 1983.

[20] K. Onaga. Optimum flows in general communication networks. *Journal of the Franklin Institute*, 283(4):308–327, 1967.

[21] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[22] T. Radzik. Approximate generalized circulation. *Technical Report93-2, Cornell Computational Optimization Project, Cornell University*, 1993.

[23] T. Radzik. Improving time bounds on maximum generalised flow computations by contracting the network. *Theoretical Computer Science*, 312(1):75–97, 2004.

[24] M. Restrepo and D. P. Williamson. A simple GAP-canceling algorithm for the generalized maximum flow problem. *Mathematical Programming*, 118(1):47–74, 2009.

[25] M. Shigeno. A survey of combinatorial maximum flow algorithms on a network with gains. *Journal of the Operations Research Society of Japan*, 47:244–264, 2004.

[26] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[27] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, pages 250–256, 1986.

[28] É. Tardos and K. D. Wayne. Simple maximum flow algorithms in lossy networks. In *Proceedings of IPCO, Lecture Notes in Computer Science*, volume 1412, pages 310–324, 1998.

[29] K. Truemper. On max flows with gains and pure min-cost flows. *SIAM Journal on Applied Mathematics*, 32(2):450–456, 1977.

[30] L. A. Végh. Concave generalized flows with applications to market equilibria. *Mathematics of Operations Research*, 2014. (to appear).

[31] K. D. Wayne. A polynomial combinatorial algorithm for generalized minimum cost flow. *Mathematics of Operations Research*, pages 445–459, 2002.