
Proceedings Paper:
Feldmann, A.E. orcid.org/0000-0001-6229-5332, Könemann, J., Pashkovich, K. et al. (1 more author) (2016) Fast approximation algorithms for the generalized survivable network design problem. In: Hong, S., (ed.) Leibniz International Proceedings in Informatics, LIPIcs. 27th International Symposium on Algorithms and Computation (ISAAC 2016), 12-14 Dec 2016, Sydney, Australia. Leibniz International Proceedings in Informatics, 64 . Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik , Dagstuhl, Germany , 33.1-33.12. ISBN 978-3-95977-026-2

https://doi.org/10.4230/LIPIcs.ISAAC.2016.33

Fast Approximation Algorithms for the Generalized Survivable Network Design Problem

Andreas Emil Feldmann*[1], Jochen Könemann[2], Kanstantsin Pashkovich[3], and Laura Sanità[4]

1 **Dept. of Applied Mathematics (KAM), Charles University, Prague, Czech Republic**
feldmann.a.e@gmail.com
2 **Dept. of Combinatorics and Optimization, University of Waterloo, Canada**
jochen@uwaterloo.ca
3 **Dept. of Combinatorics and Optimization, University of Waterloo, Canada**
kpashkovich@uwaterloo.ca
4 **Dept. of Combinatorics and Optimization, University of Waterloo, Canada**
laura.sanita@uwaterloo.ca

Abstract

In a standard *f-connectivity* network design problem, we are given an undirected graph $G = (V, E)$, a *cut-requirement function* $f : 2^V \to \mathbb{N}$, and non-negative costs $c(e)$ for all $e \in E$. We are then asked to find a minimum-cost vector $x \in \mathbb{N}^E$ such that $x(\delta(S)) \geq f(S)$ for all $S \subseteq V$. We focus on the class of such problems where f is a *proper* function. This encodes many well-studied NP-hard problems such as the *generalized survivable network design* problem.

In this paper we present the first *strongly polynomial time* FPTAS for solving the LP relaxation of the standard IP formulation of the f-connectivity problem with general proper functions f. Implementing Jain's algorithm, this yields a strongly polynomial time $(2 + \epsilon)$-approximation for the generalized survivable network design problem (where we consider *rounding up* of rationals an arithmetic operation).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization

Keywords and phrases strongly polynomial runtime, generalized survivable network design, primal-dual method

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.33

1 Introduction

The input to a typical *network design* problem consists of a directed or undirected graph $G = (V, E)$, non-negative unit-capacity *installation costs* $c(e)$ for all $e \in E$, and a collection of *connectivity requirements* among the vertices in V. The goal is then to find a minimum-cost capacity installation in G that satisfies the connectivity requirements. The above abstract problem class captures many practically relevant optimization problems, many of which are NP-hard. Therefore, maybe not surprisingly, there has been a tremendous amount of research in the area of approximation algorithms for network design problems throughout

* Supported by project CE-ITI (GAČR no. P202/12/G061) of the Czech Science Foundation.

the last four decades (e.g., see [17, 18]).

$$\begin{aligned} \min \ & \sum_{e \in E} c(e)x(e) && \text{(IP)} \\ \text{s.t. } & x(\delta(S)) \geq f(S) \ \ \forall S \subset V \\ & x \geq \not\vdash, x \text{ integer.} \end{aligned}$$

Connectivity requirements can be modelled in many ways, but we will adopt the *f-connectivity* viewpoint in this paper. Here, one is given a *cut-requirement function* $f : 2^V \to \mathbb{N}$, and one wants to find a minimum-cost non-negative integer vector x such that for each $S \subseteq V$, the sum of variables $x(e)$ for edges e crossing the cut S is at least $f(S)$. In other words, we are interested in problems that can be encoded by integer program (IP); here $\delta(S)$ denotes the set of edges incident to a vertex in S and a vertex outside S, and $x(\delta(S)) := \sum_{e \in \delta(S)} x(e)$.

Restricting even further, we will henceforth only be concerned with instances of (IP) where f is *proper*, that is, f satisfies the three properties of *maximality* (i.e., $f(A \cup B) \leq \max\{f(A), f(B)\}$ for all disjoint sets $A, B \subseteq V$), *symmetry* (i.e., $f(S) = f(V \setminus S)$ for all $S \subseteq V$), and $f(V) = 0$. Program (IP) with proper cut-requirement function f captures (among others) the special case where the goal is to find a minimum-cost network that has $r(u, v)$ edge-disjoint paths connecting any pair u, v of vertices (for given non-negative integer parameters r). The implicit cut-requirement function in this case is then given by $f(S) := \max_{u \in S, v \in V \setminus S} r(u, v)$ for all $S \subseteq V$.

Based on the *primal-dual* method, Goemans and Williamson [12] first gave a $2\mathcal{H}(f_{\max})$-approximation algorithm for (IP) with proper cut-requirement functions where one is allowed to pick edges multiple times in the solution. Goemans et al. [11] later obtained the same performance ratio for the setting where multiple copies of edges are not allowed. More recently, in a breakthrough result, Jain [19] obtained a 2-approximation for the more general class of *skew-supermodular* cut-requirement functions based on *iterative rounding*.

$$\begin{aligned} \min \ & \sum_{e \in E} c(e)x(e) && \text{(LP}_1\text{)} \\ \text{s.t. } & x(\delta(S)) \geq f(S) - z(\delta(S)) \ \ \forall S \subseteq V \\ & x(e) = 0 \ \ \forall e \in I \\ & x \geq \not\vdash \end{aligned}$$

Jain's algorithm iteratively fixes the value of a subset of variables in (IP). To aid this, he first defines a slightly more general version of the IP, where the value of certain variables is fixed. Specifically, for a set $I \subseteq E$ of edges, assume that the value of variable $x(e)$ is fixed to $z(e) \in \mathbb{N}$. The LP relaxation of the IP for the corresponding residual problem is given in (LP$_1$). Jain's key observation was that the extreme points of the feasible region of (LP$_1$) are *sparse*, and have at least one variable with value at least 1/2.

Capitalizing on this insight, his algorithm then iteratively solves $O(|V|)$ instantiations of (LP$_1$) while intermittently rounding up the values of large variables in the computed solutions. In order to solve (LP$_1$) one needs to employ the Ellipsoid method [16] together with a polynomial-time seperation oracle for the LP's constraints (see [7]).

Our work is motivated by *Open Problem 4* in Williamson and Shmoys' recent book [23] where the authors point out that solving (LP$_1$) for general (proper) functions f may be computationally quite demanding despite the fact that it can be done efficiently in a theoretical sense. The authors leave as an open problem the design of a *primal-dual* 2-approximation for the survivable network design problem. Our main result is a replacement of the Ellipsoid-based exact LP-solver calls in Jain's algorithm by approximate ones that are based on the

(in a sense) primal-dual *multiplicative-weights* method of [10]. We realize that the likely intended meaning of *primal-dual* in Williamson and Shmoys' open problem statement is the *primal-dual method for approximation algorithms* (as in [12]). However we believe that the contribution made in this paper is in line with the motivation given for Open Problem 4 in [23]: we substantially speed up LP computations in Jain's algorithm at the expense of an inconsequential loss in performance guarantee of the algorithm.

▶ **Theorem 1.** *For any $\epsilon > 0$, there is a $(1+\epsilon)$-approximation algorithm for* (LP_1) *that runs in* strongly polynomial time[1] *independently of the values of c and f.*

In contrast to our result, Jain [19] observes that the relaxations of (IP) and (LP_1) are of a *combinatorial* nature, and hence can be solved in strongly polynomial-time via Tardos' algorithm [22] *whenever* their number of variables and constraints are polynomially bounded (in the problem dimension). For example, (IP) and (LP_1) have an equivalent compact representation when $f(S) = \max_{u\in S, v\notin S} r(u,v)$ for all $S \subseteq V$. We also note that one can argue that the Ellipsoid method applied to (LP_1) and the linear relaxation of (IP) terminates in a strongly polynomial number of steps *whenever* function $f(S)$ is polynomially bounded (in the problem dimension), for all $S \subseteq V$ as this implies small encoding-length of vertices of the feasible region of (IP) and (LP_1).

To achieve the result in Theorem 1, we rely on the *multiplicative weights method* of Garg and Könemann [10] (henceforth referred to as GK). This is a natural idea as (LP_1) belongs to the class of *positive covering LPs*. As such, [10] applies to the LP dual of (LP_1). The algorithm can therefore be used to compute an approximate pair of primal and dual solutions in strongly-polynomial time as long as we are able to provide it with a strongly-polynomial time (approximation) algorithm for the so called *shortest row problem*. For (LP_1) this boils down to computing

$$\min_{\substack{f(S)-z(\delta(S))\geq 1\\ S\subseteq V}} \frac{x(\delta(S))}{f(S)-z(\delta(S))},$$

for given $x \in \mathbb{R}^E_+$ and $z \in \mathbb{Z}^E_+$, i.e., finding a corresponding set S. The above shortest row problem is solved directly by Gabow et al.'s strongly-polynomial time separation oracle for the constraints of (LP_1) (see [7]) in the case where $I = \emptyset$, and hence $z = \not\vdash$. Once $I \neq \emptyset$, Gabow et al.'s algorithm can not be used directly to give a strongly polynomial-time algorithm, and a more subtle approach is needed. In fact, in this case, we provide only a $(1+\zeta)$-approximate solution to the shortest row problem (for appropriate $\zeta > 0$). As is well-known (e.g., see [6, 10]), the exact shortest-row subroutine used in GK may be replaced by an α-approximate one, sacrificing a factor of α in the overall performance ratio of the algorithm in [10]. We obtain the following direct corollary of Theorem 1.

▶ **Corollary 2.** *Combining Theorem 1 with Jain's algorithm, we obtain a strongly polynomial-time*[2] *$(2+\varepsilon)$-approximation algorithm for* (IP)*, that does not use linear programming solvers.*

We once again stress that the above results hold for *any not necessarily bounded* proper cut-requirement function f.

---

[1] An algorithm is *strongly polynomial* if its number of arithmetic operations, i.e. the number of additions, subtractions, multiplications, divisions and comparisons, is bounded by a polynomial in the dimension of the problem (i.e., the number of data items in the input), and the length of the numbers occurring during the algorithm is bounded by a polynomial in the length of the input.

[2] If rounding up numbers is considered an arithmetic operation.

Further related work

The past 30 years have seen significant research on solving linear programs efficiently; e.g., see the work by Shahrokhi & Matula [21], Luby & Nisan [20], Grigoriadis & Khachian [14, 15], Young [24, 25], Garg & Könemann [10], Fleischer [6], and Iyengar & Bienstock [3]. We refer the reader to two recent surveys by Bienstock [2] and Arora, Hazan & Kale [1].

Particularly relevant to this paper is the work by Fleischer [5] who previously proposed a Lagrangian-based approximation scheme for positive covering LPs with added *variable upper bounds*. Their algorithm builds on [10] and [6], and achieves a performance ratio of $(1+\epsilon)$ for any positive ϵ using $O(\epsilon^{-2}m\log(Cm))$ calls to a separation oracle for the given covering problem; here m denotes the number of variables, and C is bounded by the maximum objective function coefficient. Garg and Khandekar [9] later addressed the same problem, and presented an improved algorithm with $O(m\epsilon^{-2}\log m+\min\{n,\log\log C\})$ calls to an oracle for the most violated constraint.

The algorithms in [5, 9] naturally apply to solving LP relaxations of various network design IPs where the multiplicity $x(e)$ of each edge e is limited to some given upper bound. As the approaches in [5, 9] need to approximate the same type of the shortest row problem, as an immediate corollary of our result, we obtain a strongly polynomial-time $(2+\varepsilon)$-approximation algorithm for (IP) with constant upper bounds on the variables. This captures in particular the interesting case in which we have *binary* constraints for x.

Finally, we also mention the work of Garg & Khandekar [8] who present a fully polyonimal-time approximation algorithm for the fractional *Steiner forest* problem. The algorithm also applies to the more general problem of finding a minimum-cost fractional hitting set of a given collection of *clutters*.

Organization

We first provide some more details on how to implement the iterative rounding algorithm of Jain. We continue and provide a detailed description of GK with approximate oracles in Section 3 for completeness, and describe the shortest-row oracles in Section 4. Finally, in Section 5 we put together all ingredients to prove our main result.

2 Iterative rounding

Recall that Jain's key structural insight was to observe that extreme points $x\in\mathbb{R}_+^E$ of (LP_1) have $x(e)\geq 1/2$ for at least one $e\in E$. Jain also noted that, in an implementation of his algorithm, the computation of extreme points may be circumvented. In fact, he suggests obtaining LP (LP_g) from (LP_1) by adding the constraint $x(g)\geq 1/2$ for some edge $g\in E$. Let opt_g be the objective function value of an optimum solution to (LP_g). Jain's structural lemma now implies that $\min_{g\in E}\mathrm{opt}_g$ is at most the optimum value of (LP_1). Jain's algorithm can now be implemented by replacing the computation of an optimum basic solution to the residual problem in each iteration, by computing optimal solutions to linear programs of type (LP_g) for all edges $g\in E$.

Of course, we can also replace computing an optimal solution to (LP_g) with computing an approximate one, at the expense of a slight increase of the final approximation factor. Jain's method in this case is summarized for completeness in Algorithm 1.

▶ **Claim 3.** *Given $\zeta>0$, Algorithm 1 is a $2(1+\zeta)^{|E|}$-approximation algorithm for* (IP). *Moreover, Algorithm 1 terminates after at most $|E|$ iterations of step 2.*

Algorithm 1 A $2(1+\zeta)^{|E|}$- approximation algorithm for (IP).

$I_0 \leftarrow \varnothing$, $z_0(e) \leftarrow 0$ for all $e \in E$, $k \leftarrow 0$
while $f(S) - z_k(\delta(S)) > 0$ for some $S \subseteq V$ **do**
  $k \leftarrow k+1$
  for all $g \in E \setminus I_{k-1}$ **do**
    find a $(1+\zeta)$-approximation $x_{k,g}$ for (LP_g) with $z := z_{k-1}$ and $I := I_{k-1}$
  end for
  let x_k be a vector $x_{k,g}$ corresponding to $\min_{g \in E \setminus I_{k-1}} \sum_{e \in E} c(e) x_{k,g}(e)$
  $I_k \leftarrow I_{k-1} \cup \{e \in E \setminus I_{k-1} \,|\, x_k(e) = 0 \text{ or } x_k(e) \geq 1/2\}$
  for all $e \in E$, let $z_k(e) \leftarrow \lceil x_k(E) \rceil$ if $x_k(e) \geq 1/2$, and $z_k(e) \leftarrow z_{k-1}(e)$ otherwise
end while
return z_k

Proof. Jain's structural lemma immediately implies that $I_{k-1} \subsetneq I_k \subseteq E$ for every k. Hence the total number of iterations is at most $|E|$. In iteration k we fix the values of variable $x(e)$, $e \in I_k \setminus I_{k-1}$, and due to the definition of $I_k \setminus I_{k-1}$ we have $z_k(e) \leq 2x_k(e)$, $e \in I_k \setminus I_{k-1}$. The remaining values $x_k(e)$, $e \in E \setminus I_k$ form a valid solution for (LP_1) with $z(e) := z_k(e)$, $e \in E$, which is solved with approximation guarantee $(1+\zeta)$ in the $(k+1)$-st iteration. Since there are at most $|E|$ iterations and in step 5 the found solution is a $(1+\zeta)$-approximation of (LP_g), we know that the objective value of the output is at most $2(1+\zeta)^{|E|}$ times the objective value of the linear relaxation of (IP), finishing the proof. ◀

Note that by Claim 3, if $\zeta \leq \ln(1+\varepsilon)/|E|$ then Algorithm 1 gives a $2(1+\varepsilon)$-approximation for (IP).

3 Multiplicative weights method

In this section, we briefly review the multiplicative weights method [10] of GK, when applied to a positive covering LP of the form

$$\begin{aligned} \min \quad & \sum_{j \in [n]} c(j)x(j) && (\mathrm{LP}_2) \\ \text{s.t.} \quad & \sum_{j \in [n]} A(i,j)x(j) \geq b(i) \quad \forall i \in [m], \\ & x \geq 0\,, \end{aligned}$$

where $A(i,j) \geq 0$ for all $i \in [m]$, $j \in [n]$, $b(i) > 0$ for all $i \in [m]$ and $c(j) > 0$ for all $j \in [n]$. Note, that the linear program (LP_g) is a positive LP of the above form when simply eliminating variables $x(e)$ for $e \in I$. In the same way, we can exclude the inequalities corresponding to $S \subseteq V$ with $f(S) - z(\delta(S)) \leq 0$.

Given $i \in [m]$ and a vector $x \in \mathbb{R}^n_+$ we define the *length* $\mathrm{len}(i,x)$ of row i with respect to x as

$$\mathrm{len}(i,x) := \sum_{j \in [n]} A(i,j)x(j)/b(i)\,, \tag{1}$$

and we denote by $\mathrm{len}(x)$ the shortest length of a row in A with respect to x, i.e. $\mathrm{len}(x) := \min_{i \in [m]} \mathrm{len}(i,x)$. Now it is straightforward to reformulate (LP_2) as

$$\min_{x \geq 0, x \neq 0} \sum_{j \in [n]} c(j)x(j)/\mathrm{len}(x)\,. \tag{2}$$

Algorithm 2 The multiplicative weights algorithm to solve (LP_2).

1: $\delta \leftarrow (1+\zeta)\big((1+\zeta)n\big)^{-\frac{1}{\zeta}}$, $x_0(j) \leftarrow \delta/c(j)$ for all $j \in [n]$, $y_0(i) \leftarrow 0$ for all $i \in [m]$, $k \leftarrow 0$
2: **while** $\sum_{j\in[n]} c(j)x_k(j) < 1$ **do**
3: $\quad k \leftarrow k+1$
4: $\quad$ determine a $(1+\zeta)$-approximation for the shortest row with respect to x_{k-1}, let it be row q_k
5: $\quad$ determine $j \in [n]$ with the minimum value $c(j)/A(q_k, j)$, let it be column p_k
6: $\quad$ **for all** $i \in [m]$ **do**

$$y_k(i) \leftarrow \begin{cases} y_{k-1}(i) + c(p_k)/A(q_k,p_k) & \text{if} \quad i = q_k \\ y_{k-1}(i) & \text{otherwise}\,. \end{cases}$$

7: $\quad$ **end for**
8: $\quad x_k(j) \leftarrow \big(1 + \zeta \frac{c(p_k)A(q_k,j)}{c(j)A(q_k,p_k)}\big)x_{k-1}(j)$ for all $j \in [n]$
9: **end while**
10: **return** $x_k/\operatorname{len}(x_k)$ corresponding to $\min_k \sum_{j\in[n]} c(j)x_k(j)/\operatorname{len}(x_k)$

The multiplicative weights method of GK applied to the dual of (LP_2) computes an approximate pair of primal and dual solutions in strongly-polynomial time, as long as it is provided with a strongly-polynomial time oracle for determining the row q of shortest length (the *shortest row*) with respect to given lengths $x \in \mathbb{R}^n_+$ as in (1).

It is implicit in the work of [10, 6] that exact oracles can be replaced by approximate ones (incurring a corresponding degradation in performance ratio, of course). Such a modification is described from a packing point of view in [4], for example. Algorithm 2 shows the pseudo code of the algorithm for completeness. In step 4 of the algorithm a $(1+\zeta)$-approximation q of the shortest row with respect to some vector $x \in \mathbb{R}^n_+$ is computed. That is, q is a row for which $\operatorname{len}(q, x) \le (1+\zeta)\operatorname{len}(x)$. Section 4 describes how to obtain this approximation in strongly-polynomial time.

We give a proof of the next lemma for completeness.

▶ **Lemma 4** (implicit in [10, 6]). *Algorithm 2 is a* $(1+4\zeta)$*-approximation for* (LP_2)*. Moreover, Algorithm 2 terminates after at most* $\frac{1}{\zeta}\log_{1+\zeta}(1+\zeta)n$ *iterations.*

Proof. Let us define $\beta := \min_k \frac{\sum_{j\in[n]} c(j)x_k(j)}{\operatorname{len}(x_k)}$, below we show that β provides a good approximation for the problem given by (2).

For every $k \ge 1$ we have

$$\sum_{j\in[n]} c(j)x_k(j) - \sum_{j\in[n]} c(j)x_{k-1}(j) =$$
$$\zeta \operatorname{len}(q_k, x_{k-1})c(p_k)b(q_k)/A(q_k,p_k) \le \zeta(1+\zeta)\operatorname{len}(x_{k-1}) \times \sum_{i\in[m]} b(i)(y_k(i) - y_{k-1}(i))\,.$$

Hence,

$$\sum_{j\in[n]} c(j)x_k(j) \le \sum_{j\in[n]} c(j)x_0(j) + \zeta(1+\zeta) \times \sum_{h\in[k]}\sum_{i\in[m]} b(i)(y_h(i) - y_{h-1}(i))\operatorname{len}(x_{h-1})\,.$$

Due to the definition of β we have $\text{len}(x_{h-1}) \leq \sum_{j\in[n]} c(j)x_{h-1}(j)/\beta$ and thus

$$\sum_{j\in[n]} c(j)x_k(j) \leq \sum_{j\in[n]} c(j)x_0(j) + \frac{\zeta(1+\zeta)}{\beta} \times \sum_{h\in[k]} \sum_{i\in[m]} b(i)(y_h(i) - y_{h-1}(i)) \sum_{j\in[n]} c(j)x_{h-1}(j)\,. \quad (3)$$

To show that the right-hand side of (3) is at most $n\delta\,\mathbf{e}^{\zeta(1+\zeta)\sum_{i\in[m]} b(i)y_k(i)/\beta}$, we use induction. Indeed, the case $k = 0$ is clear, and to show the statement consider

$$\sum_{j\in[n]} c(j)x_0(j) + \frac{\zeta(1+\zeta)}{\beta} \times \sum_{h\in[k]} \sum_{i\in[m]} b(i)(y_h(i) - y_{h-1}(i)) \sum_{j\in[n]} c(j)x_{h-1}(j)\,,$$

which equals

$$\sum_{j\in[n]} c(j)x_0(j) + \frac{\zeta(1+\zeta)}{\beta} \times \Big(\sum_{h\in[k-1]} \sum_{i\in[m]} b(i)(y_h(i) - y_{h-1}(i)) \sum_{j\in[n]} c(j)x_{h-1}(j) + \sum_{i\in[m]} b(i)\big(y_k(i) - y_{k-1}(i)\big) \sum_{j\in[n]} c(j)x_{k-1}(j)\Big)$$

Due to (3) we conclude that the last expression is at most

$$\Big(1 + \frac{\zeta(1+\zeta)}{\beta} \sum_{i\in[m]} b(i)\big(y_k(i) - y_{k-1}(i)\big)\Big) \times \Big(\sum_{j\in[n]} c(j)x_0(j) + \frac{\zeta(1+\zeta)}{\beta} \times \sum_{h\in[k-1]} \sum_{i\in[m]} b(i)\big(y_h(i) - y_{h-1}(i)\big) \sum_{j\in[n]} c(j)x_{h-1}(j)\Big)\,.$$

Using the inequality $(1+\alpha) \leq \mathbf{e}^{\alpha}$, $\alpha \in \mathbb{R}$ and the induction hypothesis we upper-bound the expression above by

$$\mathbf{e}^{\zeta(1+\zeta)\sum_{i\in[m]} b(i)\left(y_k(i)-y_{k-1}(i)\right)/\beta} \times n\delta\,\mathbf{e}^{\zeta(1+\zeta)\sum_{i\in[m]} b(i)y_{k-1}(i)/\beta} = n\delta\,\mathbf{e}^{\zeta(1+\zeta)\sum_{i\in[m]} b(i)y_k(i)/\beta}\,.$$

Now let us consider the last iteration t, where we have

$$1 \leq \sum_{j\in[n]} c(j)x_t(j) \leq n\delta\,\mathbf{e}^{\zeta(1+\zeta)\sum_{i\in[m]} b(i)y_t(i)/\beta}\,,$$

and thus

$$\frac{\beta}{\sum_{i\in[m]} b(i)y_t(i)} \leq \frac{\zeta(1+\zeta)}{\ln((n\delta)^{-1})} \quad (4)$$

whenever $n\delta < 1$.

Now let us show that $y_t/\log_{1+\zeta}\left(\frac{1+\zeta}{\delta}\right)$ is a feasible solution for the dual of (LP_2). It is enough to show that

$$\max_j \sum_{i\in[m]} \frac{A(i,j)y_t(i)}{c(j)} \leq \log_{1+\zeta} \frac{1+\zeta}{\delta}\,.$$

To see this note that for every $j \in [n]$ and every k

$$\sum_{i\in[m]} \frac{A(i,j)y_k(i)}{c(j)} - \sum_{i\in[m]} \frac{A(i,j)y_{k-1}(i)}{c(j)} = \frac{A(q_k,j)}{c(j)} \frac{c(p_k)}{A(q_k,p_k)} \leq 1\,,$$

and $\sum_{i\in[m]} \frac{A(i,j)y_0(i)}{c(j)} = 0$. On the other hand for every $j \in [n]$ and every k

$$\frac{x_k(j)}{x_{k-1}(j)} = 1 + \zeta \frac{c(p_k)A(q_k,j)}{c(j)A(q_k,p_k)} \leq 1+\zeta\,,$$

$x_0(j) = \delta/c(j)$ and due to the termination condition $x_{t-1}(j) < 1/c(j)$ and hence $x_t(j) < (1+\zeta)/c(j)$. This implies that the algorithm terminates after at most $\log_{1+\zeta} \frac{1+\zeta}{\delta}$ iterations. Thus, $y_t/\log_{1+\zeta}\left(\frac{1+\zeta}{\delta}\right)$ is a feasible solution for the dual of (LP_2).

Hence, the algorithm provides a feasible solution for (2) with value β, which is an approximation with guarantee

$$\frac{\zeta(1+\zeta)}{\ln((n\delta)^{-1})} \log_{1+\zeta} \frac{1+\zeta}{\delta} = \frac{\zeta(1+\zeta)}{\ln(1+\zeta)} \frac{\ln \frac{1+\zeta}{\delta}}{\ln((n\delta)^{-1})}\,,$$

due to (4) and the fact that $y_t/\log_{1+\zeta}\left(\frac{1+\zeta}{\delta}\right)$ is a feasible solution for the dual of (LP_2). Thus, we obtain

$$\frac{\zeta(1+\zeta)}{\ln(1+\zeta)} \frac{\ln \frac{1+\zeta}{\delta}}{\ln((n\delta)^{-1})} = \frac{\zeta(1+\zeta)}{(1-\zeta)\ln(1+\zeta)} \leq \frac{\zeta(1+\zeta)}{(1-\zeta)(\zeta-\zeta^2/2)} \leq \frac{1+\zeta}{(1-\zeta)^2}\,,$$

which is at most $(1+4\zeta)$ for $\zeta \leq 0.15$. ◀

4 The shortest row problem

In this section we describe how to (approximately) solve the shortest row problem needed in Algorithm 2 when applied to (LP_1). We start by stating the following simple remark, that we will need at the end of our analysis.

▶ Remark 5. For every $k \geq 1$ and every $S \subseteq V$, we have $f(S) - z_k(\delta(S)) \leq |E|/2$ in Algorithm 1.

Proof. Define $x \in \mathbb{R}_+^E$ by letting $x(e) := x_k(e)$ whenever $x_k(e) < 1/2$, and let $x(e) := 0$ otherwise. Then note that $|E|/2 \geq x(\delta(S)) \geq f(S) - z_k(\delta(S))$, due to the feasibility of x in (LP_1) with $z := z_k$ and $I := I_k$. ◀

Let us recall that, for given $x \in \mathbb{R}_+^E$, $z \in \mathbb{Z}_+^E$, and proper function f, the shortest row problem we need to solve is the following:

$$\min_{\substack{f(S)-z(\delta(S))\geq 1\\ S\subseteq V}} \frac{x(\delta(S))}{f(S)-z(\delta(S))}.$$

The above shortest row problem is quite easy to solve when $z = \not\vdash$. In this case, Gabow et al. [7] give a strongly-polynomial time separation oracle based on the construction of *Gomory-Hu trees* [13], as we are now going to explain.

Given a graph $G = (V, E)$ and values $x(e) \in \mathbb{R}_+$ for each $e \in E$, a *Gomory-Hu tree* [13] is a capacitated tree $T = (V, J)$ such that for any two vertices $v, u \in V$ the minimum x-value of a cut in G separating v and u equals the minimum x-value among the u-v cuts induced

by the edges of T. More concretely, let S_e and $V \setminus S_e$ induce connected components in T after removing e from T. We then have

$$\min_{\substack{u\in S, v\notin S\\ S\subseteq V}} x(\delta(S)) = \min_{\substack{u\in S_e, v\notin S_e\\ e\in J}} x(\delta(S_e))\,.$$

The next lemma shows that in order to find the shortest row in the first iteration of step 2 in Algorithm 1 (i.e. when $z = \nvdash$), it is enough to compute a Gomory-Hu tree with respect to values $x(e) \in \mathbb{R}_+$, $e \in E$.

▶ **Lemma 6** ([7]). *Given a graph $G = (V, E)$, a proper function $f : 2^V \to \mathbb{Z}_+$ and a Gomory-Hu tree $T = (V, J)$ with respect to values $x(e) \in \mathbb{R}_+$, $e \in E$, we have*

$$\min_{\substack{f(S)\neq 0\\ S\subseteq V}} x(\delta(S))/f(S) = \min_{\substack{f(S_e)\neq 0\\ e\in J}} x(\delta(S_e))/f(S_e)\,.$$

Proof. Consider $S \subseteq V$ and the edges $\delta_T(S)$ in the Gomory-Hu tree T defined by S. By definition of a Gomory-Hu tree $x(\delta(S)) \geq x(\delta(S_e))$ for every $e \in \delta_T(S)$, due to the cut in T incurred by the vertices incident to e. Thus, to prove the claim it is enough to show that

$$f(S) \leq \max_{e\in\delta_T(S)} f(S_e)\,. \tag{5}$$

To show the last inequality let $V_1,\ldots,V_k$ be vertex sets of the connected components after removing S in T. Thus, $V_1,\ldots,V_k$ form a partition of $V \setminus S$, and so $\max_{i\in[k]} f(V_i) \geq f(V \setminus S) = f(S)$. Choose $i \in [k]$. Replacing S_e by $V \setminus S_e$, we can assume that S_e and V_i are disjoint for every $e \in \delta_T(V_i)$. Thus the sets S_e with $e \in \delta_T(V_i)$ partition $V \setminus V_i$, showing that $f(V_i) \leq \max_{e\in\delta_T(V_i)} f(S_e)$. Since, $\delta_T(V_i)$, $i \in [k]$ partition $\delta(S)$ we get (5), finishing the proof. ◀

In the later iterations of steps 2 in Algorithm 1, the inequality corresponding to $S \subseteq V$ has the form $x(\delta(S)) \geq g(S)$, where $g : 2^V \to \mathbb{Z}_+$ is such that $g(S) = f(S) - z(\delta(S))$ for some $z(e) \in \mathbb{Z}_+$, $e \in E$, and a proper function f. Once $z \neq \nvdash$, $g(S)$ is not a proper function any more and unfortunately, Gabow et al.'s algorithm can not be used directly. We do not know how to solve this problem *exactly* in strongly-polynomial time, but we can approximate it using the following observation.

Fix a value $\gamma > 0$, and let us check whether the optimal solution of the shortest row problem has a value less than γ. The crucial fact is that given $x(e) \in \mathbb{R}_+$, $e \in E$ and $\gamma > 0$ checking whether

$$\min_{\substack{f(S)-z(\delta(S))\geq 1\\ S\subseteq V}} \frac{x(\delta(S))}{f(S) - z(\delta(S))} < \gamma\,.$$

is equivalent to checking whether

$$x(\delta(S))/\gamma + z(\delta(S)) < f(S)$$

for some $S \subseteq V$, i.e. it can be reduced to finding

$$\min_{\substack{f(S)\neq 0\\ S\subseteq V}} \frac{x(\delta(S))/\gamma + z(\delta(S))}{f(S)}.$$

Therefore, we can apply Lemma 6 after replacing $x(e)$ with $x(e)/\gamma + z(e)$. This enables us to use binary search to find a $(1 + \zeta)$-approximation for the shortest row indexed by

Algorithm 3 Determining upper and lower bounds $\gamma_{\min}$, $\gamma_{\max}$.

1: $G_0 = (V_0, E_0) \leftarrow G = (V, E)$, $f_0(S) \leftarrow f(S)$ for all $S \subseteq V$, $k \leftarrow 0$
2: **while** $f_k(S) - z(\delta_{G_k}(S)) > 0$ for some $S \subseteq V_k$ **do**
3: find a set $S \subseteq V_k$ such that $f_k(S) - z(\delta_{G_k}(S)) \geq 1$, let it be S_k
4: determine $e \in \delta_{G_k}(S_k)$ with maximum $x(e)$, let it be $e_k \leftarrow \{u_k, v_k\}$
5: contract e_k in G_k (keeping multiple copies of edges) to obtain G_{k+1}, and set

$$f_{k+1}(S) \leftarrow \begin{cases} f_k(S \cup \{u_k, v_k\} \setminus w_k) & \text{if} \quad w_k \in S \\ f_k(S) & \text{otherwise}\,, \end{cases}$$

for all $S \subseteq V_{k+1}$, where w_k is the vertex in G_{k+1} corresponding to the contracted edge e_k.
6: $k \leftarrow k + 1$
7: **end while**
8: $\gamma_{\min} \leftarrow \min_k 2x(e_k)/|E|$, and let p be the index for which this minimum is achieved
9: $\gamma_{\max} \leftarrow x(\delta(U_p))$, where U_p is the vertex subset of V corresponding to the vertex subset S_p of V_p
10: **return** $\gamma_{\min}$, $\gamma_{\max}$

vertex subsets whenever we have a lower bound $\gamma_{\min}$ and an upper bound $\gamma_{\max}$ on the length of the shortest row. Giving trivial bounds on such a value (e.g. 1 and $(|E| \cdot \max_S f(S))$) is of course easy. However, given an interval $[\gamma_{\min}, \gamma_{\max}]$ for binary search we have to construct a Gomory-Hu tree $\lceil \log_{1+\zeta} \gamma_{\max}/\gamma_{\min} \rceil$ times, and therefore we need that $\gamma_{\max}/\gamma_{\min}$ is independent of the size of f in order to achieve strong polynomiality. To this aim, we propose Algorithm 3.

▶ **Lemma 7.** *Algorithm 3 computes an interval $[\gamma_{\min}, \gamma_{\max}]$, which contains the shortest row length with respect to $x(e) \in \mathbb{R}_+$, $e \in E$. Moreover, $\gamma_{\max}/\gamma_{\min} \leq |E|^2/2$, and the algorithm runs in strongly-polynomial time.*

Proof. Algorithm 3 works as follows. It does a sequence of at most $|V|$ iterations. In iteration k, it takes an arbitrary subset S corresponding to a violated cut, i.e. such that $f_k(S) - z(\delta_{G_k}(S)) > 0$, and contracts the edge e_k in this cut of maximum x-value. Contracting this edge naturally yields a graph G_{k+1} and a function f_{k+1} to use in the next iteration. Note that a violated subset S can be computed efficiently given that f is a proper function [12].

Our first claim is that $\gamma_{\min}$ is a valid lower bound on the shortest row length. In other words, we claim that for every $S : f(S) - z(\delta(S)) \geq 1$, we have

$$\frac{x(\delta(S))}{f(S) - z(\delta(S))} \geq \frac{x(e_p)}{|E|/2} = \gamma_{\min}\,.$$

Due to the termination condition, for every $S \subseteq V$ with $f(S) - z(\delta(S)) \geq 1$ the edge set $\delta(S)$ contains at least one of the edges $e_1, \ldots, e_t$ selected by the algorithms during its t iterations. Therefore, $x(\delta(S)) \geq x(e_p)$, by the choice of p in step 8. Moreover, by Remark 5 $f(S) - z(\delta(S)) \leq |E|/2$. The claim then follows.

Our second claim is that $\gamma_{\max}$ is a valid upper bound on the shortest row length. To see this, note that $f(U_p) - z(\delta(U_p)) \geq 1$ because $f(U_p) = f_p(S_p)$ and $z(\delta(U_p)) = z(\delta_{G_p}(S_p))$, proving that $\gamma_{\max}$ is a valid upper bound for the shortest length of a row indexed by $S \subseteq V$.

Finally, recalling that e_p satisfies $x(e_p) = \max_{e \in \delta(U_p)} x(e)$ (step 4), we have

$$\gamma_{\max}/\gamma_{\min} = \frac{x(\delta(U_p))}{2x(e_p)/|E|} \leq |E|^2/2\,. \qquad \blacktriangleleft$$

5 Concluding remarks

We are now ready to put all pieces together and give a proof of Theorem 1 and Corollary 2 stated in the introduction.

Proof of Theorem 1. Given an $\varepsilon > 0$, we apply Algorithm 2 to (LP_1) with $\zeta = \ln(1+\varepsilon)/|E|$. Algorithm 2 in its turn approximates the shortest row at most $O((\ln|V|)/\zeta^2)$ times[3]. It makes a call to Algorithm 3, computing at most $|V|$ Gomory-Hu trees and afterwards the binary search needs $O((\ln|E|)/\zeta)$ computations of a Gomory-Hu tree in $G = (V, E)$. Recall, that $\zeta = \ln(1+\varepsilon)/|E| = \Theta(\varepsilon/|E|)$ and hence each linear program appearing in Algorithm 1 is solved in time dominated by finding $O(|E|^3(\ln|E|)^2/\varepsilon^3)$ Gomory-Hu trees. Note that a Gomory-Hu tree for $G = (V, E)$ with respect to values $x(e) \in \mathbb{R}_+$, $e \in E$ can be found by $|V|$ computations of the minimum cut in G [13], so a Gomory-Hu tree can be found in strongly-polynomial time. ◀

The number of times our algorithm solves the Gomory-Hu tree problem is substantially smaller than the corresponding number for the Ellipsoid method given the classical estimation for the encoding length of vertices or given that $\max_S f(S)$ is sufficiently large, because this number for the Ellipsoid method grows proportionally with the logarithm of $\max_S f(S)$.

Proof of Corollary 2. To obtain a $(2+\varepsilon)$-approximation for (IP) we apply Algorithm 1. Algorithm 1 solves $O(|E|^2)$ linear programs, i.e. there are $O(|E|^2)$ calls of Algorithm 1 to Algorithm 2 to solve linear programs, and it makes at most $|E|$ roundings. Considering rounding as a basic operation, the result follows. ◀

We conclude the paper with some open questions. It remains open whether one is able to provide a 2-approximation algorithm for (IP), which does not need to solve linear programs. This question is among the top 10 open questions in the theory of approximation algorithms according to Shmoys and Williamson [23]. In our opinion, a good intermediate question is whether it is possible to give an algorithm with a constant approximation guarantee such that the number of linear programs solved in its course is bounded by a constant. One way to prove this could be to exploit that after each rounding in the algorithm of Jain [19] we have a sufficiently "good" feasible point for the new linear program.

References

1 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

2 Daniel Bienstock. *Potential function methods for approximately solving linear programming problems: theory and practice*, volume 53. Springer Science & Business Media, 2006.

3 Daniel Bienstock and Garud Iyengar. Solving fractional packing problems in o ast $(1/\varepsilon)$ iterations. In *Proc. of the 36th Annual ACM Symp. on Theory of Computing*, pages 146–155. ACM, 2004.

4 Khaled Elbassioni, Kurt Mehlhorn, and Fahimeh Ramezani. Towards more practical linear programming-based techniques for algorithmic mechanism design. *SAGT*, pages 98–109, 2015.

5 Lisa Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1001–1010. Society for Industrial and Applied Mathematics, 2004.

[3] Here, we use the inequality that $1 + \zeta < \mathbf{e}^{\zeta} < 1 + \frac{7}{4}\zeta$ for $0 < \zeta < 1$.

6 Lisa K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

7 Harold N. Gabow, Michel X. Goemans, and David P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82(1-2, Ser. B):13–40, 1998. Networks and matroids; Sequencing and scheduling.

8 Naveen Garg and Rohit Khandekar. Fast approximation algorithms for fractional steiner forest and related problems. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 500–509. IEEE, 2002.

9 Naveen Garg and Rohit Khandekar. Fractional covering with upper bounds on the variables: Solving lps with negative entries. In *Algorithms–ESA 2004*, pages 371–382. Springer, 2004.

10 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007. doi:10.1137/S0097539704446232.

11 M. X. Goemans, D. B. Shmoys, A. V. Goldberg, É. Tardos, S. Plotkin, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proc. of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'94)*, pages 223–232. ACM, 1994.

12 Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

13 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 9:551–570, 1961.

14 Michael D Grigoriadis and Leonid G Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.

15 Michael D Grigoriadis and Leonid G Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{o}(\epsilon - 2knm)$ time. *Mathematical Programming*, 75(3):477–482, 1996.

16 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics: Study and Research Texts*. Springer-Verlag, Berlin, 1988. doi:10.1007/978-3-642-97881-4.

17 Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

18 Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

19 Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

20 Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 448–457. ACM, 1993.

21 Farhad Shahrokhi and David W Matula. The maximum concurrent flow problem. *Journal of the ACM (JACM)*, 37(2):318–334, 1990.

22 Eva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

23 David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, Cambridge, 2011. doi:10.1017/CBO9780511921735.

24 Neal E Young. Randomized rounding without solving the linear program. In *SODA*, volume 95, pages 170–178, 1995.

25 Neal E Young. Sequential and parallel algorithms for mixed packing and covering. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 538–546. IEEE, 2001.