

Approximating $q \rightarrow p$ Norms of Non-Negative Matrices in Nearly-Linear Time

Etienne Objois  

IRIF, Université Paris Cité, France

Adrian Vladu  

CNRS, IRIF, Université Paris Cité, France

Abstract

We provide the first nearly-linear time algorithm for approximating $\ell_{q \rightarrow p}$ -norms of non-negative matrices, for $q \geq p \geq 1$. Our algorithm returns a $(1 - \varepsilon)$ -approximation to the matrix norm in time $\tilde{O}\left(\frac{1}{q\varepsilon} \cdot \text{nnz}(\mathbf{A})\right)$, where \mathbf{A} is the input matrix, and improves upon the previous state of the art, which either proved convergence only in the limit [Boyd '74], or had very high polynomial running times [Bhaskara-Vijayraghavan, SODA '11]. Our algorithm is extremely simple, and is largely inspired from the coordinate-scaling approach used for positive linear program solvers. Our algorithm can readily be used in the [Englert-Räcke, FOCS '09] to improve the running time of constructing $O(\log n)$ -competitive ℓ_p -oblivious routings.

2012 ACM Subject Classification Theory of computation \rightarrow Routing and network design problems

Keywords and phrases matrix norm, Perron-Frobenius theory, oblivious routings, input-sparsity time, lp norm

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.69

Related Version *Extended Version*: <https://arxiv.org/abs/2503.19553>

Funding This work was partially supported by the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE48-0016 (project COMCOPT).

Acknowledgements We thank Alina Ene and Huy Lê Nguyễn for helpful conversations on approximating matrix norms.

1 Introduction

We are interested in computing the norm of matrices. We define the $\ell_{q \rightarrow p}$ -norm of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ as

$$\|\mathbf{A}\|_{q \rightarrow p} = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} \quad (1)$$

where $\|\mathbf{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$, and $q, p \geq 1$. When $q = p = 2$, the quantity $\|\mathbf{A}\|_{2 \rightarrow 2}$ corresponds to the spectral norm. In all generality, $\|\mathbf{A}\|_{q \rightarrow p}$ corresponds to the maximum stretch of the operator \mathbf{A} from the normed space ℓ_q^n to ℓ_p^m .

The $\ell_{q \rightarrow p}$ -norm of a matrix appears in different optimization problems. When $p = q$, $\|\mathbf{A}\|_{p \rightarrow p}$ is known as the p -norm of \mathbf{A} and has important applications to computing linear oblivious routings for graphs [9]. A linear oblivious routing in a graph with m edges can be implicitly represented by a square matrix \mathbf{A} with m rows and columns. For some class of linear routings, the induced norm corresponds to its competitiveness.

A big challenge is even certifying that a routing scheme is good, which requires saying that for any demand, the resulting flow is competitive. It is difficult to do this in general, since as opposed to the classical $p = \infty$ case, it is unclear how to find the worst case demand. In fact for general matrices the problem is NP-hard. For matrices arising from routing schemes,



© Etienne Objois and Adrian Vladu;

licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 69; pp. 69:1–69:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



which are entry-wise non-negative, there are polynomial time algorithms. In particular, Boyd proposed a power method-type algorithm which converged in the limit, but which lacked asymptotic bounds [6]. Later, Bhaskara and Vijayraghavan proposed a modified algorithm which provably converged in polynomial time, but with a cubic iteration number [4]. In this paper we provide a lightweight and efficient algorithm which runs in nearly linear time in input sparsity.

Noteworthy, [4] showed that by being able to approximate the p -norm of a non-negative matrix, one can compute in polynomial time $O(\log n)$ -competitive linear oblivious routings in undirected graphs when the load function is an unknown monotone norm and the aggregation function is an ℓ_p -norm on the load vector.

Routing schemes have also raised interest in the non-oblivious case, that is when the routing scheme has information on the current state of the graph (i.e. the flow on each edge). In this case, one can compute the current shortest path at any instance to minimize the congestion in the graph. Recently, an algorithm using electrical flow instead of shortest paths has been shown to be more efficient [21].

1.1 Our Contributions

We give the first input-sparsity time algorithm to approximate the $\ell_{q \rightarrow p}$ -norm of a non-negative matrix when $q \geq p \geq 1$. The result is stated in the following theorem.

► **Theorem 1** (Short version of Theorem 13). *Given a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a positive real $0 < \varepsilon \leq 1/2q$, two reals $q \geq p \geq 1$, and a guess V on $\|\mathbf{A}\|_{q \rightarrow p}$, Algorithm 1 recovers \mathbf{x} such that*

$$\frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} \geq (1 - \varepsilon)V$$

or certifies infeasibility in time $\tilde{O}\left(\frac{\text{nnz}(\mathbf{A})}{q\varepsilon}\right)^1$, where $\text{nnz}(\mathbf{A})$ corresponds to the time to perform a matrix-vector product with \mathbf{A} .

While the statement of Theorem 1 concerns approximation of a decision problem, standard techniques can transform an approximation decision algorithm into an approximation algorithm. We refer the reader to Appendix A for the details which we provide for completeness.

In the context of tree-based oblivious routing schemes, the *competitive ratio* can be written directly as an $\ell_{p \rightarrow p}$ -norm expression. If \mathbf{M} is the $|E| \times |E|$ matrix that specifies how unit demands on edges are routed, then the competitive ratio is $\max_{\|\mathbf{x}\|_p \leq 1} \|\mathbf{M}\mathbf{x}\|_p$. [4] proved it is sufficient to approximate $\|\mathbf{M}\|_{p \rightarrow p}$ a polynomial number of times to compute a $O(\log n)$ competitive tree-based oblivious routing. Hence, we can use Theorem 1 for the case $p = q$, and $\mathbf{M} \in \mathbb{R}^{m \times m}$ to improve the running time of [4] by a factor of $O(m^3)$. Moreover, if both the load function and the aggregation function are known monotone norms, we provide the first algorithm to compute an optimal oblivious linear routing. Notice that this problem is slightly different from the one solved by [4] as we do need knowledge of the load function. Hence, the following theorem.

► **Theorem 2** (Short version of Theorem 19). *Given a directed or undirected graph $G = (V, E)$, there exists an algorithm to compute an optimal oblivious routing when the cost function is a known monotone norm in time $\tilde{O}(n^6 m^3)$.*

¹ \tilde{O} hides poly logarithmic factors in $\varepsilon^{-1}, n, m, q, p$.

1.2 Our Techniques

We seek to maximize $g : \mathbf{x} \mapsto \|\mathbf{A}\mathbf{x}\|_p^q / \|\mathbf{x}\|_q^q$. Our approach is partly inspired from the long line of work on solvers for positive linear programs [17, 24, 1, 18], as well as certain non-standard instantiations of the same framework in the context of regression problems [8]. Those algorithms are *width-independent*, meaning their running time is at most linear in $\text{nnz}(\mathbf{A}) + n + m$ (at a cost of a larger dependency on the precision ε). The *width* of an instance measures the instance’s diameter in the appropriate norm. For example, for linear programs, the *width* is typically defined as the maximum absolute value that any of its linear constraint functions can attain over the search space. By design, those algorithms are different from classical multiplicative weight update algorithms as their running time does not depend on the *width* of the problem.

At each iteration, we scale by $(1 + \alpha)$ some coordinates of our iterate to ensure large progress is made. We say that a coordinate is hit at time t if we scale it between iterations t and $t + 1$. We do not directly measure the progress on g , instead, we separate the progress obtained in the denominator from the progress obtained in the numerator. At each iteration, assuming \mathbf{A} is non-negative, we want to ensure the ratio between those progresses is larger than $(1 - O(\varepsilon))^q \|\mathbf{A}\|_{q \rightarrow p}^q$ (Lemma 7 with $V = \|\mathbf{A}\|_{q \rightarrow p}$). This ensures after T iterations that $g(\mathbf{x}^{(T)})$ is a $(1 - O(1/\|\mathbf{x}^{(T)}\|_q^q))$ approximation of $\|\mathbf{A}\|_{q \rightarrow p}^q$. To do so, we consider a potential function $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (Definition 8) and hit the coordinates whose potential is above a threshold. \mathbf{A} being non-negative is crucial as it keeps the property that the norm $\mathbf{x} \mapsto \|\mathbf{A}\mathbf{x}\|_p$ is monotone.

The algorithm converges once $\|\mathbf{x}^{(T)}\|_q$ is large. If we were to hit only one coordinate of $\mathbf{x}^{(t)}$ at iteration t (say the coordinate with the largest potential), the running time would depend on the ambient dimension n which would not be a width-independent algorithm. To overcome this issue, we transform our problem into an approximate decision one. Given $V \geq 0$, a guess on $\|\mathbf{A}\|_{q \rightarrow p}$, we either want to find \mathbf{x} such that $g(\mathbf{x}) \geq (1 - \varepsilon)^q V^q$ or certify that $V > \|\mathbf{A}\|_{q \rightarrow p}$. We then change the threshold for Φ accordingly. The certification comes from the fact that at each iteration, at least one coordinate of $\Phi(\mathbf{x})$ should be above $\|\mathbf{A}\|_{q \rightarrow p}^q$ (Lemma 10). Assuming $V \leq \|\mathbf{A}\|_{q \rightarrow p}$, there should always be a coordinate with potential larger than V^q . This modification still does not give a width-independent algorithm, however, Φ behaves “nicely” under multiplicative scaling which allows us to upper bound coordinate-wise the value of $\Phi(\mathbf{x}^{(t+1)})/\Phi(\mathbf{x}^{(t)})$ (Lemma 11).

We use this upper bound to prevent new coordinates from having potential larger than V^q . This allows us to guarantee that at iteration t , a coordinate with potential larger than V^q also had potential larger than V^q at iterations $0, 1, \dots, t - 1$. Since its potential was always larger than V^q , it was always hit and so it is large. In order to prevent coordinates from having potential larger than V^q , we need to also hit coordinates with potential slightly smaller than V^q which is not an issue since we aim for an approximation. This gives a first width-independent algorithm to approximate the $\ell_{q \rightarrow p}$ -norm of a non-negative matrix in time $\tilde{O}(\text{nnz}(\mathbf{A})/q\varepsilon)$ or in parallel time $\tilde{O}(1/q\varepsilon)$ (Corollary 14).

The algorithm can also use preconditioning techniques for ℓ_p subspace embedding. We first compute a matrix \mathbf{A}' such that $\|\mathbf{A}\mathbf{x}\|_p = (1 \pm \varepsilon) \|\mathbf{A}'\mathbf{x}\|_p$ and \mathbf{A}' has fewer nonzero entries than \mathbf{A} . Lewis weights sampling introduced by Cohen and Peng [7] and improved for $p > 2$ in [23] allows to reduce the number of rows of $\mathbf{A} \in \mathbb{R}^{m \times n}$ to $\tilde{O}(n/\varepsilon^2)$ when $p \leq 2$. This preconditioning technique, paired with the fact that $\|\mathbf{A}\|_{q \rightarrow p} = \|\mathbf{A}^T\|_{p^* \rightarrow q^*}$ where $1/q + 1/q^* = 1/p + 1/p^* = 1$, allows better running time when $m \gg n$ or $n \gg m$. This result is showed in Corollary 15.

The algorithm is linked with the Perron-Frobenius theorem, in the sense that for $q = p = 2$, if \mathbf{A} is positive then the maximum of $\mathbf{x} \mapsto \|\mathbf{Ax}\|_2 / \|\mathbf{x}\|_2$ is the spectral norm of \mathbf{A} . For positive matrices, Lemma 10 is a generalized version of the Collatz-Wielandt formula which states that $\min_{\mathbf{x} > \mathbf{0}} \max_i \frac{\langle \mathbf{A} \cdot \mathbf{i}, \mathbf{Ax} \rangle}{x_i} = \|\mathbf{A}\|_{2 \rightarrow 2}^2$. We generalize this formula to ℓ_p norms, and obtain $\min_{\mathbf{x} > \mathbf{0}} \max_i \frac{\langle \mathbf{A} \cdot \mathbf{i}, (\mathbf{Ax})^{p-1} \rangle}{x_i^{p-1}} = \|\mathbf{A}\|_{p \rightarrow p}^p$. When $p \neq q$, we just add a scaling factor that depends on $\|\mathbf{x}\|_q$ to find the equality.

We then show that an application of our algorithm is to compute a $O(\log n)$ -competitive linear oblivious routing with a better running time than [4]. When the load function is a known monotone norm, we show that we can reduce the problem to finding a saddle point of a bi-linear function.

1.3 Related Works

Computing the $\ell_{q \rightarrow p}$ norm on general matrices is hard. For arbitrary matrices, there are three known easy cases: $\|\mathbf{A}\|_{2 \rightarrow 2}$, $\|\mathbf{A}\|_{q \rightarrow \infty}$ and $\|\mathbf{A}\|_{1 \rightarrow p}$ (see [22]).

On the hardness front, for arbitrary matrices, computing $\|\mathbf{A}\|_{q \rightarrow p}$ when $1 \leq p < q \leq \infty$ is NP-hard [22]. For $q = p \neq 1, 2, \infty$, even if p and the matrix are assumed to have rational entries, it is NP-hard to compute $\|\mathbf{A}\|_{p \rightarrow p}$ [12].

If we aim to find an approximation, for arbitrary matrices the problem is also hard to solve to arbitrary precision. For $q \geq p > 2$ (and $2 > q \geq p > 1$), assuming $\text{NP} \notin \text{DTIME}(n^{\text{poly} \log(n)})$, the problem cannot be approximated to a factor $2^{(\log n)^{1-\varepsilon}}$, for any constant $\varepsilon > 0$ [4]. [3] showed that under the Small-Set Expansion hypothesis, it is NP-hard to approximate $\|\mathbf{A}\|_{2 \rightarrow p}$ for arbitrary matrices \mathbf{A} when $p \geq 4$. Finally, when $2 < q < p < \infty$ (and $1 < q < p < 2$), if $\text{NP} \notin \text{BPTIME}(2^{(\log n)^{O(1)}})$, it is NP-hard to approximate $\|\mathbf{A}\|_{q \rightarrow p}$ to a factor $2^{(\log n)^{1-\varepsilon}}$ for any constant $\varepsilon > 0$ [5].

For non-negative matrices, [6] used a power iteration-type algorithm to approximate p -norm of non-negative matrices without any bounds on the time of convergence. When $q \geq p \geq 1$, [4] provided the analysis and proved that the power iteration-type algorithm from [6] converges in time $\tilde{O}(\frac{n(n+m)^2}{\varepsilon} \text{nnz}(\mathbf{A}))$. Under the same settings, [22] demonstrated that the problem is equivalent to maximizing a concave function ($\mathbf{x} \mapsto \left\| \mathbf{Ax}^{\frac{1}{q}} \right\|_p$) over the unit simplex. When $q < p$, no hardness result is known for non-negative matrices.

The non-negative assumption may be necessary to make the problem easy. Indeed, it is possible to relax MAXCUT into approximating the $\ell_{\infty \rightarrow 1}$ norm of a symmetric diagonally dominant matrix (see [22]). Since MAXCUT is known to be NP-Hard to approximate better than $16/17 \approx 0.941$ [13], and UG-Hard to approximate better than $\approx 1/0.878$ [16], this implies that there is no PTAS for the $\ell_{\infty \rightarrow 1}$ problem, even for symmetric diagonally dominant matrices.

Hence, approximating $\|\mathbf{A}\|_{\infty \rightarrow 1}$ is NP-hard even when \mathbf{A} is positive semi-definite and the precision sought is fixed. However, we can easily extend the results on non-negative matrices to matrices \mathbf{A} such that there exist two diagonal matrices $\mathbf{L} \in \mathbb{R}^{m \times m}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$ with ± 1 entries on their diagonals such that \mathbf{LAR} is non-negative [22]. Indeed, given \mathbf{L}, \mathbf{R} , we have that for any \mathbf{x} , $\|\mathbf{Ax}\|_p = \|\mathbf{LARx}\|_p$. Hence, we can apply the algorithm on \mathbf{LAR} and consider \mathbf{Rx} where \mathbf{x} is a $(1 - \varepsilon)$ -approximation of $\|\mathbf{LAR}\|_{q \rightarrow p}$.

An immediate application of approximating the ℓ_p norm of non-negative matrices is computing oblivious routings, per [9]. The main difficulty posed by their approach was efficiently certifying whether a non-negative matrix implicitly defined by the underlying graph has large or small induced ℓ_p norm, for $p > 2$. The polynomial time algorithm of [4]

could be used together with the [9] framework, at the expense of paying a large overall running time. Notably, for the case of $p = 2$, approximating the induced norm can be solved efficiently via the power method; this special case has been leveraged by [10] to achieve ℓ_∞ oblivious routings running in sub-quadratic time.

2 Preliminaries

2.1 Notations

We use lowercase bold letters for vectors and uppercase for matrices. For a vector \mathbf{z} (resp. a matrix \mathbf{A}), \mathbf{z}^T (resp. \mathbf{A}^T) denotes the transpose of \mathbf{z} (resp. of \mathbf{A}). For an integer i , \mathbf{A}_i denotes the i^{th} row of \mathbf{A} , and $\mathbf{A}_{\cdot,i}$ the i^{th} column. We say that a vector or a matrix is non-negative if all its entries are non-negative. $\langle \cdot, \cdot \rangle$ is the usual inner product and $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product. For $p \geq 1$ and a vector \mathbf{z} , we denote by $\|\mathbf{z}\|_p$ the ℓ_p -norm of \mathbf{z} defined as

$$\|\mathbf{z}\|_p := \left(\sum_i |z_i|^p \right)^{\frac{1}{p}}.$$

For $q, p \geq 1$ and a matrix \mathbf{A} , we denote by $\|\mathbf{A}\|_{q \rightarrow p}$ the $\ell_{q \rightarrow p}$ -norm of \mathbf{A} defined as

$$\|\mathbf{A}\|_{q \rightarrow p} := \max_{\mathbf{z} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{z}\|_p}{\|\mathbf{z}\|_q}.$$

$\mathbf{0}$ (resp. $\mathbf{1}$) denotes the vector with all entries equal to 0 (resp. 1). For a vector \mathbf{z} and a real q , \mathbf{z}^q is the vector where the exponent is applied coordinate wise. For a matrix \mathbf{A} , $\text{nnz}(\mathbf{A})$ denotes the number of nonzero elements in \mathbf{A} . We denote by Δ_n the non-negative simplex $\Delta_n := \{\mathbf{z} \in \mathbb{R}_{\geq 0}^n : \sum_i z_i = 1\}$.

2.2 Convex Optimization

We would like to emphasize that the problem of maximizing $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x}\|_p$ on the unit ℓ_q ball corresponds to maximizing a convex function over a convex domain which can not be solved using classical convex optimization tools. Nevertheless, when \mathbf{A} is non-negative, $q \geq p \geq 1$, we can show that by composing f with a simple concave mapping from the unit simplex Δ_n to the unit ℓ_q sphere we obtain a concave function that we would like to maximize over a convex domain [22]. To intuit, one should remark that the maximum of f is achieved by a vector in the positive orthant of the unit ℓ_q sphere which we call S_q . Moreover, S_q is not a convex set. In fact, for two different vectors $\mathbf{x}, \mathbf{y} \in S_q$, and $0 < t < 1$, we have $\|t\mathbf{x} + (1-t)\mathbf{y}\|_q < 1$, hence we are “losing” norm by taking the convex combination of two points. That means that even if f is convex, we have the potential to “increase” the value of f between any two points of S_q by taking a path in S_q . This new function has better chance to be concave than f since its value between two points of S_q is larger. One can remark that if $\mathbf{u} \in \Delta_n$, then $\mathbf{u}^{\frac{1}{q}} \in S_q$. The mapping $g : \mathbf{u} \mapsto \mathbf{u}^{\frac{1}{q}}$ describes such a path since for any $t \in [0, 1]$ and $\mathbf{u}, \mathbf{v} \in \Delta_n$, we have that $t\mathbf{u} + (1-t)\mathbf{v} \in \Delta_n$. As expected, g is entry-wise concave and calculus gives that as long as $q \geq p$, $\mathbf{u} \mapsto f(g(\mathbf{u})) = \|\mathbf{A}\mathbf{u}^{\frac{1}{q}}\|_p$ is concave. We give the proof in Subsection D.1.

► **Theorem 3** (Remark 3.4 from [22]). *Given $\mathbf{A} \in \mathbb{R}_{\geq 0}^{m \times n}$, and $q \geq p \geq 1$. We have $f : \mathbf{x} \mapsto \|\mathbf{A}\mathbf{x}^{\frac{1}{q}}\|_p$ is concave on Δ_n .*

This shows that a simple re-parametrization of the problem turns it into a concave maximization problem, which can be efficiently solved using a cutting plane method [14], leading to a running time of $\tilde{O}(n \cdot \text{nnz}(\mathbf{A}) + n^3)$. However, as cubic runtime may turn out to be prohibitive, we will focus on more efficient algorithms at the expense of a linear dependence in the error tolerance of the provided solution.

2.3 Power Iteration

The previous state-of-the-art algorithm was provided in [4]. Let

$$f(\mathbf{x}) = \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q}.$$

In order to compute $\|\mathbf{A}\|_{q \rightarrow p}$, one needs to maximize f over \mathbb{R}^n . If we assume \mathbf{A} is non-negative, then [4] provide an algorithm with running time $\tilde{O}(\frac{n(n+m)^2}{\varepsilon} \text{nnz}(\mathbf{A}))$. To find the maximum of f , we can compute its gradient ∇f . We have

$$\frac{\partial f}{\partial x_i} = \frac{\|\mathbf{x}\|_q \|\mathbf{A}\mathbf{x}\|_p^{1-p} \langle \mathbf{A}_{\cdot, i}, |\mathbf{A}\mathbf{x}|^{p-1} \rangle - \|\mathbf{A}\mathbf{x}\|_p \|\mathbf{x}\|_q^{1-q} |x_i|^{q-1}}{\|\mathbf{x}\|_q^2}.$$

At the optimum, $\nabla f = \mathbf{0}$ which can be written as

$$|x_i|^{q-1} = \frac{\|\mathbf{x}\|_q^q}{\|\mathbf{A}\mathbf{x}\|_p^p} \mathbf{A}^T |\mathbf{A}\mathbf{x}|^{p-1}.$$

Since \mathbf{A} is non-negative, we know there is a maximum \mathbf{x} with non-negative coordinates. [4] then define the operator $S : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}^n$ such that $S(\mathbf{x}) = (\mathbf{A}^T (\mathbf{A}\mathbf{x})^{p-1})^{\frac{1}{q-1}}$. It is easy to show that if \mathbf{x} is such that $S(\mathbf{x}) \propto \mathbf{x}$, then \mathbf{x} is a critical point and $\|\mathbf{A}\|_{q \rightarrow p} = \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q}$. Consider the two potentials $m(\mathbf{x}) := \min_i S(\mathbf{x})_i/x_i$ and $M(\mathbf{x}) = \max_i S(\mathbf{x})_i/x_i$, [4] showed the following lemma.

► **Lemma 4** (Lemma 3.3, [4]). *For any non-negative matrix \mathbf{A} and positive vector \mathbf{x} , we have*

$$m(\mathbf{x})^{q-1} \leq \frac{\|\mathbf{A}\mathbf{x}\|_p^p}{\|\mathbf{x}\|_q^q} \leq \|\mathbf{A}\|_{q \rightarrow p}^p \|\mathbf{x}\|_q^{p-q} \leq M(\mathbf{x})^{q-1}.$$

At optimum, $m(\mathbf{x}^*)^{q-1} = M(\mathbf{x}^*)^{q-1} = \|\mathbf{A}\mathbf{x}^*\|_p^p / \|\mathbf{x}^*\|_q^q$. Hence, the goal is to reduce the ratio $M(\mathbf{x})/m(\mathbf{x})$. [4] showed that iteratively applying S such that $\mathbf{x}^{(t+1)} = S(\mathbf{x}^{(t)})$ outputs $\mathbf{x}^{(T)}$ a $(1 - \varepsilon)$ -approximation when $T = \tilde{O}(\frac{n(n+m)^2}{\varepsilon})$. Hence, the total running time is $\tilde{O}(\frac{n(n+m)^2}{\varepsilon} \text{nnz}(\mathbf{A}))$. In this paper, we provide a much simpler algorithm with running time $\tilde{O}(\text{nnz}(\mathbf{A})/q\varepsilon)$.

2.4 Lewis Weights Sampling

Cohen and Peng introduced in [7] a fast algorithm to compute a matrix \mathbf{A}' such that $\|\mathbf{A}\mathbf{x}\|_p \approx_{1+\varepsilon} \|\mathbf{A}'\mathbf{x}\|_p$ for any vector \mathbf{x} . The algorithm computes \mathbf{A}' , a matrix containing few rescaled rows of \mathbf{A} . The algorithm is of particular interest in our situation when $\mathbf{A} \in \mathbb{R}^{m \times n}$ is such that $m \gg n$. For $1 \leq p \leq 2$, $\tilde{O}(\frac{n}{\varepsilon^2})$ rows are sufficient. Moreover, the cost of such an algorithm has only time complexity $\tilde{O}(\text{nnz}(\mathbf{A}) + n^\omega)$ where ω is the time complexity of matrix multiplication. Formally, this gives the following theorem.

► **Theorem 5** (Theorem 1.3 and A.2 from [23]). *Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $p \geq 1$, one can compute \mathbf{A}' such that with probability over $1 - \delta$, we have*

$$\forall \mathbf{x} \in \mathbb{R}^n, (1 - \varepsilon) \|\mathbf{A}' \mathbf{x}\|_p \leq \|\mathbf{A} \mathbf{x}\|_p \leq (1 + \varepsilon) \|\mathbf{A}' \mathbf{x}\|_p.$$

in time $\tilde{O}(\text{nnz}(\mathbf{A}) + n^\omega + \frac{n^{\max\{1, p/2\}}}{\varepsilon^2})$. Moreover, \mathbf{A}' has at most

$$O\left(\frac{n^{\max\{1, p/2\}}}{\varepsilon^2} \left((\log n)^2 \log m + \log \frac{1}{\delta}\right)\right)$$

rows.

Since the algorithm just samples and rescales rows of \mathbf{A} , if \mathbf{A} is non-negative, then so is \mathbf{A}' . Hence, it makes Lewis weights sampling a possible preconditioning of the input.

3 Efficiently Computing Induced $\ell_{q \rightarrow p}$ -Norms of Non-Negative Matrices

3.1 A Simple $\tilde{O}\left(\frac{n}{q\varepsilon} \text{nnz}(\mathbf{A})\right)$ Algorithm

We want to solve the following approximate decision problem, which we show is sufficient to solve the approximate optimization problem in Appendix A.

► **Definition 6.** *Given $V \geq 0$, we solve the approximate decision problem if we either provide \mathbf{x} such that*

$$\frac{\|\mathbf{A} \mathbf{x}\|_p}{\|\mathbf{x}\|_q} \geq (1 - \varepsilon)V,$$

or certify that $\|\mathbf{A}\|_{q \rightarrow p} < V$.

We seek to maximize a function f of the form $f : \mathbf{x} \mapsto \frac{f_n(\mathbf{x})}{f_d(\mathbf{x})}$ where both f_n and f_d are convex functions. Hence, we have

$$\frac{f_n(\mathbf{x} + \boldsymbol{\delta}) - f_n(\mathbf{x})}{f_d(\mathbf{x} + \boldsymbol{\delta}) - f_d(\mathbf{x})} \geq \frac{\langle \nabla f_n(\mathbf{x}), \boldsymbol{\delta} \rangle}{\langle \nabla f_d(\mathbf{x} + \boldsymbol{\delta}), \boldsymbol{\delta} \rangle}.$$

If we take $f_d : \mathbf{x} \mapsto \|\mathbf{x}\|_q$, we have $\nabla f_d(\mathbf{x} + \boldsymbol{\delta}) = \left(\frac{\mathbf{x} + \boldsymbol{\delta}}{\|\mathbf{x} + \boldsymbol{\delta}\|_q}\right)^{q-1}$ which is not optimal to upper bound by $\nabla f(\mathbf{x})$. Instead, we consider $g_n : \mathbf{x} \mapsto \|\mathbf{A} \mathbf{x}\|_p^q$, $g_d : \mathbf{x} \mapsto \|\mathbf{x}\|_q^q$ and we want to maximize $g : \mathbf{x} \mapsto \left(\frac{\|\mathbf{A} \mathbf{x}\|_p}{\|\mathbf{x}\|_q}\right)^q$. We now obtain a tighter bound

$$\frac{\|\mathbf{A}(\mathbf{x} + \boldsymbol{\delta})\|_p^q - \|\mathbf{A} \mathbf{x}\|_p^q}{\|\mathbf{x} + \boldsymbol{\delta}\|_q^q - \|\mathbf{x}\|_q^q} \geq \frac{\langle \nabla g_n(\mathbf{x}), \boldsymbol{\delta} \rangle}{\langle \nabla g_d(\mathbf{x} + \boldsymbol{\delta}), \boldsymbol{\delta} \rangle} = \|\mathbf{A} \mathbf{x}\|_p^{q-p} \frac{\langle \mathbf{A}^T (\mathbf{A} \mathbf{x})^{p-1}, \boldsymbol{\delta} \rangle}{\langle (\mathbf{x} + \boldsymbol{\delta})^{q-1}, \boldsymbol{\delta} \rangle}.$$

This bound will be used to design an iterate that satisfies the following lemma. We prove Lemma 7 in Subsection E.1.

► **Lemma 7.** *Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x}^{(0)} \in \mathbb{R}^n$ such that $\|\mathbf{x}^{(0)}\|_q = 1$, $0 < q, p$, $0 < \varepsilon \leq 2/q$ and $V \geq 0$. Assume at each iteration we have*

$$\frac{\|\mathbf{A} \mathbf{x}^{(t+1)}\|_p^q - \|\mathbf{A} \mathbf{x}^{(t)}\|_p^q}{\|\mathbf{x}^{(t+1)}\|_q^q - \|\mathbf{x}^{(t)}\|_q^q} \geq \left(\left(1 - \frac{\varepsilon}{2}\right)V\right)^q,$$

69:8 Approximating $q \rightarrow p$ Norms of Non-Negative Matrices in Nearly-Linear Time

then, once $\|\mathbf{x}^{(T)}\|_q^q \geq \frac{4}{q\varepsilon}$, we have

$$\frac{\|\mathbf{Ax}^{(T)}\|_p}{\|\mathbf{x}^{(T)}\|_q} \geq (1 - \varepsilon)V.$$

With scaling updates, i.e. $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \boldsymbol{\delta}^{(t)}$ where $\delta_i^{(t)} \in \{0, \alpha x_i^{(t)}\}$ we obtain the following inequality.

$$\frac{\|\mathbf{Ax}^{(t+1)}\|_p^q - \|\mathbf{Ax}^{(t)}\|_p^q}{\|\mathbf{x}^{(t+1)}\|_q^q - \|\mathbf{x}^{(t)}\|_q^q} \geq \frac{1}{(1 + \alpha)^{q-1}} \|\mathbf{Ax}^{(t)}\|_p^{q-p} \frac{\langle \mathbf{A}^T (\mathbf{Ax}^{(t)})^{p-1}, \boldsymbol{\delta}^{(t)} \rangle}{\langle (\mathbf{x}^{(t)})^{q-1}, \boldsymbol{\delta}^{(t)} \rangle}.$$

To simplify the notation, we define a vector of potentials $\Phi(\mathbf{x}) \in \mathbb{R}^n$. Those potentials have nice properties, for instance, the direction of the gradient of g can be extracted from the values of the potentials. Indeed, they are made such that $\nabla g(\mathbf{x}) = \frac{\nabla g_d(\mathbf{x})}{g_d(\mathbf{x})} \circ (\Phi(\mathbf{x}) - g(\mathbf{x})\mathbf{1})$. That means when the potential is larger than $g(\mathbf{x})$, the gradient is positive.

► **Definition 8.** Given a positive vector $\mathbf{x} \in \mathbb{R}^n$, we define the potentials of \mathbf{x} as the vector $\Phi(\mathbf{x}) \in \mathbb{R}^n$ such that

$$\Phi(\mathbf{x})_k = \|\mathbf{Ax}\|_p^{q-p} \frac{\langle \mathbf{A}_{\cdot,k}, (\mathbf{Ax})^{p-1} \rangle}{x_k^{q-1}}.$$

The algorithm does not follow the traditional gradient descent framework, however each iteration still follows the direction of the gradient. We want our iterate to satisfy the following corollary of Lemma 7.

► **Corollary 9.** Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x}^{(0)} \in \mathbb{R}^n$ such that $\|\mathbf{x}^{(0)}\|_q = 1$, $0 < q, p$, $0 < \varepsilon < 2/q$ and $V \geq 0$, assume the following invariant is satisfied at each iteration

$$\delta_i^{(t)} = \alpha x_i^{(t)} \implies \frac{1}{(1 + \alpha)^{q-1}} \Phi(\mathbf{x})_i \geq \left(\left(1 - \frac{\varepsilon}{2}\right) V \right)^q, \quad (2)$$

then, once $\|\mathbf{x}^{(T)}\|_q^q \geq \frac{4}{q\varepsilon}$, we have

$$\frac{\|\mathbf{Ax}^{(T)}\|_p}{\|\mathbf{x}^{(T)}\|_q} \geq (1 - \varepsilon)V.$$

For the iterate to satisfy the invariant in Corollary 9, there must always be a potential larger than V^q . This is guaranteed by the following lemma from [4].

► **Lemma 10** (Lemma 3.3 from [4]). Let \mathbf{A} be a non-negative matrix and $q \geq p \geq 1$. Given $\mathbf{x} > \mathbf{0}$, there is a coordinate k such that $\Phi(\mathbf{x})_k \geq \|\mathbf{A}\|_{q \rightarrow p}^q$.

As long as $V \leq \|\mathbf{A}\|_{q \rightarrow p}$, there always is a coordinate with potential above V^q . Setting $\alpha = \varepsilon/2$, at time t we scale coordinates with potential larger than V^q so that our iterate satisfies Invariant (2). At each iteration, a different coordinate may be scaled which means $\|\mathbf{x}^{(T)}\|_q$ increases quite slowly. We only have the following lower bound $\|\mathbf{x}^{(T)}\|_q^q \geq (1 + \alpha)^{T/n}$, which requires $T = \tilde{O}(n/q\varepsilon)$ iterations are required to solve the approximate decision problem.

3.2 A Width-Independent Algorithm

In order to achieve an efficient algorithm, it is necessary to improve the number of iterations by a factor of n . The challenge we had to deal before was that the ℓ_q norm of $\mathbf{x}^{(t)}$ was not increasing rapidly enough. To overcome this issue, we will modify the algorithm to ensure there is a coordinate k such that $\mathbf{x}_k^{(t)}$ is large. Notice that Lemma 10 guarantees that there is at least one coordinate with potential larger than V^q . The new invariant we would like to implement is that the set of coordinates with potential larger than V^q does not take any new elements. This requires to also scale coordinates with a potential slightly smaller than V^q which does not prevent satisfying the first Invariant (2). In order to do so, we need to analyze how the potentials evolve between two iterations.

► **Lemma 11.** *Given a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $q \geq p \geq 1$ and a positive vector \mathbf{x} , $\alpha > 0$. Let $\boldsymbol{\delta}$ be such that $\delta_i = \alpha x_i$ or $\delta_i = 0$, we have*

1. *If $\delta_i \neq 0$, then $\Phi(\mathbf{x} + \boldsymbol{\delta})_i \leq \Phi(\mathbf{x})_i$.*
2. *If $\delta_i = 0$, then $\Phi(\mathbf{x} + \boldsymbol{\delta})_i \leq (1 + \alpha)^{q-1} \Phi(\mathbf{x})_i$.*

Proof. No matter the value of δ_i , we have

$$\Phi(\mathbf{x} + \boldsymbol{\delta})_i = \|\mathbf{A}(\mathbf{x} + \boldsymbol{\delta})\|_p^{q-p} \frac{\langle \mathbf{A}_{\cdot, i}, (\mathbf{A}(\mathbf{x} + \boldsymbol{\delta}))^{p-1} \rangle}{(\mathbf{x} + \boldsymbol{\delta})_i^{q-1}} \leq (1 + \alpha)^{q-1} \|\mathbf{A}\mathbf{x}\|_p^{q-p} \frac{\langle \mathbf{A}_{\cdot, i}, (\mathbf{A}\mathbf{x})^{p-1} \rangle}{(\mathbf{x})_i^{q-1}}.$$

If $\delta_i = \alpha x_i$, then $\Phi(\mathbf{x} + \boldsymbol{\delta})_i \leq \Phi(\mathbf{x})_i$, otherwise, $\Phi(\mathbf{x} + \boldsymbol{\delta})_i \leq (1 + \alpha)^{q-1} \Phi(\mathbf{x})_i$. ◀

As mentioned earlier, if we consider a multiplicative scaling $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \boldsymbol{\delta}^{(t)}$ where $\delta_i^{(t)} \neq 0$ if and only if the potential of coordinate i is larger than θ for some θ , then it creates a glass ceiling where no new coordinates can have potential noticeably larger than θ . We quantify this upper bound in the following corollary.

► **Corollary 12.** *Assume at each iteration we scale coordinates with potential larger than θ , then the set of coordinates with potential larger than $(1 + \alpha)^{q-1} \theta$ is decreasing.*

Proof. Let $C^{(t)}$ be the set of coordinates with potential larger than $(1 + \alpha)^{q-1} \theta$ at time t . For a coordinate i that is not in $C^{(t)}$, we have that if it is scaled at time t , then by Lemma 11, $\Phi(\mathbf{x}^{(t+1)})_i \leq \Phi(\mathbf{x}^{(t)})_i$, hence i is not part of $C^{(t+1)}$. Moreover, if i is not scaled, then we know $\Phi(\mathbf{x}^{(t)})_i \leq \theta$, hence using Lemma 11, $\Phi(\mathbf{x}^{(t+1)})_i \leq (1 + \alpha)^{q-1} \theta$. Thus, $i \notin C^{(t+1)}$. ◀

We choose α and the threshold such that the glass ceiling is equal to V^q . That means that no new coordinates can have their potential larger than V^q . At the same time, we need to be careful about our choice of α and threshold so that Invariant (2) is still satisfied.

Assume we hit coordinates with potential larger than $((1 - \varepsilon/4)V)^q$ with $\alpha = \varepsilon/8$, we ensure that Invariant (2) is satisfied since

$$\frac{1}{(1 + \frac{\varepsilon}{8})^{q-1}} \left(\left(1 - \frac{\varepsilon}{4}\right) V \right)^q \geq \left(\left(1 - \frac{\varepsilon}{2}\right) V \right)^q.$$

Moreover, using Corollary 12, we ensure that the set of coordinates with potential larger than $(1 + \varepsilon/8)^{q-1} (1 - \varepsilon/4)^q V^q < V^q$ is decreasing. We also know that at the end of the algorithm at least one coordinate has potential larger than V^q using Lemma 10. Hence, those coordinates were scaled at each iteration and are equal to $(1 + \alpha)^T \mathbf{x}^{(0)}$. This gives the following theorem which we prove in Subsection E.2.

69:10 Approximating $q \rightarrow p$ Norms of Non-Negative Matrices in Nearly-Linear Time

► **Theorem 13.** *Given a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a positive real $0 < \varepsilon \leq 1/2q$, two reals $q \geq p \geq 1$, and a guess V on $\|\mathbf{A}\|_{q \rightarrow p}$, Algorithm 1 recovers \mathbf{x} such that*

$$\frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} \geq (1 - \varepsilon)V$$

or certifies infeasibility in time

$$O\left(\frac{1}{q\varepsilon} \log\left(\frac{n}{q\varepsilon}\right) (\text{nnz}(\mathbf{A}) + m \log p)\right).$$

■ **Algorithm 1** Approximating $\ell_{q \rightarrow p}$ -norm of non-negative matrices.

input $\mathbf{A} \in \mathbb{R}_{\geq 0}^{m \times n}$, V , ε , and $q \geq p \geq 1$.
output \mathbf{x} such that $\frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} \geq (1 - \varepsilon)V$ or infeasibility certificate.
 $\mathbf{x} \leftarrow n^{-1/q} \mathbf{1}$
 $\alpha \leftarrow \varepsilon/8$
while $\frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} < (1 - \varepsilon)V$ **do**
 $\delta_i \leftarrow \begin{cases} \alpha x_i & \text{if } \Phi(\mathbf{x})_i \geq ((1 - \varepsilon/4)V)^q \\ 0 & \text{otherwise} \end{cases}$
 $\mathbf{x} \leftarrow \mathbf{x} + \delta$
 if for all i , $\Phi(\mathbf{x})_i < V^q$ **then**
 return “infeasible”
 end if
end while
return \mathbf{x}

The algorithm can be parallelized since computing the potentials $\Phi(\mathbf{x})$ requires computing two matrix-vector multiplications. In the PRAM model, matrix-vector multiplication runs in $O(n^2/P)$ time on a CRCW PRAM and $O(n^2/P + \log n)$ on an EREW PRAM with P processors. With $P = \Theta(n^2)$ processors, the depth becomes $\tilde{O}(1)$, hence we give as a corollary the running time obtained if we perform those operations in parallel.

► **Corollary 14.** *Given a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a positive real $0 < \varepsilon \leq 1/2q$, two reals $q \geq p \geq 1$, and a guess V on $\|\mathbf{A}\|_{q \rightarrow p}$, it is possible to compute \mathbf{x} such that $\|\mathbf{A}\mathbf{x}\|_p / \|\mathbf{x}\|_q \geq (1 - \varepsilon) \|\mathbf{A}\|_{q \rightarrow p}$ in parallel time $\tilde{O}(\frac{1}{q\varepsilon})$ with total work $\tilde{O}(\frac{\text{nnz}(\mathbf{A})}{q\varepsilon})$.*

Using the Lewis weights sampling procedure from [7, 23], we can obtain the following corollary which we prove in Subsection E.3.

► **Corollary 15.** *Given a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a positive real $0 < \varepsilon \leq 1/2q$, two reals $q \geq p \geq 1$, and a guess V on $\|\mathbf{A}\|_{q \rightarrow p}$, by first preconditioning the input using Theorem 5, Algorithm 1 returns \mathbf{x} such that*

$$\frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_q} \geq (1 - \varepsilon)V$$

or certifies infeasibility in time

$$\tilde{O}\left(\frac{r}{q\varepsilon^3} n^{\max\{\frac{p}{2}, 1\}} + \text{nnz}(\mathbf{A}) + n^\omega\right).$$

Where r is the maximum number of nonzero entries in a row of \mathbf{A} .

References

- 1 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering lp solvers: Achieving width-independence and-convergence. *Mathematical Programming*, 175:307–353, 2019. doi:10.1007/S10107-018-1244-X.
- 2 Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 383–388, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780599.
- 3 Boaz Barak, Fernando G.S.L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, STOC'12. ACM, May 2012. doi:10.1145/2213977.2214006.
- 4 Aditya Bhaskara and Aravindan Vijayaraghavan. Approximating matrix p-norms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 497–511, USA, 2011. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973082.40.
- 5 Vijay Bhattiprolu, Mrinalkanti Ghosh, Venkatesan Guruswami, Euiwoong Lee, and Madhur Tulsiani. Approximability of $p \rightarrow q$ matrix norms: generalized krivine rounding and hypercontractive hardness. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1358–1368, USA, 2019. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975482.83.
- 6 David W. Boyd. The power method for lp norms. *Linear Algebra and its Applications*, 9:95–101, 1974. doi:10.1016/0024-3795(74)90029-9.
- 7 Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. *CoRR*, abs/1412.0588, 2014. arXiv:1412.0588.
- 8 Alina Ene and Adrian Vladu. Improved convergence for ℓ_1 and ℓ_∞ regression via iteratively reweighted least squares. In *International Conference on Machine Learning*, pages 1794–1801. PMLR, 2019.
- 9 Matthias Englert and Harald Räcke. Oblivious routing for the lp-norm. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–40, 2009. doi:10.1109/FOCS.2009.52.
- 10 Gramoz Goranci, Monika H Henzinger, Harald Räcke, Sushant Sachdeva, and AR Sricharan. Electrical flows for polylogarithmic competitive oblivious routing. In *15th Innovations in Theoretical Computer Science Conference*, volume 287, 2024.
- 11 Anupam Gupta, Mohammad Hajiaghayi, and Harald Räcke. Oblivious network design. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979, January 2006. doi:10.1145/1109557.1109665.
- 12 Julien M. Hendrickx and Alex Olshevsky. Matrix p-norms are np-hard to approximate if $p \neq 1, 2, \infty$, 2010. arXiv:0908.1397.
- 13 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001. doi:10.1145/502090.502098.
- 14 Arun Jambulapati, James R. Lee, Yang P. Liu, and Aaron Sidford. Sparsifying generalized linear models. *CoRR*, abs/2311.18145, 2023. doi:10.48550/arXiv.2311.18145.
- 15 Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games and its applications. *CoRR*, abs/2004.04250, 2020. arXiv:2004.04250.
- 16 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.
- 17 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, 1993. doi:10.1145/167088.167211.

- 18 Michael W. Mahoney, Satish Rao, Di Wang, and Peng Zhang. Approximating the solution to mixed packing and covering lps in parallel $o(\epsilon^{-3})$ time. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2016.52.
- 19 H. Racke. Minimizing congestion in general networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 43–52, 2002. doi:10.1109/SFCS.2002.1181881.
- 20 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08*, pages 255–264, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374415.
- 21 Ali Kemal Sinop, Lisa Fawcett, Sreenivas Gollapudi, and Kostas Kollias. Robust routing using electrical flows. *ACM Trans. Spatial Algorithms Syst.*, 9(4), November 2023. doi:10.1145/3567421.
- 22 Daureen Steinberg. Computation of matrix norms with application to robust optimization. 2005. URL: <https://www2.isye.gatech.edu/~nemirovs/Daureen.pdf>.
- 23 David P. Woodruff and Taisuke Yasuda. Online lewis weight sampling, 2022. doi:10.48550/arXiv.2207.08268.
- 24 Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 538–546. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959930.

A From Approximate Decision to Approximate Optimization

Our algorithm solves an approximate decision problem. When performing the binary search for the guess value V , a precision of ϵ is not required. We provide here a standard relaxation that transforms an approximate decision algorithm into an approximate optimization one.

First, notice that with $\kappa := \max_{i,j} \mathbf{A}_{i,j}$, we have for any $q \geq p \geq 1$, $\kappa \leq \|\mathbf{A}\|_{q \rightarrow p} \leq \kappa n^{1-\frac{1}{q}} m^{\frac{1}{p}}$. Given a guess value V and a precision ϵ , Theorem 13 gives that either $\|\mathbf{A}\|_{q \rightarrow p}$ is in $[(1-\epsilon)V, +\infty)$ or $[0, V]$. We initialize our search interval as $[\kappa, n^{1-\frac{1}{q}} m^{\frac{1}{p}} \kappa]$.

Given a search interval $[L, U]$, we let $V = \sqrt{LU}$ and $\tilde{\epsilon} := \min\{\frac{1}{2q}, (\frac{U}{L})^{\frac{1}{6}} - 1\}$. We run Algorithm 1 with $\epsilon = \tilde{\epsilon}$ and V as the guess value. If the algorithm returns a vector \mathbf{x} such that $\frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_q} \geq (1-\tilde{\epsilon})V$, we update $L = V(1-\tilde{\epsilon})$, otherwise we update $U = V$. We repeat this process until $U/L \leq 1/(1-\frac{\epsilon}{4})$. When this happens, we chose $V = L$ and use precision $\frac{\epsilon/4}{1+\epsilon/4}$. Since we maintain that L is feasible and U is infeasible, we obtain \mathbf{x} such that

$$\frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_q} \geq \left(1 - \frac{\epsilon/4}{1+\epsilon/4}\right) L \geq \frac{1-\frac{\epsilon}{4}}{1+\frac{\epsilon}{4}} U \geq (1-\epsilon) \|\mathbf{A}\|_{q \rightarrow p}.$$

We now analyze the cost of the search. While $\frac{U}{L} > \left(1 + \frac{1}{2q}\right)^6$, we invoke the algorithm with precision $\frac{1}{2q}$ and $\log U/L$ is divided by a constant fraction. Hence, this process is repeated at most $O(\log \log(n^{1-\frac{1}{q}} m^{\frac{1}{p}}))$ times. Moreover, when $\frac{U}{L} \leq \left(1 + \frac{1}{2q}\right)^6$, we use precision $\exp((\log U/L)/6) - 1 = \Theta(\log U/L)$. Since the running time of an iteration is proportional to $\frac{1}{\epsilon}$, the total running time of this part of the search is dominated by the last one with precision $\tilde{\epsilon} = O(\epsilon)$. Hence, the total running time is $O\left(\mathcal{T}\left(\frac{1}{2q}\right) \log \log(n^{1-\frac{1}{q}} m^{\frac{1}{p}}) + \mathcal{T}(\epsilon)\right)$ where $\mathcal{T}(\epsilon)$ is the running time of Algorithm 1 with precision ϵ .

B Multicommodity Flows

B.1 Definitions

B.1.1 Multicommodity Flow Problem

We describe the general multicommodity flow framework introduced by Gupta et al. in [11]. We are given $G = (V, E)$ an unweighted graph with n vertices and m edges and \mathbf{R} a collection of routing requests $\mathbf{R} = \langle R_i = (s_i, t_i, d_i, k_i) \rangle$. Each routing request R_i consists of a source-target pair (s_i, t_i) , a non-negative amount of traffic d_i and a type of routing $k_i \in \llbracket 1, K \rrbracket$. One can think of k_i as a quality of service for instance. Each routing request is called a *commodity* hence this problem is called the *multicommodity* flow problem.

The cost of a multicommodity flow is a function of the load. For each edge $e \in E$, $\mathbf{G}_{e,k}$ denotes the amount of flow of type k that is sent along e . Each type of routing can induce different load on the same edge. Given a function $l : \mathbb{R}^K \rightarrow \mathbb{R}$, we can define the load $L(e)$ of the edge

$$L(e) = l([\mathbf{G}_{e,1}, \dots, \mathbf{G}_{e,K}]).$$

The cost is calculated by aggregating the edges via a function $\text{AGG} : \mathbb{R}^m \rightarrow \mathbb{R}$

$$\text{COST} = \text{AGG}([L(e_1), \dots, L(e_m)]).$$

This problem captures a lot of problems, for $l = \sum$ and $L = \max$ for instance, this captures the congestion of the framework.

Instead of using a collection of routing requests \mathbf{R} , we consider a demand matrix \mathbf{D} with $n(n-1)$ rows and K columns. Each row corresponds to a directed pair of vertices and each column to a type of routing. The value of the matrix at row i and column k is the non-negative amount of traffic of type k that needs to be routed from the source-target pair of the i -th line. We denote \mathcal{D} the set of all demand matrices². This notation allows constructing linear oblivious routings as matrices (hence linear).

B.1.2 Linear Oblivious Routings

The multicommodity flow problem allows for a rather simple routing construction, for each source-target pair of vertices (i, j) we route according to a precomputed unit non-negative flow $\mathbf{f}_{(i,j)} \in \mathbb{R}^m$ from i to j . Those flows can be concatenated to form a non-negative matrix $\mathbf{A} \in \mathbb{R}^{m \times n(n-1)}$. Given a demand matrix $\mathbf{D} \in \mathcal{D}$, we call \mathbf{F} the flow induced by routing the demand \mathbf{D} with the oblivious routing \mathbf{A} . This flow matrix has k columns and m rows. Each one of its column is obtained by multiplying \mathbf{A} with a column of \mathbf{D} : $\mathbf{F}_{\cdot,i} = \mathbf{A}\mathbf{D}_{\cdot,i}$. For the sake of simplicity, we write $\mathbf{F} = \mathbf{A}(\mathbf{D})$. This flow matrix induces the load $L(e) = l(\mathbf{A}(\mathbf{D})_e)$ on edge e . We call \mathbf{A} a linear oblivious routing. Let P_2^n be the set of oriented pairs of vertices and P_3^n the set of oriented triplets of vertices.

² Hence $\mathcal{D} = \{\mathbf{D} \in \mathbb{R}_{\geq 0}^{n(n-1) \times K} : \mathbf{D} \neq \mathbf{0}\}$

► **Definition 16.** A matrix \mathbf{A} is a linear oblivious routing if and only if it is a solution of the following system.

$$\begin{aligned} \forall e \in E, \forall (i, j) \in P_2^n : & \mathbf{A}_{e,(i,j)} \geq 0 \\ \forall (i, j) \in P_2^n : & \sum_{e \in \text{OUT}(i)} \mathbf{A}_{e,(i,j)} - \sum_{e \in \text{IN}(i)} \mathbf{A}_{e,(i,j)} = 1 \\ \forall (k, i, j) \in P_3^n : & \sum_{e \in \text{OUT}(k)} \mathbf{A}_{e,(i,j)} - \sum_{e \in \text{IN}(k)} \mathbf{A}_{e,(i,j)} = 0 \end{aligned}$$

We call \mathcal{R}_{lin} the set of all linear oblivious routings in the graph G . A linear oblivious routing \mathbf{A} on a demand \mathbf{D} creates a flow $\mathbf{A}(\mathbf{D})$ with cost

$$\text{COST}(\mathbf{A}(\mathbf{D})) = \text{AGG}(L(l(\mathbf{A}(\mathbf{D})))).$$

In this report, we focus on the case where COST is a monotone norm $\|\cdot\|$ from $\mathbb{R}^{m \times K}$ to \mathbb{R} .

B.1.3 Optimal Flows and Competitive Ratio

A multicommodity flow \mathbf{F} induces a cost $\text{AGG}(L(l(\mathbf{F})))$. The optimal cost of a demand matrix is $\text{OPT}(\mathbf{D}) := \min_{R \in \mathcal{R}_{\text{lin}}} \text{COST}(R\mathbf{D})$. The competitive ratio of the routing \mathbf{A} is its worst performance against OPT.

► **Definition 17.** The competitive ratio of \mathbf{A} is defined as

$$\text{COMPETITIVE-RATIO}(\mathbf{A}) = \max_{\mathbf{D} \in \mathcal{D}} \frac{\text{COST}(\mathbf{A}(\mathbf{D}))}{\text{OPT}(\mathbf{D})}.$$

Since OPT is linear, we may rescale the COST function to ensure that if $\text{COST}(\mathbf{D}) \leq 1$, then $\max_{i,j} \mathbf{D}_{i,j} \leq 1$. This is possible since all norms are equivalent in finite dimension, moreover, this does not change the value of the competitive ratio.

► **Definition 18.** The competitive ratio of \mathbf{A} is

$$\text{COMPETITIVE-RATIO}(\mathbf{A}) = \max_{\mathbf{D} \in \text{OPT}_{\leq 1}} \|\mathbf{A}(\mathbf{D})\|$$

where $\text{OPT}_{\leq 1} := \{\mathbf{D} \in \mathcal{D} : \text{OPT}(\mathbf{D}) \leq 1\}$.

An optimal linear oblivious routing is a linear oblivious routing with the lowest competitive ratio among the competitive ratio of the others.

B.2 From Undirected to Directed Graphs

Consider an undirected graph $G = (V, E)$ with n vertices and m edges. We follow the same transformation as of [2]. For any undirected edge (u, v) , [2] consider the directed gadget u, v, x, y with 5 directed edges. The five directed edges are $e_1 = (u, x), e_2 = (v, x), e_3 = (y, u), e_4 = (y, v)$ and $e_5 = (x, y)$. We call e_5 the replacing edge of (u, v) . The new graph is a directed one, moreover, a linear oblivious routing on the new directed graph gives a linear oblivious routing on the undirected graph.

C Computing an Optimal Oblivious Routing for a Fixed Monotone Norm

We explain in Subsection B.1 what multicommodity flows are and the assumptions we are making. Oblivious routings have been computed under two different regimes. The first regime is when the aggregating function is an ℓ_p -norm, and the load function is unknown. In that case, linear routings with $O(\log n)$ competitive ratio exists [19, 20, 9, 4]. When p is not 1, 2 or ∞ , the running time is polynomial with an extremely large exponent. The analysis made by [4] requires precision ε smaller than m^{-90} for approximating the p -norm of the matrix. This time complexity is too large for real world usage. In the second regime, one assumes the aggregating function, and the load function are known. They both make the cost function, hence in this regime, we assume the cost function is a known monotone norm. This regime was analyzed when the cost function was the ℓ_∞ norm [2], we extend it to any monotone norm. Our goal is to find an optimal linear oblivious routing, i.e. a linear oblivious routing \mathbf{A}^* such that:

$$\forall \mathbf{A} \in \mathcal{R}_{\text{lin}}, \text{COMPETITIVE-RATIO}(\mathbf{A}^*) \leq \text{COMPETITIVE-RATIO}(\mathbf{A}).$$

In the following, we assume the graph to be directed. Undirected graphs can be transformed in directed graphs, hence our results are also valid for undirected graphs (c.f. Subsection B.2 for more details).

► **Theorem 19.** *For any directed or undirected graph $G = (V, E)$ and any cost function that is a monotone norm on the load vector, it is possible to compute a linear oblivious routing \mathbf{A} such that*

$$\text{COMPETITIVE-RATIO}(\mathbf{A}) \leq \min_{\mathbf{A}} \text{COMPETITIVE-RATIO}(\mathbf{A}) + \varepsilon$$

in time $\tilde{O}(n^6 m^3 \log \frac{n}{\varepsilon})$ with high probability in n .

To construct the linear oblivious routing from Theorem 19, we will use the following lemma where \mathcal{X} represents a relaxation of the set of linear routings and \mathcal{Y} a relaxation of the set of demand vectors.

► **Lemma 20.** *There exist two convex sets $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^{m \times n(n-1)}$ such that $\mathcal{R}_{\text{lin}} \subseteq \mathcal{X}$ and given $\hat{\mathbf{A}} \in \mathcal{X}$ and $\hat{\mathbf{Q}} \in \mathcal{Y}$ satisfying*

$$\max_{\mathbf{Q} \in \mathcal{Y}} \langle \hat{\mathbf{A}}, \mathbf{Q} \rangle_F - \min_{\mathbf{A} \in \mathcal{X}} \langle \mathbf{A}, \hat{\mathbf{Q}} \rangle_F \leq \varepsilon,$$

we can construct in polynomial time a linear oblivious routing $\tilde{\mathbf{A}}$ such that

$$\text{COMPETITIVE-RATIO}(\tilde{\mathbf{A}}) \leq \min_{\mathbf{A}} \text{COMPETITIVE-RATIO}(\mathbf{A}) + \varepsilon.$$

To see how Lemma 20 is useful, notice that using a convex-concave game solver we can find $\hat{\mathbf{A}}$ and $\hat{\mathbf{Q}}$ satisfying the condition of Lemma 20. The next theorem is a convex-concave game solver from [15] that we will use to prove the running time of Theorem 19.

► **Theorem 21** (Theorem C.9 from [15]). *Given convex sets $\mathcal{X} \subset B(0, R) \subset \mathbb{R}^a$ and $\mathcal{Y} \subset B(0, R) \subset \mathbb{R}^b$ such that both \mathcal{X} and \mathcal{Y} contain a ball of radius r . Let $H(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ be an L -Lipschitz function that is convex in \mathbf{x} and concave in \mathbf{y} . For any $0 < \varepsilon \leq \frac{1}{2}$, we can find $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ such that*

$$\max_{\mathbf{y} \in \mathcal{Y}} H(\hat{\mathbf{x}}, \mathbf{y}) - \min_{\mathbf{x} \in \mathcal{X}} H(\mathbf{x}, \hat{\mathbf{y}}) \leq \varepsilon L r$$

69:16 Approximating $q \rightarrow p$ Norms of Non-Negative Matrices in Nearly-Linear Time

in time

$$O\left((a+b)^3 \log\left(\frac{a+bR}{\varepsilon r}\right) + (a+b)\mathcal{J} \log\left(\frac{a+bR}{\varepsilon r}\right)\right)$$

with high probability in $a+b$ where \mathcal{J} is the cost of computing sub-gradient ∇f .

C.1 Relaxing the Problem

C.1.1 Relaxing the Set of Linear Routings

Notice that \mathcal{R}_{lin} as defined in Definition 16 is convex, however it does not contain a ball of radius r for $r > 0$. Hence, we relax the problem to solving on \mathcal{X} where

$$\mathcal{X} := \left\{ \mathbf{A} \in \mathbb{R}^{m \times n(n-1)} : \begin{array}{ll} \forall e \in E, (i, j) \in V \times V & \mathbf{A}_{e, (ij)} \geq 0 \\ \wedge \forall e \in E, (i, j) \in V \times V & \mathbf{A}_{e, (ij)} \leq 2 \\ \wedge \forall (i, j) \in V \times V & \sum_{e \in \text{OUT}(i)} \mathbf{A}_{e, (ij)} - \sum_{e \in \text{IN}(i)} \mathbf{A}_{e, (ij)} \geq 1 \\ \wedge \forall (k, i, j) \in V \times V \times V & \sum_{e \in \text{OUT}(k)} \mathbf{A}_{e, (ij)} - \sum_{e \in \text{IN}(k)} \mathbf{A}_{e, (ij)} \geq 0 \end{array} \right\}.$$

We have that $\mathcal{R}_{\text{lin}} \subseteq \mathcal{X}$, and \mathcal{X} contains a ball of radius of $r = m^{-O(1)}$ and is inside $B(0, 2)$. The following claim is a result of $\|\cdot\|$ being monotone.

▷ Claim 22. Given a demand matrix \mathbf{D} , we have

$$\min_{\mathbf{A} \in \mathcal{R}_{\text{lin}}} \|\mathbf{AD}\| = \min_{\mathbf{A} \in \mathcal{X}} \|\mathbf{AD}\|.$$

C.1.2 Relaxing the Set of Demand Vectors

Let $\text{OPT}_{\leq 1}$ be the set of demand matrices \mathbf{D} such that $\min_{\mathbf{A} \in \mathcal{R}_{\text{lin}}} \|\mathbf{AD}\| \leq 1$. Using Claim 22, we have that $\text{OPT}_{\leq 1}$ can also be defined as the set of demand matrices \mathbf{D} such that $\min_{\mathbf{A} \in \mathcal{X}} \|\mathbf{AD}\| \leq 1$. Since \mathcal{X} only contains non-negative matrices, we have with S_* the set of non-negative matrices with $\mathbb{R}^{m \times K}$ coordinates and dual norm less than 1 that $\|\mathbf{AD}\| = \max_{\mathbf{G} \in S_*} \langle \mathbf{A}, \mathbf{GD}^T \rangle_F$. Let $\mathcal{Y}_1 := \{\mathbf{GD}^T : \mathbf{G} \in S_*, \mathbf{D} \in \text{OPT}_{\leq 1}\}$, we want to find $\min_{\mathbf{A} \in \mathcal{X}} \max_{\mathbf{Q} \in \mathcal{Y}_1} \langle \mathbf{A}, \mathbf{Q} \rangle_F$. However, \mathcal{Y}_1 is not convex, we relax it to $\mathcal{Y} = \text{conv}(\mathcal{Y}_1)$. The following claim is a consequence of $\mathbf{Q} \mapsto \langle \mathbf{A}, \mathbf{Q} \rangle_F$ being linear.

▷ Claim 23. Given a matrix \mathbf{A} from \mathcal{X} , we have

$$\max_{\mathbf{Q} \in \mathcal{Y}_1} \langle \mathbf{A}, \mathbf{Q} \rangle_F = \max_{\mathbf{Q} \in \mathcal{Y}} \langle \mathbf{A}, \mathbf{Q} \rangle_F.$$

C.1.3 Solving the Relaxed Problem

► Lemma 24. Assume $\|\cdot\|$ is a monotonic norm, let $\|\cdot\|_*$ be its dual norm. Define

$$\mathcal{X} := \left\{ \mathbf{A} \in \mathbb{R}^{m \times n(n-1)} : \begin{array}{ll} \forall e \in E, (i, j) \in V \times V & \mathbf{A}_{e, (ij)} \geq 0 \\ \wedge \forall e \in E, (i, j) \in V \times V & \mathbf{A}_{e, (ij)} \leq 2 \\ \wedge \forall (i, j) \in V \times V & \sum_{e \in \text{OUT}(i)} \mathbf{A}_{e, (ij)} - \sum_{e \in \text{IN}(i)} \mathbf{A}_{e, (ij)} \geq 1 \\ \wedge \forall (k, i, j) \in V \times V \times V & \sum_{e \in \text{OUT}(k)} \mathbf{A}_{e, (ij)} - \sum_{e \in \text{IN}(k)} \mathbf{A}_{e, (ij)} \geq 0 \end{array} \right\},$$

and

$$\mathcal{Y} := \text{conv}\{\mathbf{GD}^T : \mathbf{D} \in \text{OPT}_{\leq 1}, \mathbf{G} \in S_*\}.$$

Given $\widehat{\mathbf{A}} \in \mathcal{X}$ and $\widehat{\mathbf{Q}} \in \mathcal{Y}$ such that

$$\max_{\mathbf{Q} \in \mathcal{Y}} \langle \widehat{\mathbf{A}}, \mathbf{Q} \rangle_F - \min_{\mathbf{A} \in \mathcal{X}} \langle \mathbf{A}, \widehat{\mathbf{Q}} \rangle_F \leq \varepsilon,$$

we can construct in polynomial time a linear oblivious routing $\widetilde{\mathbf{A}}$ such that

$$\text{COMPETITIVE-RATIO}(\widetilde{\mathbf{A}}) \leq \min_{\mathbf{A}} \text{COMPETITIVE-RATIO}(\mathbf{A}) + \varepsilon.$$

To see how Lemma 24 is helpful, notice that from $\widehat{\mathbf{A}}$, we can construct $\widetilde{\mathbf{A}}$ that satisfies Definition 16 by reducing the coordinates of $\widehat{\mathbf{A}}$. Since the cost function is monotone, for any $\mathbf{Q} \in \mathcal{Y}$, $\langle \widetilde{\mathbf{A}}, \mathbf{Q} \rangle \leq \langle \widehat{\mathbf{A}}, \mathbf{Q} \rangle$. Moreover, $\min_{\mathbf{A} \in \mathcal{X}} \langle \mathbf{A}, \widehat{\mathbf{Q}} \rangle_F \leq \text{COMPETITIVE-RATIO}(\mathbf{A}^*)$ since the minimizer over \mathcal{X} is in \mathcal{R}_{lin} . Hence, we have $\text{COMPETITIVE-RATIO}(\widetilde{\mathbf{A}}) \leq \text{COMPETITIVE-RATIO}(\mathbf{A}^*) + \varepsilon$. Moreover, we get $\widehat{\mathbf{A}}, \widehat{\mathbf{Q}}$ by using Theorem 21.

D Proofs from Section 2

D.1 Proof of Theorem 3

► **Lemma 25.** For any non-negative vector $\mathbf{z} \in \mathbb{R}_{\geq 0}^n$, and $q \geq p \geq 1$, we have $g_{\mathbf{z}} : \mathbf{u} \mapsto \langle \mathbf{z}, \mathbf{u}^{1/q} \rangle^p$ is concave on $\mathbb{R}_{\geq 0}^n$.

Proof of Lemma 25. Given a non-negative vector \mathbf{z} , let $f_{\mathbf{z}} : \mathbf{x} \mapsto \langle \mathbf{z}, \mathbf{x}^{1/q} \rangle^q$. Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}_{\geq 0}^n$ and $t \in [0, 1]$. Define $\mathbf{x}_i := tz_i^q \mathbf{u}_i$ and $\mathbf{y}_i := (1-t)z_i^q \mathbf{v}_i$. We have

$$\begin{aligned} f_{\mathbf{z}}(t\mathbf{u} + (1-t)\mathbf{v}) &= \left(\sum_{i=1}^n z_i (t\mathbf{u}_i + (1-t)\mathbf{v}_i)^{1/q} \right)^q \\ &= \left(\sum_{i=1}^n (tz_i^q \mathbf{u}_i + (1-t)z_i^q \mathbf{v}_i)^{1/q} \right)^q \\ &= \left(\sum_{i=1}^n (\mathbf{x}_i + \mathbf{y}_i)^{1/q} \right)^q \\ &\geq \left(\sum_{i=1}^n \mathbf{x}_i^{1/q} \right)^q + \left(\sum_{i=1}^n \mathbf{y}_i^{1/q} \right)^q \\ &= t \left(\sum_{i=1}^n z_i \mathbf{u}_i^{1/q} \right)^q + (1-t) \left(\sum_{i=1}^n z_i \mathbf{v}_i^{1/q} \right)^q \\ &= t f_{\mathbf{z}}(\mathbf{u}) + (1-t) f_{\mathbf{z}}(\mathbf{v}). \end{aligned}$$

Where the inequality follows from Minkowski's inequality. Moreover, since $q \geq p$, $x \mapsto {}^{q/p}\sqrt{x}$ is concave and non-decreasing, hence $g_{\mathbf{z}} = {}^{q/p}\sqrt{f_{\mathbf{z}}}$ is concave. ◀

We are now ready to prove Theorem 3.

Proof. Let $g : \mathbf{x} \mapsto \|\mathbf{A}\mathbf{x}^{1/q}\|_p^p$. Let \mathbf{z}_i be the i^{th} row of \mathbf{A} . We have $g(\mathbf{x}) = \sum_{i=1}^m g_{\mathbf{z}_i}(\mathbf{x})$. Hence, by Lemma 25, g is concave. Moreover, since $x \mapsto {}^{q/p}\sqrt{x}$ is concave and non-decreasing, we have that $\mathbf{x} \mapsto {}^{q/p}\sqrt{g(\mathbf{x})} = \|\mathbf{A}\mathbf{x}^{1/q}\|_p$ is concave. ◀

E Proofs from Section 3**E.1** Proof of Lemma 7

Proof. We have

$$\begin{aligned}
\frac{\|\mathbf{Ax}^{(T)}\|_q^q}{\|\mathbf{x}^{(T)}\|_q^q} &= \frac{\|\mathbf{Ax}^{(0)}\|_p^q + \sum_{t=0}^{T-1} \|\mathbf{Ax}^{(t+1)}\|_p^q - \|\mathbf{Ax}^{(t)}\|_p^q}{\|\mathbf{x}^{(T)}\|_q^q} \\
&\geq \frac{\|\mathbf{Ax}^{(0)}\|_p^q}{\|\mathbf{x}^{(T)}\|_q^q} + \left(1 - \frac{\varepsilon}{2}\right)^q \frac{\sum_{t=0}^{T-1} \|\mathbf{x}^{(t+1)}\|_q^q - \|\mathbf{x}^{(t)}\|_q^q}{\|\mathbf{x}^{(T)}\|_q^q} \\
&\geq \left(1 - \frac{\varepsilon}{2}\right)^q \left(1 - \frac{\|\mathbf{x}^{(0)}\|_q^q}{\|\mathbf{x}^{(T)}\|_q^q}\right) \\
&= \left(1 - \frac{\varepsilon}{2}\right)^q \left(1 - \frac{1}{\|\mathbf{x}^{(T)}\|_q^q}\right).
\end{aligned}$$

Hence, once $1 - \frac{1}{\|\mathbf{x}^{(T)}\|_q^q} \geq (1 - \varepsilon/2)^q$, we have

$$\frac{\|\mathbf{Ax}^{(T)}\|_p}{\|\mathbf{x}^{(T)}\|_q} \geq (1 - \varepsilon)V.$$

Since $\varepsilon q \leq 2$, we have $(1 - \varepsilon/2)^q \leq 1 - \varepsilon q/4$, hence it is sufficient to have $\|\mathbf{x}^{(T)}\|_q^q \geq 4/q\varepsilon$. ◀

E.2 Proof of Theorem 13

Proof. First, if the algorithm outputs “infeasible”, we know by Lemma 10 that it means $V > \|\mathbf{A}\|_{q \rightarrow p}$. Hence, assuming the algorithm does not return “infeasible”, we will prove it outputs \mathbf{x} such that $\|\mathbf{Ax}\|_p / \|\mathbf{x}\|_q \geq (1 - \varepsilon)V$.

First, we prove that the invariant of Corollary 9 is satisfied. At iteration t , if coordinate i is scaled, then

$$\frac{1}{(1 + \alpha)^{q-1}} \Phi(\mathbf{x}^{(t)})_i \geq \frac{1}{(1 + \frac{\varepsilon}{8})^{q-1}} \left(1 - \frac{\varepsilon}{4}\right)^q \geq \left(1 - \frac{\varepsilon}{2}\right)^q.$$

Moreover, using Corollary 12, we know that the set of coordinates with potential larger than $\left(1 - \frac{\varepsilon}{4}\right)^q (1 + \frac{\varepsilon}{8})^{q-1} < V^q$ is decreasing. However, using Lemma 10, we know at least one coordinate has potential larger than V^q at iteration T . Hence, this coordinate always had potential larger than V^q . Assume k is such a coordinate, we have

$$\|\mathbf{x}^{(T)}\|_q^q \geq \left(\mathbf{x}_k^{(T)}\right)^q = (1 + \alpha)^{qT} n^{-1}.$$

With $T \geq \frac{8}{q\varepsilon} \log \frac{4n}{q\varepsilon}$, we have $(1 + \alpha)^{qT} n^{-1} \geq \frac{4}{q\varepsilon}$ which implies

$$\frac{\|\mathbf{Ax}^{(T)}\|_p}{\|\mathbf{x}^{(T)}\|_q} \geq (1 - \varepsilon)V$$

using Corollary 9. ◀

E.3 Proof of Corollary 15

Proof. Using Theorem 5, with $\tilde{\varepsilon} = \varepsilon/4$, we obtain a matrix \mathbf{A}' such that $(1 - \tilde{\varepsilon}) \|\mathbf{A}'\mathbf{x}\|_p \leq \|\mathbf{Ax}\|_p \leq (1 + \tilde{\varepsilon}) \|\mathbf{A}'\mathbf{x}\|_p$. Computing this matrix cost $\tilde{O}(\text{nnz}(\mathbf{A}) + n^\omega)$. Since \mathbf{A} is non-negative, then so is \mathbf{A}' as it corresponds to rescaled rows of \mathbf{A} . Hence, we can use Theorem 13 on \mathbf{A}' with $\tilde{\varepsilon}$ to obtain \mathbf{x} such that $\|\mathbf{A}'\mathbf{x}\|_p \geq (1 - \tilde{\varepsilon}) \|\mathbf{A}'\|_{q \rightarrow p}$. We can bound $\|\mathbf{Ax}\|_p$ by $\|\mathbf{A}'\|_{q \rightarrow p}$.

$$\|\mathbf{Ax}\|_p \geq (1 - \tilde{\varepsilon}) \|\mathbf{A}'\mathbf{x}\|_p \geq (1 - \tilde{\varepsilon})^2 \|\mathbf{A}'\|_{q \rightarrow p}.$$

Moreover, we can bound $\|\mathbf{A}'\|_{q \rightarrow p}$ by $\|\mathbf{A}\|_{q \rightarrow p}$.

$$\|\mathbf{A}'\|_{q \rightarrow p} = \max_{\|\mathbf{y}\|_q \leq 1} \|\mathbf{A}'\mathbf{y}\|_p \geq \max_{\|\mathbf{y}\|_q \leq 1} \frac{1}{1 + \tilde{\varepsilon}} \|\mathbf{Ay}\|_p = \frac{1}{1 + \tilde{\varepsilon}} \|\mathbf{A}\|_{q \rightarrow p}.$$

Hence, $\|\mathbf{Ax}\|_p \geq \frac{(1 - \tilde{\varepsilon})^2}{1 + \tilde{\varepsilon}} \|\mathbf{A}\|_{q \rightarrow p} \geq (1 - \varepsilon) \|\mathbf{A}\|_{q \rightarrow p}$.

Moreover, \mathbf{A}' has $\tilde{O}\left(\frac{n^{\max\{1, p/2\}}}{\varepsilon^2}\right)$ rescaled rows of \mathbf{A} , hence $\text{nnz}(\mathbf{A}') \leq \tilde{O}\left(\frac{r}{\varepsilon^2} n^{\max\{1, p/2\}}\right)$ where r is the maximum number of nonzero in a row of \mathbf{A} . \blacktriangleleft