Strictly Balancing Matrices in Polynomial Time Using Osborne's Iteration

Rafail Ostrovsky[1]
Department of Computer Science, University of California Los Angeles, USA
rafail@cs.ucla.edu

Yuval Rabani[2]
The Rachel and Selim Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
yrabani@cs.huji.ac.il

Arman Yousefi[3]
Department of Computer Science, University of California Los Angeles, USA
armany@cs.ucla.edu

Abstract

Osborne's iteration is a method for balancing $n \times n$ matrices which is widely used in linear algebra packages, as balancing preserves eigenvalues and stabilizes their numeral computation. The iteration can be implemented in any norm over $\mathbb{R}^n$, but it is normally used in the L_2 norm. The choice of norm not only affects the desired balance condition, but also defines the iterated balancing step itself.

In this paper we focus on Osborne's iteration in any L_p norm, where $p < \infty$. We design a specific implementation of Osborne's iteration in any L_p norm that converges to a strictly ϵ-balanced matrix in $\tilde{O}(\epsilon^{-2}n^9K)$ iterations, where K measures, roughly, the *number of bits* required to represent the entries of the input matrix.

This is the first result that proves a variant of Osborne's iteration in the L_2 norm (or any L_p norm, $p < \infty$) strictly balances matrices in polynomial time. This is a substantial improvement over our recent result (in SODA 2017) that showed weak balancing in L_p norms. Previously, Schulman and Sinclair (STOC 2015) showed strict balancing of another variant of Osborne's iteration in the L_∞ norm. Their result does not imply any bounds on strict balancing in other norms.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Numerical Linear Algebra, Optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.93

[1] Research supported in part by NSF grant 1619348, DARPA, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government.
[2] Research supported in part by ISF grant 956-15, by BSF grant 2012333, and by I-CORE Algo.
[3] Research supported in part by NSF grant 1619348, DARPA, US-Israel BSF grant 2012366.

© Rafail Ostrovsky, Yuval Rabani, and Arman Yousefi;
licensed under Creative Commons License CC-BY
45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 93; pp. 93:1–93:11
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

1.1 Problem Statement and Motivation

This paper analyzes the convergence properties of Osborne's celebrated iteration [10] for balancing matrices. Given a norm $\|\cdot\|$ in $\mathbb{R}^n$, an $n \times n$ matrix A is balanced if and only if for all i, the i-th row of A and the i-th column of A have the same norm. The problem of balancing a matrix A is to compute a diagonal matrix D such that DAD^{-1} is balanced. The main motivation behind this problem is that balancing a matrix does not affect its eigenvalues, and balancing matrices in the L_2 norm increases the numerical stability of eigenvalue computations [10, 9]. Balancing also has a positive impact on the computational time needed for computing eigenvalues ([9, section 1.4.3]). In practice, it is sufficient to get a good approximation to the balancing problem. For $\alpha \geq 1$, a matrix $B = DAD^{-1}$ is an α-approximation to the problem of balancing A if and only if for all i, the ratio between the maximum and minimum of the norms of the i-th row and column is bounded by α. It is desirable to achieve $\alpha = 1 + \epsilon$ for some small $\epsilon > 0$. A matrix B that satisfies this relaxed balancing condition is also said to be strictly ϵ-balanced.

Osborne's iteration attempts to compute the diagonal matrix D by repeatedly choosing an index i and balancing the i-th row and column (this multiplies the i-th diagonal entry of D appropriately). Osborne proposed this iteration in the L_2 norm, and suggested round-robin choice of index to balance. However, other papers consider the iteration in other norms and propose alternative choices of index to balance [12, 15, 11]. Notice that a change of norm not only changes the target balance condition, but also changes the iteration itself, as in each step a row-column pair is balanced in the given norm. An implementation of Osborne's iteration is used in most numerical algebra packages, including MATLAB, LAPACK, and EISPACK, and is empirically efficient (see [9, 16] for further background). The main theoretical question about Osborne's iteration is its rate of convergence. How many rounds of the iteration are provably sufficient to get a strictly ϵ-balanced matrix?

1.2 Our Results

We consider Osborne's iteration in L_p norms for finite p. We design a new simple choice of the iteration (i.e., a rule to choose the next index to balance), and we prove that this variant provides a polynomial time approximation scheme to the balancing problem. More specifically, we show that in the L_1 norm, our implementation converges to a strictly ϵ-balanced matrix in $O\left(\epsilon^{-2} n^9 \log(wn/\epsilon) \log w / \log n\right)$ iterations, where $\log w$ is a lower bound on the number of bits required to represent the entries of A (exact definitions await Section 2). The time complexity of these iterations is $O\left(\epsilon^{-2} n^{10} \log(wn/\epsilon) \log w\right)$ arithmetic operations over $O(n \log(w/\epsilon))$-bit numbers. This result implies similar bounds for any L_p norm where p is fixed, and in particular the important case of $p = 2$. This is because applying Osborne's iteration in the L_p norm to $A = (a_{ij})_{n\times n}$ is equivalent to applying the iteration in the L_1 norm to $(a_{ij}^p)_{n\times n}$. Of course, the bit representation complexity of the matrix, and thus the bound on the number of iterations, grows by a factor of p.

Our results give the first theoretical analysis that indicates that Osborne's iteration in the L_2 norm, or any L_p norm for finite p, is indeed efficient in the worst case. This partially resolves the question that has been open since 1960 because we actually analyze a variant of Osborne's iteration which is different from the original iteration in the way it picks the next index to balance. Previously, Schulman and Sinclair [15] analyzed yet another variant of Osborne's iteration and answered this question only for the L_∞ norm. Concerning the

convergence rate for the L_p norms discussed here, we recently published a result [11] that considers a much weaker notion of approximation. The previous result only shows the rate of convergence to a matrix that is approximately balanced in an average sense. The matrix might still have row-column pairs that are highly unbalanced. The implementations in the common numerical linear algebra packages use as a stopping condition the strict notion of balancing, and not this weaker notion. We discuss previous work in greater detail below.

1.3 Previous Work

Osborne [10] studied the L_2 norm version of matrix balancing, proved the uniqueness of the L_2 solution, designed the iterative algorithm discussed above, and proved that it converges in the limit to a balanced matrix (without bounding the convergence rate). Parlett and Reinsch [12] generalized Osborne's iteration to other norms. Their implementation is the one widely used in practice (see Chen [3, Section 3.1], also the book [13, Chapter 11] and the code in [1]). Grad [6] proved convergence in the limit for the L_1 version (again without bounding the running time), and Hartfiel [7] showed that the L_1 solution is unique. Eaves et al. [5] gave a characterization of balanceable matrices. Kalantari et al. [8] gave an algorithm for ϵ-balancing a matrix in the L_1 norm. The algorithm reduces the problem to unconstrained convex optimization and uses the ellipsoid algorithm to approximate the optimal solution. This generates a weakly ϵ-balanced matrix, which satisfies the following definition. Given $\epsilon > 0$, a matrix $A = (a_{ij})_{n\times n}$ is weakly ϵ-balanced if and only if $\sqrt{\sum_{i=1}^{n}(\|a_{.,i}\| - \|a_{i,.}\|)^2} \leq \epsilon \cdot \sum_{i,j} |a_{i,j}|$. Compare this with the stronger condition of being strictly ϵ-balanced, which we use in this paper, and numerical linear algebra packages use as a stopping condition. This condition requires that for every $i \in \{1, 2, \ldots, n\}$, $\max\{\|a_{.,i}\|, \|a_{i,.}\|\} \leq (1+\epsilon) \cdot \min\{\|a_{.,i}\|, \|a_{i,.}\|\}$. In L_∞, Schneider and Schneider [14] gave a polynomial time algorithm that exactly balances a matrix. The algorithm does not use Osborne's iteration. Its running time was improved by Young et al. [17]. Both algorithms rely on iterating over computing a minimum mean cycle in a weighted strongly connected digraph, then contracting the cycle. Schulman and Sinclair [15] were the first to provide a quantitative bound on the running time of Osborne's iteration. They proposed a carefully designed implementation of Osborne's iteration in the L_∞ norm that strictly ϵ-balances an $n \times n$ matrix A in $O(n^3 \log(\varrho n/\epsilon))$ iterations, where ϱ measures the initial L_∞ imbalance of A. Their proof is an intricate case analysis. Following that work, in [11] we showed that several implementations of Osborne's iteration in L_p norms, including the original implementation, converge to a weakly ϵ-balanced matrix in polynomial time (which, in fact, can be either nearly linear in number of non-zero entries of matrix A or nearly linear in $1/\epsilon$). Very recently, Cohen et al. [4] gave an interior point algorithm that weakly ϵ-balances matrices in the L_1 norm in time $\tilde{O}(m^{3/2} \log(1/\epsilon))$, where m is the number of non-zero entries of A. Their results also apply to matrix scaling, a problem similar to matrix balancing, for which Allen-Zhu et al. [2] independently gave similar results. Notice that in the results of [4], due to the logarithmic dependence on ϵ one can choose ϵ that is exponentially small in the input representation, and thus get an algorithm for strict balancing with better worst case running time than our claimed analysis of Osborne's iteration. We note, however, that the purpose of our analysis is not to present a new algorithm for matrix balancing, but rather to provide some worst case guarantee for the time it takes some implementation of Osborne's iteration to achieve strict balancing. Osborne's iteration and strict balancing are the heuristic and stopping condition used in practice to balance matrices. We also note that by the lower bound proved in our previous paper [11], Osborne's iteration does not guarantee convergence in time polynomial in $\log(1/\epsilon)$ (the lower bound is $\Omega(1/\sqrt{\epsilon})$).

1.4 Our Contribution

The result of [11] is derived by observing that any implementation of Osborne's iteration can be interpreted as an application of coordinate descent to optimize the convex function from [8]. This is the starting point of this paper, but to make the approach guarantee strict balancing, we need to revise substantially previous implementations using novel algorithmic ideas. Nevertheless, the implementation itself remains a rather simple rule for choosing the next index to balance.

The main difficulty with respect to previous work is the following. The convergence rate of coordinate descent can be bounded effectively as long as there is a choice of coordinate (i.e., index) for which the drop in the objective function in a single step is non-negligible compared with the current objective value. But if this is not the case, then one can argue only about the balance of each index relative to the sum of norms of all rows and columns. Indices that have relatively heavy weight (row norm + column norm) will indeed be balanced at this point. However, light-weight indices may be highly unbalanced. The naive remedy to this problem is to work down by scales. After balancing the matrix globally, heavy-weight indices are balanced, approximately, so they can be left alone, deactivated. Now there are light-weight indices that have become heavy-weight with respect to the remaining active nodes, so we can continue balancing the active indices until the relatively heavy-weight among them become approximately balanced, and so forth. The problem with the naive solution is that balancing the active indices shifts the weights of both active and inactive indices, and they move out of their initial scale. This movement need not complicate the balancing rule—we can keep balancing far-from-balanced nodes in the current scale. However, if the scale sets of indices keep changing, it is harder to argue that the process converges. Shifting between scales is precisely what our algorithm and proof deal with. Light-weight indices that have become heavy-weight are easy to handle. They can keep being active. Heavy-weight indices that have become light-weight cannot continue to be inactive, because they are no longer guaranteed to be approximately balanced. Thus, in order to analyze convergence effectively, we need to bound the number (and global effect on weight) of these reactivation events.

2 Preliminaries

The input is a real square matrix $A = (a_{ij})_{n\times n}$. We denote the i-th row of such a matrix by $a_{i,.}$ and the i-th column by $a_{.,i}$. We also use the notation $[n] = \{1, 2, \ldots, n\}$. The matrix A is *balanced* in the L_p norm iff $\|a_{.,i}\|_p = \|a_{i,.}\|_p$ for every index $i \in [n]$. Since the condition for being balanced depends neither on the signs of the entries of A nor on the diagonal values, we will assume without loss of generality that A is non-negative with zeroes on the diagonal.

An invertible diagonal matrix $D = \texttt{diag}(d_1, \cdots, d_n)$ *balances* A in the L_p norm iff DAD^{-1} is balanced in the L_p norm. A matrix A is *balanceable* iff there exists an invertible diagonal matrix D that balances A. Balancing a matrix $A = (a_{ij})_{n\times n}$ in the L_p norm is equivalent to balancing the matrix $(a_{ij}^p)_{n\times n}$ in the L_1 norm. Therefore, for the rest of the paper we focus on balancing matrices in the L_1 norm.

We use $a_{\min}$ to denote the minimum non-zero entry of A. We also define $w = \frac{1}{a_{\min}} \cdot \sum_{ij} a_{ij}$.

▶ **Definition 1.** Given $\epsilon > 0$ and an $n \times n$ matrix A, we say that the index i of A (where $i \in [n]$) is *ϵ-balanced* iff

$$\frac{\max\{\|a_{.,i}\|_1, \|a_{i,.}\|_1\}}{\min\{\|a_{.,i}\|_1, \|a_{i,.}\|_1\}} \leq 1 + \epsilon.$$

We say that A is *strictly ϵ-balanced* iff every index i of A is ϵ-balanced.

Any implementation of Osborne's iteration can be thought of as computing vectors $\mathbf{x^{(t)}} \in \mathbb{R}^n$ for $t = 1, 2, \ldots$, where iteration t is applied to the matrix $(a_{ij}^{(t)}) = DAD^{-1}$ for $D = \texttt{diag}(e^{x_1^{(t)}}, e^{x_2^{(t)}}, \ldots, e^{x_n^{(t)}})$. Thus, for all i, j, $a_{ij}^{(t)} = a_{ij} \cdot e^{x_i^{(t)} - x_j^{(t)}}$. Initially, $\mathbf{x}^{(1)} = (0, 0, \ldots, 0)$. A balancing step of the iteration chooses an index i, then sets $x_i^{(t+1)} = x_i^{(t)} + \frac{1}{2} \cdot \left(\ln \|a_{.,i}^{(t)}\|_1 - \ln \|a_{i,.}^{(t)}\|_1\right)$, and for all $j \neq i$, keeps $x_j^{(t+1)} = x_j^{(t)}$. For $\mathbf{x} \in \mathbb{R}^n$, we denote the sum of entries of the matrix DAD^{-1} for $D = \texttt{diag}(e^{x_1}, e^{x_2}, \ldots, e^{x_n})$ by $f(\mathbf{x}) = f_A(\mathbf{x}) = \sum_{ij} a_{ij} \cdot e^{x_i - x_j}$. For any $n \times n$ non-negative matrix $B = (b_{ij})$, we denote by G_B the weighted directed graph with node set $\{1, 2, \ldots, n\}$, arc set $\{(i, j) : b_{ij} > 0\}$, where an arc (i, j) has weight b_{ij}. We will assume henceforth that the undirected version of G_A is connected, otherwise we can handle each connected component separately. We quote a few useful lemmas. The references contain the proofs.

▶ **Lemma 2** (Theorem 1 in Kalantari et al. [8]). *The input matrix A is balanceable if and only if G_A is strongly connected. Moreover, DAD^{-1} is balanced in the L_1 norm if and only if $D = \texttt{diag}(e^{x_1^*}, e^{x_2^*}, \ldots, e^{x_n^*})$, where $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ minimizes $f(\mathbf{x})$ over $\mathbf{x} \in \mathbb{R}^n$.*

Notice that f is a convex function and the gradient $\nabla f(\mathbf{x})$ of f at $\mathbf{x}$ is given by

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \sum_{j=1}^{n} a_{ij} \cdot e^{x_i - x_j} - \sum_{j=1}^{n} a_{ji} \cdot e^{x_j - x_i},$$

the difference between the total weight of arcs leaving node i and the total weights of arcs going into node i in the graph of DAD^{-1} for $D = \texttt{diag}(e^{x_1}, e^{x_2}, \ldots, e^{x_n})$. If DAD^{-1} is balanced then the arc weights $a_{ij} \cdot e^{x_i - x_j}$ form a valid circulation in the graph G_A, since the gradient has to be 0. Some properties of f are given in the following lemma.

▶ **Lemma 3** (Lemmas 2.1 and 2.2 in Ostrovsky et al. [11]). *If $\mathbf{x}'$ is derived from $\mathbf{x}$ by balancing index i of a matrix $B = (b_{ij})_{n \times n}$, then $f(\mathbf{x}) - f(\mathbf{x}') = (\sqrt{\|b_{.,i}\|_1} - \sqrt{\|b_{i,.}\|_1})^2$. Also, for all $\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{n}{2} \cdot \|\nabla f(\mathbf{x})\|_1$.*

We also need the following absolute bounds on the arc weights.

▶ **Lemma 4** (Lemma 3.2 in Ostrovsky et al. [11]). *Suppose that a matrix B is derived from a matrix A through a sequence of balancing operations. Then, for every arc (i, j) of G_B,*

$$\left(\frac{a_{\min}}{\sum_{ij} a_{ij}}\right)^n \cdot \sum_{ij} a_{ij} \leq b_{ij} \leq \sum_{ij} a_{ij}.$$

(Notice that the arcs of G_B are identical to the arcs of G_A.)

Finally, we prove the following global condition on indices being ϵ-balanced.

▶ **Lemma 5.** *Consider a matrix $B = DAD^{-1} = (b_{ij})_{n \times n}$, where $D = \texttt{diag}(e^{x_1}, e^{x_2}, \ldots, e^{x_n})$, that was derived from A by a sequence of zero or more balancing operations. Let $\epsilon \in (0, 1/2]$, and put $\epsilon' = \frac{\epsilon^2}{64n^4}$. Suppose that $\|\nabla f_A(\vec{0})\|_1 \leq \epsilon' \cdot f_A(\vec{0})$. Then, for every $i \in [n]$ we have the following implication. If $\|b_{.,i}\|_1 + \|b_{i,.}\|_1 \geq \frac{1}{8n^3} \cdot f_A(\mathbf{x})$, then index i is ϵ-balanced in B.*

Proof. We will show the contrapositive claim that if a node is not ϵ-balanced then it must have low weight (both with respect to B). Let i be an index that is not ϵ-balanced in B. Without loss of generality we may assume that the in-weight is larger than the out-weight,

so $\|b_{.,i}\|_1/\|b_{i,.}\|_1 > 1+\epsilon$. Consider what would happen if we balance index i in B, yielding a vector $\mathbf{x}'$ that differs from $\mathbf{x}$ only in the i-th coordinate.

$$\begin{aligned} f_A(\mathbf{x}) - f_A(\mathbf{x}') &= \left(\sqrt{\|b_{.,i}\|_1} - \sqrt{\|b_{i,.}\|_1}\right)^2 \\ &> \|b_{.,i}\|_1 \cdot \left(1 - \sqrt{\frac{1}{1+\epsilon}}\right)^2 \\ &> \frac{\epsilon^2}{16} \cdot (\|b_{.,i}\|_1 + \|b_{i,.}\|_1)\,, \end{aligned} \tag{1}$$

where the equation follows from Lemma 3 and the last inequality uses the fact that $\epsilon \le \frac{1}{2}$. On the other hand, we have

$$\begin{aligned} f_A(\mathbf{x}) - f_A(\mathbf{x}') &\le f_A(\vec{0}) - f(\mathbf{x}^*) \\ &\le \frac{n}{2} \cdot \|\nabla f_A(\vec{0})\|_1 \\ &\le \frac{n}{2} \cdot \epsilon' \cdot f_A(\vec{0}) \\ &= \frac{\epsilon^2}{128n^3} \cdot f_A(\vec{0}). \end{aligned} \tag{2}$$

where the first inequality follows from the the fact that every balancing step decreases f_A, the second inequality follows from Lemma 3, the third inequality follows from the assumption on $f_A(\vec{0})$, and the last equation follows from the choice of ϵ'. Combining the bounds on $f_A(\mathbf{x}) - f_A(\mathbf{x}')$ in Equations (1) and (2) gives

$$\|b_{.,i}\|_1 + \|b_{i,.}\|_1 < \frac{1}{8n^3} \cdot f_A(\vec{0}),$$

and this completes the proof. ◀

3 Strict Balancing

In this section we present a variant of Osborne's iteration and prove that it converges in polynomial time to a strictly ϵ-balanced matrix. The algorithm, a procedure named StrictBalance, is defined in pseudocode labeled Algorithm 1 on page 7. Lemma 5 above motivates the main idea of contracting heavy nodes in step 14 of StrictBalance.

Our main theorem is

▶ **Theorem 6.** *StrictBalance(A, ϵ) returns a strictly ϵ-balanced matrix $B = DAD^{-1}$ after at most $O\left(\epsilon^{-2}n^9 \log(wn/\epsilon)\log w/\log n\right)$ balancing steps, using $O\left(\epsilon^{-2}n^{10}\log(wn/\epsilon)\log w\right)$ arithmetic operations over $O(n\log(w/\epsilon))$-bit numbers.*

The proof of Theorem 6 uses a few arguments, given in the following lemmas. A *phase* of StrictBalance is one iteration of the outer while loop. Notice that in the beginning of this loop the variable s indexes the phase number (i.e., $s-1$ phases were completed thus far). Also in the beginning of the inner while loop the variable t indexes the total iteration number from all phases (i.e., $t-1$ balancing operations from all phases were completed thus far).

We identify outer loop iteration s with an interval $[t_s, t_{s+1}) = \{t_s, t_s+1, \ldots, t_{s+1}-1\}$ of the inner loop iterations executed during phase s. We denote by $\mathcal{B}_{s,t}$ the value of $\mathcal{B}_s$ in the beginning of the inner while loop iteration number t (dubbed time t). If $t \in [t_j, t_{j+1})$, then $\mathcal{B}_{s,t}$ is defined only for $s \le j$. We also use $G^{(\mathcal{B}_{s,t})}$ to denote the graph that is obtained by contracting the nodes of set $\mathcal{B}_{s,t}$ in graph G_A. Also $f^{(\mathcal{B}_{s,t})}$ is the function corresponding to

Algorithm 1 StrictBalance(A, ϵ)

Input: Matrix $A \in \mathbb{R}^{n\times n}, \epsilon$
Output: A strictly ϵ-balanced matrix

1: $\mathcal{B}_1 = \varnothing$, $\tau_1 = 0$, $s = 1$, $\epsilon' = \epsilon^2/64n^4$, $\mathbf{x^{(1)}} = (0,\dots,0)$, $t = 1$
2: **while** $\mathcal{B}_s \neq [n]$ and there is $i \in [n]$ that is not ϵ-balanced **do**
3: Define $f^{(\mathcal{B}_s)} : \mathbb{R}^n \to \mathbb{R}$, $f^{(\mathcal{B}_s)}(\mathbf{x}) = \sum_{i,j:i\notin\mathcal{B}_s \text{ or } j\notin\mathcal{B}_s} a_{ij}e^{x_i-x_j}$
4: **while** $\dfrac{\|\nabla f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}})\|_1}{f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}})} > \epsilon'$ **do**
5: Pick $i = \arg\max_{i\notin\mathcal{B}_s}\left\{\left(\sqrt{\|a^{(t)}_{.,i}\|_1} - \sqrt{\|a^{(t)}_{i,.}\|_1}\right)^2\right\}$
6: Balance ith node: $\mathbf{x^{(t+1)}} = \mathbf{x^{(t)}} + \alpha_t\mathbf{e}_i$, where $\alpha_t = \ln\sqrt{\|a^{(t)}_{.,i}\|_1/\|a^{(t)}_{i,.}\|_1}$
7: $t \leftarrow t+1$
8: **if** $s > 1$ and $\|a^{(t)}_{.,i}\|_1 + \|a^{(t)}_{i,.}\|_1 < \tau_s$ for some $i \in \mathcal{B}_s \setminus \mathcal{B}_{s-1}$ **then**
9: $\mathcal{B}_s = \mathcal{B}_s \setminus \{i \notin \mathcal{B}_{s-1} : \|a^{(t)}_{.,i}\|_1 + \|a^{(t)}_{i,.}\|_1 < \tau_s\}$
10: Redefine $f^{(\mathcal{B}_s)} : \mathbb{R}^n \to \mathbb{R}$, $f^{(\mathcal{B}_s)}(\mathbf{x}) = \sum_{i,j:i\notin\mathcal{B}_s \text{ or } j\notin\mathcal{B}_s} a_{ij}e^{x_i-x_j}$
11: **end if**
12: **end while**
13: $\tau_{s+1} = \dfrac{1}{4n^3}f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}})$
14: $\mathcal{B}_{s+1} = \mathcal{B}_s \cup \left\{i : \|a^{(t)}_{.,i}\|_1 + \|a^{(t)}_{i,.}\|_1 \geq \tau_{s+1}\right\}$
15: $s \leftarrow s+1$
16: **end while**
17: **return** the resulting matrix

graph $G^{(\mathcal{B}_{s,t})}$ and $f^{(\mathcal{B}_{s,t})}(\mathbf{x^{(t)}})$ denotes the sum of weights of arcs of graph $G^{(\mathcal{B}_{s,t})}$ at time t. If set B_s is unchanged during an interval and there is no confusion, we may use $G^{(\mathcal{B}_s)}$ instead of $G^{(\mathcal{B}_{s,t})}$. Particularly we use $f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}})$ instead of $f^{(\mathcal{B}_{s,t})}(\mathbf{x^{(t)}})$. We refer to the quantity $\|a^{(t)}_{.,i}\|_1 + \|a^{(t)}_{i,.}\|_1$ as the *weight* of node i at time t.

▶ **Lemma 7.** *For every phase $s \geq 1$, for every $t \geq t_{s+1}$, $\mathcal{B}_{s,t} = \mathcal{B}_{s,t_{s+1}}$.*

Proof. The claim follows easily from the fact that any iteration $t \geq t_{s+1}$ belongs to a phase $s' > s$, so $\mathcal{B}_{s,t_{s+1}} \cap (\mathcal{B}_{s',t} \setminus \mathcal{B}_{s'-1,t}) = \emptyset$, and by line 8 and 9 of StrictBalance none of the nodes in $\mathcal{B}_{s,t_{s+1}}$ will be removed. ◀

▶ **Lemma 8.** *For all $s > 1$, for all $t \in [t_s, t_{s+1})$, $f^{(\mathcal{B}_{s,t})}(\mathbf{x^{(t)}}) \leq (n - |\mathcal{B}_{s,t}|)\cdot\tau_s$.*

Proof. Let $t_s = t_{s,1} < t_{s,2} < t_{s,3} < \cdots < t_{s,\ell_s}$ denote the time steps before which $\mathcal{B}_s$ changes during phase s. For simplicity, we abuse notation and use $\mathcal{B}_{s,j}$ instead of $\mathcal{B}_{s,t_{s,j}}$. Clearly $\mathcal{B}_{s,1} \supseteq \mathcal{B}_{s,2} \dots \supseteq \mathcal{B}_{s,\ell_s}$, because we only remove nodes from $\mathcal{B}_s$ once it is set. Fix $s > 1$. We prove this lemma by induction on $r \in \{1, 2, \dots, \ell_s\}$. For the basis, let $r = 1$. Clearly, by

the way the algorithm sets $\mathcal{B}_s$ before time $t_{s,1}$, all nodes with weight $\geq \tau_s$ are in $\mathcal{B}_s$, and therefore every node $i \notin \mathcal{B}_s$ has weight at most τ_s, so the lemma follows. Now, assume that the lemma is true for every $t \leq t_{s,r}$, we show that the lemma holds for every $t \leq t_{s,r+1}$. If $t \in [t_{s,r}, t_{s,r+1})$, then $\mathcal{B}_{s,t} = \mathcal{B}_{s,t_{s,r}}$, and we have:

$$f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t})}) \leq f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t_{s,r}})}) \leq \left(n - \left|\mathcal{B}_{s,t_{s,r}}\right|\right) \cdot \tau_s = (n - |\mathcal{B}_{s,t}|) \cdot \tau_s.$$

The first inequality holds because balancing operations from time $t_{s,r}$ to time t only reduce the value of $f^{(\mathcal{B}_s)}$, and the second inequality holds by the induction hypothesis.

Just before iteration $t = t_{s,r+1}$, the set $\mathcal{B}_s$ changes, and one or more nodes are removed from it. However, every removed node has weight at most τ_s, and its removal does not change the weights of the other nodes in $[n] \setminus \mathcal{B}_s$. Therefore, if k nodes are removed from $\mathcal{B}_s$,

$$f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t_{s,r+1}})}) \leq \left(n - \left|\mathcal{B}_{s,t_{s,r}}\right|\right) \cdot \tau_s + k \cdot \tau_s = \left(n - \left|\mathcal{B}_{s,t_{s,r+1}}\right|\right) \cdot \tau_s.$$

This completes the proof. ◀

▶ **Corollary 9.** *For all $s > 1$, $f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t_{s+1}})}) \leq \frac{1}{4n^2} \cdot f^{(\mathcal{B}_{s-1})}(\mathbf{x}^{(\mathbf{t_s})})$. If $s > 2$, then $\tau_s \leq \frac{\tau_{s-1}}{4n^2}$.*

Proof. Notice that

$$f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t_{s+1}})}) \leq n \cdot \tau_s = \frac{1}{4n^2} \cdot f^{(\mathcal{B}_{s-1})}(\mathbf{x}^{(\mathbf{t_s})}),$$

where the inequality follows from Lemma 8, and the equation follows from line 13 of StrictBalance. This proves the first assertion. As for the second assertion, notice that if $s > 2$ then $s - 1 > 1$, so using line 13 of StrictBalance and Lemma 8 again,

$$\tau_s = \frac{1}{4n^3} \cdot f^{(\mathcal{B}_{s-1})}(\mathbf{x}^{(\mathbf{t_s})}) \leq \frac{1}{4n^3} \cdot n\tau_{s-1} = \frac{1}{4n^2} \cdot \tau_{s-1},$$

as stipulated. ◀

▶ **Lemma 10.** *For every phase $s > 1$, for every $t \geq t_s$, all the nodes in $\mathcal{B}_{s,t}$ have weight $\geq \tau_s/2$ and are ϵ-balanced at time t.*

Proof. Fix $s > 1$ and let $i \in \mathcal{B}_{s,t}$. Without loss of generality $i \notin \mathcal{B}_{s-1,t}$, otherwise we can replace s with $s - 1$. (Recall that $\mathcal{B}_1 = \emptyset$ at all times.) Also note that it must be the case that $i \in \mathcal{B}_{s,t_s}$, because $\mathcal{B}_s$ does not accumulate additional nodes after being created. If $t \in [t_s, t_{s+1}]$, then lines 13-14 and 8-9 of StrictBalance guarantee that if $i \in \mathcal{B}_{s,t} \setminus \mathcal{B}_{s-1,t}$, then its weight at time t is at least τ_s.

Otherwise, consider $t > t_{s+1}$ and let $s' > s$ be the phase containing t. Consider a phase $j > s$. By Lemma 8 the total weight of $f^{(\mathcal{B}_j)}$ during phase j is at most $n\tau_j$, and $f^{(\mathcal{B}_j)}$ never drops below 0. So, the total weight that a node $i \in \mathcal{B}_j$ can lose (which is at most the total weight that $f^{(\mathcal{B}_j)}$ can lose) is at most $n\tau_j$. By Corollary 9, for every $j > s$, $\tau_{j+1} \leq \frac{\tau_j}{4n^2}$. Now, suppose that t is an iteration in phase $s' > s$. Then, the weight of i at time t is at least

$$\tau_s - \sum_{j=s+1}^{s'} n\tau_j \geq \tau_s \cdot \left(1 - n \cdot \sum_{k=1}^{s'-s} (2n)^{-2k}\right) \geq \frac{\tau_s}{2}.$$

Thus we have established that at any time $t \geq t_s$, if $i \in \mathcal{B}_{s,t}$ then its weight is at least $\frac{\tau_s}{2} = \frac{1}{8n^3} f^{(\mathcal{B}_{s-1})}(\mathbf{x}^{(\mathbf{t_s})})$. By line 4 of StrictBalance, $\|\nabla f^{(\mathcal{B}_{s-1,t_s})}(\mathbf{x}^{(\mathbf{t_s})})\|_1 \leq \epsilon' \cdot f^{(\mathcal{B}_{s-1,t_s})}(\mathbf{x}^{(\mathbf{t_s})})$. By Lemma 7, $\mathcal{B}_{s-1}$ does not change in the interval $[t_s, t]$. Therefore, we conclude from Lemma 5 that i is ϵ-balanced at time t. ◀

▶ **Lemma 11.** *Suppose that $t < t'$ satisfies $[t, t') \subseteq [t_s, t_{s+1})$, and furthermore, during the iterations in the interval $[t, t')$ the set $\mathcal{B}_s$ does not change (it could change after balancing step $t' - 1$). Then, the length of the interval*

$$t' - t = O\left(\epsilon^{-2} n^7 \log(wn/\epsilon)\right).$$

Proof. Rename the nodes so that $\mathcal{B}_{s,t} = \mathcal{B}_{s,t'-1} = \{p, p+1, \ldots, n\}$. The assumption that $\mathcal{B}_s$ does not change during the interval $[t, t')$ means that the weights of all the nodes $p, p+1, \ldots, n$ remain at least τ_s for the duration of this interval. During the interval $[t, t')$, the graph $G^{(\mathcal{B}_s)}$ (which remains fixed) is obtained by contracting the nodes $p, p+1, \ldots, n$ in G_A. So $G^{(\mathcal{B}_s)}$ has p nodes $1, 2, \ldots, p-1, p$, where the last node p is the contracted node. In each iteration in the interval $[t, t')$, one of the nodes $1, 2, \ldots, p-1$ is balanced. Consider some time step $t'' \in [t, t')$, and let I_i and O_i, respectively, denote the current sums of weights of the arcs of $G^{(\mathcal{B}_s)}$ into and out of node i, respectively. Let $j \in [p-1]$ be the node that maximizes $\frac{(I_j - O_j)^2}{I_j + O_j}$. We have

$$\begin{aligned}
f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')}) - f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}''+\mathbf{1})}) &= \max_{i \in [p-1]} \left(\sqrt{I_i} - \sqrt{O_i}\right)^2 \geq \left(\sqrt{I_j} - \sqrt{O_j}\right)^2 \\
&\geq \frac{(I_j - O_j)^2}{2(I_j + O_j)} \geq \frac{\sum_{i=1}^{p-1}(I_i - O_i)^2}{2\sum_{i=1}^{p-1}(I_i + O_i)} \\
&\geq \frac{\left(\sum_{i=1}^{p-1} |I_i - O_i|\right)^2}{2n\sum_{i=1}^{p}(I_i + O_i)} \geq \frac{\left(\sum_{i=1}^{p} |I_i - O_i|\right)^2}{8n\sum_{i=1}^{p}(I_i + O_i)} \\
&= \frac{1}{16n} \cdot \frac{\|\nabla f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})\|_1^2}{f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})}. \qquad (3)
\end{aligned}$$

The first equation follows from the choice of i in line 5 StrictBalance, and Lemma 3. The third inequality follows from an averaging argument and the choice of j. The fourth inequality uses Cauchy-Schwarz. The last inequality holds because $\sum_{i=1}^{p}(I_i - O_i) = 0$, so $|I_p - O_p| = \left|\sum_{i=1}^{p-1}(I_i - O_i)\right| \leq \sum_{i=1}^{p-1} |I_i - O_i|$, and therefore $\sum_{i=1}^{p} |I_i - O_i| \leq 2\sum_{i=1}^{p-1} |I_i - O_i|$.

Since the interval $[t, t')$ is contained in phase s, the stopping condition for the phase does not hold, so

$$\frac{\|\nabla f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})\|_1}{f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})} > \epsilon' = \frac{\epsilon^2}{64n^4}.$$

Therefore,

$$\begin{aligned}
f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')}) - f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}''+\mathbf{1})}) &\geq \frac{1}{16n} \cdot \frac{\|\nabla f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')}\|_1^2}{f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})} \\
&> \frac{\epsilon'}{16n} \cdot \|\nabla f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')})\|_1 \\
&\geq \frac{\epsilon'}{8n^2} \cdot (f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*)),
\end{aligned}$$

where the last inequality follows from Lemma 3. Rearranging the terms gives

$$f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}''+\mathbf{1})}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*) \leq \left(1 - \frac{\epsilon'}{8n^2}\right) \cdot (f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}'')}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*)).$$

Iterating for T step yields

$$f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t}+\mathbf{T})}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*) \leq \left(1 - \frac{\epsilon'}{8n^2}\right)^T \cdot (f^{(\mathcal{B}_s)}(\mathbf{x}^{(\mathbf{t})}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*)).$$

Now, by Lemma 4, we have that $f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*) \le f^{(\mathcal{B}_s)}(\mathbf{x^{(t)}}) \le \sum_{i,j=1}^n a_{ij}$, and for all t'', $f^{(\mathcal{B}_s)}(\mathbf{x^{(t'')}}) \ge \frac{1}{w^n}\sum_{i,j=1}^n a_{ij}$. Therefore, if $t' - t \ge \frac{8n^2}{\epsilon'} \cdot \ln\left(16nw^n/(\epsilon')^2\right) + 1$, then

$$f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*) \le \left(\frac{\epsilon'}{4\sqrt{n}}\right)^2 \cdot \frac{1}{w^n} \cdot \sum_{i,j=1}^n a_{ij} \le \left(\frac{\epsilon'}{4\sqrt{n}}\right)^2 \cdot f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}}).$$

Therefore,

$$\begin{aligned}\frac{1}{16n} \cdot \frac{\|\nabla f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}})\|_1^2}{(f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}}))^2} &\le \frac{f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}}) - f^{(\mathcal{B}_s)}(\mathbf{x^{(t')}})}{f^{(\mathcal{B})}(\mathbf{x^{(t'-1)}})} \\ &\le \frac{f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}}) - f^{(\mathcal{B}_s)}(\mathbf{x}^*)}{f^{(\mathcal{B})}(\mathbf{x^{(t'-1)}})} \le \left(\frac{\epsilon'}{4\sqrt{n}}\right)^2,\end{aligned}$$

where the first inequality follows from (3), and the second inequality holds because $f^{(\mathcal{B}_s)}(\mathbf{x}^*) \le f^{(\mathcal{B}_s)}(\mathbf{x^{(t'-1)}})$. We get that $\frac{\|\nabla f^{(\mathcal{B})}(\mathbf{x^{(t'-1)}})\|_1}{f^{(\mathcal{B})}(\mathbf{x^{(t'-1)}})} \le \epsilon'$, in contradiction to our assumption that the phase does not end before the start of iteration t'. ◀

▶ **Corollary 12.** *In any phase, the number of balancing steps is at most $O\left(\epsilon^{-2}n^8\log(wn/\epsilon)\right)$.*

Proof. In the beginning of phase s the set $\mathcal{B}_s$ contains at most $n-1$ nodes. Partition the phase into intervals $[t, t')$ where $\mathcal{B}_s$ does not change during an interval, but does change between intervals. By Lemma 11, each interval consists of at most $O\left(\epsilon^{-2}n^7\log(wn/\epsilon)\right)$ balancing steps. Since nodes that are removed from $\mathcal{B}_s$ between intervals are never returned to $\mathcal{B}_s$, the number of such intervals is at most $n-1$. Hence, the total number of balancing steps in the phase is at most $O\left(\epsilon^{-2}n^8\log(wn/\epsilon)\right)$. ◀

▶ **Lemma 13.** *The total number of phases of the algorithm is $O(n\log w/\log n)$.*

Proof. Let $s > 2$ be a phase of the algorithm and $t \in [t_s, t_{s+1})$. By Lemma 4, $f^{(\mathcal{B}_{s,t})}(\mathbf{x^{(t)}}) \ge \frac{1}{w^n} \cdot \sum_{ij} a_{ij}$. On the other hand, by Lemma 8 and Corollary 9, $\tau_s \le \frac{1}{(4n^2)^{s-2}} \cdot \tau_2 \le \frac{1}{(4n^2)^{s-2}} \cdot \sum_{ij} a_{ij}$, and $f^{(\mathcal{B}_{s,t})}(\mathbf{x^{(t)}}) \le n\tau_s$. Combining these gives $\frac{1}{w^n} \cdot \sum_{ij} a_{ij} \le n\tau_s \le \frac{n}{(4n^2)^{s-2}} \cdot \sum_{ij} a_{ij}$ which implies that $s \le \frac{\log(nw^n)}{\log(4n^2)} + 2$. ◀

Proof of Theorem 6. By Lemma 13, for some $s = O(n\log w/\log n)$, StrictBalance terminates, so $\mathcal{B}_{s,t_s} = [n]$. By Corollary 12, the number of balancing steps in a phase is at most $O\left(\epsilon^{-2}n^8\log(wn/\epsilon)\right)$. Therefore, the total number of balancing steps is at most $O\left(\epsilon^{-2}n^9\log(wn/\epsilon)\log w/\log n\right)$. These steps require at most $O\left(\epsilon^{-2}n^{10}\log(wn/\epsilon)\log w\right)$ arithmetic operations. When the algorithm terminates at time t_s, all the nodes are in $\mathcal{B}_{s,t_s}$, and by Lemma 10 they are all ϵ-balanced, so the matrix is strictly ϵ-balanced.

Thus, assuming exact arithmetics with infinite precision we have shown that the algorithm converges to a strictly ϵ-balanced matrix in the claimed number of arithmetic operations. To show that the algorithm still works if all numbers are represented with only $O(n\log(w/\epsilon))$ bits, we apply an analysis similar to the one in section 3 of [11]. ◀

References

1 EISPACK implementation. URL: `http://www.netlib.org/eispack/balanc.f`.

2 Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *Proc. of the 58th Ann. IEEE Symp. on Foundations of Computer Science*, 2017.

3 T.-Y. Chen. Balancing sparse matrices for computing eigenvalues. Master's thesis, UC Berkeley, May 1998.

4 Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *Proc. of the 58th Ann. IEEE Symp. on Foundations of Computer Science*, 2017.

5 B. C. Eaves, A. J. Hoffman, U. G. Rothblum, and H. Schneider. Line-sum-symmetric scalings of square nonnegative matrices. In *Mathematical Programming Essays in Honor of George B. Dantzig Part II*, pages 124–141. Springer, 1985.

6 J. Grad. Matrix balancing. *The Computer Journal*, 14(3):280–284, 1971.

7 D. J. Hartfiel. Concerning diagonal similarity of irreducible matrices. In *Proceedings of the American Mathematical Society*, pages 419–425, 1971.

8 B. Kalantari, L. Khachiyan, and A. Shokoufandeh. On the complexity of matrix balancing. *SIAM Journal on Matrix Analysis and Applications*, 118(2):450–463, 1997.

9 D. Kressner. *Numerical methods for general and structured eigenvalue problems*. Princeton University Press, 2005.

10 E. E. Osborne. On pre-conditioning of matrices. *Journal of the ACM (JACM)*, 7(4):338–345, 1960.

11 Rafail Ostrovsky, Yuval Rabani, and Arman Yousefi. Matrix balancing in lp norms: Bounding the convergence rate of osborne's iteration. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 154–169, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039697`.

12 B. N. Parlett and C. Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numerische Mathematik*, 13(4):293–304, 1969.

13 W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing, 3rd Edition*. Cambridge University Press, 2007.

14 H. Schneider and M. H. Schneider. Max-balancing weighted directed graphs and matrix scaling. *Mathematics of Operations Research*, 16(1):208–222, 1991. `doi:10.1287/moor.16.1.208`.

15 L. J. Schulman and A. Sinclair. Analysis of a classical matrix preconditioning algorithm. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 831–840, 2015.

16 L. N. Trefethen and M. Embree. *Spectra and pseudospectra: The behavior of nonnormal matrices and operators*. Springer, 2005.

17 N. E. Young, R. E. Tarjan, and J. B. Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks*, 21(2):205–221, 1991. `doi:10.1002/net.3230210206`.