A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents

Nathan Linial [*] Alex Samorodnitsky [†] Avi Wigderson [‡]

Abstract

We present a deterministic strongly polynomial algorithm that computes the permanent of a nonnegative $n \times n$ matrix to within a multiplicative factor of e^n. To this end we develop the first strongly polynomial-time algorithm for matrix scaling - an important nonlinear optimization problem with many applications. Our work suggests a simple new (slow) polynomial time decision algorithm for bipartite perfect matching, conceptually different from classical approaches.

[*]Hebrew University. Work supported in part by a grant of the Binational Israel-US Science Foundation.

[†]Hebrew University

[‡]Hebrew University. Work partially supported by grant 032-7736 from the Israel Academy of Sciences. Part of this work was done during a visit to the Institute for Advanced Study, under the support of a Sloan Foundation grant 96-6-2.

1 Introduction

1.1 The permanent

Background: Let $A = (a_{ij})$ be an $n \times n$ matrix. The number

$$per(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)},$$

where S_n is the symmetric group on n elements, is called the permanent of A. For a $0,1$ matrix A, $per(A)$ counts the number of perfect matchings in G, the bipartite graph represented by A. We freely interchange between a $0,1$ matrix and the corresponding graph.

It is $\#P$-hard to compute the permanent of a nonnegative (even $0,1$) matrix [31], and so it is unlikely to be efficiently computable exactly for all matrices. In fact, even the existence of an efficient algorithm that computes the permanent of an exponentially small fraction (appropriately defined) of all $0,1$ matrices implies a collapse of the polynomial hierarchy [9]. Planar graphs provide the only interesting class of matrices for which a (beautiful) polynomial-time algorithm is known for the permanent [20].

The realistic goal, then, is to try and *approximate* the permanent efficiently as well as possible, for large classes of matrices.

There are deep and interesting upper and lower bounds on the permanent in certain classes of matrices. The most well known is Egorychev's [7] and Falikman's [8] resolution of van der Waerden's conjecture: The permanent of a doubly stochastic $n \times n$ matrix is at least $\frac{n!}{n^n} \geq e^{-n}$. Bregman [5] resolved the Minc conjecture and proved a tight upper bound on the permanent of a zero-one matrix with given row sums. Both bounds are easy to compute, and indeed were the starting point of our approach.

Efficient poly-time probabilistic algorithms that approximate the permanent extremely tightly ($1+\epsilon$ factor) were developed for several classes of graphs, e.g., dense graphs and random graphs [18], and others. This line of work has led to an $exp(O(\sqrt{n}))$-time probabilistic algorithm that achieves $(1+\epsilon)$-approximation for every graph [19].

How well can the permanent be approximated in polynomial time? The first result that provides a $2^{O(n)}$-factor approximation for *arbitrary positive matrices* was recently obtained by Barvinok [2]. His work can be viewed as a continuation of earlier papers where certain random variables are shown to be unbiased estimators for the permanent. Barvinok introduced new such estimators for which he was able to establish a strong enough concentration of measure. He went on [3], [4] to develop a probabilistic polynomial time algorithm with a c^n-factor approximation, $c \approx 1.31$, currently the record in this area.

We remark [3] that any polynomial-time $2^{n^{1-\alpha}}$ approximation, $0 < \alpha \leq 1$ yields a polynomial-time $(1+\epsilon)$-approximation.

Our results: We achieve $O(e^n)$-factor approximation *deterministically.* Our algorithm is strongly polynomial.

Theorem 1.1: *There is a function f, such that*

$$per(A) \leq f(A) \leq e^n per(A)$$

holds on every nonnegative $n \times n$ matrix A. The function f is computable in $\tilde{O}(n^5)$ elementary operations.

Our approach to this problem is completely different from the previously taken routes. It involves a natural reduction technique between problem instances: *scaling.* Observe the following linearity of permanents: Multiplying a row or column by a constant c, multiplies the permanent by c as well. More generally, we say that a matrix B is a scaling of A (by positive vectors $x, y \in (\Re^+)^n$) if $B = XAY$, where $X = diag(x)$ and $Y = diag(y)$ are diagonal matrices with x (resp. y) on their diagonal (these being the factors that multiply the rows and the columns respectively). As observed, $per(B) = (\prod_i x_i)(\prod_i y_i)per(A)$. Thus scaling reductions not only allow us to compute $per(A)$ from $per(B)$, but in fact any k-factor approximation of $per(B)$ efficiently yields the same approximation for $per(A)$.

The idea, then, is to scale an input matrix A into a matrix B whose permanent we can efficiently approximate. A natural strategy is to seek an efficient algorithm for scaling A to a *doubly stochastic* B. For suppose we succeed: the permanent of B is clearly at most 1, and $per(B) \geq n!/n^n > e^{-n}$ by the lower bound of [7, 8]. Consequently, $per(A)$ is also approximated to within an e^n factor, as claimed.

Note that such a scaling may not always exist - when $per(A) = 0$. Moreover, even if scaling exists, the scaling vectors x, y may have irrational coordinates, so we may have to settle for an *approximately* doubly stochastic matrix. The scaling algorithm must, therefore, be accompanied by approximate versions of the van der Waerden bound, and indeed we prove results of the following type (see also proposition 5.1).

Lemma 1.2: *Let B be a nonnegative $n \times n$ matrix, in which all row sums are 1, and where no column sum exceeds $1 + \frac{1}{n^2}$, then $per(B) \geq e^{-(n+1)}$.*

So we want to efficiently scale A to an almost doubly stochastic matrix. This scaling problem that so naturally arose from our considerations, turned out to have been studied in other contexts as well. The next subsection briefly describes these as well as scaling algorithms - old and new.

1.2 Matrix Scaling

Background

Our discussion will be restricted to square matrices, though everything generalizes to rectangular matrices (and some of it even to multidimensional arrays).

Let $r, c \in (\Re^+)^n$ be two positive vectors with $\sum r_i = \sum c_j$. A matrix B is an (r, c)-matrix if r and c are the vectors of row and columns sums of B respectively. An (r, c)-scaling of a matrix A consists of two vectors x, y for which $B = XAY$ is an (r, c)-matrix. (Again $X = diag(x)$ and $Y = diag(y)$.) Given A, r, c one is led to consider the existence of a scaling, its uniqueness, and of course the complexity of deciding and finding (or approximating) such a scaling. Note that scaling to a doubly stochastic matrix is exactly $(\bar{1}, \bar{1})$-scaling. Since multiplying the vectors r and c by the same constant does not change the problem, we will assume $\sum r_i = \sum c_j = n$ (to make it consistent with the doubly stochastic case).

The (r, c)-scaling problem arose in such diverse contexts as statistics [30, 10], numerical analysis [32], operations research [28], image reconstruction [17], engineering [6], and as we were not surprised to discover, permanents [12]. Here are a couple of quick examples: To solve the linear system of equations $Av = u$, we consider instead the scaled system $B(Yv) = X^{-1}u$. A judicious choice of scaling can increase numerical stability, and indeed this method is used for matrix preconditioning [32]. In computer tomography and some statistical applications we wish to reconstruct a (sometimes multidimensional) array of new data, from complete old data and some additional information. Different dimensions represent essential (spatial or statistical) parameters. The assumption is that data was modified as a result of a (multiplicative) change in "intensity" or "importance" of the parameters, leading naturally to the scaling problem. Finally, observe also that scaling does not affect the zero/nonzero pattern of the matrix, and this (along with linear constraints on the marginals and positivity requirements) is a natural constraint in various optimization problems.

The mathematical literature abounds with equivalent formulations of the problem, connections with other problems, and various necessary and sufficient conditions for existence and uniqueness of an (r, c)-scaling (see e.g. [29] and the references therein). Again our discussion is very brief: The max-flow-min-cut theorem provides a necessary and sufficient condition for the existence of scaling. The scaling vectors are then the (unique) optimal solution of a nonlinear program which can be made convex (but ugly) after appropriate changes of variables.

Finally, the algorithmics: Here the literature seems quite sparse. We note that (r, c)-scalability can be decided in polynomial time, using network flows. Thus we will assume that the input matrix A is (r, c)-scalable, and the task is to find (or approximate to a desired accuracy) the target matrix and the scaling vectors.

The original paper by Sinkhorn [30] already introduced an iterative algorithm for this task[1]. The idea is simple – apply a sequence of scalings to the given matrix A, alternately normalizing the row sums to r (say in odd steps) and the column sums to c (in even steps). Let A_t be the matrix generated after t iterations. Sinkhorn proved that this sequence converges to an (r, c)-matrix, but gave no bound on convergence rates. It should be clear that each iteration is computationally trivial, and that the final scaling vectors are obtained by keeping a running product of the scaling vectors used at odd and even steps.

To quantify convergence rates, let ϵ be the desired accuracy (say in L_∞), and $L(A)$ the binary input length (i.e. log of the ratio of the largest to smallest nonzero entries) of A.

The convergence rate of this procedure was first considered by Franklin and Lorenz [11]. They showed that each step in Sinkhorn's method is a contraction map in the Hilbert projective metric. By estimating the contraction constant in terms of $L(A)$, they concluded that the number of iterations is bounded by

$$O(L(A) \cdot 1/\epsilon).$$

This shows that Sinkhorn's is an approximation scheme, but of course not polynomial in $\log(1/\epsilon)$.

Moreover, simple examples show that the dependence on the input length cannot be improved. This is the best existing bound for the general (r, c)-scaling problem.

For the $(\overline{1}, \overline{1})$-scaling problem, Kalantari and Kachiyan [21] have recently developed an approximation scheme based on convex optimization via the ellipsoid method. Their algorithm requires

$$O(L(A)n^4 \log(1/\epsilon))$$

arithmetic operations on integers of length $O(L(A) \log(n/\epsilon))$. This is thus a polynomial time approximation scheme, which is, however, not strongly polynomial.

This still leaves open the polynomiality of the general (r, c)-problem, and the quest for a strongly polynomial scheme, both of which we accomplish in the present paper.

Our results: We develop two *strongly* polynomial algorithms. The first is restricted to the $(\overline{1}, \overline{1})$-scaling problem, and the other works for the general (r, c)-scaling problem.

For $(\overline{1}, \overline{1})$-scaling our approach is this: We start with a certain preprocessing step that's followed by Sinkhorn's procedure. This, along with a different convergence analysis, yield a strongly polynomial scheme. Our preprocessing is itself also a scaling step that is attained through a weighted matching algorithm. Following this step, there is a generalized diagonal of A each element of which is largest in its respective row. This guarantees a lower bound of n^{-n} on the permanent. The analysis then proves (and this

[1]Sinkhorn suggested his procedure (sometimes called the RAS procedure) only for the $(\overline{1}, \overline{1})$-scaling problem he was interested in, but it was naturally extended to the general problem.

is simple) that every step in Sinkhorn's procedure increases the permanent. Moreover, this increase can be quantified in terms of the distance of the matrix from being doubly stochastic. This allows us to bound the number of iterations till an ϵ approximation is attained by

$$O((n/\epsilon)^2)$$

Note that the number of iterations does not involve $L(A)$ (Naturally the arithmetic operations are applied to integers of length $L(A)^{O(1)}$.) However, it is only polynomial in $1/\epsilon$, so this is not a fully polynomial approximation scheme. Still, for the purpose of approximating the permanent $\epsilon = n^{-2}$ suffices, and Theorem 1.1, (with a total running time of $O(n^6)$) follows. These results are described in Section 3.

For the general (r, c)-scaling problem, we develop a new algorithm. Like Sinkhorn's it is iterative, and every odd step normalizes the row sums to r. In the even steps, however, a more elaborate scaling procedure is performed on the columns. This scheme allows us to quantify the progress we make: After odd steps, when row sums are as designated, the Euclidean distance of the vector of current columns sums to its target c shrinks by a constant factor. This comprises a fully polynomial approximation scheme that is also strongly polynomial, with the number of iterations bounded by

$$\tilde{O}(n^7 \log(1/\epsilon)).$$

These results are described in section 4. [2]

2 Preliminaries

We start with a general overview of the setting in which our algorithms work and of the features they share. We also discuss the complexity matters, and introduce the necessary notation.

First we describe the setting in which our algorithms are applicable. Let A be a nonnegative $n \times n$ matrix and let $r = (r_1, ..., r_n)$, $c = (c_1, ..., c_n)$ be two positive vectors. We say that A is *an* (r, c) *matrix* if the i-th row sum of A is r_i and the j-th column sum is c_j.

Next we would like to introduce the notion of (r, c)-scalability. It turns out that there are several pertinent definitions, which we now present.

The matrix A is said to be *exactly* (r, c)*-scalable*, if there exist positive finite n-tuples $(x_1, ..., x_n)$, $(y_1, ..., y_n)$ such that $B = (b_{ij}) = (x_i a_{ij} y_j)$ is an (r, c) matrix.

[2] Leonid Gurvits [15] had pointed out to us, that algorithms of similar spirit are presented, without performance analysis, in [27].

We are also interested in a weaker notion of approximate scalability: Given an $\epsilon > 0$, we say that B is an "$\epsilon - (r,c)$" if B's row sums are given by the vector r, whereas B's columns sums c'_j, satisfy [3]:

$$\sum_{j=1}^{n}(c'_j - c_j)^2 \leq \epsilon. \tag{1}$$

We say that A is $\epsilon - (r,c)$-*scalable* if there exist positive finite n-tuples $(x_1,...,x_n)$, $(y_1,...,y_n)$ such that $B = (b_{ij}) = (x_i a_{ij} y_j)$ is an $\epsilon - (r,c)$ matrix.

Since our official business (in the doubly stochastic case) is in approximating the permanent, we will be satisfied if we can scale A *arbitrarily close* to a (r,c)-matrix. This leads to a notion of an almost (r,c) scalable matrix, which turns out to be precisely the notion we need, and therefore deserves a formal definition:

Definition 2.1: A nonnegative $n \times n$ matrix is *almost* (r,c)-*scalable* if it is $\epsilon - (r,c)$ scalable for any $\epsilon > 0$. ∎

The following Proposition [29] characterizes exact and approximate scalability.

Proposition 2.2: *A nonnegative matrix A is exactly (r,c)-scalable iff for every zero minor $Z \times L$ of A,*

1. $$\sum_{i \in Z^c} r_i \geq \sum_{j \in L} c_j \quad \textit{equivalently} \quad \sum_{i \in Z} r_i \leq \sum_{j \in L^c} c_j.$$
2. *Equality in 1 holds iff the $Z^c \times L^c$ minor is all zero as well.*

A nonnegative matrix A is almost (r,c)-scalable iff condition 1 holds.

In particular, a nonnegative matrix A is almost $(\overline{1},\overline{1})$-scalable iff $Per(A) > 0$.

Note that the (r,c) scaling algorithm presented in Theorem 4.5 gives an alternative proof of the sufficiency of the first condition for almost-(r,c) scalability. Indeed, the algorithm accepts as input a matrix satisfying condition 1 of the proposition and an $\epsilon > 0$, and proceeds to (efficiently) scale the matrix to an $\epsilon - (r,c)$ matrix.

We observe that almost (r,c)-scalability of A can be efficiently checked.

Lemma 2.3: *The question whether a given matrix A satisfies condition 1 of proposition 2.2 can be decided in polynomial time (via a max-flow formulation [14]).*

[3] Note it is always possible to normalize the row sums to the desired r_i. (Or, alternatively, normalize the column sums. Doing both is the challenge.) Thus, henceforth all our matrices are row-normalized.

Proof: For completeness' sake, here is the translation to a flow problem:
Let F be a graph with $2n+2$ vertices, of which two are special: the source and the sink. Corresponding to the rows of A are n vertices and the remaining n vertices correspond to the columns. The source is adjacent to every row vertex of F and the n edges have capacities $r_1, ..., r_n$. In the same way, every column vertex of F is adjacent to the sink with edge capacities $c_1, ..., c_n$. There is an edge from the i-th row vertex to the j-th column vertex iff $A_{ij} > 0$ and such edges have infinite capacities. The max-flow, min-cut theorem easily implies that the maximal flow in F is $\sum_i r_i = \sum_j c_j$ iff condition 1 holds. ∎

We proceed to describe two scaling algorithms. We call our $(\overline{1}, \overline{1})$ scaling algorithm **DSS** - for Doubly Stochastic Scaling, and the general (r, c) scaling algorithm is **RCS** - for (r, c)-Scaling.

Given a matrix A to be scaled, the two algorithms operate by producing a sequence of matrices $A = A_0, A_1, .., A_N$, where $A_{t+1} = f(A_t)$, $t = 1, ..$, and the last matrix A_N is nearly (r, c). [4] The function f is specific to the algorithm and is denoted by $f_{\mathbf{DSS}}$ and $f_{\mathbf{RCS}}$ correspondingly. The matrices A_t are always row-normalized, i.e., the i-th rows sum of A_t is r_i, for all t and n. An *iteration* of the algorithm proceeds from A_t to A_{t+1}, and therefore N is the number of iterations required for the approximate scaling. To quantify the notion of "proximity", we define $N(\epsilon)$ to be the number of iterations required for ϵ-*scaling*, namely so that A_N is an $\epsilon - (r, c)$ matrix.

Let I denote the number of arithmetic operations needed for one iteration of the algorithm. Then the total complexity of the ϵ-scaling problem is $N_{\mathbf{DSS}}(\epsilon) \cdot I_{\mathbf{DSS}}$ in the doubly stochastic case, and $N_{\mathbf{RCS}}(\epsilon) \cdot I_{\mathbf{RCS}}$ in the general case. We also set $\mathcal{L}(\epsilon)$ denote the maximal length of numbers encountered in the ϵ-scaling process. (Recall that $L(A)$ denotes the binary input length of A.)

3 Doubly stochastic scaling algorithm

In this section we present a modification of Sinkhorn's matrix scaling algorithm, which, given an $(\overline{1}, \overline{1})$ scalable, nonnegative $n \times n$ matrix converges rapidly to an *almost doubly stochastic matrix*. ("Almost" is quantified later on, but the idea is that an approximate version of the van der Waerden conjecture holds.) After preprocessing the input matrix, this algorithm performs the Sinkorn algorithm, namely repeatedly normalizes the columns and the rows.

[4] Henceforth, in this section, we refer only to (r, c) scaling, which includes the doubly stochastic scaling as a special case.

3.1 The DSS Algorithm

DSS(A)

Input: $(\overline{1}, \overline{1})$-scalable matrix A.

1. $A_1 = \text{Preprocess}(A)$
2. For $t = 1, 2, \ldots, N = O(n^2 \log(n))$ do $\qquad A_{t+1} = f_{\mathbf{DSS}}(A_t)$.
3. **Output** A_N.

Preprocess(A)

Find a permutation $\sigma \in S_n$, such that σ is the "heaviest" generalized diagonal of A, i.e.,

$$\prod_{i=1}^{n} A_{i\sigma(i)} = \max_{\tau \in S_n} \prod_{i=1}^{n} A_{i\tau(i)}.$$

Find a positive diagonal matrix Y, such that the matrix $B = AY$, satisfies:

$$\forall \ \ 1 \leq i, j \leq n \qquad B_{i\sigma(i)} \geq B_{ij}. \tag{2}$$

Normalize the rows of B.

Output A_1.

$\underline{f_{DSS}(\cdot)}$.

Input Nonnegative matrix A_t with unit row sums and positive column sums.

1. Normalize(columns)
2. Normalize(rows)
3. **Output** A_{t+1}.

3.2 Theorems

Theorem 3.1: *(The preprocessing step)*
Let A be a nonnegative matrix in which σ is a "heaviest" generalized diagonal, i.e.,

$$\prod_{i=1}^{n} A_{i\sigma(i)} \geq \prod_{i=1}^{n} A_{i\tau(i)}$$

for every permutation τ. Then, there exists a positive diagonal matrix Y such that $B = AY$, satisfies:

$$\forall \ \ 1 \leq i, j \leq n \qquad B_{i\sigma(i)} \geq B_{ij}. \tag{3}$$

The diagonal σ and the matrix Y can be found in $O(n^5 \log n)$ arithmetic operations.

Theorem 3.2: *(Number of iterations)*
The number $N_{\mathbf{DSS}}(\frac{1}{n\log n})$ *of* **DSS** *iterations required for* $\frac{1}{n\log n}$*-scaling is at most* $O(n^2\log(n))$*.*

Theorem 3.3: *(Iteration cost)*
The number of arithmetic operations per each iteration of **DSS** *is at most* $O(n^2)$*.*

Theorem 3.4: *(Number size)*
The maximal length $\mathcal{L}_{\mathbf{DSS}}(\frac{1}{n\log n})$ *of numbers that appear in the* $\frac{1}{n\log n}$*-scaling algorithm is at most poly*$(L(A),n))$*.*

Corollary 3.5: *Given an* $(\overline{1},\overline{1})$*-scalable matrix* A*, modified Sinkorn's algorithm solves the* $\frac{1}{n\log n}$*-scaling problem for* A *in at most* $O(n^5\log(n))$ *elementary operations on numbers of length at most poly* $(L(A),n))$*.*

Remark 3.6: Below (Proposition 5.1) we show that if A if $\frac{1}{n\log n}$-doubly stochastic then $per(A)\geq(\frac{1}{e})^{n+o(n)}$. Therefore Corollary 3.5 enables us to use a version of the van der Waerden conjecture. ∎

Remark 3.7: The number of iterations of the *Unmodified* Sinkorn's procedure required to solve the $\frac{1}{n\log n}$-scaling problem may be arbitrarily large. This is demonstrated in the following example:

Example 3.8: Let A be a 3×3 matrix,

$$A=\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \alpha & \alpha & 1-2\alpha \\ \alpha & \alpha & 1-2\alpha \end{pmatrix},$$

with (arbitrarily) small positive α. This matrix satisfies the conditions of proposition 2.2, so Sinkhorn's algorithm converges to a doubly stochastic matrix. However, note that $per(A)=O(\alpha)$ and that each application of **DSS** increases the permanent by a factor no bigger than 5. Consequently, it takes at least $\Omega(\log(\frac{1}{\alpha}))$ steps, for the matrix A to become close enough to doubly stochastic. ∎

∎

3.3 Proofs

Proof: Of Theorem 3.1
Given A, a "heaviest" generalized diagonal σcorresponds to a maximum-weight perfect matching in a bipartite graph, so σ may be found in at most $O(n^3)$ arithmetic operations [1, 13]. It is convenient (and kosher) to assume henceforth that σ is the identity permutation.

We turn to the main part of the theorem, namely to show that the matrix Y exists and can be found in polynomial time.

We start with the existential part. Define the matrix $\hat{A}$ via $\hat{A}_{ij} = \log(A_{ij})$, $1 \leq i, j \leq n$, (with the convention $\log(0) = -\infty$) and restate the theorem thus: Let $\hat{A}$ be an $n \times n$ matrix where $\sum_{i=1}^{n} \hat{A}_{ii} \geq \sum_{i=1}^{n} \hat{A}_{i\tau(i)}$ for every permutation τ. Then there exist numbers $\mu_1, ..., \mu_n$, such that

$$\forall \quad 1 \leq i, j \leq n, \qquad \hat{A}_{ii} + \mu_i \geq \hat{A}_{ij} + \mu_j$$

i.e.

$$\mu_i - \mu_j \geq \hat{A}_{ij} - \hat{A}_{ii}. \tag{4}$$

Note, that we are only concerned with indices i, j for which $\hat{A}_{ij} > -\infty$, since in the other case the inequalities obviously hold. By LP-duality, if this system of inequalities is inconsistent, this may be established by linearly combining these inequalities. So, assume that there exist nonnegative weights w_{ij}, so that

1. $\sum w_{ij}(\mu_i - \mu_j) = 0$, i.e., the nonconstant terms are eliminated, and
2. $\sum w_{ij}(\hat{A}_{ij} - \hat{A}_{ii}) > 0$, i.e., the combination of constant terms is positive.

The first condition implies that the map which assigns value w_{ij} to the edge (i, j), is a circulation in K_n. By the Circulation Theorem, w is a positive combination of the characteristic functions of directed cycles: $w = \sum_{\gamma \in \Gamma} \lambda_\gamma 1_\gamma$, where each $\gamma \in \Gamma$ is a directed cycle and the λ_γ are positive. But then, the second condition implies that for some $\gamma \in \Gamma$:

$$\sum_{i \in \gamma} \hat{A}_{i\gamma(i)} > \sum_{i \in \gamma} \hat{A}_{ii}.$$

Define a permutation $\tau \in S_n$, by setting $\tau(i) = \gamma(i)$ for $i \in \gamma$, and $\tau(i) = i$ otherwise. Clearly, $\sum_{i=1}^{n} \hat{A}_{i\tau(i)} > \sum_{i=1}^{n} \hat{A}_{ii}$, contrary to our assumptions.

We return to the computational part of the theorem. Note, that (4) is a system of $m = n^2$ linear inequalities with only two variables per inequality. Such a system is solvable [26] by a *strongly polynomial algorithm* in $O(mn^3 \log(n)) = O(n^5 \log(n))$ arithmetic operations. ∎

Proof: Of Theorem 3.2
Theorem 3.1 shows how to perform the *Preprocessing Step* in the Modified Sinkhorn's algorithm. Our gain is that following this preprocessing, the permanent of the resulting matrix A_1 is $\geq \frac{1}{n^n}$, since all the elements on the σ-diagonal of A_1 are $\geq \frac{1}{n}$.

Now, recall that A_t is the matrix generated from A_1 after $t-1$ iterations of **DSS**. $A_{t+1} = f_{\mathbf{DSS}}(A_t)$. Our goal is to estimate the convergence rate of A_t to the set of doubly stochastic matrices. We measure our progress with the help of a potential function, namely, $per(A_t)$.

Lemma 3.9: *Let B be a nonnegative matrix with unit row sums. Then*

$$per(\mathbf{DSS}(B)) \geq per(B)$$

and the inequality is strict unless B is doubly stochastic.

Proof: Let C be obtained by normalizing B's columns, and $D = \mathbf{DSS}(B)$ be the row normalization of C. We claim $per(D) \geq per(C) \geq per(B)$, and the inequalities are strict unless $B = C = D$, are doubly stochastic.

Let c_j be the j-th column sum of B. All c_j are positive, and $\sum_{j=1}^n c_j = n$. Clearly, $per(C) = \frac{per(B)}{\prod_{j=1}^n c_j}$. The arithmetic-geometric inequality applied to $\{c_j\}_{j=1}^n$ yields $\prod_{j=1}^n c_j \leq 1$, with equality only if all the c_j are 1. The same argument proves also the second claim, switching rows and columns. ∎

We have seen, in fact, that each iteration of **DSS** multiplies the permanent by at least $\frac{1}{\prod_{j=1}^n c_j}$. The convergence rate can be estimated through the following lemma:

Lemma 3.10: *Let $x_1, ..., x_n$ be positive reals with $\sum_{i=1}^n x_i = n$, and $\sum_{i=1}^n (x_i - 1)^2 = \Delta$. Then*

$$\prod_{i=1}^n x_i \leq 1 - \frac{\Delta}{2} + O(\Delta^{\frac{3}{2}}).$$

Proof: Let $x_i - 1 = \rho_i$. Then $\sum_{i=1}^n \rho_i = 0$, and $\sum_{i=1}^n \rho_i^2 = \Delta$.
It is easily verifiable, that $1 + t \leq e^{t - \frac{t^2}{2} + \frac{t^3}{3}}$, for all real t, therefore:

$$\prod_{i=1}^n x_i = \prod_{i=1}^n (1 + \rho_i) \leq \exp(\sum_{i=1}^n \rho_i - \frac{1}{2}\sum_{i=1}^n \rho_i^2 + \frac{1}{3}\sum_{i=1}^n \rho_i^3) \leq$$

$$\leq \exp(-\frac{1}{2}\sum_{i=1}^n \rho_i^2 + \frac{1}{3}(\sum_{i=1}^n \rho_i^2)^{\frac{3}{2}}) =$$

$$= \exp(-\frac{\Delta}{2} + \frac{1}{3}\Delta^{\frac{3}{2}}) = 1 - \frac{\Delta}{2} + O(\Delta^{\frac{3}{2}}).$$

∎

Now we have to show that $N_{\mathbf{DSS}}((\frac{1}{n\log n})) \leq O(n^2\log^2 n)$. Indeed, as long as Condition (1), with $\epsilon = \frac{1}{n\log n}$, doesn't hold, each iteration of **DSS** multiplies the permanent by at least $1+\Omega(\frac{1}{n\log n})$. The bound on N follows, since $per(A_1) \geq \frac{1}{n^n}$ and for any t, $per(A_t) \leq \prod_{i=1}^n r_i = 1$. ∎

Proof: Of Theorem 3.3
The statement of the theorem follows immediately from the definition of the **DSS**-iteration. ∎

Proof: Of Theorem 3.4
To perform the preprocessing, we solve a special system of n^2 linear inequalities $a_{ii}y_i \geq a_{ij}y_j$, for $1 \leq i,j \leq n$. As already mentioned, we employ here the *strongly polynomial algorithm* from [26].

Now, let $f_{\mathbf{DSS}}$ act on $A = (a_{ij})$ and produce $A' = (a'_{ij})$. We observe that for our purposes it suffices to remember only the first $O(\log(n/\epsilon))$ significant bits of a_{ij} (see lemma 4.8 for a proof).

Since all the entries in A, A' are numbers between zero and one, every iteration of **DSS** decreases an entry by a multiplicative factor of at most n^2, and therefore increases the input length by at most $2\log n$. ∎

4 Strongly polynomial scaling algorithm

In this section we present a strongly polynomial algorithm, which, given an (r,c)-scalable nonnegative $n \times n$ matrix A and an $\epsilon > 0$, solves the ϵ-scaling problem for A.

4.1 The RCS Algorithm

RCS(A)

Input: Nonnegative vectors r and c; an almost (r,c)-scalable matrix $A = A_1$ and an $\epsilon > 0$.

1. For $t = 1, 2, \ldots N = O(n^5\log(\frac{n}{\epsilon}))$ do $\quad A_{t+1} = f_{\mathbf{RCS}}(A_t)$.
2. **Output** A_N.

$f_{RCS}(\cdot)$.

Input An almost (r,c)-scalable matrix A_t with row sums r_i and positive column sums c'_j.

1. Sort the differences $d_j = c'_j - c_j$. (We assume, for convenience, that they are already ordered: $d_1 \leq d_2 ... \leq d_n$.)

2. If $\sum_{j=1}^{n} d_j^2 \leq \epsilon$ **Stop**.
Remark: in this case we are done.

3. Otherwise, let j_0 be the index where the largest gap occurs between consecutive d_j, and set:

$$G := d_{j_0+1} - d_{j_0} = max_j\{d_{j+1} - d_j\}.$$

4. Find δ_0, the smallest positive δ, such that $|a_{\mu\nu}(\delta) - a_{\mu\nu}| = \frac{G}{8n}$ for some indices μ and ν. Here $A_t(\delta) = (a_{ij}(\delta))$ is the matrix obtained from A_t by multiplying each *lacunary column* $j \in L := \{1...j_0\}$ by a factor $1 + \delta$, and then renormalizing the rows.

5. **Output** $A_{t+1} = A_t(\delta_0)$.

4.2 Theorems

Theorem 4.1: *Existence of* δ_0
An **RCS** *iteration can always be performed, i.e., the positive real* δ_0 *in step 4 of the iteration is well defined.*

Theorem 4.2: *Number of iterations*
The number of **RCS** *iterations required for* ϵ*-scaling is at most* $N_{\mathbf{RCS}}(\epsilon) \leq O(n^5 \log(\frac{n}{\epsilon}))$.

Theorem 4.3: *Iteration cost*
Each iteration of **RCS** *involves at most* $O(n^2 \log(n))$ *elementary operations.*

Theorem 4.4: *Number size*
The maximal length $\mathcal{L}_{\mathbf{RCS}}(\epsilon)$ *of numbers that appear in the* ϵ*-scaling algorithm is at most poly* $\Big(L(A), n, \log(\frac{1}{\epsilon}\Big)$.

Corollary 4.5: *Given an* (r, c)*-scalable matrix* A*,* **RCS** *algorithm solves the* ϵ*-scaling problem for* A *in at most* $O(n^7 \log(n) \log(\frac{n}{\epsilon}))$ *elementary operations on numbers of length at most poly* $\Big(L(A), n, \log(\frac{1}{\epsilon})\Big)$.

4.3 Proofs

Proof: Of Theorem 4.1
For notational convenience we prove the existence and the uniqueness of δ_0 for the first

iteration. In this case, A_t is simply A. Let δ be a positive real number. The expression for the (i,l) entry in $A(\delta)$ is:

$$a_{il}(\delta) = \begin{cases} (1+\delta)\cdot\frac{a_{il}r_i}{r_i+\delta w_i} & \text{if } l \in L \\ \frac{a_{il}r_i}{r_i+\delta w_i} & \text{if } l \notin L \end{cases},$$

where $w_i = \sum_{j\in L} a_{ij}$ is the contribution of the lacunary columns to the i-th row sum in A. Since all $w_i \le r_i$, the function $a_{ij}(\delta)$ is nondecreasing for $j \in L$, and nonincreasing for $j \notin L$. Therefore the quantity $d_j(\delta) = c'_j(\delta) - c_j$ is nondecreasing for $j \in L$, and nonincreasing for $j \notin L$. We show that as δ grows, at least one of these quantities reaches the midpoint M of the largest gap G.

Lemma 4.6: *Let A be a nonnegative (r,c)-scalable $n \times n$ matrix with row sums r_i and column sums c'_j. Then, there are $j \in L$ and $k \notin L$ and a (possibly infinite) $\delta > 0$ for which $d_j(\delta) = d_k(\delta)$. Consequently, there is an index l and $\delta > 0$ for which $d_l(\delta) = M$.*

Proof: Let Z be the set of rows i for which $w_i = 0$. Note that $a_{ij} = 0$ for $i \in Z$ and $j \in L$, so for $l \in L$, $c'_l(\delta) = (1+\delta)\cdot\sum_{i\in Z^c}\frac{a_{ij}r_i}{r_i+\delta w_i}$. Therefore, for $l \in L$, when $\delta \to \infty$,

$$d_l(\delta) \to \sum_{Z^c}\frac{a_{il}r_i}{w_i} - c_l \quad .$$

Summing these expressions for all $l \in L$ we conclude:

$$\sum_{l\in L} d_l(\delta) \to \sum_{l\in L}(\sum_{Z^c}\frac{a_{il}r_i}{w_i} - c_l) = \sum_{i\in Z^c} r_i - \sum_{l\in L} c_l \ge 0.$$

The inequality being from Proposition 2.2. On the other hand, for $j \notin L$, as $\delta \to \infty$,

$$d_j(\delta) \to \sum_{Z} a_{ij} - c_j$$

whence

$$\sum_{j\notin L} d_j(\delta) \to \sum_{j\notin L}\sum_{i\in Z} a_{ij} - \sum_{j\notin L} c_j = \sum_{i\in Z} r_i - \sum_{j\notin L} c_j \le 0$$

again by Proposition 2.2.

Therefore two indices $j \in L$, $k \notin L$ exist for which $\lim_{\delta\to\infty}(d_j(\delta) - d_k(\delta)) \ge 0$. ∎

Recall that M is the middle of the interval $[d_{j0}, d_{j0+1}]$. The Lemma tells us, that there is a column l and a possibly infinite positive δ such that $d_l(\delta) = M$, implying $|d_l(\delta) - d_l| \ge \frac{G}{2}$. It follows, that $|d_l(\delta) - d_l| \ge \frac{G}{4}$, for some positive *finite* δ. Therefore there is a row i, for which $|a_{il}(\delta) - a_{il}| \ge \frac{G}{4n}$. This of course implies the existence of δ_0, the smallest positive real for which $|a_{\mu\nu}(\delta) - a_{\mu\nu}| = \frac{G}{8n}$ for some indices μ and ν. (Note, that we settle for a step - $\frac{G}{8n}$ - that is smaller than what can be attained, namely $\frac{G}{4n}$. This choice is intended to control the length of the binary representation of the matrix

$A(\delta)$ - see Theorem 4.4). Note also, that δ_0 is *uniquely* specified in view of the fact that the functions $a_{ij}(\delta)$ are monotone and continuous. ∎

Proof: Of Theorem 4.2
In order to find out how many iterations of **RCS** are required for the solution of the ϵ-scaling problem, we have to estimate the rate of convergence of the sequence $\{A_t\}$ to the set of (r,c)-matrices. We assess our progress, through the decrease of the l_2 norm $\|d\|_2 = \sqrt{\sum_{j=1}^n d_j^2}$. We now show that our choice of δ_0 in the fourth step of the **RCS** iteration suits our needs. Once again we assume, for convenience, that we deal with the *first iteration*

Lemma 4.7:

$$\|d(\delta_0)\|_2^2 \leq \|d\|_2^2 \cdot (1 - \Omega(n^{-5})).$$

Proof: First show that

$$\|d(\delta_0)\|_2^2 \leq \|d\|_2^2 - G^2/64n^2.$$

Let $\mathbf{j} = (1, ..., 1)$. Since both inner products $< d(\delta_0), \mathbf{j} >$ and $< d, \mathbf{j} >$ are 0, it follows that

$$\|d\|_2^2 - \|d(\delta_0)\|_2^2 = \|d - M \cdot \mathbf{j}\|_2^2 - \|d(\delta_0) - M \cdot \mathbf{j}\|_2^2, \tag{5}$$

and so we may consider the change in $\|d - M \cdot \mathbf{j}\|_2$. The definition of δ_0 implies for all i, that $a_{ij} \leq a_{ij}(\delta_0) \leq a_{ij} + \frac{G}{8n}$, for all $j \in L$, and $a_{ik} \geq a_{ik}(\delta_0) \geq a_{ik} - \frac{G}{8n}$ for all $k \notin L$. Therefore, for all $j \in L, k \notin L$, $d_j \leq d_j(\delta_0) \leq M \leq d_k(\delta_0) \leq d_k$, implying that $|d_j(\delta) - M| \leq |d_j - M|$ for all $1 \leq j \leq n$. Our gain comes from the ν-th coordinate, and is, at least $\frac{G^2}{64n^2}$.

To conclude, we need to compare between G and $\|d\|_2^2$. Recall that $\sum d_j = 0$ and $|d_j - d_{j-1}| \leq G$ for all j. The maximum of $\|d\|_2^2$ under these conditions is attained when $d_j - d_{j-1} = G$ for all j, in which case $\|d\|_2^2 = \Theta(n^3 G^2)$. In other words, $G \geq \Omega(\|d\|_2 \cdot n^{-3/2})$. This, together with (5) implies the claim of the lemma. ∎

It immediately follows, that in $N = O(n^5 \log(\frac{n}{\epsilon}))$ iterations we have $\|d\|_2^2 \leq \epsilon$, and therefore A_N is an $\epsilon - (r,c)$ matrix. ∎

Proof: Of Theorem 4.3
To perform the t-th iteration of **RCS** we have to find δ_0 and then compute the matrix $A_t(\delta_0)$. Given δ_0, the computation of $A_t(\delta_0)$ requires $O(n^2)$ elementary operations. To find δ_0 we define δ_{ij} for each pair of indices i, j via $|a_{ij}(\delta_{ij}) - a_{ij}| = \frac{G}{8n}$ which is a linear equation in δ_{ij}, and $\delta_0 = \min_{ij} \delta_{ij}$. Consequently, this involves only $O(n^2 \log(n))$ elementary operations. ∎

Proof: Of Theorem 4.4
Let an iteration **RCS** act on $A_t = (a_{ij})$ and produce $A_{t+1} = (a'_{ij})$. Since all the entries

in A_t, A_{t+1} are numbers between zero and one, we only have to worry about an entry in A_{t+1} suddenly becoming very small - thus requiring a long representation.

This is probably the right moment to specify that the entries of A_t are represented in floating point. Namely, a_{ij} is represented as (b_{ij}, α_{ij}), where an integer b_{ij} and $\frac{1}{2} < \alpha_{ij} \le 1$ are uniquely determined via $a_{ij} = 2^{-b_{ij}} \alpha_{ij}$.

Our first observation is, that for our purposes it suffices to remember only the first few bits of α_{ij}. This is the contents of the following lemma.

Lemma 4.8: *Truncating all but the most significant* $12\log(\frac{n}{\epsilon})$ *bits in every* α_{ij} *changes the permanent by a multiplicative factor of at most* $e^{O(\epsilon^{12}/n^{11})}$*; and changes* $\|d\|_2^2 = \sum_{j=1}^{n}(c'_j - c_j)^2$ *by an additive term of at most* $O(\frac{\epsilon^{12}}{n^9})$*.*

Proof: The truncation affects each entry by a multiplicative factor of at most $1 + \frac{\epsilon^{12}}{n^{12}}$. This gives the first claim of the lemma.

Since all entries are bounded above by n, the truncation affects each entry by an additive term of at most $\frac{\epsilon^{12}}{n^{11}}$. Therefore each c'_j is changed by at most $\frac{\epsilon^{12}}{n^{10}}$ and the second claim of the lemma follows. ∎

Corollary 4.9: *The "truncated"* **RCS** *algorithm terminates in at most* $O(n^5\log(\frac{n}{\epsilon}))$ *iterations.*

It follows that the representation length required will be polynomial in $\log(b_{ij})$, $\log(n)$ and $\log(\frac{1}{\epsilon})$. It therefore suffices to control the growth of $B = \max_{ij} b_{ij}$. Namely, to prove the proposition we have to verify that $\log(B) < poly(L(A), n)$.

Let $\chi(A)$ be the smallest non-zero entry of A. We show:

$$\chi(A') > \Omega\left(\frac{\chi^2(A)}{n}\right). \tag{6}$$

Consequently, after t iterations $B \le O\left(2^t \cdot (2^{L(A)} + t\log(n))\right)$. Since **RCS** is iterated only poly(n) times, this suffices for our purposes.

Recall, that we know exactly how the entries change:

$$a'_{il} = \begin{cases} (1+\delta)\cdot\frac{a_{il}r_i}{r_i+\delta w_i} & \text{if } l \in L \\ \frac{a_{il}r_i}{r_i+\delta w_i} & \text{if } l \notin L \end{cases}, \tag{7}$$

where δ is the smallest positive real for which

$$|a'_{\mu\nu} - a_{\mu\nu}| = \frac{G}{8n}. \tag{8}$$

for some indices μ and ν. It is only for $l \notin L$ that a_{il} decreases, so we have to bound a'_{il} for $l \notin L$ from below. By lemma 4.6 there exist indices s, t for which

$$|a'_{st} - a_{st}| = \frac{G}{4n}. \tag{9}$$

We will assume that the equality in (9) is obtained for $t \in L$ (the other case is quite similar). It is convenient at this point to introduce the following notation: For any pair of indices i and j and for any positive Δ, let us denote by $\delta_{ij}(\Delta)$ the minimal positive x such that $|a_{ij}(x) - a_{ij}| = \Delta$, *if such an x exists.*

By (9) and the explicit formula (7) we conclude:

$$\delta_{st}(\frac{G}{4n}) = \frac{Gr_s}{4na_{st}(r_s - w_s) - Gw_s} < +\infty.$$

In particular, the denominator $\rho = 4na_{st}(r_s - w_s) - Gw_s$ is strictly positive, i.e. $4na_{st}(r_s - w_s) > Gw_s$. This implies:

$$\delta = \delta_{\mu\nu}(\frac{G}{8n}) \le \delta_{st}(\frac{G}{8n}) = \frac{Gr_s}{8na_{st}(r_s - w_s) - Gw_s} \le \frac{Gr_s}{Gw_s} = \frac{r_s}{w_s}.$$

Therefore, for any $1 \le i \le n$, $l \notin L$

$$a'_{il} = \frac{a_{il}r_i}{r_i + \delta w_i} \ge \frac{a_{il}}{\delta \cdot \frac{w_i}{r_i} + 1} \ge \Omega\left(\frac{a_{il}}{\delta}\right) \ge \Omega\left(\frac{a_{il}w_s}{r_s}\right) \ge \Omega\left(\frac{a_{il}a_{st}}{r_s}\right).$$

Since $r_s \le n$, we obtain $a'_{il} \ge \Omega\left(\frac{\chi^2(A)}{n}\right)$, proving (6), and we are done. ∎

5 Approximating the permanent

In this section we prove Theorem 1.1. Since our algorithms produce matrices that are only approximately doubly stochastic, we need lower bounds for the permanent of such matrices.

Proposition 5.1: *Let A be a nonnegative $n \times n$ matrix in which all row sums are 1. Let c_j be the j-th column sum of A and let $\epsilon > 0$. If*

$$\sum_{j=1}^{n}(c_j - 1)^2 < \frac{1}{n^{1+\epsilon}},$$

then

$$per(A) \ge \Omega\left(\frac{1}{e^{n(1+n^{-\frac{\epsilon}{2}})}}\right).$$

In particular,
$$\sum_{j=1}^{n}(c_j-1)^2 < \frac{1}{n\log n},$$
implies
$$per(A) \geq \Omega\left(\frac{1}{e^{n(1+\frac{1}{\sqrt{\log n}})}}\right).$$

Proof: We start with a simple lemma.

Lemma 5.2: *With the same notations, if*
$$\sum_{j=1}^{n}(c_j-1)^2 < \frac{1}{n},$$
then
$$per(A) > 0.$$

Proof: If $per(A) = 0$, then by König-Hall:
$$A = \begin{pmatrix} O & B \\ C & D \end{pmatrix}, \tag{10}$$
where O is an $s \times n-s+1$ zero submatrix. Since all row sums are 1, the sum of all entries in B is s. Since A is nonnegative, it implies $\sum_{j=n-s+2}^{n} c_j \geq s$. By Cauchy-Schwartz:
$$\sum_{j=0}^{n}(c_j-1)^2 \geq \sum_{j=n-s+2}^{n}(c_j-1)^2 \geq$$
$$\frac{1}{s-1}(\sum_{j=n-s+2}^{n}(c_j-1))^2 \geq \frac{1}{s-1} > \frac{1}{n}.$$
∎

We return to the proof of the proposition. We claim that A can be expressed as $A = D + Z$. Here $D = \lambda\Delta$ with Δ doubly stochastic and $\lambda \geq 0$; Z is nonnegative, and $per(Z) = 0$. The only problem in finding such a decomposition is in satisfying the requirement $per(Z) = 0$. So, suppose that we have such a decomposition with $per(Z) > 0$. Then, there is a permutation π, with $\min_i Z_{i,\pi(i)} = \alpha > 0$. Let $P = P_\pi$ be the corresponding permutation matrix. Note that $D' = D + \alpha P$ is also a positive multiple of a doubly stochastic matrix. Let $Z' = Z - \alpha P$. Replace the representation $A = D + Z$ by $A = D' + Z'$. After a finite number of repetitions, a decomposition $A = D_0 + Z_0 = \lambda\Delta_0 + Z_0$ with $per(Z_0) = 0$ is obtained. If $\lambda = 1$, then $Z_0 = 0$, the zero matrix, A is doubly stochastic, and $per(A) \geq \frac{n!}{n^n}$, so we are done. If $\lambda < 1$, consider the matrix $B = \frac{Z_0}{1-\lambda}$. The row sums in B are 1 and its column sums are $c'_j = \frac{c_j - \lambda}{1-\lambda}$. Clearly,

$$\sum_{j=1}^{n}(c'_j - 1)^2 = \frac{1}{(1-\lambda)^2} \cdot \sum_{j=1}^{n}(c_j - 1)^2 < \frac{1}{n^{1+\epsilon}(1-\lambda)^2}.$$

On the other hand, $per(B) = 0$, so lemma 5.2 implies $\frac{1}{n^{1+\epsilon}(1-\lambda)^2} > \frac{1}{n}$. That is, $\lambda > 1 - n^{-\frac{\epsilon}{2}}$. The proof is concluded by observing that

$$per(A) \geq per(D_0) \geq \lambda^n per(\frac{D_0}{\lambda}) \geq \Omega(e^{-n^{1-\frac{\epsilon}{2}}}) \cdot \frac{n!}{n^n}$$
$$\geq \Omega\left(\frac{1}{e^{n(1+n^{-\frac{\epsilon}{2}})}}\right).$$

∎

Proposition 5.1 together with the estimates on the running time of the **DSS**algorithm imply:

Corollary 5.3: *Let A be a nonnegative $n \times n$ matrix with a positive permanent. Then scaling factors $x_1, ..., x_n$ and $y_1, ..., y_n$ may be found, such that the matrix $B = (b_{ij}) = (x_i a_{ij} y_j)$ is nearly doubly stochastic. Specifically*

$$(\frac{1}{e})^{n+o(n)} \leq per(B) \leq 1.$$

The scaling factors can be found in at most $O(n^5 \log^2 n)$ elementary operations.

Since $per(B) = \prod_{i=1}^{n} x_i \prod_{j=1}^{n} y_j \; per(A)$ the corollary implies theorem 1.1, and we are done.

5.1 Perfect Matchings

The goal of this short subsection is to emphasize the following curious and potentially promising aspect of this work. Our new scaling algorithm may be used to decide whether a given bipartite graph has a perfect matching. The approach we use is conceptually different from known methods. [5] Moreover, it is certainly the simplest to describe and code. We cannot resist the temptation to present it here - it is a simple exercise to prove its correctness, which otherwise might be derived from corollary 3.2 and lemma 5.2. Note that no preprocessing is required here, since the permanent of a $0-1$ matrix, if nonzero, is already known up to a multiplicative factor of n^n.

A simple perfect matching algorithm

[5] Essentially the same algorithm has been independently developed in [16].

Given a 0-1 matrix A
Begin
for $n^2 \log(n)$ iterations do
    Normalize(columns);
    Normalize(rows);
    If($\sum_{j=1}^{n}(c_j - 1)^2 < \frac{1}{n}$)
        return(*Perfect Matching*);
return(*No Perfect Matchings*);
End

The maximum matching problem [25] is, of course a classical question in the theory of algorithms. Among the more recent interesting findings is the fact that this problem is in *RNC* [24, 23]. It is a major challenge to find an *NC* algorithm for it. While our work is very far from resolving this difficulty, it may shed some new light, and may be useful in discovering alternative routes to deciding the existence of perfect matchings.

The above algorithm is polynomial-time, though much inferior to standard algorithms. We note that each iteration of the present scheme can be carried out in *NC*. If one can find another iterative *NC*-realizable scheme that requires only *polylogarithmic* many iterations, this would finally put the decision problem for perfect matchings in *NC*.

6 Conclusions

With respect to matrix scaling, we have hardly begun investigating the advantage in applying our new algorithms rather than existing ones. Nor have we studied the possible extensions of our algorithm to more general scaling and nonlinear optimization problems from this literature.

With respect to permanent approximation, it is again but a first step on a path that advocates simple reduction of the input matrix into one that may be easily approximated. An obvious challenge is of course to develop a polynomial time $(1+\epsilon)$-factor approximation scheme. At present even finding a $(1+\epsilon)^n$-factor approximation scheme seems hard and challenging. This is at present even elusive for $0, 1$ matrices with a constant number of 1's in every row and column.

7 Acknowledgement

We are grateful to Yuri Rabinovitch for opening our eyes to the fact that what we were doing was actually *matrix scaling*. We also thank Leonid Gurvits for interesting remarks.

References

[1] M. O. Ball and U. Derigs, An analysis of alternate strategies for implementing matching algorithms, *Networks* 13, 517-549, 1983.

[2] A. I. Barvinok, Computing Mixed Discriminants, Mixed Volumes, and Permanents, *Discrete & Computational Geometry* , 18 (1997), 205-237.

[3] A. I. Barvinok, A simple polynomial time algorithm to approximate the permanent within a simply exponential factor, Preprint, 1998.

[4] A. I. Barvinok, A simple polynomial time algorithm to approximate permanents and mixed discriminants within a simply exponential factor, Preprint, 1998.

[5] L. M. Bregman, Certain properties of nonnegative matrices and their permanents, *Soviet Math. Dokl.* 14, 945-949, 1973.

[6] D. T. Brown, A note on approximations of discrete probability distributions, *Inform. and Control*, 2, 386-392, 1959.

[7] G.P. Egorychev, The solution of van der Waerden's problem for permanents, *Advances in Math.*, 42, 299-305, 1981.

[8] D. I. Falikman, Proof of the van der Waerden's conjecture on the permanent of a doubly stochastic matrix, *Mat. Zametki* 29, 6: 931-938, 957, 1981, (in Russian).

[9] U. Feige and C. Lund. On the hardness of computing the permanent of random matrices, *STOC* 24, 643-654, 1992.

[10] S. E. Fienberg, *The Analysis of Cross Classified Data*, MIT press, Cambridge, MA, 1977.

[11] J. Franklin and J. Lorenz, On the scaling of multidimensional matrices, *Linear Algebra Appl.* 114/115, 717-735, 1989.

[12] S. Friedland, C. Li, H. Schneider, Additive Decomposition of Nonnegative Matrices with Applications to Permanents and Scaling, *Linear and Multilinear Algebra*,23, 63–78, 1988.

[13] Z. Galil, S. Micali and H. Gabow, Priority queues with variable priority and an $O(EV \log V)$ algorithm for finding maximal weighted matching in a general graph, *FOCS* 23, 255-261, 1982.

[14] A. V. Goldberg and R. E. Tarjan, A new approach to the maximum-flow problem, *J. Assoc. Comp. Mach.*, 35, 921–940, 1988.

[15] L. Gurvits, Private communication, 1998.

[16] L. Gurvits, P. N. Yianilos, The deflation-inflation method for certain semidefinite programming and maximum determinant completion problems, Preprint, 1998.

[17] G. T. Herman and A. Lint, Iterative reconstruction algorithms, *Comput. Biol. Med.* 6, 276, 1976.

[18] M. Jerrum and A. Sinclair, Approximating the permanent, *SIAM J. Comput.*, 18, 1149-1178, 1989.

[19] M. Jerrum and U. Vazirani, A mildly exponential approximation algorithm for the permanent, *Algorithmica*, 16(4/5), 392-401, 1996.

[20] P. W. Kasteleyn, The statistics of dimers on a lattice 1. The number of dimer arrangements on a quadratic lattice. *Physica*, 27, 1209–1225, 1961.

[21] B. Kalantari and L Khachian, On the complexity of nonnegative matrix scaling, *Linear Algebra Appl.*, 240, 87-104, 1996.

[22] N. Karmarkar, R. Karp, R. Lipton, L. Lovasz and M. Luby, A Monte-Carlo algorithm for estimating the permanent, *SIAM Journal on Computing*, 22(2), 284-293, 1993.

[23] R. M. Karp, E. Upfal and A. Wigderson, Are search and decision problems computationally equivalent? *STOC* 17, 464-475, 1985.

[24] L. Lovasz, On determinants, matchings and random algorithms, *Fundamentals of computing theory*, edited by L. Budach, Akademia-Verlag, Berlin 1979.

[25] L. Lovasz and M. D. Plummer, **Matching Theory**, North Holland, Amsterdam 1986.

[26] N. Megiddo, Towards a genuinly polynomial algorithm for linear programming, *SIAM Journal on Computing*, 12(2), 347-353, 1983.

[27] B. N. Parlett, T. L. Landis, Methods for Scaling to Doubly Stochastic Form, *Linear Algebra Appl.* 48, 53-79, 1982.

[28] T. E. S. Raghavan, On pairs of multidimensional matrices, *Linear Algebra Appl.* 62, 263-268, 1984.

[29] U. Rothblum and H. Schneider, Scaling of matrices which have prespecified row sums and column sums via optimization, *Linear Algebra Appl.* 114/115, 737-764, 1989.

[30] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, *Ann. Math. Statist.* 35, 876-879, 1964.

[31] L. G. Valiant, The complexity of computing the permanent, *Theoretical Computer Science*, 8(2), 189-201, 1979.

[32] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Her Majesty's Stationery Office, England, 1963.