

Approximating the Permanent in $O^*(n^7)$ Time

Ivona Bezáková* Daniel Štefankovič* Vijay V. Vazirani† Eric Vigoda†

November 3, 2004

Abstract

The first polynomial-time algorithm to approximate (with arbitrary precision) the permanent of a non-negative matrix was presented by Jerrum, Sinclair and Vigoda. They designed a simulated annealing algorithm with a running time of $O^*(n^{26})$ for 0/1 matrices. Subsequently, they improved their analysis, resulting in a $O^*(n^{10})$ time algorithm. We present a $O^*(n^7)$ time algorithm. Our improvement comes from an improved “cooling schedule” for the simulated annealing algorithm, and a refined analysis of the underlying Markov chain.

1 Introduction

The study of the permanent has a long history in many fields, including Mathematics [13] and Physics [10]. Within Computer Science, this problem has occupied a special place – its study, especially the computation of 0/1 permanents (which is also the same as computing the number of perfect matchings in a bipartite graph), has led to fundamental progress on the complexity of counting problems, including introduction of the class $\#\mathbf{P}$ [15], the relationship between the complexities of approximate counting and random generation for self-reducible problems [8], the Markov chain Monte Carlo (MCMC) method for random generation [1] and the relationship between conductance and mixing time of Markov chains [5].

The permanent of a non-negative matrix A , of size $n \times n$, is defined as

$$\text{per}(A) = \sum_{\pi \in S_n} \prod_i A(i, \pi(i)),$$

where S_n is the set of permutations of $\{1, 2, \dots, n\}$. Viewing A as the adjacency matrix of a bipartite graph with $n + n$ vertices, the permanent of A equals the sum of weighted perfect matchings in this graph. Hence, computing the permanent of 0/1 matrices is equivalent to computing the number of perfect matchings for bipartite graphs.

In a breakthrough result, Jerrum and Sinclair [5] showed that a Markov chain proposed by Broder [1] yields an fpras (fully polynomial randomized approximation scheme) for computing the number of perfect matchings in a bipartite graph provided the ratio of near-perfect to perfect matchings in the

*Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: {ivona, stefanko}@cs.uchicago.edu. I.B. was supported by NSF grant CCR-0237834.

†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332. Email: {vazirani, vigoda}@cc.gatech.edu. V.V. is supported by NSF grants CCR-0311541 and CCR-0220343. E.V. is supported by NSF grant CCR-0237834.

given graph is polynomially bounded. In particular, this yields an fpras for computing the permanent of dense 0/1 matrices (matrices having more 1's than 0's in every row and column). Extending this to arbitrary 0/1 matrices remained open for over a decade. The only progress in the interim was a mildly-exponential approximation algorithm, having a running time of $\exp(O(\sqrt{n}))$ [9].

Jerrum, Sinclair and Vigoda [6] presented the first fpras for arbitrary non-negative matrices. Their approach was a simulated annealing algorithm with a running time of $O^*(n^{26})$ for 0/1 matrices¹. More recently, they improved the analysis of their algorithm, resulting in $O^*(n^{10})$ time algorithm [7]. We present an $O^*(n^7)$ time algorithm for the permanent of 0/1 matrices. Our algorithm also extends to non-negative matrices.

Improving the running time of the fpras for the permanent is not only of theoretical importance, but it is also of certain practical importance. Randomly sampling 0/1 contingency tables, which is a fundamental problem in Statistics [3], can be reduced to the permanent.

Approximating the permanent, together with estimating the volume of a convex body have occupied a central place within MCMC theory. It is worth noting that the first algorithm for estimating the volume of Dyer, Frieze and Kannan, had a running time of $O^*(n^{23})$ calls to a separation oracle. A long series of papers has culminated in the recent $O^*(n^4)$ algorithm by Lovász and Vempala [12]. A key idea in the final improvement of Lovász and Vempala is an improved “cooling schedule” in a simulated annealing algorithm. A similar idea plays a prominent role in our work as well.

Going beyond dense permanents to arbitrary 0/1 permanents requires dealing with the fact that the number of near-perfect matchings may be exponentially more than the number of perfect matchings in the associated bipartite graph. The algorithm of [7] deals with this by constructing a Markov chain whose stationary probability distribution satisfies the following property: the total probability of all near-perfect matchings having holes (or unmatched vertices) at u, v is the same as the total probability of all perfect matchings, for each vertex pair u, v . Moreover, the probability of any two matchings (perfect or near-perfect) having the same hole pattern is the same. Then, sampling from this distribution $O(n^2 \log n)$ times ensures getting a random perfect matching with high probability. However, the problem of obtaining such a chain seems as hard as the problem of computing the permanent itself!

The clever idea in [7] is to start with the chain for the complete bipartite graph and gradually “fade away” non-edges (i.e., decrease their weight), each time updating the transition probabilities so that the above property is satisfied in a weighted sense. When all the non-edges are fully faded away, the required property is satisfied. This is a simulated annealing algorithm where the temperature corresponds to the weight of non-edges. Thus, at high temperature the chain is walking on perfect matchings of the complete bipartite graph, and as the temperature decreases we prefer only those matchings which are valid in the input graph.

The $O^*(n^{10})$ running time of [7] is accounted for as follows: They establish a bound of $O^*(n^6)$ on the mixing time of their basic Markov chain. The non-edges are faded away over $O(n^2 \log n)$ phases. In each phase, $O(\log n)$ samples of matchings (perfect and near-perfect) of each hole pattern are needed; there are a total of $n^2 + 1$ hole patterns.

Our improvement comes about via two ideas. The mixing time of the basic Markov chain of [7] is established via a canonical path argument due to Jerrum and Sinclair [5] and comes about by establishing an upper bound on the number of canonical paths using a single transition of the chain. Using new combinatorial properties of perfect and near-perfect matchings, we improve this upper bound by a factor of n^2 , thereby showing that the basic Markov chain has a mixing time of

¹The O^* notation hides log factors and the dependence on the error parameter.

$O^*(n^4)$. A further factor of n improvement comes about by using an improved scheme for fading away non-edges. We give a “cooling schedule” that drops the weights of non-edges in $O(n \log^2 n)$ phases. Our cooling schedule is non-uniform: it starts by decreasing weights slowly and gradually accelerates the rate.

Theorem 1. *For all $\epsilon > 0$, there exists a randomized algorithm to approximate, within a factor $(1 \pm \epsilon)$, the permanent of a 0/1 $n \times n$ matrix A in time $O(n^7 \log^4(n) + n^6 \log^3(n)\epsilon^{-2})$. The algorithm extends to arbitrary matrices with non-negative entries.*

2 Algorithm

Much of our algorithm is similar to the algorithm of [7]. We will highlight the differences as they appear. Before presenting the algorithm we need some notation.

2.1 Preliminaries

Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = n$. We will let $u \sim v$ denote the fact that $(u, v) \in E$. For $u \in V_1, v \in V_2$ we will have a positive real number $\lambda(u, v)$ called the *activity* of (u, v) . If $u \sim v$, $\lambda(u, v) = 1$ throughout the algorithm, and otherwise, $\lambda(u, v)$ starts at 1 and drops to $1/n!$ as the algorithm evolves. The activities allow us to work on the complete graph on V_1 and V_2 .

Let \mathcal{P} denote the set of perfect matchings (recall that we are working on the complete graph now), and let $\mathcal{N}(u, v)$ denote the set of near-perfect matchings with holes (or unmatched vertices) at u and v . Similarly, let $\mathcal{N}(x, y, w, z)$ denote the set of matchings that have holes only at the vertices x, y, w, z . Let \mathcal{N}_i denote the set of matchings with exactly i unmatched vertices. The set of states of the Markov chain is $\Omega = \mathcal{P} \cup \mathcal{N}_2$. For any matching M , denote its activity as

$$\lambda(M) := \prod_{(u,v) \in M} \lambda(u, v).$$

For a set S of matchings, let $\lambda(S) := \sum_{M \in S} \lambda(M)$. For $u \in V_1, v \in V_2$ we will have a positive real number $w(u, v)$ called the *weight* of the hole pattern u, v . Given weights w , the weight of a matching $M \in \Omega$ is

$$w(M) := \begin{cases} \lambda(M)w(u, v) & \text{if } M \in \mathcal{N}(u, v), \text{ and} \\ \lambda(M) & \text{if } M \in \mathcal{P}. \end{cases}$$

The weight of a set S of matchings is

$$w(S) := \sum_{M \in S} w(M).$$

For given activities, the *ideal weights* on hole patterns are the following:

$$w^*(u, v) = \frac{\lambda(\mathcal{P})}{\lambda(\mathcal{N}(u, v))} \tag{1}$$

Note that for the ideal weights all the $\mathcal{N}(u, v)$ and \mathcal{P} have the same weight. Hence, $w^*(\Omega) = (n^2 + 1)\lambda(\mathcal{P})$.

For the purposes of the proof, we need to extend the weights to 4-hole matchings. Let

$$w^*(x, y, w, z) = \frac{\lambda(\mathcal{P})}{\lambda(\mathcal{N}(x, y, w, z))}$$

and for $M \in \mathcal{N}(x, y, w, z)$, let

$$w^*(M) = \lambda(M)w^*(x, y, w, z).$$

2.2 Markov chain definition

At the heart of the algorithm lies a Markov chain MC , which was used in [7], and a slight variant was used in [1, 5]. Let $\lambda : V_1 \times V_2 \rightarrow \mathbb{R}_+$ be the activities and $w : V_1 \times V_2 \rightarrow \mathbb{R}_+$ be the weights. The state space is Ω , the set of all perfect and near-perfect matchings of the complete bipartite graph on V_1, V_2 . The stationary distribution π is proportional to w , i.e., $\pi(M) = w(M)/Z$ where $Z = \sum_{M \in \Omega} w(M)$.

The transitions $M_t \rightarrow M_{t+1}$ of the Markov chain MC are defined as follows:

1. If $M_t \in \mathcal{P}$, choose an edge e uniformly at random from M_t . Set $M' = M_t \setminus e$.
2. If $M_t \in \mathcal{N}(u, v)$, choose vertex x uniformly at random from $V_1 \cup V_2$.
 - (a) If $x \in \{u, v\}$, let $M' = M \cup (u, v)$.
 - (b) If $x \in V_2$ and $(w, x) \in M_t$, let $M' = M \cup (u, x) \setminus (y, x)$.
 - (c) If $x \in V_1$ and $(x, z) \in M_t$, let $M' = M \cup (x, v) \setminus (x, z)$.
 - (d) Otherwise, let $M' = M_t$.
3. With probability $\min\{1, w(M')/w(M_t)\}$, set $M_{t+1} = M'$; otherwise, set $M_{t+1} = M_t$.

Note, cases 1 and 2a move between perfect and near-perfect matchings, whereas cases 2b and 2c move between near-perfect matchings with different hole patterns.

The key technical theorem is that the Markov chain quickly converges to the stationary distribution π if the weights w are close to the ideal weights w^* . The mixing time $\tau(\delta)$ is the time needed for the chain to be within variation distance δ from the stationary distribution.

Theorem 2. *Assuming the weight function w satisfies inequality*

$$w^*(u, v)/2 \leq w(u, v) \leq 2w^*(u, v) \tag{2}$$

for every $(u, v) \in V_1 \times V_2$ with $\mathcal{M}(u, v) \neq \emptyset$, then the mixing time of the Markov chain MC is bounded above by $\tau(\delta) = O(n^4(\ln(1/\pi(M)) + \log \delta^{-1}))$.

This theorem improves the mixing time bound by $O(n^2)$ over the corresponding result in [7]. The theorem will be proved in Section 5.

We will run the chain with weights w close to w^* , and then we can use samples from the stationary distribution to redefine w so that they are arbitrarily close to w^* . Note, $\pi(\mathcal{P}) = \lambda(\mathcal{P})/Z$. The key observation is that

$$w^*(u, v) = \frac{\pi(\mathcal{P})w(u, v)}{\pi(\mathcal{N}(u, v))} \tag{3}$$

Given weights w which are a rough approximation to w^* , identity (3) implies an easy method to recalibrate weights w to an arbitrarily close approximation to w^* . We generate many samples from the stationary distribution, and observe the number of perfect matchings in our samples versus the number of near-perfect matchings with holes u, v . By generating sufficiently many samples, we can estimate $\pi(\mathcal{P})/\pi(\mathcal{N}(u, v))$ within an arbitrarily close factor, and hence we can estimate $w^*(u, v)$ (via (3)) within an arbitrarily close factor.

From the argument of [7, pages 11-12], it follows that $O(n^2 \log(1/\hat{\eta}))$ samples of the Markov chain MC (with $\delta = O(1/n^2)$) are enough to obtain a $\sqrt{2}$ -approximation w' of w^* with probability $\geq 1 - \hat{\eta}$. Starting the Markov chain anew for each sample would require $O(n^5 \log n)$ steps of the chain per sample. This can be improved using warm starts resulting in amortized $O(n^4 \log n)$ steps of the chain per sample, see [7, Section 6].

2.3 Algorithm for estimating ideal weights

In this section we present an $\tilde{O}(n^7)$ algorithm for estimating the ideal weights w^* . The algorithm will be used in Section 7 to approximate the permanent of a 0-1 matrix. The algorithm can be generalized to compute the permanent of general non-negative matrices, the necessary modifications are described in Section 8.

The algorithm runs in phases, each characterized by a parameter $\hat{\lambda}$. In every phase,

$$\lambda(e) = \begin{cases} 1 & \text{for } e \in E \\ \hat{\lambda} & \text{for } e \notin E \end{cases} \quad (4)$$

We start with $\hat{\lambda} = 1$ and slowly decrease $\hat{\lambda}$ until it reaches its target value $1/n!$.

In [7], $O(n^2 \log n)$ phases are required. A straightforward way to achieve this is to decrease $\hat{\lambda}$ by a factor $(1 - 1/3n)$ between phases. This ensures that the weight of any matching changes by at most a factor $(1 - 1/3n)^n \leq \exp(1/3) < \sqrt{2}$.

We use only $O(n \log^2 n)$ phases by progressively decreasing $\hat{\lambda}$ by a larger amount per phase. Initially we decrease $\hat{\lambda}$ by a factor of roughly $(1 - 1/3n)$ per phase, but during the final phases we decrease $\hat{\lambda}$ by a constant factor per phase.

At the start of each phase we have a set of weights satisfying (2), for all u, v , with high probability. Applying Theorem 2 we generate many samples from the stationary distribution. Using these samples and (3), we refine the weights so that the following holds, for all u, v , with high probability,

$$\frac{w^*(u, v)}{\sqrt{2}} \leq w(u, v) \leq \sqrt{2}w^*(u, v) \quad (5)$$

This allows us to decrease $\hat{\lambda}$ while keeping the condition (2) satisfied.

Here is the pseudocode of our algorithm. The algorithm outputs w which is a 2-approximation of the ideal weights w^* with probability $\geq 1 - \eta$. The quantities S and T satisfy $S = O(n^2(\log n + \log \eta^{-1}))$ and $T = O(n^4 \log n)$.

Algorithm for approximating ideal weights of 0-1 matrices:

Initialize $\hat{\lambda} = 1$ and $i = n - 2$.

Initialize $w(u, v) \leftarrow n$ for all $(u, v) \in V_1 \times V_2$.

While $\hat{\lambda} > 1/n!$ do:

Take S samples from MC with parameters λ, w , using a warm start simulation

(i. e., initial matchings for the simulation are the final matchings from the previous simulation). We use T steps of the MC per sample, except for the first sample which needs $O(Tn \log n)$ steps.

Use the samples to obtain estimates $w'(u, v)$ satisfying condition (5), for all u, v . The algorithm fails

(i. e., (5) is not satisfied) with small probability.

If $\hat{\lambda} > n^{-n/i}$, set $\hat{\lambda} = \hat{\lambda} 2^{-1/2(i+2)}$.

Else set $\hat{\lambda} = n^{-n/i}$ and decrement i by 1.

If $\hat{\lambda} < 1/n!$, set $\hat{\lambda} = 1/n!$.

Set $w(u, v) = w'(u, v)$ for all $u \in V_1, v \in V_2$.

Output the final weights $w(u, v)$.

Note the number of phases is $O(n \log^2 n)$. To see this, consider the time interval in the algorithm when i is fixed and let q_i be the total number of decrements of $\hat{\lambda}$ during this time interval. For $i \in \{1, \dots, n-3\}$ we have the following bound on q_i :

$$q_i \leq 2 \log_2 \left(\frac{n^{-n(i+2)/(i+1)}}{n^{-n(i+2)/i}} \right) \leq \frac{4}{i} n \log_2 n.$$

Since $(n^{-n/(n-2)})^n \geq n^{-2n}$ for $n \geq 4$, we can bound $q_n \leq 4n \log_2 n$.

Putting it all together,

$$\sum_{i=1}^{n-2} q_i \leq \sum_{i=1}^{n-3} \frac{4}{i} n \log_2 n + 4n \log_2 n = O(n \log^2 n).$$

Therefore, the algorithm consists of $O(n \log^2 n)$ phases. The total running time is $O(STn \log^2 n) = O(n^7 \log^4 n)$. In Section 6 we prove that our weights at the start of each phase satisfy (2) assuming that the estimates w' satisfied condition (5) throughout the execution of the algorithm. In Section 7 we show how to use the (constant factor) estimates of the ideal weights to obtain a $(1 \pm \epsilon)$ -approximation of the permanent.

3 Canonical Paths for Proving Theorem 2

We bound the mixing time by the canonical paths method. For $(I, F) \in \Omega \times \mathcal{P}$, we will define a *canonical path* from I to F , denoted, $\gamma(I, F)$, which is of length $\leq n$. The path is along transitions of the Markov chain. We then bound the weighted sum of canonical paths (or “flow”) through any transition. More precisely, for a transition $T = M \rightarrow M'$, let

$$\rho(T) = \sum_{\substack{(I, F) \in \Omega \times \mathcal{P}: \\ T \in \gamma(I, F)}} \frac{\pi(I)\pi(F)}{\pi(M)P(M, M')},$$

denote the *congestion* through the transition T , where $P(M, M')$ denotes the probability of the transition T . Let

$$\rho = \max_T \rho(T).$$

Then (see [14, 2]) for any initial state M_0 , the mixing time is bounded as

$$\tau_{M_0}(\delta) \leq \frac{7n\rho}{\pi(\mathcal{P})} (\ln \pi(M_0)^{-1} + \ln \delta^{-1})$$

The factor $1/\pi(\mathcal{P})$ comes from restricting to $F \in \mathcal{P}$, see Lemma 9 in [7]. When the weights w satisfy (2), we have $\pi(\mathcal{P}) = \Omega(1/n^2)$. Thus, to prove Theorem 2 we need to prove $\rho(T) = O(n)$ for every transition T .

We define the canonical paths now, and defer the bound on the congestion to Section 5, after presenting some combinatorial lemmas in Section 4. We will assume that the vertices of G are numbered. If $I \in \mathcal{P}$, then $I \oplus F$ consists of even length cycles. Let us assume that the cycles are numbered according to the smallest numbered vertex contained in them. The path $\gamma(I, F)$ “corrects” these cycles in order. Let v_0, v_1, \dots, v_{2k} be a cycle C , where v_0 is the smallest numbered vertex in C and $(v_0, v_1) \in I$. The path starts by unmatching edge (v_0, v_1) and successively interchanging edge (v_{2i-1}, v_{2i}) for edge (v_{2i}, v_{2i+1}) . Finally it adds edge (v_{2k-1}, v_{2k}) to the matching.

If $I \in \mathcal{N}(w, z)$, then there is an augmenting path from w to z in $I \oplus F$. The canonical path starts by augmenting I along this path by first exchanging edges and finally adding the last edge. It then “corrects” the even cycles in order.

For a transition $T = M \rightarrow M'$, we need to bound the number of canonical paths passing thru T . We partition these paths into $2n^2 + 1$ sets,

$$cp_T = \{(I, F) \in \mathcal{P} \times \mathcal{P} : \gamma(I, F) \ni T\},$$

And, for all w, z ,

$$cp_T^{w,z} = \{(I, F) \in \mathcal{N}(w, z) \times \mathcal{P} : \gamma(I, F) \ni T\}.$$

4 Key Technical Lemmas

The following Lemmas will be used to analyze the congestion through a transition. Much weaker versions of these Lemmas were used in the earlier work of [7]. In particular, Lemma 4 below improves on Lemma 7 in [7] by constructing more efficient mappings, and thereby helps put a tighter upper bound on the congestion through a transition of the Markov chain. We first present our mappings in the simpler setting of Lemma 3 and later use them to prove Lemma 4.

The related lemmas in [7] did not contain the sum in the left-hand side, and were a factor of 2 smaller in the right-hand side. Our improvement comes from looking at the appropriate sum, and only losing a factor of 2.

Lemma 3. *Let $u, w \in V_1, v, z \in V_2$ be distinct vertices. Then,*

1.

$$\sum_{x,y:(u,y),(x,v) \in E} |\mathcal{N}(u, v)| |\mathcal{N}(x, y)| \leq 2|\mathcal{P}|^2.$$

2.

$$\sum_{x:(x,v) \in E} |\mathcal{N}(u, v)| |\mathcal{N}(x, z)| \leq 2|\mathcal{N}(u, z)| |\mathcal{P}|.$$

3.

$$\sum_{x,y:(u,y),(v,x) \in E} |\mathcal{N}(u,v)| |\mathcal{N}(x,y,w,z)| \leq 2|\mathcal{N}(w,z)| |\mathcal{P}|.$$

The basic intuition for the proofs of these inequalities is straightforward. For example consider the first inequality. Take matchings $M \in \mathcal{N}(u,v)$, $M' \in \mathcal{N}(x,y)$. The set $M \cup M' \cup (u,y) \cup (v,x)$ consists of a set of alternating cycles. Hence, this set can be broken into a pair of perfect matchings. One of the perfect matchings contains the edge (u,y) and one matching contains the edge (v,x) . Hence, given the pair of perfect matchings, we can deduce the original unmatched vertices (by guessing which of the two edges incident to u), and thereby reconstruct M and M' . This outlines the approach for proving Lemma 3.

Proof. 1. We will construct a one-to-one map:

$$f_1 : \mathcal{N}(u,v) \times \bigcup_{x,y:(u,y),(v,x) \in E} \mathcal{N}(x,y) \rightarrow \mathcal{P} \times \mathcal{P} \times b,$$

where b is a bit, i.e., b is 0/1.

Let $L_0 \in \mathcal{N}(u,v)$ and $L_1 \in \bigcup_{x,y:(u,y),(v,x) \in E} \mathcal{N}(x,y)$. In $L_0 \oplus L_1$ the four vertices u, v, x, y each have degree one, and the remaining vertices have degree zero or two. Hence these four vertices are connected by two disjoint paths. Since $|L_0| = |L_1|$, the two paths must be of the same parity. The edges (u,y) and (v,x) are in neither matching, and so $(L_0 \oplus L_1) \cup \{(u,y), (v,x)\}$ contains an even cycle, say C , containing (u,y) and (v,x) . We will partition the edges of $L_0 \cup L_1 \cup \{(u,y), (v,x)\}$ into two perfect matchings as follows. Let M_0 contain the edges of L_0 outside of C and alternate edges of C starting with edge (u,y) . M_1 will contain the remaining edges. Bit b is set to 0 if $(x,v) \in M_0$ and to 1 otherwise. This defines the map f_1 .

Next, we show that f_1 is one-to-one. Let M_0 and M_1 be two perfect matchings and b be a bit. If u and v are not in one cycle in $M_0 \oplus M_1$ then (M_0, M_1, b) is not mapped onto by f_1 . Otherwise, let C be the common cycle containing u and v . Let y be the vertex matched to u in M_0 . If $b = 0$, denote by x the vertex that is matched to v in M_0 ; else denote by x the vertex that is matched to v in M_1 . Let L_0 contain the edges of M_0 outside C and let it contain the near-perfect matching in C that leaves u and v unmatched. Let L_1 contain the edges of M_1 outside C and let it contain the near-perfect matching in C that leaves x and y unmatched. It is easy to see that $f_1(L_0, L_1) = (M_0, M_1, b)$.

2. We will construct a one-to-one map:

$$f_2 : \mathcal{N}(u,v) \times \bigcup_{x:(v,x) \in E} \mathcal{N}(x,y) \rightarrow \mathcal{N}(u,y) \times \mathcal{P} \times b.$$

Let $L_0 \in \mathcal{N}(u,v)$ and $L_1 \in \bigcup_{x:(v,x) \in E} \mathcal{N}(x,y)$. As before, u, v, x, y are connected by two disjoint paths of the same parity in $L_0 \oplus L_1$ and $(v,x) \notin L_0 \cup L_1$. Hence, $L_0 \cup L_1 \cup \{(v,x)\}$ contains an odd path from u to y , say P . Construct $M_0 \in \mathcal{N}(u,y)$ by including all edges of L_0 not on P and alternate edges of P , leaving u, y unmatched. Let $M_1 \in \mathcal{P}$ consist of the remaining edges of $L_0 \cup L_1 \cup \{(v,x)\}$. Let $b = 0$ if $(v,x) \in M_0$, and to 1 otherwise. Clearly, path P appears in $M_0 \oplus M_1$, and as before, L_0 and L_1 can be retrieved from (M_0, M_1, b) .

3. We will construct a one-to-one map:

$$f_3 : \mathcal{N}(u, v) \times \bigcup_{x, y: (u, y), (v, x) \in E} \mathcal{N}(x, y, w, z) \rightarrow \mathcal{N}(w, z) \times \mathcal{P} \times b.$$

Let $L_0 \in \mathcal{N}(u, v)$ and $L_1 \in \bigcup_{x, y: (u, y), (v, x) \in E} \mathcal{N}(x, y, w, z)$. Consider $L_0 \oplus L_1$. There are two cases. If there are two paths connecting the four vertices u, v, x, y (and a separate path connecting w and z), then the mapping follows using the construction given in 1. The second case is that $L_0 \oplus L_1$ contains three disjoint paths: u to w , v to y , and x to z . The first two are even paths and the third is odd. Now, $L_0 \cup L_1 \cup \{(u, y), (v, x)\}$ contains an odd path, say P , from w to z . Now, the mapping follows using the construction given in 2. □

The following lemma is an extension of the previous lemma, which serves as a warm-up. This lemma is used to bound the congestion.

Lemma 4. *Let $u, w \in V_1$, $v, z \in V_2$ be distinct vertices. Then,*

1.

$$\sum_{x \in V_1, y \in V_2} \lambda(u, y) \lambda(x, v) \lambda(\mathcal{N}(u, v)) \lambda(\mathcal{N}(x, y)) \leq 2\lambda(\mathcal{P})^2.$$

2.

$$\sum_{x \in V_1} \lambda(x, v) \lambda(\mathcal{N}(u, v)) \lambda(\mathcal{N}(x, z)) \leq 2\lambda(\mathcal{N}(u, z)) \lambda(\mathcal{P}).$$

3.

$$\sum_{x \in V_1, y \in V_2} \lambda(u, y) \lambda(x, v) \lambda(\mathcal{N}(u, v)) \lambda(\mathcal{N}(x, y, w, z)) \leq 2\lambda(\mathcal{N}(w, z)) \lambda(\mathcal{P}).$$

Proof. We will use the mappings f_1, f_2, f_3 constructed in Lemma 3. Observe that since mapping f_1 constructs matchings M_0 and M_1 using precisely the edges of L_0, L_1 and the edges $(u, y), (v, x)$, it satisfies

$$\lambda(u, y) \lambda(x, v) \lambda(L_0) \lambda(L_1) = \lambda(M_0) \lambda(M_1).$$

Summing over all pairs of matchings in

$$\mathcal{N}(u, v) \times \bigcup_{x, y: (u, y), (v, x) \in E} \mathcal{N}(x, y)$$

we get the first inequality. The other two inequalities follow in a similar way using mappings f_2 and f_3 . □

5 Bounding Congestion: Proof of Theorem 2

We bound the congestion separately for transitions which move between near-perfect matchings (Cases 2b and 2c), and transitions which move between a perfect and near-perfect matching. Our goal for this section will be to prove for every transition $T = M \rightarrow M'$,

$$\sum_{\substack{(I, F) \in \Omega \times \mathcal{P}: \\ T \in \gamma(I, F)}} w^*(I) w^*(F) w^*(M) = O(w^*(\Omega)). \quad (6)$$

At the end of the section we will prove that this easily implies the desired bound on the congestion.

The following lemma converts into a more manageable form, the weighted sum of I, F pairs which contain a transition of the first type.

Lemma 5. *Let $T = M \rightarrow M'$ be a transition which moves between near-perfect matchings (i.e., Case 2b or 2c). Let $M \in \mathcal{N}(u, v), M' \in \mathcal{N}(u, v')$, $u \in V_1, v, v' \in V_2$, and $M' = M \setminus (v', x) \cup (v, x)$ for some $x \in V_1$. Then, the following hold:*

1.

$$\sum_{(I,F) \in cp_T} \lambda(I)\lambda(F) \leq \sum_{y \in V_2} \lambda(\mathcal{N}(x, y))\lambda(u, y)\lambda(x, v)\lambda(M)$$

2. For all $z \in V_2$,

$$\sum_{(I,F) \in cp_T^{u,z}} \lambda(I)\lambda(F) \leq \lambda(\mathcal{N}(x, z))\lambda(v, x)\lambda(M)$$

3. For all $w \in V_1, w \neq u$ and $z \in V_2, z \neq v, v'$,

$$\sum_{(I,F) \in cp_T^{w,z}} \lambda(I)\lambda(F) \leq \sum_{y \in V_2} \lambda(\mathcal{N}(w, z, x, y))\lambda(u, y)\lambda(v, x)\lambda(M)$$

Proof. 1. We will first construct a one-to-one map

$$\eta_T : cp_T \rightarrow \bigcup_{x,y:(u,y),(v,x) \in E} \mathcal{N}(x, y).$$

Let $I, F \in \mathcal{P}$ and $(I, F) \in cp_T$. Let S be the set of cycles in $I \oplus F$. Order the cycles in S using the convention given in Section 3. Clearly, u, v, x lie on a common cycle, say $C \in S$, in $I \oplus F$. Since T lies on the canonical path from I to F , M has already corrected cycles before C and not yet corrected cycles after C in S . Let y be a neighbor of u on C . Define $M'' \in \mathcal{N}(x, y)$ to be the near-perfect matching that picks edges as follows: outside C , it picks edges $(I \cup F) - M$, and on C it picks the near perfect-matching leaving x, y unmatched. Define $\eta_T(I, F) = M''$.

Clearly, $(M \oplus M'') \cup \{(u, v), (x, y)\}$ consists of the cycles in S , and I and F can be retrieved from M, M'' by considering the order defined on S . This proves that the map constructed is one-to-one. Since the union of edges in I and F equals the edges in $M \cup M'' \cup \{(u, v), (x, y)\}$,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(u, y)\lambda(x, v).$$

Summing over all $(I, F) \in cp_T$ we get the first inequality.

2. For all $z \in V_2$, we will first construct a one-to-one map

$$\eta_T^{u,z} : cp_T^{u,z} \rightarrow \mathcal{N}(x, z).$$

Let $I \in \mathcal{N}(u, z), F \in \mathcal{P}$ and $(I, F) \in cp_T^{u,z}$. Let S be the set of cycles and P be the augmenting path from u to z in $I \oplus F$. Clearly, v, x lie on P . M has “corrected” part of the path P and none of the cycles in S . It contains the edges of I from z to v and the edges of F from x to u . Also, it contains the edges of I from the cycles in S , as well as the edges in $I \cap F$.

Construct matching $M'' \in \mathcal{N}(x, z)$ as follows. It contains the edges of F from the cycles in S , the edges $I \cap F$ and $(P - \{(x, v)\} - M)$. Define $\eta_T^{u,z}(I, F) = M''$. It is easy to see that $M \cup M'' = I \cup F \cup \{(x, v)\}$. Therefore,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(x, v).$$

Furthermore, I, F can be retrieved from M, M'' . Hence, summing over all $(I, F) \in cp_T^{u,z}$ we get the second inequality.

3. For all $w \in V_1, w \neq u$ and $z \in V_2, z \neq v, v'$, we will first construct a one-to-one map

$$\eta_T^{w,z} : cp_T^{w,z} \rightarrow \bigcup_{y:u \sim y} \mathcal{N}(w, z, x, y).$$

Let $I \in \mathcal{N}(w, z), F \in \mathcal{P}$ and $(I, F) \in cp_T^{w,z}$. Let S be the set of cycles and P be the augmenting path from w to z in $I \oplus F$. Clearly, u, v, x lie on a common cycle, say $C \in S$, in $I \oplus F$. and M has already “corrected” P and so looks like F on P . Construct $M'' \in \mathcal{N}(w, z, x, y)$ as follows. On P , it looks like I . Outside $P \cup C$, it picks edges $(I \cup F) - M$, and on C it picks the near perfect-matching leaving x, y unmatched. Define $\eta_T^{w,z}(I, F) = M''$. It is easy to see that $M \cup M'' = I \cup F \cup \{(u, y), (x, v)\}$. Therefore,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(u, y)\lambda(x, v).$$

Furthermore, I, F can be retrieved from M, M'' . Hence, summing over all $(I, F) \in cp_T^{w,z}$ we get the third inequality. □

We now prove (6) for the first type of transitions. The proof applies Lemma 5 and then Lemma 4. We break the statement of (6) into two cases depending on whether I is a perfect matching or a near-perfect matching.

Lemma 6. *For a transition $T = M \rightarrow M'$ which moves between near-perfect matchings (i.e., Case 2b or 2c), the congestion from $(I, F) \in \mathcal{P} \times \mathcal{P}$ is bounded as*

$$\sum_{(I, F) \in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \leq \frac{2w^*(\Omega)}{n^2} \quad (7)$$

And, the congestion from $(I, F) \in \mathcal{N}_2 \times \mathcal{P}$ is bounded as

$$\sum_{w \in V_1, z \in V_2} \sum_{(I, F) \in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \leq 3w^*(\Omega) \quad (8)$$

Proof. The transition T is sliding an edge, let x denote the pivot vertex, let $M \in \mathcal{N}(u, v), M' \in \mathcal{N}(u, v'), u \in V_1, v, v' \in V_2$. Thus, $M' = M \setminus (v', x) \cup (v, x)$ for some $x \in V_1$. The encodings from Lemma 5 will always contain x as a hole.

We begin with the proof of (7).

$$\begin{aligned}
& \sum_{(I,F) \in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \\
&= \sum_{(I,F) \in cp_T} \lambda(I)\lambda(F) \frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{P})} \\
&\leq \sum_{y \in V_2} \frac{\lambda(\mathcal{N}(x,y))\lambda(u,y)\lambda(x,v)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{P})} \quad \text{by Lemma 5} \\
&\leq 2\lambda(\mathcal{P}) \quad \text{by Lemma 4} \\
&= \frac{2w^*(\Omega)}{n^2 + 1}
\end{aligned}$$

This completes the proof of (7). We now prove (8) in two parts. This first bound covers the congestion due to the first part of the canonical paths from a near-perfect matching to a perfect matching – unwinding the augmenting path. The second bound covers the second part of these canonical paths when we unwind the alternating cycle(s). During the unwinding of the augmenting path, one of the holes of the transition is the same as one of the holes of the initial near-perfect matching. This is what characterizes the first versus the second part of the canonical path.

$$\begin{aligned}
& \sum_{z \in V_2} \sum_{(I,F) \in cp_T^{u,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \\
&= \sum_{z \in V_2} \sum_{(I,F) \in cp_T^{u,z}} \lambda(I)\lambda(F) \frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{N}(u,z))} \\
&\leq \sum_{z \in V_2} \frac{\lambda(\mathcal{N}(x,z))\lambda(v,x)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{N}(u,z))} \quad \text{by Lemma 5} \\
&\leq \sum_{z \in V_2} 2\lambda(\mathcal{P}) \quad \text{by Lemma 4} \\
&= \frac{2n}{n^2 + 1} w^*(\Omega) \\
&\leq w^*(\Omega)
\end{aligned}$$

Finally, bounding the congestion from the unwinding of the alternating cycle(s) on canonical

paths from near-perfect matchings to perfect matchings,

$$\begin{aligned}
& \sum_{\substack{w \in V_1, z \in V_2: \\ w \neq u}} \sum_{(I,F) \in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \\
&= \sum_{\substack{w \in V_1, z \in V_2: \\ w \neq u}} \sum_{(I,F) \in cp_T^{w,z}} \lambda(I)\lambda(F) \frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{N}(w,z))} \\
&\leq \sum_{\substack{w \in V_1, z \in V_2: \\ w \neq u}} \sum_{y \in V_2} \frac{\lambda(\mathcal{N}(w,z,x,y))\lambda(u,y)\lambda(v,x)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{N}(w,z))} \quad \text{by Lemma 5} \\
&\leq \sum_{\substack{w \in V_1, z \in V_2: \\ w \neq u}} 2\lambda(\mathcal{P}) \quad \text{by Lemma 4} \\
&\leq 2w^*(\Omega)
\end{aligned}$$

□

We now follow the same approach as Lemmas 5 and 6 to prove (6) for transitions moving between a perfect and near-perfect matching. The proofs in this case are easier.

Lemma 7. *For a transition $T = M \rightarrow M'$ which adds or subtracts an edge (i.e., Case 1 or 2a), let N denote the near-perfect matching of M and M' . Then,*

$$\sum_{(I,F) \in cp_T} \lambda(I)\lambda(F) \leq \lambda(\mathcal{P})\lambda(u,v)\lambda(N).$$

And, for all $w \in V_1, z \in V_2$,

$$\sum_{(I,F) \in cp_T^{w,z}} \lambda(I)\lambda(F) \leq \lambda(\mathcal{N}(w,z))\lambda(u,v)\lambda(N).$$

Proof. Let P denote the perfect matching of M and M' . Define $\eta = \eta_T^{w,z} : cp_T^{w,z} \rightarrow \mathcal{N}(w,z)$ as

$$\eta(I,F) = I \cup F \setminus P.$$

The mapping satisfies $\lambda(I)\lambda(F) = \lambda(P)\lambda(\eta(I,F))$. Note, $\lambda(P) = \lambda(N)\lambda(u,v)$. Since the mapping is one-to-one, summing over all $N' \in \mathcal{N}(w,z)$ proves the lemma for all w,z . The proof is identical for cp_T with the observation that when $I \in \mathcal{P}$, we have $I \cup F \setminus P$ is in \mathcal{P} . □

Lemma 8. *For a transition $T = M \rightarrow M'$ which adds or subtracts an edge (i.e., Case 1 or 2a), the congestion from $(I,F) \in \Omega \times \mathcal{P}$ is bounded as*

$$\sum_{w,z} \sum_{(I,F) \in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \leq w^*(\Omega) \quad (9)$$

$$\sum_{(I,F) \in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \leq \frac{w^*(\Omega)}{n^2} \quad (10)$$

Proof. Let $M \in \mathcal{N}(u, v)$ and $M' \in \mathcal{P}$, thus the transition adds the edge (u, v) . The proof for the transition which subtracts the edge will be analogous. The proof is a simplified version of Lemma 6, since the encoding is simpler in this case (see Lemma 7 versus Lemma 5).

Observe that for any x, y ,

$$\lambda(x, y)\lambda(\mathcal{N}(x, y)) \leq \lambda(\mathcal{P}) \quad (11)$$

We begin with the proof of (9).

$$\begin{aligned} \sum_{w,z} \sum_{(I,F) \in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} &= \sum_{w,z} \sum_{(I,F) \in cp_T^{w,z}} \lambda(I)\lambda(F) \frac{\lambda(\mathcal{N}(u, v))}{\lambda(M)\lambda(\mathcal{N}(w, z))} \\ &\leq \sum_{w,z} \lambda(u, v)\lambda(\mathcal{N}(u, v)) \quad \text{by Lemma 7} \\ &\leq w^*(\Omega) \quad \text{by (11)} \end{aligned}$$

We now prove (10).

$$\begin{aligned} \sum_{(I,F) \in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} &= \sum_{(I,F) \in cp_T} \lambda(I)\lambda(F) \frac{\lambda(\mathcal{N}(u, v))}{\lambda(M)\lambda(\mathcal{P})} \\ &\leq 2\lambda(u, v)\lambda(\mathcal{N}(u, v)) \quad \text{by Lemma 7} \\ &\leq \lambda(\mathcal{P}) \quad \text{by (11)} \end{aligned}$$

□

Proof of Theorem 2. Inequality (2) implies for any set of matchings $S \subset \Omega$, the stationary distribution $\pi(S)$ under w is within a factor 4 of the distribution under w^* . Therefore, to prove Theorem 2 it suffices to consider the stationary distribution with respect to w^* . In other words, we need to prove, for every transition T , $\rho(T) = O(n)$ where, for $M \in \Omega$, $\pi(M) = w^*(M)/w^*(\Omega)$. Then for weights satisfying (2) the congestion increases by at most a constant factor. Thus, we need to prove

$$\sum_{\substack{(I,F) \in \Omega \times \mathcal{P}: \\ T \in \gamma(I,F)}} \frac{w^*(I)w^*(F)}{w^*(M)P(M, M')} = O(nw^*(\Omega)).$$

Recall that the transitions $M_t \rightarrow M_{t+1}$ of our Markov chain are according to the Metropolis filter. From M_t , a new matching N is proposed with probability $1/4n$, and then the proposed new matching is accepted with probability $\min\{1, w^*(N)/w^*(M_t)\}$. Hence, for the transition $T = M \rightarrow M'$,

$$w^*(M)P(M, M') = \frac{1}{4n} \min\{w^*(M), w^*(M')\}.$$

Since the chain is reversible for every transition $T = M \rightarrow M'$, there is a reverse transition $T' = M' \rightarrow M$. To prove Theorem 2, it suffices to prove that for every transition $T = M \rightarrow M'$,

$$\sum_{\substack{(I,F) \in \Omega \times \mathcal{P}: \\ T \in \gamma(I,F)}} \frac{w^*(I)w^*(F)}{w^*(M)} = O(w^*(\Omega)). \quad (12)$$

Lemmas 6 and 8 imply (12) which completes the proof of the Theorem. □

6 Number of Phases

In this section we prove that the choice of $\hat{\lambda}$ from the weight-estimating algorithm ensures that (2) is satisfied in each phase. Intuitively, we need to decrease $\hat{\lambda}$ from 1 to $1/n!$. Recall that we can obtain a refined estimate of the ideal weights in each phase, see (5). We need to guarantee that the weights of two consecutive phases do not differ too much. Namely, if they are within a $\sqrt{2}$ factor of each other, together with (5) we have (2) for the next phase. As we will see shortly, for our choice of activities the ideal weights $w^*(u, v)$ are a ratio of two polynomials of degree $\leq n$ evaluated at $\hat{\lambda}$. The properties of the polynomials will allow us to partition the $[1/n!, 1]$ interval into several subintervals such that in the i -th subinterval only the first $i + 2$ terms of the polynomials are relevant. This, in turn, allows us to decrease $\hat{\lambda}$ by a higher factor in intervals with smaller number of relevant terms.

Definition 9. We say that a matching $M \in \mathcal{P}$ of a complete bipartite graph covers k edges of a graph G if the size of $M \cap E(G)$ is k . Let

$$R_G(x) = \sum_{k=0}^n p_k x^{n-k},$$

where p_k is the number of matchings in \mathcal{P} covering k edges of G .

Note that the ideal weights w^* , defined by (1), for activities given by (4) can be expressed as follows

$$w_{\hat{\lambda}}^*(u, v) = \frac{R_G(\hat{\lambda})}{R_{G \setminus \{u, v\}}(\hat{\lambda})}. \quad (13)$$

Lemma 10. Let $\hat{\lambda}_1 > \hat{\lambda}_2 > \dots > \hat{\lambda}_q$ where $\hat{\lambda}_1 = 1$, $\hat{\lambda}_q = 1/n!$ and $q = O(n \log^2 n)$ be the sequence of $\hat{\lambda}$ used by the weight-estimating algorithm. Assume that G contains a perfect matching. Then

$$\begin{aligned} R_G(\hat{\lambda}_k) &\geq R_G(\hat{\lambda}_{k+1}) \geq R_G(\hat{\lambda}_k)/\sqrt{2}, & \text{and} \\ R_{G \setminus \{u, v\}}(\hat{\lambda}_k) &\geq R_{G \setminus \{u, v\}}(\hat{\lambda}_{k+1}) \geq R_{G \setminus \{u, v\}}(\hat{\lambda}_k)/\sqrt{2} \end{aligned} \quad \text{for every } u, v. \quad (14)$$

We will prove Lemma 10 at the end of this section.

To shorten the notation, we will use w_k to denote $w_{\hat{\lambda}_k}$. Equation (13) with Lemma 10 imply the w_k^* and w_{k+1}^* are within a constant factor. Moreover if the weight-estimating algorithm does not fail, i. e., the w_k satisfy (5) then w_k and w_{k+1} are within a constant factor as well. The following corollaries are used in Section 7 for approximating the permanent once a good approximation of the ideal weights is obtained.

Corollary 11. For every u, v ,

$$\frac{1}{\sqrt{2}} w_{k+1}^*(u, v) \leq w_k^*(u, v) \leq \sqrt{2} w_{k+1}^*(u, v). \quad (15)$$

If the w_k satisfy (5) then for every u, v ,

$$\frac{1}{2\sqrt{2}} w_{k+1}(u, v) \leq w_k(u, v) \leq 2\sqrt{2} w_{k+1}(u, v). \quad (16)$$

Note that

$$w_k(\Omega) = R_G(\hat{\lambda}_k) + \sum_{u,v} R_{G \setminus \{u,v\}}(\hat{\lambda}_k) w_k(u,v).$$

Corollary 11 and Lemma 10 imply the following result.

Corollary 12. *If the weight-estimating algorithm does not fail then*

$$\frac{w_k(\Omega)}{2\sqrt{2}} \leq w_{k+1}(\Omega) \leq 2\sqrt{2}w_k(\Omega).$$

Let $M \in \Omega$ be a matching. Note that $\hat{\lambda}_{k+1} \leq \hat{\lambda}_k$ and hence $\lambda_{k+1}(M) \leq \lambda_k(M)$. For $M \in P$ we have $w_{k+1}(M) \leq w_k(M)$. If $M \in \mathcal{N}(u,v)$ then, assuming that the weight-estimating algorithm did not fail we have $w_{k+1}(M) = w_{k+1}(u,v)\lambda_{k+1}(M) \leq 2\sqrt{2}w_k(u,v)\lambda_k(M) = 2\sqrt{2}w_k(M)$. Hence we have the following observation.

Corollary 13. *Assume that the weight-estimating algorithm does not fail. Then for any matching $M \in \Omega$*

$$w_{k+1}(M) \leq 2\sqrt{2}w_k(M).$$

The rest of this section is devoted to proving Lemma 10.

The *log-derivative* of a function f is $(\log f)' = f'/f$. The log-derivative measures how quickly a function is increasing.

Definition 14. *We say that a polynomial f is dominant over a polynomial g on an interval I if $f'(x)/f(x) \geq g'(x)/g(x)$ for every $x \in I$.*

Lemma 15. *Let $f, g : I \rightarrow \mathbf{R}^+$ be two non-decreasing polynomials. If f dominates over g on I , then $f(y)/f(x) \geq g(y)/g(x)$ for every $x, y \in I$, $x \leq y$.*

We will use Lemma 15 to show that the condition (14) is satisfied for our choice of the $\hat{\lambda}_k$. Suppose that a function f dominates R_G and the $R_{G \setminus \{u,v\}}$ on interval I . Let λ_k and λ_{k+1} be in the interval I . Then $f(\lambda_{k+1}) \geq f(\lambda_k)/\sqrt{2}$ implies (14).

We partition the interval $(0, \infty)$ into subintervals I_3, \dots, I_n such that x^i dominates over every R -polynomial on the interval I_i . The $\hat{\lambda}_j$ in I_i will be such that x^i decreases by a factor $\sqrt{2}$ between consecutive $\hat{\lambda}$.

Lemma 16. *Let $g(x) = \sum_{j=0}^n a_j x^j$ be a polynomial with non-negative coefficients. Then x^n dominates g on the interval $(0, \infty)$.*

Proof. It is easy to verify that $f'_n(x)/f_n(x) \geq g'(x)/g(x)$ for every $x > 0$. □

Lemma 17. *Let $g(x) = \sum_{j=0}^n a_j x^j$ be a polynomial with non-negative integer coefficients such that $g(1) \leq n!$ and at least one of a_0, a_1, a_2 is non-zero. Then for any $i \geq 3$ the polynomial x^i dominates g on the interval $(0, n^{-n/(i-1)}]$.*

Proof. The logarithmic derivative of x^i is i/x . Hence we need to prove that $ig(x) \geq xg'(x)$ for $x \leq n^{-n/(i-1)}$.

Let d be the smallest integer such that $a_d \neq 0$. From the assumptions of the lemma $d \leq 2$. For $x \leq n^{-n/(i-1)}$ and $n \geq 2$ the following holds

$$\begin{aligned} \sum_{j=i+1}^n ja_j x^{j-d} &\leq \sum_{j=i+1}^n na_j x^{j-2} \leq \sum_{j=i+1}^n na_j n^{-n(j-2)/(i-1)} \leq \\ &\sum_{j=i+1}^n na_j n^{-n} \leq \frac{n!}{n^{n-1}} \leq 1. \end{aligned}$$

Since $i > d$, for $x \leq n^{-n/(i-1)}$ and $n \geq 2$ we have

$$xg'(x) = \sum_{j=0}^i ja_j x^j + \sum_{j=i+1}^n ja_j x^j \leq \sum_{j=d}^i ja_j x^j + a_d x^d \leq \sum_{j=d}^i ia_j x^j = ig(x).$$

□

Corollary 18. *Let G be a bipartite graph on $n + n$ vertices. For $i \geq 3$ function x^i dominates over the polynomials R_G and $R_{G \setminus \{u,v\}}$ on the interval $I'_i := (0, n^{-n/(i-1)})$. Furthermore, x^n dominates over the R on the interval $I'_n := (0, \infty)$.*

Proof. Since G contains a perfect matching, every R -polynomial has a positive low-degree coefficient. To see this, let M be a perfect matching in G . The existence of M implies that the absolute coefficient in R_G is positive. Similarly, if $(u, v) \in M$, then the absolute coefficient in $R_{G \setminus \{u,v\}}$ is positive because $M \setminus \{(u, v)\}$ is a perfect matching in $G \setminus \{u, v\}$. If $(u, v) \notin M$, let u' , resp. v' be the vertices matched to u and v in M , and let $M' = M \cup \{(v', u')\} \setminus \{(u, u'), (v, v')\}$. Depending on (v', u') being an edge in G , the cardinality of M' is either $n - 1$ or $n - 2$. Therefore either the coefficient of x^0 or x^1 in $R_{G \setminus \{u,v\}}$ is positive. Furthermore, $R_G(1)$ counts the number of all permutations of n elements, thus $R_G(1) = n!$ and $R_{G \setminus \{u,v\}}(1) = (n - 1)!$. Thus all the R -polynomials satisfy the conditions of Lemma 17. □

Proof of Lemma 10. Notice that the $\hat{\lambda}$ from the weight-estimating algorithm are such that if $\hat{\lambda}_k, \hat{\lambda}_{k+1} \in [n^{-n/(i-2)}, n^{-n/(i-1)}]$, then $\hat{\lambda}_{k+1}^i \sqrt{2} = \hat{\lambda}_k^i$. Therefore, by the previous corollary and Lemma 15 all of the R -polynomials decrease by a factor at most $\sqrt{2}$. Similarly, if $\hat{\lambda}_k, \hat{\lambda}_{k+1} \in [n^{n-2}, 1]$, then $\hat{\lambda}_{k+1}^n \sqrt{2} = \hat{\lambda}_k^n$. This proves (14). □

7 Reduction from Counting to Sampling

In this section we show how to approximate the permanent of a matrix once we have good approximations of the ideal weights. For simplicity we consider the case of 0 – 1 matrix. The argument follows the argument of Section 5 from [7].

We want to estimate the number of perfect matchings $|\mathcal{P}_G|$ in a bipartite graph G . Let $\hat{\lambda}_0 = 1 > \hat{\lambda}_1 > \dots > \hat{\lambda}_q = 1/n!$, $q = O(n \log^2 n)$ be the sequence of $\hat{\lambda}$ used in the weight-estimating algorithm. Assume that the algorithm did not fail, i. e., the hole weights w_0, \dots, w_q computed by the algorithm are within a factor of $\sqrt{2}$ from the ideal weights w_0^*, \dots, w_q^* . Recall that $w_0(\Omega) = n!(n^2 + 1)$. We can express $|\mathcal{P}_G|$ as a telescoping product:

$$|\mathcal{P}_G| = \frac{|\mathcal{P}_G|}{w_q(\Omega)} \frac{w_q(\Omega)}{w_{q-1}(\Omega)} \frac{w_{q-1}(\Omega)}{w_{q-2}(\Omega)} \dots \frac{w_1(\Omega)}{w_0(\Omega)} w_0(\Omega) = n!(n^2 + 1)\alpha^* \prod_{0 \leq k < q} \alpha_k, \quad (17)$$

where $\alpha^* = |\mathcal{P}_G|/w_q(\Omega)$ and $\alpha_k = w_{k+1}(\Omega)/w_k(\Omega)$. Note that corollary 12 implies $\alpha_k = \Theta(1)$. The quantity $w_q(\Omega)$ is within a constant factor of $(n^2 + 1)|\mathcal{P}_G|$ and hence $\alpha^* = \Theta(1/n^2)$.

Let $X_k \sim w_k$ denote a random matching chosen from the distribution defined by w_k , i. e., the probability of a matching M is $w_k(M)/w_k(\Omega)$. Let $Y_k = w_{k+1}(X_k)/w_k(X_k)$. Then Y_k is an unbiased estimator for α_k :

$$\mathbf{E}(Y_k) = \mathbf{E}_{X_k \sim w_k} \left(\frac{w_{k+1}(X_k)}{w_k(X_k)} \right) = \sum_{M \in \Omega} \frac{w_k(M)}{w_k(\Omega)} \frac{w_{k+1}(M)}{w_k(M)} = \frac{w_{k+1}(\Omega)}{w_k(\Omega)} = \alpha_k. \quad (18)$$

For $k = q$ let $Y_q = 1_{X_q \in \mathcal{P}_G}$, where $1_{M \in \mathcal{P}_G}$ is the indicator function which takes value 1 if M is a perfect matching of G , and 0 otherwise. Then Y_q is an unbiased estimator for α^* :

$$\mathbf{E}(Y_q) = \mathbf{E}_{X_q \sim w_q} (1_{X_q \in \mathcal{P}_G}) = \sum_{M \in \Omega} \frac{w_q(M)}{w_q(\Omega)} 1_{M \in \mathcal{P}_G} = \frac{|\mathcal{P}_G|}{w_q(\Omega)} = \alpha^*. \quad (19)$$

Corollary 13 implies that $0 \leq Y_k \leq 2\sqrt{2}$ and hence $\mathbf{Var}(Y_k) = O(1)$. Thus the mean \bar{Y}_k of $\Theta(q\epsilon^{-2})$ samples of Y_k has $\mathbf{Var}(\bar{Y}_k) = O(\epsilon^2/q)$. Therefore

$$\frac{\mathbf{E}(\bar{Y}_k^2)}{\mathbf{E}(\bar{Y}_k)^2} = 1 + \frac{\mathbf{Var}(\bar{Y}_k)}{\mathbf{E}(\bar{Y}_k)^2} = 1 + O(\epsilon^2/q),$$

since $\mathbf{E}(\bar{Y}_k) = \mathbf{E}(Y_k) = \Theta(1)$.

Let $Z = \prod_{k=0}^{q-1} \bar{Y}_k$. We have

$$\frac{\mathbf{E}(Z^2)}{\mathbf{E}(Z)^2} = (1 + O(\epsilon^2/q))^q = 1 + O(\epsilon^2),$$

and hence $\mathbf{Var}(Z)/(\mathbf{E}(Z))^2 = O(\epsilon^2)$. Thus by the Chebychev inequality Z is within a factor $1 \pm \epsilon/6$ from $\mathbf{E}(Z) = \prod_{k=0}^{q-1} \alpha_k$ with probability $\geq 11/12$ for appropriately chosen constants within the O notation.

Even though we cannot sample from w_k exactly, it suffices to sample the X_k (and hence Y_k) from a distribution within variation distance $O(\epsilon/q)$ from w_k . The expectation of Z will be within factor $1 \pm \epsilon/6$ from $\prod_{k=0}^{q-1} \alpha_k$ and the above variance argument remains unchanged.

Similarly to estimate α^* it is enough to take the mean of $O(n^2\epsilon^{-2})$ values of Y_q with X_q from a distribution within variation distance $O(\epsilon)$ from w_q . The result is an estimate of α^* within a factor of $1 \pm \epsilon/3$ with probability at least $11/12$.

Therefore, $n!(n^2 + 1) \prod_{k=0}^q \bar{X}_k$ estimates $|\mathcal{P}_G|$ within a factor of $1 \pm \epsilon$ with probability $\geq 5/6$. The total running time is $O(n^6 \log^3 \epsilon^{-2})$. See [7] for details.

8 Non-negative Matrices

A slight modification of our algorithm can be used to compute the permanent of a matrix $A = (a_{i,j})_{n \times n}$ with non-negative entries $a_{i,j}$. Suppose $\text{per}(A) > 0$. (The question $\text{per}(A) = 0$ can be

decided in deterministic polynomial time by finding the maximum matching in the corresponding weighted bipartite graph. The permanent is positive if and only if there exists a matching of nonzero weight.) Let $a_{\max} = \max_{i,j} a_{i,j}$ and $a_{\min} = \min_{i,j:a_{i,j}>0} a_{i,j}$. Then $\text{per}(A) \geq a_{\min}^n$. Therefore, if $a_{i,j} = 0$ we can round it to $(a_{\min}/a_{\max})^n/n!$. This change guarantees a 2-approximation of the permanent of the original matrix. We can boost this to a $(1 + \epsilon)$ -approximation using techniques described in Section 7. For the following algorithm, recall that $S = O(n^2(\log n + \log \eta^{-1}))$ and $T = O(n^4 \log n)$.

Algorithm for approximating ideal weights of non-negative matrices:

Initialize $\hat{\lambda} := a_{\max}$ and $i := n - 2$.

Initialize $w(u, v) \leftarrow na_{\max}$ for all $(u, v) \in V_1 \times V_2$.

Let $b := \left(\frac{a_{\min}}{a_{\max}}\right)^n / n!$.

For every i, j such that $a_{i,j} = 0$, set $a_{i,j} := b$.

While $\hat{\lambda} > b$ do:

Take S samples from MC with parameters λ, w , using a warm start simulation (i. e., initial matchings for the simulation are the final matchings from the previous simulation). We use T steps of the MC per sample, except for the first sample which needs $O(Tn \log n)$ steps.

Use the samples to obtain estimates $w'(u, v)$ satisfying condition (5), for all u, v , with high probability.

If $i = 1$ or $\hat{\lambda} > n^{-n/i}$, set $\hat{\lambda} := \hat{\lambda} 2^{-1/2(i+2)}$.

Else set $\hat{\lambda} := n^{-n/i}$ and decrement i by 1.

If $\hat{\lambda} < b$, set $\hat{\lambda} := b$.

Set $w(u, v) := w'(u, v)$ for all $u \in V_1, v \in V_2$.

Output the final weights $w(u, v)$.

Notice that now the number of different values of $\hat{\lambda}$ incurred by the algorithm is $O(n \log a_{\max})$ to get from $\hat{\lambda} = a_{\max}$ to $\hat{\lambda} \simeq 1$, plus $O(n \log^2 n)$ steps as before, plus $O(n \log(a_{\max}/a_{\min}))$ decrements from $\hat{\lambda} \simeq 1/n!$ until $\hat{\lambda} = b$. Therefore the running time of the algorithm is $O(ST(n \log^2 n + n \log \frac{a_{\max}}{a_{\min}})) = O(n^7 \log^2 n (\log^2 n + \log \frac{a_{\max}}{a_{\min}}))$.

Notice that the correctness of the algorithm for approximating ideal weights of non-negative matrices follows from the fact that n^n dominates over the R -polynomials on the interval $(0, \infty)$ and n^3 dominates on the interval $(0, n^{-n/2})$.

As discussed in [7], this can be converted to a strongly polynomial time algorithm for approximating the permanent by first applying the algorithm of Linial, Samorodnitsky and Wigderson [11], which converts our input matrix into a nearly doubly stochastic matrix. See Section 7 of [7] for details.

9 Discussion

The problem of constructing an fpras for approximating the number of perfect matchings in non-bipartite graphs still remains open – the algorithm of Jerrum and Sinclair can only handle graphs in which the number of near-perfect matchings is polynomially bounded in the number of perfect matchings; this includes the case of dense graphs, i.e., graphs in which each vertex has degree $> n/2$. We present below a simple idea for obtaining an fpras for graphs containing at most $O(\log n)$ odd cycles.

Let G contain $k = O(\log n)$ odd cycles, c_1, \dots, c_k . Let e_0^i, e_1^i be two adjacent edges chosen from cycle c_i (an edge may be chosen several times). For each choice of $b_1, \dots, b_k \in \{0, 1\}^k$, remove edges $e_{b_1}^1, e_{b_2}^2, \dots, e_{b_k}^k$ to obtain a bipartite graph. Let G_1, \dots, G_{2^k} be the list of graphs obtained in this manner. Estimate the number of perfect matchings in these graphs using the algorithm of [7], and let T be the total. Pick a graph, say G_i , from this list with probability proportional to the number of perfect matchings in it, and find a random perfect matching, say M in G_i . Let m be the number of graphs in the list that contain the perfect matching M . Output M with probability $1/m$. It is easy to see that this procedure outputs a random perfect matching of G .

We are not aware of any other way of counting perfect matchings in graphs containing at most $O(\log n)$ odd cycles. This raises the following question: can the set of bipartizations of G play a role in counting the number of perfect matchings in non-bipartite graphs? For bipartite graphs, can our running time be further improved to obtain a practical algorithm?

References

- [1] Andrei Z. Broder, How hard is it to marry at random? (On the approximation of the permanent), *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, ACM Press, 1986, 50–58. Erratum in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, p. 551.
- [2] Persi Diaconis and Daniel Stroock, Geometric bounds for eigenvalues of Markov chains, *The Annals of Applied Probability* **1** (1991), 36–61.
- [3] P. Diaconis and B. Efron. Testing for independence in a two-way table: new interpretations of the chi-square statistic. *Ann. Statist.*, 13(3):845–913, 1985.
- [4] M.E. Dyer, A.M. Frieze and R. Kannan, A random polynomial time algorithm for approximating the volume of convex bodies, *Journal of the Association for Computing Machinery* 38(1):1–17, 1991.
- [5] Mark Jerrum and Alistair Sinclair, Approximating the permanent, *SIAM Journal on Computing* **18** (1989), 1149–1178.
- [6] M.R. Jerrum, A. Sinclair and E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 712–721, 2001.
- [7] M.R. Jerrum, A. Sinclair and E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. To appear in *Journal of the Association for Computing Machinery*.
- [8] Mark Jerrum, Leslie Valiant and Vijay Vazirani, Random generation of combinatorial structures from a uniform distribution, *Theoretical Computer Science* **43** (1986), 169–188.
- [9] Mark Jerrum and Umesh Vazirani, A mildly exponential approximation algorithm for the permanent, *Algorithmica* **16** (1996), 392–401.
- [10] P. W. Kasteleyn, The statistics of dimers on a lattice, I., The number of dimer arrangements on a quadratic lattice, *Physica* **27** (1961), 1664–1672.

- [11] Nathan Linial, Alex Samorodnitsky and Avi Wigderson, A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents, *Combinatorica* **20** (2000), 545–568.
- [12] L. Lovász and S. Vempala, Simulated Annealing in Convex Bodies and an $O^*(n^4)$ Volume Algorithm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 650-659, 2003.
- [13] Henryk Minc, *Permanents*, Encyclopedia of Mathematics and its Applications Vol. 6, Addison-Wesley, 1982.
- [14] Alistair Sinclair, Improved bounds for mixing rates of Markov chains and multicommodity flow, *Combinatorics, Probability and Computing* **1** (1992), 351–370.
- [15] L. G. Valiant, The complexity of computing the permanent, *Theoretical Computer Science* **8** (1979), 189–201.