Computing Permanents and Counting Hamiltonian Cycles by Listing Dissimilar Vectors

Andreas Björklund
Department of Computer Science, Lund University, Sweden

Ryan Williams
Department of Electrical Engineering and Computer Science & CSAIL, MIT, Cambridge, MA, USA

Abstract

We show that the permanent of an $n \times n$ matrix over any finite ring of $r \le n$ elements can be computed with a deterministic $2^{n-\Omega(\frac{n}{r})}$ time algorithm. This improves on a Las Vegas algorithm running in expected $2^{n-\Omega(n/(r\log r))}$ time, implicit in [Björklund, Husfeldt, and Lyckberg, IPL 2017]. For the permanent over the integers of a 0/1-matrix with exactly d ones per row and column, we provide a deterministic $2^{n-\Omega(\frac{n}{d^{3/4}})}$ time algorithm. This improves on a $2^{n-\Omega(\frac{n}{d})}$ time algorithm in [Cygan and Pilipczuk ICALP 2013]. We also show that the number of Hamiltonian cycles in an n-vertex directed graph of average degree δ can be computed by a deterministic $2^{n-\Omega(\frac{n}{\delta})}$ time algorithm. This improves on a Las Vegas algorithm running in expected $2^{n-\Omega(\frac{n}{\mathrm{poly}(\delta)})}$ time in [Björklund, Kaski, and Koutis, ICALP 2017].

A key tool in our approach is a reduction from computing the permanent to listing pairs of *dissimilar* vectors from two sets of vectors, i.e., vectors over a finite set that differ in each coordinate, building on an observation of [Bax and Franklin, Algorithmica 2002]. We propose algorithms that can be used both to derandomise the construction of Bax and Franklin, and efficiently list dissimilar pairs using several algorithmic tools. We also give a simple randomised algorithm resulting in Monte Carlo algorithms within the same time bounds.

Our new fast algorithms for listing dissimilar vector pairs from two sets of vectors are inspired by recent algorithms for detecting and counting orthogonal vectors by [Abboud, Williams, and Yu, SODA 2015] and [Chan and Williams, SODA 2016].

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases permanent, Hamiltonian cycle, orthogonal vectors

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.25

Category Track A: Algorithms, Complexity and Games

Funding *Andreas Björklund*: The Swedish Research Council grant VR 2016-03855 "Algebraic Graph Algorithms".
Ryan Williams: The U.S. National Science Foundation grant CCF-1741615 "CAREER: Common Links in Algorithms and Complexity".

1 Introduction

In recent years, the apparent impossibility of finding a pair of orthogonal vectors among two sets of N Boolean vectors in $N^{2-\varepsilon}$ time (the OV problem) has been used to derive conditional hardness results for many problems (e.g., [2] to name just one). However, when the dimensionality of the vectors is at most $d\log(N)$ for a constant d, N^c-time algorithms do exist for some $c < 2$, and these algorithms for OV have been used to derive faster algorithms for other problems; a prominent example is counting satisfying assignments to sparse CNF formulas [11]. In this paper we consider a natural generalisation of the OV problem, design algorithms for it, and apply those algorithms to derive faster deterministic algorithms for two other notoriously hard, well-known problems:

1. Computing matrix permanents over finite rings and regular 0/1 matrices over the integers.
2. Counting Hamiltonian cycles in sparse directed graphs.

The natural generalisation of OV is a problem we call **listing dissimilar vector pairs**: *Given N vectors over a finite set of size r, list all pairs of vectors that differ in every coordinate.* This problem can easily be reduced to listing orthogonal vector pairs (see Proposition 9). We also propose some tailored algorithms for listing dissimilar pairs for ease of understanding. (and in the hopes that our new algorithmic ideas will lead to interesting future work). As in the literature on orthogonal vector detection and counting [1, 11], we show how fast rectangular matrix multiplication can be used to list dissimilar vector pairs; we apply a fast algorithm for *counting* (not listing) orthogonal vectors ([11]) as a black box to derandomise our application algorithms.

A key difference in our work is that our applications to counting problems require us to list *all* dissimilar vector pairs, rather than merely finding one. The task of listing OV pairs also arises in the fastest known online algorithm for Boolean matrix-vector multiplication [15].

1.1 Faster Algorithm for Ring Permanents

The permanent of a square matrix $M = \{m_{i,j}\} \in K^{n\times n}$ over a ring K is defined as

$$\operatorname{per}(M) = \sum_{\sigma\in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}, \tag{1}$$

where S_n is the symmetric group on n elements. The problem of computing permanents is known to be #P-complete, even for sparse binary (i.e., 0/1) matrices over the integers [22], and hence we only expect exponential-time algorithms for the problem in general.

An inclusion–exclusion formula by Herbert Ryser from 1963 [18] states that

$$\operatorname{per}(M) = \sum_{X\subseteq[n]} (-1)^{n-|X|} \prod_{i=1}^{n} \left(\sum_{j\in X} m_{i,j}\right). \tag{2}$$

By enumerating the subsets X in a Gray code order (i.e., an enumeration that lists each subset exactly once by only adding or removing one element at a time) one can compute each term $\prod_{i=1}^{n}\left(\sum_{j\in X} m_{i,j}\right)$ in only $O(n)$ operations per subset, leading to an algorithm using $O(n2^n)$ additions and multiplications. To date, this is still the fastest method known for computing the permanent over general rings, and it is a major open question whether there is a $O((2-\epsilon)^n)$ time algorithm for some constant $\epsilon > 0$, even for the special case of binary matrices over the integers. For other special cases, like sparse matrices or finite rings, such algorithms *do* exist, and even for the computation of binary matrices over the integers, *somewhat* faster algorithms than Ryser's are known; see the Related Work section.

In this paper, we provide a faster algorithm for permanents over finite rings.

▶ **Theorem 1.** *There is a deterministic algorithm that computes the permanent of a matrix $M \in K^{n\times n}$ over any finite ring K on $r \le n$ elements in $2^{n-\Omega(n/r)}$ time.*

The previous best bound is a $2^{n-\Omega(n/(r\log r))}$ expected time algorithm implicit in [6].[1] Note that our result is much more than a "log-shaving" of the running time in the classical sense, as we are improving the *exponent* of the running time; moreover, our new algorithm is deterministic as opposed to the previous one.

[1] The randomised algorithms in that paper are stated for computations modulo a fixed prime and their first powers, but the algorithms can be adapted for finite rings on r elements.

We also provide a faster algorithm for d-regular 0/1 matrices with exactly d ones in each row and column.

▶ **Theorem 2.** *There is a deterministic algorithm that computes the permanent of a d-regular matrix $M \in \{0,1\}^{n\times n}$ over the integers in $2^{n-\Omega(n/d^{3/4})}$ time.*

This improves on the $2^{n-\Omega(n/d)}$ time bound of an algorithm for average d ones per row by Cygan and Pilipczuk [13].

1.2 Hamiltonian cycles

A Hamiltonian cycle in a directed graph is a simple cycle through all vertices. Counting Hamiltonian cycles is #P-complete, even in planar graphs of degree at most three [21, 16]. We provide a faster algorithm for sparse graphs.

▶ **Theorem 3.** *There is a deterministic algorithm that counts the number of Hamiltonian cycles in an n-vertex directed graph of average degree δ in $2^{n-\Omega(n/\delta)}$ time.*

The previously fastest counting algorithm (sensitive to the average degree) is a Las Vegas algorithm running in expected $2^{n-\Omega(n/\operatorname{poly}(\delta))}$ time outlined in [8]. The polynomial in the exponent is at least $\delta^4 \log \delta$.[2]

1.3 Related Work

The fastest known algorithms for the $n \times n$ matrix permanent over the integers, and counting Hamiltonian cycles in an n-vertex directed graph, are those of Björklund, Kaski, and Williams [9] which run in $2^{n-\Omega\left(\sqrt{n/\log\log n}\right)}$ time.

For permanents over the integers of binary matrices with d ones per row on average, Servedio and Wan [19] showed how to compute the permanent in $2^{n-\Omega(n/\exp(d))}$ time and polynomial space. Using exponential space, Izumi and Wadayama [14] derived a $2^{n-\Omega(n/(d\log d))}$ time algorithm. Cygan and Pilipczuk [13] gave a $2^{n-\Omega(n/d)}$ time algorithm that works over any ring, where d denotes the *average* number of non-zero entries per row. Björklund, Husfeldt, and Lyckberg [6] showed that the integer permanent can be computed modulo $p^{(1-\lambda)n/p}$ in $c_{p,\lambda}^n$ time, for $c_{p,\lambda} < 2$ depending only on the fixed prime p and $\lambda > 0$.

The fastest known algorithm for detecting the Hamiltonian cycles in an undirected graph is the $O(1.657^n)$ time Monte Carlo algorithm of Björklund [4]. For directed graphs, no detection algorithm running in $O((2-\epsilon)^n)$ time for any $\epsilon > 0$ is known, although for bipartite directed graphs, Hamiltonicity can be decided in $O(1.733^n)$ time [8]. Intriguingly, computing the *parity* of the number of Hamiltonian cycles can be done in $O(1.619^n)$ time [5], and they can be counted modulo certain integers with all prime factors at most p, in $2^{n-\Omega(n/p)}$ time [8].

There are also $6^{\mathrm{pw}(G)} \operatorname{poly}(n)$ time and $15^{\mathrm{tw}(G)} \operatorname{poly}(n)$ time algorithms parameterised by the pathwidth and treewidth of the underlying graph [10].

1.4 Our techniques and contributions

For binary 0/1 matrices over the integers, Bax and Franklin [3] gave a $2^{n-\Omega(n^{1/3}/\log n)}$ expected time Las Vegas algorithm for the $n \times n$ matrix permanent. Their algorithm is slower than the state-of-the-art algorithm of [9] (based on entirely different techniques), but still uses very interesting ingredients whose potential has probably not yet been fully utilised.

[2] See Theorem 8 in Appendix B of [7] for details.

Our algorithms in this paper stem from two ideas in Bax and Franklin's paper:

1. Perturb the input in a way that does not affect the answer, but "zeroes out" all but a tiny fraction of terms in an 2^n-sized formula for computing the answer (e.g., for permanents, we will zero-out most terms in Ryser's formula (2)). Hence to evaluate the sum, it is sufficient to list only the non-zero terms, offering an approach to a faster algorithm.
2. Divide the columns of the matrix in two halves of about $n/2$ columns each, construct vectors representing the terms restricted to the halves, and find ways to "combine" pairs of vectors corresponding to non-zero terms in the exponential sum.

By design, the vector pairs corresponding to non-zero terms will be those differing in each coordinate, which we call *dissimilar pairs*. Recently, Björklund, Kaski, and Koutis, translated this idea to the problem of counting Hamiltonian cycles [8]. In this paper, we present two new methods of listing dissimilar pairs that are more efficient than the simple tabulation-based schemes used in [3] and [8]. Our first listing algorithm is deterministic:

▶ **Theorem 4.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$ with $d < \log_3(N)/20$, and an integer $s > N$, the first s dissimilar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(N\sqrt{s})$ time.*[3]

The algorithm applies fast rectangular matrix multiplication. In particular, we use:

▶ **Theorem 5** (Coppersmith, 1984 [12]). *Over any finite field $\mathbb{F}$, the number of arithmetic operations needed to multiply an $N \times N^\alpha$ sized matrix with an $N^\alpha \times N$ sized matrix, for $\alpha < 0.17$, is $N^2 \cdot \operatorname{poly}\log(N)$.*

Our second listing algorithm is randomised, and based on hashing. It is arguably much more implementable, as it does not rely on fast matrix multiplication.

▶ **Theorem 6.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$, the set of dissimilar pairs $S \subseteq \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(|S| + 2^d \cdot N)$ time with probability of success $1 - o(1)$.*

We also prove a stronger upper bound on the number of non-zero terms needed to analyse in the special case of d-regular 0/1 matrices. The result can be seen as a sparsity parameterisation of Bax and Franklin's algorithm [3].

Outline. We describe our two dissimilarity listing algorithms in Sections 2 and 3. In Section 4 we review the (randomised) reductions from the permanent and Hamiltonian cycle counting to listing dissimilar vector pairs. Finally, in Section 5 we show how the randomised reductions can be derandomised by applying an algorithm for counting OV pairs, and using the method of conditional expectation.

2 Listing dissimilar vectors with fast matrix multiplication

We first describe a deterministic algorithm for listing dissimilar pairs based on fast rectangular matrix multiplication. Given two sets $\mathcal{X}, \mathcal{Y} \subseteq [r]^d$ with $|\mathcal{X}| = |\mathcal{Y}| = N$ and a positive integer s, we want to output a set of the (lexicographically) first s pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ that are *dissimilar*, i.e., for all $i = 1, \ldots, d$, $x_i \neq y_i$.

▶ **Reminder of Theorem 4.** *Given lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$ with $d < \log_3(N)/20$, and $N < s \leq N^2$, the first s dissimilar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(N\sqrt{s})$ time.*

[3] Note that $s \leq N^2$ implies $s \leq N\sqrt{s}$, so the running time is at least s. Also note that for $s = 2^{n-\delta n}$ and $N = 2^{n/2}$ (our applications of interest), $N \cdot s^{1/2} \leq 2^{n-\delta n/2}$.

Proof. The idea of the algorithm is to partition both $\mathcal{X}$ and $\mathcal{Y}$ in $m = s^{1/4}\sqrt{N}$ pieces, each of size at most N/m, as $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_m$ and $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_m$. First, the algorithm locates all pairs $(i,j) \in [m] \times [m]$ such that $\mathcal{X}_i \times \mathcal{Y}_j$ contains a dissimilar vector pair. This is achieved for all pairs simultaneously through several rectangular matrix products. Then the algorithm "brute-forces" all pairs in those $\mathcal{X}_i \times \mathcal{Y}_j$ containing dissimilar pairs.

Given a pair $(x,y) \in \mathcal{X} \times \mathcal{Y}$, consider the following polynomial defined over $\mathbb{Z}$:

$$p(x,y) = \prod_{i=1}^{d} (x_i - y_i)^2. \tag{3}$$

Note that $p(x,y) > 0$ if (x,y) is a dissimilar vector pair, otherwise $p(x,y) = 0$.

We can write $p(x,y)$ as a sum of 3^d products:

$$p(x,y) = \sum_{z \in \{0,1,2\}^d} (-2)^{\mathrm{ones}(z)} \left(\prod_{j=1}^{d} x_j^{z_j} \right) \left(\prod_{k=1}^{d} y_k^{2-z_k} \right), \tag{4}$$

where $\mathrm{ones}(z)$ counts the number of coordinates i in z such that $z_i = 1$.

Let $c(i,j)$ be the sum of $p(x,y)$ over all pairs $(x,y) \in \mathcal{X}_i \times \mathcal{Y}_j$, and note $c(i,j) > 0$ if and only if some pair in $\mathcal{X}_i \times \mathcal{Y}_j$ is dissimilar. We wish to compute whether $c(i,j) > 0$ for all i,j. To this end, we construct an $m \times 3^d$ integer matrix $M_{\mathcal{X}}$ representing $\mathcal{X}$, with row i representing $\mathcal{X}_i$, and columns representing different terms in (4), labeled by the corresponding z-vector. Formally, the entry at row i and column $z \in \{0,1,2\}^d$ is

$$M_{\mathcal{X},i,z} = \sum_{x \in \mathcal{X}_i} (-2)^{\mathrm{ones}(z)} \prod_{j=1}^{d} x_j^{z_j}. \tag{5}$$

Similarly, we construct an $m \times 3^d$ integer matrix $M_{\mathcal{Y}}$ representing $\mathcal{Y}$:

$$M_{\mathcal{Y},i,z} = \sum_{y \in \mathcal{Y}_i} \prod_{k=1}^{d} y_k^{2-z_k}. \tag{6}$$

We consider $P = M_{\mathcal{X}} \cdot M_{\mathcal{Y}}^{\mathrm{T}}$, and observe that $P_{i,j} = c(i,j)$ for all $i,j = 1,\ldots,m$. All entries in the result are $\mathrm{poly}(\log N, \log r)$-bit non-negative integers. To see if $c(i,j) > 0$, we compute P modulo the first $\mathrm{poly}(\log N, \log r)$ primes using Theorem 5. If any of the products has $P_{i,j} \neq 0$, we know that $c(i,j) > 0$ and mark (i,j) as containing dissimilar pairs.

Next, we loop over all marked entries (i,j) of the matrix, and test every $(x,y) \in \mathcal{X}_i \times \mathcal{Y}_j$ for dissimilarity by brute force in lexicographical order. As soon as s dissimilar pairs have been listed, the algorithm terminates.

Noting that the dimensions of the matrices obey the condition for Coppersmith's algorithm (Theorem 5), i.e., $3^d < m^{0.17}$, the running time is

$$(3^d N + m^2 + s(N/m)^2)\,\mathrm{poly}(\log N, \log r) = N\sqrt{s}\,\mathrm{poly}(\log N, \log r).$$

Here, the first two summands come from building the matrices and computing the product P, and the last summand arises from the worst case of the brute-force listing part, when every $\mathcal{X}_i \times \mathcal{Y}_j$ with $c(i,j) > 0$ contains only one dissimilar pair. This concludes the proof. ◀

3 Listing dissimilar vectors by hashing

In this section, we give an alternative listing method that avoids fast rectangular matrix multiplication. However, it is randomised, and only provides Monte Carlo algorithms running in the same time as the deterministic algorithms of Theorem 1, 2, and 3.

▶ **Reminder of Theorem 6.** *Given two lists $\mathcal{X}$ and $\mathcal{Y}$ of N vectors in $[r]^d$, the set of dissimilar pairs $S \subseteq \mathcal{X} \times \mathcal{Y}$ can be listed in $\tilde{O}(|S| + 2^d \cdot N)$ time with probability of success $1 - o(1)$.*

Proof. Let $\mathcal{H}$ be the family of hash functions $h : [r] \to \{0,1\}$. Pick $t := 3 \cdot 2^d \log(N)$ vectors of d hash functions $h_j = (h_{j,1}, h_{j,2}, \ldots, h_{j,d}) \in \mathcal{H}^d$ for $j = 1, 2, \ldots, t$.

Consider a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We say (x, y) *passes the hash* j if

$$\forall i \in 1, \ldots, d, \; h_{j,i}(x_i) \neq h_{j,i}(y_i). \tag{7}$$

Note that a similar vector pair never passes any j. Let $\varphi_{j,(x,y)}$ be the indicator for the event that a dissimilar pair (x, y) passes j, then

$$\mathbb{E}[\varphi_{j,(x,y)}] = \Pr[(x, y) \text{ pass } j] = \frac{1}{2^d}. \tag{8}$$

Form the sum

$$X = \sum_{j=1}^{t} \sum_{x,y \in S} \varphi_{j,(x,y)}. \tag{9}$$

The quantity X is the number of the vector pairs in S that survive some j, counted with multiplicity when they pass several hashes. Applying linearity of expectation to (8) and (9),

$$\mathbb{E}[X] = \frac{t|S|}{2^d} = 3|S| \log(N). \tag{10}$$

Applying Markov's inequality to (10),

$$\Pr[X > 10\mathbb{E}[X]] \leq \frac{1}{10}. \tag{11}$$

Also, the probability that a particular $(x, y) \in S$ does not pass any j, is

$$\prod_{j=1}^{2^d 3 \log N} \Pr[(x, y) \text{ does not pass } j] = \left(1 - \frac{1}{2^d}\right)^{3 \cdot 2^d \log(N)} < \exp(-3 \log N). \tag{12}$$

By a union bound, the probability that some dissimilar pair does not pass any j is at most

$$N^2 \exp(-3 \log N) < \frac{1}{N}. \tag{13}$$

Suppose we list all pairs (x, y) that pass some j. From (11) and (13), we see that with probability at least $\frac{9}{10} - \frac{1}{N}$, we will list all dissimilar pairs (possibly with repetition), and we do not list more than $30 \log N$ times the number of dissimilar pairs.

Now we describe how to list these pairs. Iterate over $j \in [t]$. For each j, iterate over $y = (y_1, \ldots, y_d) \in \mathcal{Y}$, compute its hash vector $h_j(y) := (h_{j,1}(y_1), \ldots, h_{j,d}(y_d)) \in \{0,1\}^d$, and build lists $\ell_j(v)$ for all relevant vectors $v \in \{0,1\}^d$, where $\ell_j(v) := \{y \in \mathcal{Y} \mid h_j(y) = v\}$. Note that $\sum_{v \in \{0,1\}^d} |\ell_j(v)| = N$. Next, iterate over $x = (x_1, \ldots, x_d) \in \mathcal{X}$, and output the pair (x, y') for each $y' \in \ell_j(\overline{h_j(x)})$, where $\overline{h_j(x)} := (1 - h_{j,1}(x_1), \ldots, 1 - h_{j,d}(x_d))$.

Observe the running time is $\tilde{O}(tN + X) \leq \tilde{O}(N \cdot 2^d + |S|)$. ◀

4 Reductions to listing dissimilar vectors

Here we outline reductions from our two application problems to the problem of listing dissimilar vectors, roughly following Bax and Franklin [3] for the permanent, and Björklund et al. [8] for Hamiltonian cycles. We also provide bounds on the number of dissimilar pairs that are needed for the analysis of the resulting algorithms. In particular, in Section 4.2 we provide a novel stronger bound on the number of dissimilar pairs than was previously known, for the reduction from 0/1 matrix permanents with d ones in each row and column.

4.1 Reduction for the permanent

In the following, let K be a ring on r elements denoted by $e_1, \dots, e_r$.

Bax and Franklin [3] observed the following simple but intriguing fact. For any $M \in K^{n\times n}$, let 0_n be the $1 \times n$ row vector of all zeros, and let $q \in K^{n\times 1}$ be any column vector. Form the $(n+1) \times (n+1)$ matrix

$$M_q = \begin{bmatrix} M & q \\ 0_n & 1 \end{bmatrix}. \tag{14}$$

Then we have

$$\mathrm{per}(M_q) = \mathrm{per}(M). \tag{15}$$

The equation (15) follows because every summand in the permanent (1) must include the 1 on the last row of M_q, and hence cannot include any elements of the vector q. The usefulness of this simple fact can be seen from inspecting Ryser's formula (2), partitioned in the form

$$\mathrm{per}(M_q) = \sum_{X\subseteq[n]} (-1)^{n-|X|} f(M_q, X),\ f(M_q, X) = \prod_{i=1}^{n} g_i(M_q, X),\ g_i(M_q, X) = q_i + \sum_{j\in X} M_{i,j}. \tag{16}$$

Note that $f(M_q, X) = 0$ if and only if $g_i(M_q, X) = 0$ for some i. So in order to compute $\mathrm{per}(M)$, it is enough to list those subsets $X \subseteq [n]$ such that $g_i(M_q, X) \neq 0$ for all i, and accumulate their contributions. We call such X's *contributing terms*. Furthermore, we say a subset $X \subseteq [n]$ is *k-weakly contributing* if $g_i(M_q, X)$ is non-zero for all $i \leq k$.

In particular, by choosing q uniformly at random, we can easily compute the expected number of k-weakly contributing terms, as the events $g_i(M_q, X) \neq 0$ for $i = 1, \dots, k$ are mutually independent (they depend on different q_i). Let Y be the random variable equal to the number of k-weakly contributing terms under a random $q \in K^{n\times 1}$, i.e., $Y = \sum_X Y_X$ where Y_X is the indicator of whether X is k-weakly contributing. By linearity of expectation,

$$\mathbb{E}(Y) = \sum_{X\subseteq[n]} \mathbb{E}(Y_X) = 2^n \left(1 - \frac{1}{r}\right)^k < 2^n \exp(-k/r). \tag{17}$$

Thus, if we could efficiently list the k-weakly contributing terms for some $k \geq \Omega(n)$, we would have a Las Vegas algorithm running in expected $2^{n-\Omega(n/r)}$ time. In Theorem 1, we claim a *deterministic* algorithm, in which case we cannot choose q at random. We will address this issue later in Section 5. For now, we concentrate on finding the contributing terms for a fixed q. We describe next how k-weakly contributing terms can be viewed as *dissimilar vector pairs* from two sets of short vectors.

4.1.1 Contributing terms as dissimilar vectors

Here we show how to reduce the problem of listing contributing terms to the problem of listing dissimilar vectors, implying a randomised version of Theorem 1. Assume WLOG that our matrix M is $n \times n$ and n is even; recall its entries are elements from the ring $K = \{e_1, \dots, e_r\}$. We begin by partitioning the columns in two halves $L = \{1, \dots, n/2\}$ and $R = \{n/2+1, \dots, n\}$. Let $\mathcal{L}$ be the power set of L, and $\mathcal{R}$ the power set of R. Define a map $\phi : K \to [r]$ by $\phi(e_i) := i$.

For every $A \in \mathcal{L}$, construct a vector $v^A \in [r]^k$ for $i \in \{1, \dots, k\}$ as

$$v_i^A = \phi\left(\sum_{k \in A} M_{i,k}\right), \text{ for all } i = 1, \dots, k. \tag{18}$$

Similarly, but asymmetrically, for every $B \in \mathcal{R}$ we construct the vector $v^B \in [r]^k$ as

$$v_i^B = \phi\left(-q_i - \sum_{k \in B} M_{i,k}\right), \text{ for all } i = 1, \dots, k. \tag{19}$$

Recall two vectors u and v are dissimilar if they differ in each coordinate. One can easily verify for all $A \in \mathcal{L}$ and $B \in \mathcal{R}$, the two vectors v^A and v^B are dissimilar if and only if the subset $X = A \cup B$ is k-weakly contributing. Note that $\mathcal{L}$ and $\mathcal{R}$ describe $N = 2^{n/2}$ vectors each of dimension k. For $k \geq \Omega(n)$, the number of k-weakly contributing terms in (17) is (expected to be) at most $2^{n-cn/r}$ for a constant $c > 0$. In that case, the number of dissimilar vector pairs s in our instance of $O(2^{n/2})$ vectors is at most $2^{n-cn/r}$.

Given an algorithm that efficiently lists dissimilar vector pairs, we can then efficiently list all $2^{n-cn/r}$ of the k-weakly contributing terms in the modified Ryser's formula (16). Given the list of contributing terms, we can then compute the permanent via (16), in time $2^{n-cn/r} \cdot \text{poly}(n)$. Using the listing algorithm of Theorem 4 with $s = 2^{n-cn/r}$, we obtain an algorithm with running time $2^{n-\Omega(\frac{n}{r})}$ as claimed.

4.2 The permanent algorithm for regular matrices

We now turn to giving a Las Vegas version of Theorem 2, showing that for d-regular matrices the permanent can be computed in $2^{n-\Omega(n/d^{3/4})}$ time. Later in Section 5, we will outline how to derandomise the algorithm.

Let $M \in \{0,1\}^{n \times n}$ have exactly d ones in every row and column. As in the case of permanents over finite rings, we reduce to dissimilar pair listing and apply the algorithm of Theorem 4 to locate k-weakly contributing terms for the permanent of the perturbed matrix M_q of (14). However, here we will pick the vector q from a different distribution, and we may also permute the rows of M to select a suitable subset of rows to use in the reduction to dissimilar pair listing. Our choices drastically reduce the number of contributing terms.

Index the rows and columns of M by $[n]$. For $i = 1, \dots, n$, the ith row of M naturally corresponds to a d-size subset R_i of $[n]$ indicating which columns have a 1 in the row.

We first consider the case $d \geq n^{4/5}$. Here our algorithm will work as in Bax and Franklin [3] (whose analysis works for $d \in \Omega(n)$, for which the slightly better bound $2^{n-\Omega(n^{1/3})}$ can be argued). The probability for a fixed row $i \in [n]$, that the sieved row sum $|X \cap R_i|$ is deviating significantly from its expectation, is small:

$$\Pr_X\left[\left||X \cap R_i| - \frac{d}{2}\right| > d^{3/4}\right] \leq \exp(-\Omega(\sqrt{d})), \tag{20}$$

as seen by a standard Chernoff bound. Note that $\frac{n}{d^{3/4}} \leq n^{2/5} \leq \sqrt{d}$ for $d \geq n^{4/5}$. By a union bound, the probability that some row has intersection with X outside of the $2d^{3/4}$-length interval in (20) is at most $n \cdot \exp(-\Omega(\sqrt{d})) \leq \exp(-\Omega(n/d^{3/4}))$. Hence there are at most $2^{n-\Omega(n/d^{3/4})}$ such deviating subsets $X \subseteq [n]$.

Therefore, if we run our previous permanent algorithm but with the vector q taking random values in the range $[\frac{d}{2} - d^{3/4}, \frac{d}{2} + d^{3/4}]$, we see that the probability that any X within the $2d^{3/4}$-span for every row, is k-weakly contributing, is at most

$$\Pr_q[X \text{ passes}] \leq \left(1 - \frac{1}{2d^{3/4}}\right)^k . \tag{21}$$

Hence for $k \geq \Omega(n)$, the expected number of non-zero terms is only $2^{n-\Omega(n/d^{3/4})}$, so we only need to list that many dissimilar vector pairs in our reduction.

Now consider the more interesting case of $d < n^{4/5}$. First, we preprocess the matrix via a process we call P, which puts the row sets representing M into a number of ordered lists.

Process P: Start with an empty list L_1, and put the first row set R_1 in L_1. While there is another row set R_i, not yet assigned to a list, that has at most $d^{3/4}$ elements in common with the union of the row sets in L_1, insert R_i into L_1. This operation is repeated until there either are no more row sets that meet this criterion, or L_1 contains $n/(2d^{5/4})$ row sets. In the latter case we say that the list is *full*. We consider L_1 *finished*, put it aside, and start building a new, initially empty, list L_2. We insert row sets into L_2 in the same way, continue with a third list L_3 once L_2 is finished, and so on. The process P terminates when every row set has been assigned to some list.

After we have constructed the lists $L_1, L_2, \ldots$, we reorder the rows and columns of M according to the insertion order of the row sets during process P. This permutation does not change the value of the permanent. The following lemma ensures that the first $n/2$ rows of M, after the reordering step, can be partitioned into full lists.

▶ **Lemma 7.** *The first $d^{5/4}$ lists produced by process P are full, i.e., each list contains $n/(2d^{5/4})$ row sets.*

Proof. Assume there are at least $n/2 + n/(2d^{5/4})$ row sets left to place when we start populating the list L_j. After we have put t sets into a list L_j, their union covers no more than td elements. Each element of $[n]$ is covered d times in total, since M is d-regular. Thus there are at most $td^{2-3/4}$ row sets left that has an overlap of more than $d^{3/4}$ with the union.

Therefore, as long as $\frac{n}{2} + n/(2d^{5/4}) - t > td^{2-3/4}$, i.e., $t < \frac{n}{2(d^{2-3/4})}$, there is still a row set that can be put into L_j. Hence we can put at least $\frac{n}{2d^{5/4}}$ row sets into L_j, making it full. We can repeat this for $j = 1, 2, \ldots, d^{5/4}$, as there are still $n/2 + n/(2d^{5/4})$ row sets left when we start to populate $L_{d^{5/4}}$. ◀

Next we need a Chernoff-like concentration bound on the sum of variables with bounded dependence resulting from a list L_j.

▶ **Lemma 8.** *Let L be an ordered list of d-subsets $S_1, \ldots, S_m$ of $[n]$ such that for all i, $|S_i \cap (\cup_{j=1}^{i-1} S_j)| < d^{3/4}$. Pick $X \subseteq [n]$ uniformly at random. Let Z_i be the indicator variable for the event $\frac{d}{2} - 3d^{3/4} \leq |S_i \cap X| \leq \frac{d}{2} + 3d^{3/4}$. Then*

$$\Pr_X\left[\sum_{i=1}^{m} Z_i < \frac{m}{2}\right] \leq \exp(-\Omega(d^{1/2}m)). \tag{22}$$

Proof. Let $U = [\ell] = \cup_{i=1}^m S_i$ and $X_1, \ldots, X_\ell$ be indicator variables such that $X_i = 1 \Leftrightarrow i \in X$. Hence the X_i's are mutually independent with $E[X_i] = 1/2$. Construct the sets $T_i = S_i \setminus (\cup_{l=1}^{i-1} S_l)$ for $i = 1, \ldots, m$. Note that $|T_i| \geq d - d^{3/4}$ for all i, and that T_i and T_j are disjoint for $i \neq j$. Letting $Y_i = \sum_{j \in T_i} X_j$, we have

$$\Pr_X \left[\left| Y_i - \frac{d}{2} \right| > 2d^{3/4} \right] \leq \exp(-\Omega(d^{1/2})), \tag{23}$$

as seen by a standard Chernoff bound. Note that if $|Y_i - d/2| \leq 2d^{3/4}$, then also $Z_i = 1$, as there are only at most $d^{3/4}$ other elements of X that are in $S_i \setminus T_i$. Further note that the events $|Y_i - d/2| > 2d^{3/4}$ for different i are mutually independent as they depend on different mutually independent X_j's. Hence the probability that $\sum_i Z_i < m/2$ is at most $m/2 \cdot \exp(-\Omega(d^{1/2}))^{m/2} \binom{m}{m/2} \leq \exp(-\Omega(d^{1/2} m))$. ◀

We can now argue as in the dense case (where d is large). We first bound the number of X's such that for some list L_i, more than half of the row sets in the list have a deviating sieve row sum $\left| |R_j \cap X| - \frac{d}{2} \right| > 3d^{3/4}$. From Lemma 8, replacing $m = n/(2d^{5/4})$, and a union bound over all lists, we know that this happens for at most $2^{n - \Omega(n/d^{3/4})}$ X's. Thus we may restrict our analysis to the X's that are within the $6d^{3/4}$-length interval for at least half of the rows in the first $d^{5/4}$ lists (amounting to the first $n/2$ rows of the matrix, after the pre-processing reordering). Running our algorithm from before, but with the vector q taking random values in the range $[\frac{d}{2} - 3d^{3/4}, \frac{d}{2} + 3d^{3/4}]$, we see that each of these remaining well-behaved X's are k-weakly contributing with probability at most

$$\Pr_q[X \text{ passes}] \leq \left(1 - \frac{1}{6d^{3/4}}\right)^{k/2}. \tag{24}$$

Hence, for $k = \frac{n}{c}$ for some $c > 2$, we have in expectation $2^{n - \Omega(n/d^{3/4})}$ non-zero terms.

4.3 Reduction for counting Hamiltonian cycles

In the following let $G' = (V', A')$ be the directed input graph on $n = |V'|$ vertices in which we want to count Hamiltonian cycles. Let d_v for $v \in V'$ be the out-degree of the vertex v, and let $\delta = |A'|/n$ be the average out-degree of G'. It will be convenient to work on a slightly modified graph $G = (V, E)$ constructed from G' as follows: pick an arbitrary vertex $s' \in V'$, and obtain G from G' by replacing s' with two new vertices s and t (i.e., $V = V' \setminus \{s'\} \cup \{s, t\}$), where s retains all outgoing arcs from s', i.e. $(s', u) \in A' \Leftrightarrow (s, u) \in A$, and t retains all incoming arcs to s', i.e. $(u, s') \in A' \Leftrightarrow (u, t) \in A$. Note that the Hamiltonian paths from s to t in G are in one-to-one correspondence with the Hamiltonian cycles in G', and that the average degree of G is not larger than that of G'. In the following, we consider the problem of counting the s-t Hamiltonian paths on the modified $n + 1$ vertex graph G.

4.3.1 Random columns and contributing terms

Björklund, Kaski, and Koutis [8] observed that the number of Hamiltonian cycles in a directed graph can be evaluated as an inclusion–exclusion summation over a determinant of a polynomial matrix representing the graph. We will use their construction, and restate the main ideas here. The Laplacian of the graph G, is a $(n + 1) \times (n + 1)$ polynomial matrix with rows and columns indexed by the vertices V, in the variables x_v for $v \in V$:

$$L(G)_{i,j} = \begin{cases} \sum_{(u,v)\in A} x_u & \text{if } i = j = v \\ -x_u & \text{if } i = u, j = v, (u,v) \in A \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The Laplacian *punctured at the vertex* $s \in V$, is the matrix $L(G)_s$ obtained by omitting row and column s from $L(G)$. By Tutte's directed version of the Matrix-Tree theorem of Kirchhoff [20], we know that $\det(L(G)_s)$ is a polynomial where each term corresponds to a directed spanning out-branching rooted at s. Denote by $\text{hp}(G)_{s,t}$ the number of Hamiltonian paths starting in s and ending in t. By the principle of inclusion–exclusion, we have[4]

$$\text{hp}(G)_{s,t} = \sum_{\substack{x:(V\setminus\{t\})\to\{0,1\} \\ x_t=1}} (-1)^{n-1-|x|} \det\left(L(G)_s(x)\right). \tag{26}$$

The summation is over all 2^{n-1} assignments x with the restriction that $x_t = 1$. Next, we mimic the Bax and Franklin idea for computing permanents by perturbing the matrix (of (15)): we shall parametrise the Laplacian matrices so that in expectation, many summands in the above formula are zeroed-out. To this end we introduce fresh random variables $q_v \in \{0, 1, \ldots, d_v\}$ for $v \in V$, and define the *q-perturbed Laplacian* of G as

$$L_q(G)_{i,j} = \begin{cases} \sum_{(u,v)\in A} x_u - q_v & \text{if } i = j = v \\ -x_u & \text{if } i = u, j = v, (u,v) \in A \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

The extra q_v variables may be thought of as weighted arcs originating from t: these arcs cannot be used by any of the Hamiltonian paths from s to t. The point of our perturbation is that irrespective of q, we can still compute the number of Hamiltonian paths:

$$\text{hp}(G')_{s',t'} = \sum_{\substack{x:(V\setminus\{t\})\to\{0,1\} \\ x_t=1}} (-1)^{n-1-|x|} \det\left(L_q(G)_s(x)\right). \tag{28}$$

However, in expectation, many assignments x may yield $\det\left(L_q(G)_s(x)\right) = 0$, particularly when a row in $L_q(G)_s(x)$ is all-zeroes. Observe that a row in $L_q(G)_s(x)$ is all-zero if and only if for some $v \in V \setminus \{s\}$ we have $x_v = 0$ and $\sum_{(u,v)\in A} x_u = q_v$. Let ε_v be a Boolean variable that is true if and only if $x_v = 0$ and $\sum_{(u,v)\in A} x_u = q_v$; i.e., $L_q(G)_s(x)$ is all-zero in the row indexed by v. In analogy with the case of the permanent, we say that an assignment x is *contributing* if ε_v is false for all vertices $v \in V \setminus \{s\}$, and x is *k-weakly contributing* if ε_v is false for the first k vertices. As in the case of the permanent, it is sufficient to list the k-weakly contributing terms for some k to compute the number of Hamiltonian paths in G (and hence the Hamiltonian cycles in G') through the formula of (28).

We can easily compute the expected number of contributing terms for a $q \in \{0, 1, \ldots, d_v\}^n$ picked uniformly at random, since the events ε_v are mutually independent (they depend on different q-values). The probability that $L_q(G)_s(x)$ is all-zero in the row v, with $x_v = 0$, is

$$\Pr_q[\varepsilon_v] = \frac{1}{(d_v + 1)}. \tag{29}$$

We will assign $k := cn$ for some $c < 1/2$, and assume WLOG that the vertices are sorted by increasing out-degree. This means by an averaging argument that for all $i \le k$, $d_i < 2\delta$.

[4] See [8] for a proof of (26).

Let Z_x be the indicator variable that the assignment x is k-weakly contributing, under a randomly chosen q. Let Z be the random variable equal to the number of k-weakly contributing terms under a random q, i.e., $Z = \sum_x Z_x$. By linearity of expectation, and Jensen's inequality for concave functions,

$$\mathbb{E}(Z) = \sum_{X \subseteq V\setminus\{s\}} \prod_{v \in X\cap[k]} \left(1 - \frac{1}{d_v+1}\right) \leq 2^n \left(1 - \frac{1}{2\delta+1}\right)^k \leq 2^n \cdot \exp(-\Omega(k/\delta)). \quad (30)$$

Thus, the expected number of k-weak contributors behaves similarly as in previous cases.

4.3.2 Contributing terms as dissimilar vectors

We now turn to describing how the contributing terms can be encoded as dissimilar vectors. Assume WLOG that n is even, and identify $V \setminus \{s\}$ with the set $[n]$. Partition the vertices into $L = \{1, \ldots, n/2\}$ and $R = \{n/2+1, \ldots, n\}$, letting $\mathcal{L}$ and $\mathcal{R}$ be the power sets of L and R, respectively. For every $C \in \mathcal{L}$, we construct the vector $v^C \in [2\delta+1]^k$ for $i = 1, \ldots, k$ as

$$v_i^C = \begin{cases} \sum_{(i,w)\in(A\cap C)} 1 & \text{if } i \notin C, \\ \star & \text{otherwise.} \end{cases} \quad (31)$$

where $\star$ is an extra symbol encoded as 2δ. Similarly, but asymmetrically, noting in particular that $i \notin R$ because $k < n/2$, for every $D \in \mathcal{R}$ we construct the k-length vector v^D as

$$v_i^D = q_i - \sum_{(i,w)\in(A\cap D)} 1, \text{ for all } i = 1, \ldots, k. \quad (32)$$

It is readily verified that for $C \in \mathcal{L}$ and $D \in \mathcal{R}$, the two vectors v^C and v^D are dissimilar if and only if the first k columns of $L_q(G)(x')$ are non-zero, where the Boolean vector x' is defined as $x'_v := 1 \iff v \in C \cup D$. Hence, the k-weakly contributing assignments x are precisely those corresponding to dissimilar pairs (v^C, v^D).

Note that $\mathcal{L}$ and $\mathcal{R}$ each contain $N = 2^{n/2}$ vectors of dimension k. Using a sufficiently fast algorithm for listing dissimilar vector pairs, we can enumerate all k-weakly contributing terms in (28) in $N^{2-\Omega(1/\delta)}$ time. Once we have a list of all the $2^n \cdot \exp(-\Omega(k/\delta))$ contributing terms, the number of Hamiltonian cycles can be computed with (28) in time $2^n \cdot \exp(-\Omega(k/\delta))$ as well. From (30), we know that the upper bound s on the number of dissimilar pairs can be set to $2^{n-cn/\delta}$, for some positive $c > 0$. Applying the dissimilar pair listing algorithm of Theorem 4, we arrive at the running time $2^{n-\Omega(\frac{n}{\delta})}$. This concludes the algorithm.

5 Derandomisation

Our deterministic algorithm for dissimilar pair listing from Section 2 can be used to list k-weakly contributing terms efficiently in all our desired applications (Theorems 1, 2, and 3), provided that the number of contributing terms is not much more than their expected number would be on a random vector q.

To make all of our application algorithms fully deterministic, we must provide a deterministic procedure for setting the vector q so that the number of resulting terms is bounded by the expectation. This can be done by using other known algorithmic tools, along with the well-known method of conditional expectation (see e.g., [17]).

It turns out that a fast deterministic algorithm for counting dissimilar pairs will suffice. This can be obtained by reducing our problem to counting orthogonal pairs:

▶ **Proposition 9.** *Counting the dissimilar pairs on two sets of N vectors in $[r]^d$ can be reduced in $O(Nrd)$ time to counting orthogonal pairs on two sets of N vectors in $\{0,1\}^{rd}$.*

Proof. Let $e_1, \ldots, e_r \in \{0,1\}^r$ be the standard basis vectors, where e_i is 1 in the ith component and is 0 everywhere else. Map every vector $x \in [r]^d$ to the Boolean vector

$$\rho(x) = [e_{x_1}\ e_{x_2}\ \cdots\ e_{x_d}].$$

That is, $x' \in \{0,1\}^{rd}$ is obtained by *concatenating* the d standard basis vectors corresponding to the entries of x. Note $x, y \in [r]^d$ are dissimilar $\iff$ $\rho(x)$ and $\rho(y)$ are orthogonal. ◀

Applying the proposition, we can count dissimilar pairs by counting OV pairs:

▶ **Theorem 10** (Chan and Williams [11]). *For every $c \le 2^{o(\sqrt{\log N})}$, there is a deterministic algorithm that for two sets of N vectors from $\{0,1\}^{c \log N}$, counts the orthogonal vector pairs in $N^{2-1/O(\log c)}$ time.*

An immediate corollary is that, for $d \le \log(N)$, we can deterministically count all dissimilar pairs over N vectors in $[r]^d$ in only $N^{2-1/O(\log r)}$ time. We can use this counting algorithm to *deterministically* search for a vector q that has at most as many k-weakly contributing terms as the expected number for a random q. We describe the procedure generically, as it is essentially the same for all three applications in the paper:

> Iterate over $j = 1, \ldots, k$. Suppose we have determined the first $j-1$ components of our vector q, and we wish to determine the jth component, q_j. Let us inductively suppose that there are at most $2^n \cdot \prod_{i=1}^{j-1}(1 - 1/C_i)$ contributing terms remaining (where the C_i depend on the application), and let there be C_j possible values for q_j. Construct and compute C_j distinct instances of dissimilar pair counting with j-dimensional vectors, corresponding to the C_j different values for q_j. Finally, set q_j to be the value which minimises the number of dissimilar pairs obtained.

Since we always choose q_j to minimise the number of dissimilar pairs, and we know a random setting of q_j reduces the number by a $(1 - 1/C_j)$ fraction in expectation (by our analyses in previous sections), our k-dimensional vector q produces a number of k-weakly contributing terms which is at most the expectation. Finally, note that this initial procedure for selecting the vector q is much faster than the overall running time in our applications.

This concludes our derandomisation, and the proofs of Theorems 1, 2, and 3.

References

1 Amir Abboud, Ryan Williams, and Huacheng Yu. More Applications of the Polynomial Method to Algorithm Design. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

2 Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 51–58, New York, NY, USA, 2015. ACM.

3 Bax and Franklin. A Permanent Algorithm with $\exp[\Omega(N^{1/3}/2\ln N)]$ Expected Speedup for 0-1 Matrices. *Algorithmica*, 32(1):157–162, January 2002.

4 Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

5 Andreas Björklund and Thore Husfeldt. The Parity of Directed Hamiltonian Cycles. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 727–735, 2013. `doi:10.1109/FOCS.2013.83`.

6 Andreas Björklund, Thore Husfeldt, and Isak Lyckberg. Computing the permanent modulo a prime power. *Inf. Process. Lett.*, 125:20–25, 2017. `doi:10.1016/j.ipl.2017.04.015`.

7 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and Out-Branchings via Generalized Laplacians. *CoRR*, abs/1607.04002, 2016. `arXiv:1607.04002`.

8 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and Out-Branchings via Generalized Laplacians. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 91:1–91:14, 2017. `doi:10.4230/LIPIcs.ICALP.2017.91`.

9 Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya sets for polynomial evaluation and faster computation of fermionants. *Algorithmica*, September 2018.

10 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic Single Exponential Time Algorithms for Connectivity Problems Parameterized by Treewidth. *Inf. Comput.*, 243(C):86–111, August 2015.

11 Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-smolensky. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1246–1255, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics.

12 Don Coppersmith. Rapid Multiplication of Rectangular Matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. `doi:10.1137/0211037`.

13 Marek Cygan and Marcin Pilipczuk. Faster Exponential-time Algorithms in Graphs of Bounded Average Degree. *Inf. Comput.*, 243(C):75–85, August 2015.

14 Taisuke Izumi and Tadashi Wadayama. A New Direction for Counting Perfect Matchings. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 591–598, Washington, DC, USA, 2012. IEEE Computer Society.

15 Kasper Green Larsen and Ryan Williams. Faster Online Matrix-vector Multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2182–2189, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

16 Maciej Liśkiewicz, Mitsunori Ogihara, and Seinosuke Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theoretical Computer Science*, 304(1):129–156, 2003.

17 Prabhakar Raghavan. Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs. *J. Comput. Syst. Sci.*, 37(2):130–143, October 1988.

18 Herbert John Ryser. *Combinatorial Mathematics.* Mathematical Association of America, 1963.

19 Rocco A. Servedio and Andrew Wan. Computing sparse permanents faster. *Information Processing Letters*, 96(3):89–92, 2005.

20 W. T. Tutte. The dissection of equilateral triangles into equilateral triangles. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(4):463–482, 1948.

21 Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8:410–421, 1979.

22 L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.