
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Björklund, Andreas; Kaski, Petteri

Counting short vector pairs by inner product and relations to the permanent

Published in:
48th International Colloquium on Automata, Languages, and Programming, ICALP 2021

DOI:
[10.4230/LIPIcs.ICALP.2021.29](https://doi.org/10.4230/LIPIcs.ICALP.2021.29)

Published: 01/07/2021

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Björklund, A., & Kaski, P. (2021). Counting short vector pairs by inner product and relations to the permanent. In N. Bansal, E. Merelli, & J. Worrell (Eds.), *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021* Article 29 (Leibniz International Proceedings in Informatics, LIPIcs; Vol. 198). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPIcs.ICALP.2021.29>

Counting Short Vector Pairs by Inner Product and Relations to the Permanent

Andreas Björklund¹ ✉

Lund, Sweden

Petteri Kaski ✉

Department of Computer Science, Aalto University, Espoo, Finland

Abstract

Given as input two n -element sets $\mathcal{A}, \mathcal{B} \subseteq \{0, 1\}^d$ with $d = c \log n \leq (\log n)^2 / (\log \log n)^4$ and a target $t \in \{0, 1, \dots, d\}$, we show how to count the number of pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with integer inner product $\langle x, y \rangle = t$ deterministically, in $n^2 / 2^{\Omega(\sqrt{\log n \log \log n / (c \log^2 c)})}$ time. This demonstrates that one can solve this problem in deterministic subquadratic time almost up to $\log^2 n$ dimensions, nearly matching the dimension bound of a subquadratic randomized detection algorithm of Alman and Williams [FOCS 2015]. We also show how to modify their randomized algorithm to count the pairs w.h.p., to obtain a fast randomized algorithm.

Our deterministic algorithm builds on a novel technique of reconstructing a function from sum-aggregates by prime residues, or *modular tomography*, which can be seen as an additive analog of the Chinese Remainder Theorem.

As our second contribution, we relate the fine-grained complexity of the task of counting of vector pairs by inner product to the task of computing a zero-one matrix permanent over the integers.

2012 ACM Subject Classification Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases additive reconstruction, Chinese Remainder Theorem, counting, inner product, modular tomography, orthogonal vectors, permanent

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.29

Category Track A: Algorithms, Complexity and Games

Related Version *Previous Version:* <https://arxiv.org/abs/2007.14092>

Acknowledgements We thank Virginia Vassilevska Williams and Ryan Williams for many useful discussions. We also thank the anonymous reviewers for their useful remarks.

1 Introduction

The Inner Product and the Size of Preimages. The inner product map $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$ of two d -dimensional vectors $x = (x_1, x_2, \dots, x_d)$ and $y = (y_1, y_2, \dots, y_d)$ is one of the cornerstones of linear algebra and its applications. For example, when x and y are vectors of observations normalized to zero mean and unit standard deviation, $\langle x, y \rangle$ is the Pearson correlation between x and y . As such, it is a fundamental computational and data-analytical task to efficiently gain information about the *preimages* of the inner product map; for example, to highlight pairs of similar or dissimilar observables between two families of n observables.

¹ This work was carried out while A.B. was employed as a researcher at Lund University, Department of Computer Science, and the major part of the writeup was carried out while A.B. was employed as a researcher at Ericsson Research



The protagonist of this paper is the following counting problem ($\#$ INNERPRODUCT):

Given as input a target $t \in \{0, 1, \dots, d\}$ and two n -element sets $\mathcal{A} \subseteq \{0, 1\}^d$ and $\mathcal{B} \subseteq \{0, 1\}^d$, count the number of vector pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with integer inner product $\langle x, y \rangle = t$.

From a complexity-theoretic standpoint, this problem generalizes many conjectured-hard problems in the study of fine-grained complexity – such as the $t = 0$ special case, the *orthogonal vector counting* ($\#$ OV) problem – as well as generalizing fundamental application settings, such as similarity search in Hamming spaces. While it is immediate that subquadratic scalability in n is obtainable when $d = o(\log n)$, our interest in this paper is to obtain an improved understanding of the fine-grained complexity landscape for *moderately short* vectors, specifically for d at most poly-logarithmic in n .

Subquadratic Scaling for Moderately Short Vectors. Our main positive result establishes deterministic subquadratic scalability for $\#$ INNERPRODUCT up to d growing essentially as the square of the logarithm of n :

► **Theorem 1** (Main; Subquadratic Scaling for $\#$ INNERPRODUCT). *There exists a deterministic algorithm that, given as input a target $t \in \{0, 1, \dots, c \log n\}$ and two n -element sets $\mathcal{A}, \mathcal{B} \subseteq \{0, 1\}^{c \log n}$ with $4 \leq c \leq \frac{\log n}{(\log \log n)^4}$, outputs the number of pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with $\langle x, y \rangle = t$ in time*

$$n^2 / 2^{\Omega\left(\sqrt{\frac{\log n \log \log n}{c \log^2 c}}\right)}. \quad (1)$$

The algorithm in Theorem 1 is based on a novel technique of reconstructing a function from its sum-aggregates by prime residue, which can be seen as an *additive* analog of the Chinese Remainder Theorem and may be of independent interest (cf. Sect. 2).

We also show how a randomized algorithm for the decision problem of checking for a pair of vectors whose Hamming distance is less than a target by Alman and Williams [6], can with a small modification be turned into an algorithm for $\#$ INNERPRODUCT.

► **Theorem 2** (Randomized Subquadratic Scaling for $\#$ INNERPRODUCT). *There exists a randomized algorithm that w.h.p., given as input a target $t \in \{0, 1, \dots, c \log n\}$ and two n -element sets $\mathcal{A}, \mathcal{B} \subseteq \{0, 1\}^{c \log n}$ with $4 \leq c \leq \frac{\log n}{(\log \log n)^3}$, outputs the number of pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with $\langle x, y \rangle = t$ in time*

$$n^2 / 2^{\Omega\left(\frac{\log n}{c \log^2 c}\right)}. \quad (2)$$

While the randomized algorithm in Theorem 2 is faster than the deterministic one in Theorem 1, we stress that as far as we know no deterministic algorithm in subquadratic time was previously known for $\#$ INNERPRODUCT, even for $O(\log n)$ dimensions. In particular, derandomizing Theorem 2 while retaining subquadratic time seems challenging, even though some progress on the amount of randomness needed in the algorithm has been made, cf. Theorem 1.1 in [3].

Our further objective is to understand the fine-grained complexity of $\#$ INNERPRODUCT in relation to that of $\#$ OV and other counting problems. For $d = O(\log n)$, it is known that these problems are truly-subquadratically related; indeed, Chen and Williams [15] give a parsimonious reduction for the detection variants of these two problems. That is, if $\#$ OV can be solved in $n^{2-\omega(1)}$ time, then so can $\#$ INNERPRODUCT. However, while

there is a subquadratic time algorithm for $\#OV$ whose running time scales as good as $n^{2-\Omega(1/\log c)}$ [14], the reduction of Chen and Williams [15] does not immediately give a non-trivial algorithm for $\#INNERPRODUCT$. Indeed, the fastest known algorithm for the decision version $INNERPRODUCT$ utilizes probabilistic polynomials for symmetric Boolean functions with optimal dependence on the degree and error [6], and does not go via fast OV algorithms and the reduction above. In Theorem 2, we show how a simple modification to the algorithm in Alman and Williams [6] can turn their algorithm into a counting one. We note that while Alman, Chan, and Williams [3] later presented a deterministic algorithm based on Chebyshev polynomials over the reals for minimum/maximum Hamming weight pair, with the same running time as the randomized one in [6], that deterministic algorithm, or the even faster randomized one they presented, can not be turned into one for $\#INNERPRODUCT$ by our suggested modification alone.

Lower Bounds via the Permanent. The running times (1) and (2) would, at least at first, appear to leave room for improvement. Indeed, the running time (2) is considerably worse than the running time $n^{2-\Omega(1/\log c)}$ obtained by Chan and Williams [14] for $\#OV$. We proceed to show that this intuition might be misleading, since such scalability would imply the existence of considerably faster algorithms for a canonical hard problem in exponential-time complexity. Accordingly, to gain insight into the complexity of $\#INNERPRODUCT$ and $\#OV$ when $d = \omega(\log n)$, we introduce our second protagonist (R -PERMANENT):

Given as input an $n \times n$ matrix M with entries m_{ij} in a ring R for $i, j \in [n]$, compute the *permanent*

$$\text{per } M = \sum_{\sigma \in S_n} \prod_{i \in [n]} m_{i, \sigma(i)},$$

where S_n is the group of all permutations of $[n] = \{1, 2, \dots, n\}$.

Ryser's algorithm from 1963 computes the permanent with $O(n2^n)$ arithmetic operations in R [22]. It is a major open problem whether this can be improved to $O(c^n)$ for some constant $c < 2$. Even improving the running time to less than 2^n operations has been noted as a challenge by Knuth in the *Art of Computer Programming* [19, Exercise 4.6.4.11]. Valiant in 1979 famously proved that the permanent is $\#P$ -complete even when restricted to $m_{ij} \in \{0, 1\}$ and evaluated over the ring of integers [24]; this version of the problem can be interpreted as counting the perfect matchings in a balanced bipartite graph having the matrix as its biadjacency matrix. For zero-one inputs over the integers, *somewhat* faster algorithms are known (cf. related work below); to the best of our knowledge, the current champion for zero-one matrices computes the permanent in $2^{n-\Omega(\sqrt{n/\log \log n})}$ time [12].

As our second contribution, we relate the fine-grained scalability of $\#INNERPRODUCT$ and $\#OV$ to the task of computing the permanent of a zero-one matrix over the integers. In particular, our first result shows that if we could solve $\#INNERPRODUCT$ as fast as the fastest currently known algorithms for $\#OV$ [14], then we would immediately obtain a much faster algorithm for the zero-one matrix permanent:

► **Theorem 3** (Lower Bound for $\#INNERPRODUCT$ via Integer Permanent). *If there exists an algorithm for solving $\#INNERPRODUCT$ for N vectors from $\{0, 1\}^{c \log N}$ in time $N^{2-\Omega(1/\log c)}$, then there exists an algorithm solving the permanent of an $n \times n$ zero-one matrix over the integers in time $2^{n-\Omega(n/\log n)}$.*

Thus, despite the true-subquadratic equivalence for $d = O(\log n)$ [15], it would appear that $\# \text{INNERPRODUCT}$ and $\# \text{OV}$ have different complexity characteristics when $d = \omega(\log n)$.

Our next result shows that a modest improvement in fine-grained scalability of $\# \text{OV}$ would likewise imply much faster algorithms for the permanent.

► **Theorem 4** (Lower Bound for $\# \text{OV}$ via Integer Permanent). *If there exists an algorithm for solving $\# \text{OV}$ for N vectors from $\{0, 1\}^{c \log N}$ in time $N^{2 - \Omega(1/\log^{1-\epsilon} c)}$ for some $\epsilon > 0$, then there exists an algorithm solving the permanent of an $n \times n$ zero-one matrix over the integers in time $2^{n - \Omega(n/\log^{2/\epsilon-2} n)}$.*

We note that such fast algorithms for $\# \text{OV}$ would already disprove the so-called Super Strong ETH, that k -CNFSAT on n variables has no $2^{n - n/o(k)}$ -time algorithm, by the reduction to OV by Williams [26] after sparsification [17]. The present result merely adds to the list of consequences of faster algorithms for $\# \text{OV}$.

Methodology and Organization of the Paper. The key methodological contribution underlying our main algorithmic result (Theorem 1) is a novel additive analog of the Chinese Remainder Theorem (Lemma 5 developed in Sect. 2 independently of the application), which enables us to recover the number of pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with $\langle x, y \rangle = t$ from counts of pairs (x, y) satisfying $\langle x, y \rangle \equiv r \pmod{p}$ for multiple small primes p and residues $r \in \{0, 1, \dots, p-1\}$. In particular, the crux of the algorithmic speedup lies in the observation that to recover the count associated with a target $0 \leq t \leq d$, primes up to roughly \sqrt{d} suffice by Lemma 5. To obtain the counts of pairs in each residue class r modulo p , we employ the polynomial method with modulus-amplifying polynomials of Beigel and Tarui [9] to accommodate the counts under a prime-power modulus, with fast rectangular matrix multiplication of Coppersmith [16] as the key subroutine implementing the count; this latter part of the algorithm design developed in Sect. 3 follows well-known techniques in fine-grained algorithm design (e.g. [3]). Similarly, the randomized algorithm design in Theorem 2 follows by a minor adaptation of the probabilistic-polynomial techniques of Alman and Williams [6] to a counting context; a proof is relegated to Sect. 4.

Our two lower-bound reductions, Theorem 3 and Theorem 4, rely on reducing an $m \times m$ integer permanent first via the Chinese Remainder Theorem into permanents modulo multiple primes p with $p \leq m \ln m$, and then using algebraic splitting via Ryser’s formula [22] to obtain short-vector instances of $\# \text{INNERPRODUCT}$ and $\# \text{OV}$, respectively. For $\# \text{INNERPRODUCT}$ and Theorem 3, the split employs a novel discrete-logarithm version of Ryser’s formula modulo p to arrive at two collections of vectors whose counts of pairs with specific inner products enable recovery of the permanent modulo p ; the proof is presented in Sect. 5. For $\# \text{OV}$ and Theorem 4, the split analogously employs Ryser’s formula modulo p but with a more intricate vector-coding of group residues modulo p to obtain the desired correspondence with counts of pairs of orthogonal vectors; we relegate the proof to Sect. 6.

Related Work and Further Applications. As regards exact and approximate inner products, Abboud, Williams, and Yu [1] used the polynomial method to construct a randomized subquadratic time algorithm for OV. Chan and Williams [14] derandomized the algorithm and showed that it could also solve the counting problem $\# \text{OV}$. The first result that addressed an inner product different from zero, was the randomized algorithm for minimum Hamming weight pair by Alman and Williams [6]. Subsequently, Alman, Chan, and Williams [3] found an even faster randomized as well as a deterministic subquadratic algorithm matching [6].

A number of studies address approximate versions of inner-product counting in subquadratic time, such as the detection of outlier correlations and offline computation of approximate nearest neighbors, including Valiant [23], Karppa, Kaski, and Kohonen [18], Alman [2], and Alman, Chan, and Williams [4]. All the algorithms use fast rectangular matrix multiplication.

As regards permanents, Bax and Franklin presented a randomised $2^{n-\Omega(n^{1/3}/\log n)}$ expected time algorithm for the 0/1-matrix permanent [8]. Björklund [10] derived a faster and deterministic $2^{n-\Omega(\sqrt{n/\log n})}$ time algorithm. The algorithm was subsequently improved to a deterministic $2^{n-\Omega(\sqrt{n/\log \log n})}$ time algorithm by Björklund, Kaski, and Williams [12].

For the computation of an integer matrix permanent modulo a prime power $p^{\lambda n/p}$ for any constant $\lambda < 1$, Björklund, Husfeldt, and Lyckberg [11] derived a $2^{n-\Omega(n/(p \log p))}$ time algorithm. For the computation of a matrix permanent over an arbitrary ring R on r elements, Björklund and Williams [13] gave a deterministic $2^{n-\Omega(\frac{n}{r})}$ time algorithm.

The problem #INNERPRODUCT has various applications in combinatorial algorithms. For example, it can be used to count the satisfying assignments to a SYM \circ AND formula (cf. Sect. 7), or compute the weight enumerator polynomial of a linear code (cf. Sect. 7). We would also like to highlight a recent application of deterministic algorithms for #OV [14] in efficient construction of rigid matrices [5].

2 Reconstruction from Sum-Aggregates by Prime Residue

This section develops the main methodological contribution of this work. Namely, we show that a complex-valued function $f : D \rightarrow \mathbb{C}$ can be reconstructed from its sum-aggregates by prime residue when the domain D is a prefix of the set of nonnegative integers. In essence, reconstruction of a function from its sum-aggregates can be viewed as an *additive* analog of the Chinese Remainder Theorem; that is, we obtain reconstruction up to the *sum* of the prime moduli – in the precise sense of (3) below – whereas the Chinese Remainder Theorem enables reconstruction up to the product of the moduli.²

In our application of counting pairs of vectors by inner product, we let f be a counting function such that $f(\ell)$ counts the number of pairs $(x, y) \in \mathcal{A} \times \mathcal{B}$ with $\langle x, y \rangle = \ell$. Reconstruction from sum-aggregates then enables us to recover f by counting the number of pairs (x, y) with $\langle x, y \rangle \equiv r \pmod{p}$ for small primes p and residues $r \in \{0, 1, \dots, p-1\}$; we postpone the details of this application to Sect. 3 and first proceed to study reconstructibility.

Sum-Aggregation by Prime Residue. Let p_1, p_2, \dots, p_m be distinct prime numbers and let

$$D \subseteq \{0, 1, \dots, s_m - 1\} \quad \text{where} \quad s_m = 1 + \sum_{b=1}^m (p_b - 1). \quad (3)$$

Letting f_ℓ be shorthand for $f(\ell)$, we show that we can recover f from the sequence of its *sum-aggregates*

$$F_{br} = \sum_{\substack{\ell \in D \\ \ell \equiv r \pmod{p_b}}} f_\ell \quad \text{for each residue } r \in \{0, 1, \dots, p_b - 1\} \text{ and each } b \in \{1, 2, \dots, m\}. \quad (4)$$

² Here it should be noted that the scope of the classical Chinese Remainder Theorem is also somewhat more restricted than our present setting; indeed, the Chinese Remainder Theorem does not enable the reconstruction of an arbitrary function f but rather is restricted to reconstruction in the case when f is known to vanish in all but one point of D . We also note that the present additive setting can be captured algebraically through generalized Chinese remaindering in polynomial rings; here we refer to the alternative proof of Lemma 5 at the end of this section.

To start with, we observe that this sequence is linearly dependent. Indeed, define

$$F_{01} = \sum_{\ell \in D} f_\ell \quad (5)$$

and observe that for each $b \in \{1, 2, \dots, m\}$ we have the linear relation $F_{01} = \sum_{r=0}^{p_b-1} F_{br}$.

To obtain an equivalent and – as we will shortly show – linearly independent sequence, take the sequence formed by the sum F_{01} followed by F_{br} for each *nonzero* residue $r \in \{1, 2, \dots, p_b - 1\}$ and each $b \in \{1, 2, \dots, m\}$. Let us write F for this sequence of length s_m . By extending the domain of the function f with zero-values $f_\ell = 0$ as needed, we can also assume that $D = \{0, 1, \dots, s_m - 1\}$ in what follows.

Sum-Aggregation as a Linear System – Modular Tomography. Let us now study reconstruction of f from F . From (4) and (5) we observe that to reconstruct f from F it suffices to solve the linear system

$$F = Af, \quad (6)$$

where A is the $s_m \times s_m$ *nonzero residue aggregation matrix* whose entries are defined for all $b \in \{0, 1, 2, \dots, m\}$, $i \in \{1, 2, \dots, p_b - 1\}$, and $\ell \in \{0, 1, \dots, s_m - 1\}$ by the rule

$$A_{bi,\ell} = \begin{cases} 1 & \text{if } b = 0; \\ 1 & \text{if } b \geq 1 \text{ and } i \equiv \ell \pmod{p_b}; \\ 0 & \text{if } b \geq 1 \text{ and } i \not\equiv \ell \pmod{p_b}, \end{cases} \quad (7)$$

where we have assumed for convenience that $p_0 = 2$. Indeed, we readily verify from (4), (5), and (7) that

$$F_{bi} = \sum_{\ell=0}^{s_m-1} A_{bi,\ell} f_\ell$$

holds for each $b \in \{0, 1, \dots, m\}$ and $i \in \{0, 1, \dots, p_b - 1\}$. When we want to stress the m selected primes, we write A^{p_1, p_2, \dots, p_m} for the matrix A .

The row-banded structure given by (7) is perhaps easiest illustrated with a small example. Below we display the matrix A for the primes $p_1 = 2$, $p_2 = 3$, and $p_3 = 5$:

$$A^{2,3,5} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (8)$$

Observe in particular that the first band $b = 0$ corresponds to the sum (5) and the subsequent bands $b \in \{1, 2, \dots, m\}$ each correspond to one of the primes p_1, p_2, \dots, p_m so that the $p_b - 1$ rows inside each band correspond to the sum-aggregates (4) of the $p_b - 1$ *nonzero* residue classes modulo p_b .

Our main technical lemma establishes that the matrix A is invertible, thus enabling reconstruction of f from F . Or, what is the same, we can recover f by *modular tomography* from the sequence F of *tomographs*; indeed, from a tomographic standpoint we can by (7) and (8) view each $b \in \{1, 2, \dots, m\}$ as corresponding to a bundle of pairwise disjoint parallel lines aggregating the data f , with $i \in \{1, 2, \dots, p_b - 1\}$ indexing the lines in a bundle.

► **Lemma 5** (Reconstruction from Sum-Aggregates by Prime Residue). *The nonzero residue aggregation matrix A^{p_1, p_2, \dots, p_m} is invertible whenever p_1, p_2, \dots, p_m are distinct primes.*

The key idea in the proof is to decompose A^{p_1, p_2, \dots, p_m} over the complex numbers into the product of a near-block-diagonal matrix with near-Vandermonde blocks and a Vandermonde matrix, both of which are then shown to have nonzero determinant. We will also present an alternative proof using elementary commutative algebra.

Proof of the Reconstruction Lemma. We now prove Lemma 5. We show that for distinct primes p_1, p_2, \dots, p_m the matrix $A = A^{p_1, p_2, \dots, p_m}$ is invertible over rational numbers. Our strategy is to show that $A = UV$ for two complex matrices U and V that both have nonzero determinant. Indeed, the near-cyclic banded structure of A suggests that one should pursue a decomposition in terms of block-structured near-Vandermonde matrices. Let us first define the matrices U and V , then present a small example, and then complete the proof.

We will use the following standard facts about complex roots of unity. For a positive integer N , let us write $\omega_N = \exp\left(\frac{2\pi\Im}{N}\right)$, where $\Im = \sqrt{-1}$ is the imaginary unit. For all $m \in \mathbb{Z}$ we have

$$\frac{1}{N} \sum_{j=0}^{N-1} \omega_N^{jm} = \begin{cases} 1 & \text{if } m \equiv 0 \pmod{N}; \\ 0 & \text{if } m \not\equiv 0 \pmod{N}. \end{cases} \quad (9)$$

The matrix U will use a $(m+1) \times (m+1)$ block structure that is similar to the $(m+1)$ -band structure of A , but now the structure is used both for rows and columns. Again for convenience we assume $p_0 = 2$. The matrix U is defined for all $b \in \{0, 1, 2, \dots, m\}$, $i \in \{1, 2, \dots, p_b - 1\}$, $d \in \{0, 1, \dots, m\}$, and $k \in \{1, 2, \dots, p_d - 1\}$ by the rule

$$U_{bi, dk} = \begin{cases} 1 & \text{if } d = 0 \text{ and } b = 0; \\ \frac{1}{p_b} & \text{if } d = 0 \text{ and } b \geq 1; \\ 0 & \text{if } d \geq 1 \text{ and } b \neq d; \\ \frac{1}{p_b} \omega_{p_b}^{-ik} & \text{if } d \geq 1 \text{ and } b = d. \end{cases} \quad (10)$$

The matrix V is a Vandermonde matrix with $(m+1)$ -banded structure defined for all $d \in \{0, 1, \dots, m\}$, $k \in \{1, 2, \dots, p_d - 1\}$, and $\ell \in \{0, 1, \dots, s_m - 1\}$ by the rule

$$V_{dk, \ell} = \begin{cases} 1 & \text{if } d = 0; \\ \omega_{p_d}^{k\ell} & \text{if } d \geq 1. \end{cases} \quad (11)$$

Before proceeding with the proof that $A = UV$, let us present an example for the primes $p_1 = 2$, $p_2 = 3$, and $p_3 = 5$; recall (8). We have

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}_{A^{2,3,5}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2}\omega^{-1.1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3}\omega^{-1.1} & \frac{1}{3}\omega^{-1.2} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3}\omega^{-2.1} & \frac{1}{3}\omega^{-2.2} & 0 & 0 & 0 & 0 \\ \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5}\omega^{-1.1} & \frac{1}{5}\omega^{-1.2} & \frac{1}{5}\omega^{-1.3} & \frac{1}{5}\omega^{-1.4} \\ \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5}\omega^{-2.1} & \frac{1}{5}\omega^{-2.2} & \frac{1}{5}\omega^{-2.3} & \frac{1}{5}\omega^{-2.4} \\ \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5}\omega^{-3.1} & \frac{1}{5}\omega^{-3.2} & \frac{1}{5}\omega^{-3.3} & \frac{1}{5}\omega^{-3.4} \\ \frac{1}{5} & 0 & 0 & 0 & \frac{1}{5}\omega^{-4.1} & \frac{1}{5}\omega^{-4.2} & \frac{1}{5}\omega^{-4.3} & \frac{1}{5}\omega^{-4.4} \end{bmatrix}_{U^{2,3,5}} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \omega_{2,0}^{-1} & \omega_{2,1}^{-1} & \omega_{2,2}^{-1} & \omega_{2,3}^{-1} & \omega_{2,4}^{-1} & \omega_{2,5}^{-1} & \omega_{2,6}^{-1} & \omega_{2,7}^{-1} \\ \omega_{3,0}^{-1} & \omega_{3,1}^{-1} & \omega_{3,2}^{-1} & \omega_{3,3}^{-1} & \omega_{3,4}^{-1} & \omega_{3,5}^{-1} & \omega_{3,6}^{-1} & \omega_{3,7}^{-1} \\ \omega_{3,0}^{-1} & \omega_{3,1}^{-1} & \omega_{3,2}^{-1} & \omega_{3,3}^{-1} & \omega_{3,4}^{-1} & \omega_{3,5}^{-1} & \omega_{3,6}^{-1} & \omega_{3,7}^{-1} \\ \omega_{5,0}^{-1} & \omega_{5,1}^{-1} & \omega_{5,2}^{-1} & \omega_{5,3}^{-1} & \omega_{5,4}^{-1} & \omega_{5,5}^{-1} & \omega_{5,6}^{-1} & \omega_{5,7}^{-1} \\ \omega_{5,0}^{-1} & \omega_{5,1}^{-1} & \omega_{5,2}^{-1} & \omega_{5,3}^{-1} & \omega_{5,4}^{-1} & \omega_{5,5}^{-1} & \omega_{5,6}^{-1} & \omega_{5,7}^{-1} \\ \omega_{5,0}^{-1} & \omega_{5,1}^{-1} & \omega_{5,2}^{-1} & \omega_{5,3}^{-1} & \omega_{5,4}^{-1} & \omega_{5,5}^{-1} & \omega_{5,6}^{-1} & \omega_{5,7}^{-1} \\ \omega_{5,0}^{-1} & \omega_{5,1}^{-1} & \omega_{5,2}^{-1} & \omega_{5,3}^{-1} & \omega_{5,4}^{-1} & \omega_{5,5}^{-1} & \omega_{5,6}^{-1} & \omega_{5,7}^{-1} \end{bmatrix}_{V^{2,3,5}}. \quad (12)$$

The main technical aspect of the proof that $A = UV$ is to partition the index $\ell \in \{0, 1, \dots, s_m - 1\}$ to the $m + 1$ bands. Towards this end, define for each $c \in \{0, 1, \dots, m\}$ the prefix-sum

$$\underline{s}_c = \begin{cases} 0 & \text{if } c = 0; \\ 1 + \sum_{\ell=1}^{c-1} (p_c - 1) & \text{if } c \geq 1. \end{cases}$$

In particular, for every $\ell \in \{0, 1, \dots, s_m - 1\}$, we observe that there exist unique $c \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, p_c - 1\}$ such that

$$\ell = j - 1 + \underline{s}_c. \quad (13)$$

We are now ready to show that $A = UV$. Let $b \in \{0, 1, \dots, m\}$, $i \in \{0, 1, \dots, p_b - 1\}$, and $\ell \in \{0, 1, \dots, s_k - 1\}$ be arbitrary. Let $c \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, p_c - 1\}$ be uniquely determined from ℓ by (13). From (10), (11), (9), and (7) we observe that

$$\begin{aligned} & \sum_{d=0}^m \sum_{k=0}^{p_d-1} U_{bi,dk} V_{dk,\ell} = \\ & = \begin{cases} 1 & \text{if } b = 0; \\ \frac{1}{p_b} \left(1 + \sum_{k=1}^{p_b-1} \omega_{p_b}^{-ik+k(j-1+\underline{s}_b)} \right) & \text{if } b \geq 1 \text{ and } i \equiv j - 1 + \underline{s}_b = \ell \pmod{p_b}; \\ \frac{1}{p_b} \sum_{k=0}^{p_b-1} \omega_{p_b}^{k(j-i-1+\underline{s}_b)} = 1 & \text{if } b \geq 1 \text{ and } i \equiv j - 1 + \underline{s}_b = \ell \pmod{p_b}; \\ \frac{1}{p_b} \left(1 + \sum_{k=1}^{p_b-1} \omega_{p_b}^{-ik+k(j-1+\underline{s}_b)} \right) & \text{if } b \geq 1 \text{ and } i \not\equiv j - 1 + \underline{s}_b = \ell \pmod{p_b}; \\ \frac{1}{p_b} \sum_{k=0}^{p_b-1} \omega_{p_b}^{k(j-i-1+\underline{s}_b)} = 0 & \text{if } b \geq 1 \text{ and } i \not\equiv j - 1 + \underline{s}_b = \ell \pmod{p_b}. \end{cases} \\ & = A_{bi,\ell}. \end{aligned}$$

Thus, $A = UV$ holds. It remains to show that both matrices U and V have nonzero determinant over the complex numbers. Starting with the Vandermonde matrix V , let $\nu_0 = 1$ and $\nu_\ell = \omega_{p_c}^j$ for $\ell \in \{1, 2, \dots, s_m - 1\}$, where $c \in \{0, 1, \dots, m\}$ and $j \in \{1, 2, \dots, p_c - 1\}$ are uniquely determined from ℓ by (13). In particular, we observe that V is a Vandermonde matrix with $D = s_m - 1$ and $V = [\nu_i^j]_{i=0,1,\dots,D}^{j=0,1,\dots,D}$. The Vandermonde determinant formula

thus gives $\det V = \sum_{0 \leq k < \ell \leq D} (\nu_\ell - \nu_k)$. Furthermore, this determinant is nonzero because p_1, p_2, \dots, p_m are distinct primes and thus $\nu_0, \nu_1, \dots, \nu_D$ are distinct. Next, let us consider the matrix U defined by (10). At this point it may be useful to revisit the structure of U via the example (12). We observe that the block-diagonal of U with $b = c \geq 1$ consists of matrices that each decompose into the product of a $(p_b - 1) \times (p_b - 1)$ diagonal matrix with diagonal entries $\frac{1}{p_b} \omega_{p_b}^{-i}$ for $i \in \{1, 2, \dots, p_b - 1\}$ and a $(p_b - 1) \times (p_b - 1)$ Vandermonde matrix with a nonzero determinant since $\omega_{p_b}^{-i}$ for $i \in \{1, 2, \dots, p_b - 1\}$ are distinct. Thus, since the determinant of U is the product of the determinants of the block-matrices on the diagonal, each of which is nonzero, the determinant of U is nonzero. It follows that A is invertible and thus given F we can solve for f via (6). This completes the proof of Lemma 5. ◀

An Alternative Proof. Let us sketch an alternative algebraic proof for Lemma 5. Let us view $f = \sum_{\ell=0}^{s_m-1} f_\ell x^\ell \in \mathbb{C}[x]$ as a univariate polynomial in the indeterminate x over the complex numbers. For each $b \in \{1, 2, \dots, m\}$, we observe that the polynomial remainder of f divided by $x^{p_b} - 1$ satisfies $f \bmod (x^{p_b} - 1) = \sum_{i=0}^{p_b-1} F_{bi} x^i \in \mathbb{C}[x]$. Furthermore, for distinct $b, c \in \{1, 2, \dots, m\}$ we have that the sets of complex roots of $x^{p_b} - 1$ and $x^{p_c} - 1$ are disjoint apart from the common root $x = 1$; that is, $(x^{p_b} - 1)/(x - 1)$ and $(x^{p_c} - 1)/(x - 1)$ are coprime polynomials. Thus, by (3) and the Chinese Remainder Theorem in $\mathbb{C}[x]$, we can recover f from the sequence of remainders $f \bmod (x^{p_b} - 1)$ for $b \in \{1, 2, \dots, m\}$; cf. von zur Gathen and Gerhard [25] for the algebraic background and associated near-linear-time algorithmic toolbox. ◀

3 Counting Pairs of Zero-One Vectors by Inner Product

This section documents our main algorithm and proves Theorem 1. Let κ be a parameter that satisfies, with foresight,

$$4 \leq \kappa \leq \frac{\log n}{(\log \log n)^4}. \quad (14)$$

Let $a^{(1)}, a^{(2)}, \dots, a^{(n)} \in \{0, 1\}^d$ and $b^{(1)}, b^{(2)}, \dots, b^{(n)} \in \{0, 1\}^d$ be given as input with $d \leq \kappa \log n$. We want to compute for each $t \in \{0, 1, \dots, d\}$ the count

$$f_t = |\{(i, j) \in \{1, 2, \dots, n\}^2 : \langle a^{(i)}, b^{(j)} \rangle = t\}|.$$

Our high-level approach will be to use Lemma 5 and (6) to solve for the counts f_0, f_1, \dots, f_d using as input counts that have been sum-aggregated by prime residue. More precisely, we will work with prime moduli p_1, p_2, \dots, p_m and develop an algorithm that computes, for given further input $p \in \{p_1, p_2, \dots, p_m\}$ and $r \in \{0, 1, \dots, p - 1\}$, the sum-aggregated count

$$F_{pr} = |\{(i, j) \in \{1, 2, \dots, n\}^2 : \langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}\}|.$$

The detailed choices for m and the primes p_1, p_2, \dots, p_m will be presented later.

The Residue-Indicator Polynomial. Assume p and r have been given. We will rely on the polynomial method, and accordingly we first build a standard polynomial that indicates the residue r modulo p in a pair of vectors.

Let $x = (x_1, x_2, \dots, x_d)$ and $y = (y_1, y_2, \dots, y_d)$ be two vectors of indeterminates. By Fermat's little theorem, the $2d$ -indeterminate polynomial

$$G_{p,r}(x, y) = 1 - \left(\sum_{k=1}^d x_k y_k - r \right)^{p-1} \quad (15)$$

29:10 Counting Short Vector Pairs by Inner Product and Relations to the Permanent

satisfies for all $i, j \in \{1, 2, \dots, n\}$ the indicator property

$$G_{p,r}(a^{(i)}, b^{(j)}) \equiv \begin{cases} 1 \pmod{p} & \text{if } \langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}; \\ 0 \pmod{p} & \text{if } \langle a^{(i)}, b^{(j)} \rangle \not\equiv r \pmod{p}. \end{cases} \quad (16)$$

We observe that $G_{p,r}$ has degree $2p - 2$.

Modulus Amplification for Zero-One Residues. To enable taking the sum of a large number of indicators, we make use of the *modulus amplifying polynomials* of Beigel and Tarui [9].

► **Theorem 6** (Modulus amplification; Beigel and Tarui [9]). *For $h \in \mathbb{Z}_{\geq 1}$, define the polynomial*

$$A_h(z) = 1 - (1 - z)^h \sum_{j=0}^{h-1} \binom{h+j-1}{j} z^j. \quad (17)$$

Then, for all $m \in \mathbb{Z}_{\geq 2}$ and $s \in \mathbb{Z}$, we have

- (i) $s \equiv 0 \pmod{m}$ implies $A_h(s) \equiv 0 \pmod{m^h}$, and
- (ii) $s \equiv 1 \pmod{m}$ implies $A_h(s) \equiv 1 \pmod{m^h}$.

We observe that A_h has degree $2h - 1$. Composing (17) and (15), we obtain the amplified residue-indicator polynomial

$$G_{p,r}^h(x, y) = A_h(G_{p,r}(x, y)). \quad (18)$$

From (16) and Theorem 6, we observe the amplified indicator property

$$G_{p,r}^h(a^{(i)}, b^{(j)}) \equiv \begin{cases} 1 \pmod{p^h} & \text{if } \langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}; \\ 0 \pmod{p^h} & \text{if } \langle a^{(i)}, b^{(j)} \rangle \not\equiv r \pmod{p}. \end{cases} \quad (19)$$

Furthermore, we observe that $G_{p,r}^h$ has degree $(2h - 1)(2p - 2)$.

Multilinear Reduct and Bounding the Number of Monomials. For a nonnegative integer e , define $\underline{e} = 0$ if $e = 0$ and $\underline{e} = 1$ if $e \geq 1$. For a monomial $x_1^{e_1} x_2^{e_2} \cdots x_d^{e_d} y_1^{f_1} y_2^{f_2} \cdots y_d^{f_d}$, define the *multilinear reduct* by

$$\underline{x_1^{e_1} x_2^{e_2} \cdots x_d^{e_d} y_1^{f_1} y_2^{f_2} \cdots y_d^{f_d}} = x_1^{\underline{e}_1} x_2^{\underline{e}_2} \cdots x_d^{\underline{e}_d} y_1^{\underline{f}_1} y_2^{\underline{f}_2} \cdots y_d^{\underline{f}_d}.$$

For a polynomial $Q(x, y)$, define the multilinear reduct $\underline{Q}(x, y)$ by taking the multilinear reduct of each monomial $Q(x, y)$ and simplifying. Since $a^{(i)}$ and $b^{(j)}$ are $\{0, 1\}$ -valued vectors, over the integers we have

$$Q(a^{(i)}, b^{(j)}) = \underline{Q}(a^{(i)}, b^{(j)}). \quad (20)$$

Furthermore, if Q has degree D , then \underline{Q} has at most $\sum_{j=0}^D \binom{2d}{j}$ monomials. In particular, we observe that $\underline{G}_{p,r}^h$ has at most $\sum_{j=0}^{4hp} \binom{2d}{j}$ monomials.

Split-Monomial Form of the Multilinear Reduct. Suppose that the multilinear reduct $\underline{G}_{p,r}^h(x, y)$ has exactly M monomials with the representation

$$\underline{G}_{p,r}^h(x, y) = \sum_{k=1}^M \gamma^{(k)} x_1^{e_1^{(k)}} x_2^{e_2^{(k)}} \cdots x_d^{e_d^{(k)}} y_1^{f_1^{(k)}} y_2^{f_2^{(k)}} \cdots y_d^{f_d^{(k)}}. \quad (21)$$

For $I, J \subseteq \{1, 2, \dots, n\}$ and $k \in \{1, 2, \dots, M\}$, define

$$L_{I,k} = \sum_{i \in I} (a_1^{(i)})^{e_1^{(k)}} (a_2^{(i)})^{e_2^{(k)}} \cdots (a_d^{(i)})^{e_d^{(k)}} \gamma^{(k)}, \quad R_{J,k} = \sum_{j \in J} (b_1^{(j)})^{f_1^{(k)}} (b_2^{(j)})^{f_2^{(k)}} \cdots (b_d^{(j)})^{f_d^{(k)}}. \quad (22)$$

From (22), (21), (20), and (19), we have

$$\begin{aligned} \sum_{k=1}^M L_{I,k} R_{J,k} &= \sum_{i \in I} \sum_{j \in J} \underline{G}_{p,r}^h(a^{(i)}, b^{(j)}) \\ &\equiv |\{(i, j) \in I \times J : \langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}\}| \pmod{p^h}. \end{aligned} \quad (23)$$

In particular, assuming that $|I||J| \leq p^h - 1$, from (23) it follows that $\sum_{k=1}^M L_{I,k} R_{J,k}$ computed modulo p^h recovers the number of pairs $(i, j) \in I \times J$ with $\langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}$.

We now move from deriving the polynomial and its properties to describing the algorithm.

Algorithm for the Prime-Residue Count. The algorithm will rely on (23) via fast rectangular matrix multiplication to count the number of pairs $(i, j) \in \{1, 2, \dots, n\}^2$ that satisfy $\langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}$.

The algorithm first computes the explicit M -monomial representation of the polynomial $\underline{G}_{p,r}^h$ in (21). More precisely, the algorithm evaluates (15), (17), and (18) in explicit monomial representation, taking multilinear reducts with respect to the variables $x_1, x_2, \dots, x_d, y_1, y_2, \dots, y_d$ whenever possible. This results in the set

$$\{(k, \gamma^{(k)}, e_1^{(k)}, e_2^{(k)}, \dots, e_d^{(k)}, f_1^{(k)}, f_2^{(k)}, \dots, f_d^{(k)}) : k \in \{1, 2, \dots, M\}\}. \quad (24)$$

Next, the algorithm constructs two rectangular matrices S and T , with the objective of making use of the following algorithm of Coppersmith [16].

► **Theorem 7** (Coppersmith [16]). *Given an $N \times \lfloor N^{0.17} \rfloor$ matrix S and an $\lfloor N^{0.17} \rfloor \times N$ matrix T as input, the matrix product ST over the integers can be computed in $O(N^2 \log^2 N)$ arithmetic operations.*

Towards this end, let g be a positive integer whose value we will fix later. Introduce two set partitions of $\{1, 2, \dots, n\}$ with cells

$$I_1, I_2, \dots, I_{\lceil n/g \rceil} \subseteq \{1, 2, \dots, n\} \quad \text{and} \quad J_1, J_2, \dots, J_{\lceil n/g \rceil} \subseteq \{1, 2, \dots, n\},$$

respectively, so that $|I_u| = g$ and $|J_v| = g$ for $u, v \in \{1, 2, \dots, \lceil n/g \rceil\}$. Indeed, we thus have $|I_u||J_v| \leq g^2$ for all $u, v \in \{1, 2, \dots, \lceil n/g \rceil\}$, so (23) applied to I_u and J_v modulo p^h recovers the number of pairs $(i, j) \in I_u \times J_v$ with $\langle a^{(i)}, b^{(j)} \rangle \equiv r \pmod{p}$, assuming that $g^2 \leq p^h - 1$, which will be justified by our eventual choice of g .

Now let $N = \lceil n/g \rceil$ and define the $N \times M$ and $M \times N$ matrices S and T by setting $S_{uk} = L_{I_u,k}$ and $T_{kv} = R_{J_v,k}$ for $u, v \in \{1, 2, \dots, \lceil n/g \rceil\}$ and $k \in \{1, 2, \dots, M\}$. Concretely, the algorithm computes S and T from the given input one entry at a time using the computed monomial list (24) and the formulas (22) for $I = I_u$ and $J = J_v$ for each $u, v \in \{1, 2, \dots, \lceil n/g \rceil\}$ and $k = 1, 2, \dots, M$. The algorithm then multiplies S and T to obtain the product matrix ST modulo p^h , where we assume that each entry of ST is reduced to $\{0, 1, \dots, p^h - 1\}$. Finally, the algorithm outputs the sum $F_{pr} = \sum_{u=1}^{\lceil n/g \rceil} \sum_{v=1}^{\lceil n/g \rceil} (ST)_{uv}$.

29:12 Counting Short Vector Pairs by Inner Product and Relations to the Permanent

Parameterizing the Algorithm. Let us now fix precise values for the parameters g , h , and p that were left open. First, to apply the algorithm in Theorem 7 to the matrices S and T , we need $M \leq N^{0.17} = \lceil n/g \rceil^{0.17}$. Subject to the assumption $g \leq n^{0.1}$ – to be justified later – it will be sufficient to show that $M \leq n^{0.15}$. We recall that $M \leq \sum_{j=0}^{4hp} \binom{2d}{j}$ and $d \leq \kappa \log n$. With foresight, let us set

$$\beta_\kappa = \frac{K}{\log \kappa}, \quad (25)$$

where $K > 0$ is a small constant that will be fixed later. In particular, since $\kappa \geq 4$, we have the upper bound

$$\beta_\kappa \log \frac{\kappa}{\beta_\kappa} = K - \frac{K \log K}{\kappa} + \frac{K \log \log \kappa}{\kappa} \leq \frac{5K}{4} \quad (26)$$

which we can make an arbitrarily small and positive by choosing a small enough K . Let us assume – to be justified later – that $p = o(\beta_\kappa \log n)$. Taking

$$h = \left\lfloor \beta_\kappa \frac{\log n}{p} \right\rfloor \quad (27)$$

we have, for all large enough n ,

$$\begin{aligned} M &\leq \sum_{j=0}^{4hp} \binom{2d}{j} \leq 4hp \binom{2d}{4hp} \leq 4hp \left(\frac{2ed}{4hp} \right)^{4hp} \\ &\leq 4 \left(\beta_\kappa \frac{\log n}{p} + 1 \right) p \left(\frac{2e\kappa \log n}{4(\beta_\kappa \frac{\log n}{p} - 1)p} \right)^{4(\beta_\kappa \frac{\log n}{p} + 1)p} \\ &= 4(\beta_\kappa \log n + p) \left(\frac{2e\kappa \log n}{4(\beta_\kappa \log n - p)} \right)^{4(\beta_\kappa \log n + p)} \\ &\leq (5\beta_\kappa \log n) \left(\frac{2e\kappa}{3\beta_\kappa} \right)^{5\beta_\kappa \log n} \leq n^{0.15}, \end{aligned} \quad (28)$$

where the last inequality follows by (26) and choosing K small enough. Thus, Theorem 7 applies, subject to the assumptions $g \leq n^{0.1}$, $g^2 \leq p^h - 1$, and $p = o(\beta_\kappa \log n)$, which still need to be established. Before this, we digress to further preliminaries to enable reconstruction.

Preliminaries on Asymptotics of Primes. In what follows let us write p_j for the j th prime number with $j = 1, 2, \dots$; that is, $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and so forth. Asymptotically, from the Prime Number Theorem we have $p_m \sim m \ln m$ (e.g. Rosser and Schoenfeld [21]), and the sum of the first m primes satisfies $\sum_{j=1}^m p_j \sim \frac{1}{2}m^2 \ln m$ (cf. Bach and Shallit [7]), where we write $f(m) \sim g(m)$ if $\lim_{m \rightarrow \infty} \frac{f(m)}{g(m)} = 1$.

When evaluated for the first m primes, the reconstruction parameter (3) thus satisfies

$$s_m = 1 + \sum_{j=1}^m (p_j - 1) \sim \frac{p_m^2}{2 \ln p_m}. \quad (29)$$

We are now ready to continue parameterization of the algorithm.

Further Parameterization of the Algorithm. Let m be a positive integer whose value will be fixed shortly. The algorithm will work with p_1, p_2, \dots, p_m , the first m prime numbers. To reconstruct inner products of length- d zero-one vectors over the integers, we need $d + 1 \leq s_m$, which for $d \leq \kappa \log n$ and (29) means

$$\frac{p_m^2}{2 \ln p_m} \sim \kappa \log n.$$

From Bertrand's postulate it thus follows that choosing the least m so that

$$2\sqrt{\kappa(\ln n) \ln \ln n} \leq p_m \leq 4\sqrt{\kappa(\ln n) \ln \ln n} \quad (30)$$

implies that we have $d + 1 \leq s_m$ for all large enough n and thus reconstruction is feasible. The choice (30) also justifies our earlier assumption made in the context of (27) and (28) that $p_j = o(\beta_\kappa \log n)$ for all $j \in \{1, 2, \dots, m\}$; indeed, from (14) and (25), we have

$$\beta_\kappa \log n = \frac{K \log n}{\log \kappa}$$

and thus from (14) and (30) we observe that

$$\frac{p_j}{\beta_\kappa \log n} \leq \frac{4\kappa^{1/2}(\log \kappa)(\ln n)^{1/2}(\ln \ln n)^{1/2}}{K \log n} = o(1).$$

Let us next choose the parameter g . Using $p_j = o(\beta_\kappa \log n)$ again, we have

$$p_j^{h_j} = p_j^{\left\lfloor \beta_\kappa \frac{\log n}{p_j} \right\rfloor} \geq p_j^{\beta_\kappa \frac{\log n}{p_j} - 1} \geq p_j^{\beta_\kappa \frac{\log n}{2p_j}} = 2^{\beta_\kappa \frac{\log n}{2p_j} \log p_j} = n^{\beta_\kappa \frac{\log p_j}{2p_j}}.$$

Since $p_1 < p_2 < \dots < p_m$, for $j \in \{1, 2, \dots, m\}$ thus $p_j^{h_j} \geq n^{\beta_\kappa \frac{\log p_m}{2p_m}}$. It follows that choosing

$$g = \left\lfloor \sqrt{n^{\beta_\kappa \frac{\log p_m}{2p_m}} - 1} \right\rfloor \quad (31)$$

justifies our assumption $g^2 \leq p_j^{h_j} - 1$ for $j \in \{1, 2, \dots, m\}$. The final assumption $g \leq n^{0.1}$ is justified by observing that $\frac{\log p_m}{2p_m}$ is a decreasing function of m and observing that $\beta_\kappa = o(1)$ by (14) and (25). The algorithm is now parameterized. Let us next analyse its running time.

Running Time Analysis. First, let us seek control on N as a function of n . From (30) and (31), we have

$$g \geq \sqrt{n^{\beta_\kappa \frac{2 \log 2 + \log \kappa + \log \ln n + \log \ln \ln n}{16 \sqrt{\kappa \ln n \ln \ln n}}} - 1} - 1.$$

This together with (14) gives us the crude lower bound

$$g = \exp\left(\Omega\left(\beta_\kappa \sqrt{\frac{(\ln n) \ln \ln n}{\kappa}}\right)\right).$$

We thus have

$$N^2 = \lceil n/g \rceil^2 = n^{2-\Omega\left(\beta_\kappa \sqrt{\frac{\ln \ln n}{\kappa \ln n}}\right)}.$$

Recalling (28), we observe that the time to compute the M -monomial list (24) can be bounded by $n^{0.31}$ because the algorithm is careful to take multilinear reducts and thus at no stage of evaluating (15), (17), and (18) the number of monomials increases above $(n^{0.15})^2 = n^{0.30}$. Since $\log p_j^{h_j} = h_j \log p_j = \lfloor \beta_\kappa \frac{\log n}{p_j} \rfloor \log p_j = O(\log n)$, the arithmetic over the integers and modulo $p_j^{h_j}$ for each $j = 1, 2, \dots, m$ runs in time polylogarithmic in n for each arithmetic operation executed by the algorithm. Because the algorithm in Theorem 7 runs in $O(N^2 \log^2 N)$ arithmetic operations, we observe that the polylogarithmic terms are subsumed by the asymptotic notation and the entire algorithm for computing F_{pr} for given $p \in \{p_1, p_2, \dots, p_m\}$ and $r \in \{0, 1, \dots, p-1\}$ runs in time

$$n^{2-\Omega\left(\beta_\kappa \sqrt{\frac{\log \log n}{\kappa \log n}}\right)} = n^{2-\Omega\left(\sqrt{\frac{\log \log n}{\kappa (\log \kappa)^2 \log n}}\right)}. \quad (32)$$

From (30) we observe that the required repeats for different p and r result in multiplicative polylogarithmic terms in n and are similarly subsumed to result in total running time of the form (32). This completes the proof of Theorem 1. \blacktriangleleft

4 A Faster Randomized Algorithm for #InnerProduct

This section sketches a proof for Theorem 2. We follow the algorithm outlined in Alman and Williams [6]. We note that by their Theorem 1.2, there are probabilistic polynomials over any field with error ϵ of degree $O(\sqrt{n \log(1/\epsilon)})$. In their Theorem 4.2, they have a probabilistic OR-construction that takes the disjunction of a random set of s^2 pairs of vector inner products as

$$q(x_1, y_1, x_2, y_2, \dots, x_s, y_s) = 1 + \prod_{k=1}^2 \left(1 + \sum_{(i,j) \in R_k} (1 + p(x_{i,1} + y_{i,1}, x_{i,2} + y_{j,2}, \dots, x_{i,s} + y_{j,s})) \right),$$

where p is a probabilistic threshold polynomial over \mathbb{F}_2 of error $\epsilon = s^{-3}$, and $R_k \subseteq [s]^2$ for $k = 1, 2$ are sieve subsets drawn uniformly at random. This construction can be used to detect w.h.p. if there is a pair in the s^2 -sized batch whose difference Hamming weight is less than the threshold. By repeated computations with new p 's and R_k 's, a majority vote for the batch can be chosen as the correct answer, again w.h.p. for all batches.

We implement the following change of q to get an #INNERPRODUCT algorithm. We take p to be a probabilistic polynomial of error $\epsilon = s^{-3}$ for the symmetric function $\llbracket \sum_{i=1}^n z_i = t \rrbracket$, over a field of characteristic $> s^2$. We then construct q as

$$q(x_1, y_1, x_2, y_2, \dots, x_s, y_s) = \sum_{(i,j) \in [s]^2} p(x_{i,1}y_{i,1}, x_{i,2}y_{j,2}, \dots, x_{i,s}y_{j,s}). \quad (33)$$

Since the characteristic of the field is large enough, (33) is equal to the number of pairs in the s^2 -sized batch that has inner product equal to t with probability at least $1 - s^2\epsilon \geq 1 - \frac{1}{s}$, a similar bound on the probability as in Theorem 4.2. Also, the degree of the polynomials is only a factor 2 larger. As with the original algorithm, if we repeat this enough times and take the majority in each batch, we get the correct number of pairs with t as inner product in all batches. By summing these final majority numbers over the integers, we obtain the output. We note that the parameters of the error and the degree has only changed by a constant, and hence that all calculations of the running time and the error bound of the original algorithm carries through also for our modification of the algorithm. This completes the proof sketch. \blacktriangleleft

5 A Lower Bound for #InnerProduct via Zero-One Permanents

This section proves Theorem 3; the proof of Theorem 4 is presented in Sect. 6.

Throughout this section we let M be an $n \times n$ matrix with entries $m_{ij} \in \{0, 1\}$ for $i, j \in \{1, 2, \dots, n\}$. For convenience, let us write $[n] = \{1, 2, \dots, n\}$. Recalling Ryser's formula, we have

$$\text{per } M = (-1)^n \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{i \in [n]} \sum_{j \in S} m_{ij}. \quad (34)$$

First Reduction: Chinese Remaindering. Since it is immediate that $0 \leq \text{per } M \leq n!$, it suffices to compute the permanent modulo small primes p and then assemble the result over the integers via the Chinese Remainder Theorem. Let us first state and prove a crude upper bound on the size of the primes needed. For a positive integer m , let us write $m\#$ for the product of all prime numbers at most m .

► **Lemma 8.** *For all sufficiently large n , we have $n! \leq (n \ln n)\#$.*

Proof. Recall that for a positive integer m we write $m\#$ for the product of all prime numbers at most m . For $m \geq 563$, we have $\ln m\# > m(1 - \frac{1}{2 \ln m})$; cf. Rosser and Schoenfeld [21]. For the factorial function, for $n \geq 1$, Robbins [20] shows that $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\alpha_n}$ with $\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$, which gives us the comparatively crude upper bound $\ln n! < (n + \frac{1}{2}) \ln n - n + 1$ for $n \geq 1$. We want $\ln n! < \ln m\#$. Accordingly, it suffices to have $m \geq 563$ and $(n + \frac{1}{2}) \ln n - n + 1 < m(1 - \frac{1}{2 \ln m})$. It is immediate that $m \geq n \ln n$ suffices for $m \geq 563$, which completes the proof. ◀

Thus, it suffices to work with all primes p with $p \leq n \ln n$ in what follows.

A Reduction from Zero-One Permanent to #InnerProduct. This section starts our work towards Theorem 3 without yet parameterizing the reduction in detail. Let a prime $2 \leq p \leq n \ln n$ be given. We seek to compute $\text{per } M$ modulo p . Fix a primitive root $g \in \{1, 2, \dots, p-1\}$ modulo p . For an integer a with $a \not\equiv 0 \pmod{p}$, let us write $\text{dlog}_{p,g} a$ for the *discrete logarithm* of a relative to g modulo p . That is, $\text{dlog}_{p,g} a$ is the unique integer in $\{0, 1, \dots, p-2\}$ that satisfies $g^{\text{dlog}_{p,g} a} \equiv a \pmod{p}$. Working modulo p and collecting the outer sum in (34) by the sign $\sigma \in \{-1, 1\}$ and the *nonzero* products by their discrete logarithm, we have

$$\text{per } M \equiv (-1)^n \sum_{e=0}^{p-2} g^e (w_1^{(e)} - w_{-1}^{(e)}) \pmod{p},$$

where

$$w_\sigma^{(e)} = \left| \left\{ S \subseteq [n] : (-1)^{|S|} = \sigma \text{ and } \text{dlog}_{p,g} \prod_{i \in [n]} \sum_{j \in S} m_{ij} \equiv e \pmod{p-1} \right\} \right|$$

for $\sigma \in \{-1, 1\}$ and $e \in \{0, 1, \dots, p-2\}$. Thus, to compute $\text{per } M$ modulo p it suffices to compute the coefficients $w_\sigma^{(e)}$. Towards this end, suppose that $n \geq 4$ is even and let

$$L = \{1, 2, \dots, n/2\} \quad \text{and} \quad R = \{n/2, n/2 + 1, \dots, n\}.$$

For $\sigma_L, \sigma_R \in \{1, -1\}$, let

$$w_{\sigma_L, \sigma_R}^{(e)} = \left| \left\{ S \subseteq [n] : (-1)^{|S \cap L|} = \sigma_L, (-1)^{|S \cap R|} = \sigma_R \text{ and } \text{dlog}_{p,g} \prod_{i \in [n]} \sum_{j \in S} m_{ij} \equiv e \pmod{p-1} \right\} \right|$$

29:16 Counting Short Vector Pairs by Inner Product and Relations to the Permanent

Clearly $w_{\sigma}^{(e)} = \sum_{\sigma_L, \sigma_R \in \{-1, 1\}} w_{\sigma_L, \sigma_R}^{(e)}$, so it suffices to focus on computing $w_{\sigma_L, \sigma_R}^{(e)}$ in what follows. Define the set families

$$\mathcal{L}_{\sigma_L} = \{A \subseteq L : (-1)^{|A|} = \sigma_L\} \quad \text{and} \quad \mathcal{R}_{\sigma_R} = \{B \subseteq R : (-1)^{|B|} = \sigma_R\}$$

with $|\mathcal{L}_{\sigma_L}| = |\mathcal{R}_{\sigma_R}| = 2^{n/2-1}$. Next we will define two families of length- d zero-one vectors whose pair counts by inner product will enable us to recover the coefficients $w_{\sigma_L, \sigma_R}^{(e)}$. The structure of the vectors will be slightly elaborate, so let us first define an index set D for indexing the $|D| = d$ dimensions. Let

$$D = \{(i, \ell, r, k) \in [n] \times \{0, 1, \dots, p-1\} \times \{0, 1, \dots, p-1\} \times [np] : \\ \ell + r \not\equiv 0 \pmod{p} \text{ implies } k \leq \text{dlog}_{p,g}(\ell + r)\}.$$

We have $d = n^2 p^2 + np(p-1)(p-2)/2 < n^4 (\ln n)^3$.

For $A \in \mathcal{L}_{\sigma_L}$ and $B \in \mathcal{R}_{\sigma_R}$, define the vectors $\lambda(A) \in \{0, 1\}^D$ and $\rho(B) \in \{0, 1\}^D$ for all $(i, \ell, r, k) \in D$ by the rules

$$\lambda(A)_{i\ell rk} = \begin{cases} 1 & \text{if } \ell \equiv \sum_{j \in A} m_{ij} \pmod{p}; \\ 0 & \text{otherwise;} \end{cases} \quad \text{and} \quad \rho(B)_{i\ell rk} = \begin{cases} 1 & \text{if } r \equiv \sum_{j \in B} m_{ij} \pmod{p}; \\ 0 & \text{otherwise.} \end{cases}$$

To study the inner product $\langle \lambda(A), \rho(B) \rangle$ it will be convenient to work with Iverson's bracket notation. Namely, for a logical proposition P , let

$$\llbracket P \rrbracket = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{if } P \text{ is false.} \end{cases}$$

Over the integers, we now have

$$\begin{aligned} \langle \lambda(A), \rho(B) \rangle &= \sum_{(i, \ell, r, k) \in D} \lambda(A)_{i\ell rk} \rho(B)_{i\ell rk} \\ &= \sum_{(i, \ell, r, k) \in D} \llbracket \ell \equiv \sum_{j \in A} m_{ij} \pmod{p} \rrbracket \llbracket r \equiv \sum_{j \in B} m_{ij} \pmod{p} \rrbracket \\ &= \sum_{i \in [n]} \sum_{\substack{\ell, r=0 \\ \ell+r \not\equiv 0 \pmod{p}}}^{p-1} \llbracket \ell \equiv \sum_{j \in A} m_{ij} \pmod{p} \rrbracket \llbracket r \equiv \sum_{j \in B} m_{ij} \pmod{p} \rrbracket \text{dlog}_{p,g}(\ell + r) \\ &\quad + \sum_{i \in [n]} \sum_{\ell=0}^{p-1} \llbracket \ell \equiv \sum_{j \in A} m_{ij} \pmod{p} \rrbracket \llbracket p - \ell \equiv \sum_{j \in B} m_{ij} \pmod{p} \rrbracket np \\ &= \begin{cases} \sum_{i \in [n]} \text{dlog}_{p,g} \sum_{j \in A \cup B} m_{ij} & \text{if } \prod_{i \in [n]} \sum_{j \in A \cup B} m_{ij} \not\equiv 0 \pmod{p}; \\ \geq np & \text{if } \prod_{i \in [n]} \sum_{j \in A \cup B} m_{ij} \equiv 0 \pmod{p}. \end{cases} \end{aligned} \quad (35)$$

In particular, letting $f_{\sigma_L, \sigma_R, t} = |\{(A, B) \in \mathcal{L}_{\sigma_L} \times \mathcal{R}_{\sigma_R} : \langle \lambda(A), \rho(B) \rangle = t\}|$, it follows immediately from (35) that we have $w_{\sigma_1, \sigma_2}^{(e)} = \sum_{t=0, t \equiv e \pmod{p-1}}^{n(p-2)} f_{\sigma_L, \sigma_R, t}$, which enables us to recover per M from the counts of pairs in $\mathcal{L}_{\sigma_L} \times \mathcal{R}_{\sigma_R}$ by inner product.

Completing the Proof of Theorem 3. Suppose we have an algorithm for #INNERPRODUCT that runs in $N^{2-\Omega(1/\log c)}$ time when given an input of N vectors from $\{0, 1\}^{c \log N}$. Take $N = 2^{n/2-1}$ and observe that $\log N = n/2 - 1$. The reduction from previous section has $d \leq n^4 (\ln n)^3$ and thus we can take $c = (n \ln n)^3$ and thus solve $n \times n$ zero-one permanent in time $N^{2-\Omega(1/\log c)} = 2^{n-\Omega(n/\log n)}$. This completes the proof of Theorem 3. \blacktriangleleft

6 A Lower Bound for #OV via Zero-One Permanents

A Reduction from Zero-One Permanent to #OV. This section starts our work towards Theorem 4 without yet parameterizing the reduction in detail. As in Sect. 5, it suffices to describe how to compute $\text{per } M$ modulo a given prime p with $2 \leq p \leq n \ln n$.

Let g be a positive integer parameter, which we assume divides n . For $h \in [g]$, let $V_h = \{i \in [n] : (h-1)n/g + 1 \leq i \leq hn/g\}$ be a partition of the rows of M into g groups, each of size n/g . Again from Ryser's formula, we have

$$\text{per } M = (-1)^n \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{h \in [g]} \prod_{i \in V_h} \sum_{j \in S} m_{ij}.$$

Grouping by sign $\sigma \in \{-1, 1\}$ and per-group residues $r \in \{0, 1, \dots, p-1\}^g$, we thus have

$$\text{per } M \equiv (-1)^n \sum_{r \in \{0, 1, \dots, p-1\}^g} (t_{1,r} - t_{-1,r}) \prod_{h=1}^g r_h \pmod{p}, \quad (36)$$

where $t_{\sigma,r} = |\{S \subseteq [n] : (-1)^{|S|} = \sigma \text{ and } \prod_{i \in V_h} \sum_{j \in S} m_{ij} \equiv r_h \pmod{p} \text{ for each } h \in [g]\}|$. Observe that given all the counts $t_{\sigma,r}$, it takes $O(p^g g)$ operations modulo p to compute the permanent modulo p via (36), which is less than $2^n n$ when $g < n/\log p$. We continue to describe how to get the counts $t_{\sigma,r}$ via orthogonal-vector counting.

Assuming that $n \geq 4$ is even, introduce again the split

$$L = \{1, 2, \dots, n/2\} \quad \text{and} \quad R = \{n/2, n/2 + 1, \dots, n\}.$$

Let the residue vector $r \in \{0, 1, \dots, p-1\}^g$ be fixed. For $\sigma_L, \sigma_R \in \{1, -1\}$, let

$$t_{\sigma_L, \sigma_R, r} = |\{S \subseteq [n] : (-1)^{|S \cap L|} = \sigma_L, (-1)^{|S \cap R|} = \sigma_R, \\ \text{and } \prod_{i \in V_h} \sum_{j \in S} m_{ij} \equiv r_h \pmod{p} \text{ for each } h \in [g]\}|.$$

Clearly $t_{\sigma,r} = \sum_{\substack{\sigma_L, \sigma_R \in \{-1, 1\} \\ \sigma_L \sigma_R = \sigma}} t_{\sigma_L, \sigma_R, r}$, so it suffices to focus on computing $t_{\sigma_L, \sigma_R, r}$ in what follows. We again work with the set families

$$\mathcal{L}_{\sigma_L} = \{A \subseteq L : (-1)^{|A|} = \sigma_L\} \quad \text{and} \quad \mathcal{R}_{\sigma_R} = \{B \subseteq R : (-1)^{|B|} = \sigma_R\}.$$

Let $D = [g] \times \{0, 1, \dots, p-1\}^{n/g}$. We have $d = |D| = gp^{n/g}$.

For $A \in \mathcal{L}_{\sigma_L}$ and $B \in \mathcal{R}_{\sigma_R}$, define the vectors $\lambda(A) \in \{0, 1\}^D$ and $\rho(B) \in \{0, 1\}^D$ for all $(h, u) \in D$ by the rules

$$\lambda(A)_{hu} = \begin{cases} 1 & \text{if we have } \sum_{j \in A} m_{ij} \equiv u_{i-(h-1)n/g} \pmod{p} \text{ for all } i \in V_h; \\ 0 & \text{otherwise} \end{cases}$$

and

$$\rho(B)_{hu} = \begin{cases} 0 & \text{if } \prod_{i \in V_h} (u_{i-(h-1)n/g} + \sum_{j \in B} m_{ij}) \equiv r_h \pmod{p}; \\ 1 & \text{otherwise.} \end{cases} \quad (37)$$

Over the integers, from (37) we now have

$$\begin{aligned}
\langle \lambda(A), \rho(B) \rangle &= \sum_{(h,u) \in D} \lambda(A)_{hu} \rho(B)_{hu} \\
&= \sum_{h \in [g]} \sum_{u \in \{0,1,\dots,p-1\}^{n/g}} \prod_{i \in V_h} \left[\sum_{j \in A} m_{ij} \equiv u_{i-(h-1)n/g} \pmod{p} \right] \\
&\quad \left[\prod_{i \in V_h} (u_{i-(h-1)n/g} + \sum_{j \in B} m_{ij}) \not\equiv r_h \pmod{p} \right] \\
&= \sum_{h \in [g]} \left[\prod_{i \in V_h} \left(\sum_{j \in A} m_{ij} + \sum_{j \in B} m_{ij} \right) \not\equiv r_h \pmod{p} \right] \\
&= \begin{cases} 0 & \text{if we have } \prod_{i \in V_h} \sum_{j \in A \cup B} m_{ij} \equiv r_h \pmod{p} \text{ for each } h \in [g]; \\ \geq 1 & \text{otherwise.} \end{cases} \quad (38)
\end{aligned}$$

In particular, we have $t_{\sigma_L, \sigma_R, r} = |\{(A, B) \in \mathcal{L}_{\sigma_L} \times \mathcal{R}_{\sigma_R} : \langle \lambda(A), \rho(B) \rangle = 0\}|$, which enables us to recover per M from the counts of orthogonal pairs in $\mathcal{L}_{\sigma_L} \times \mathcal{R}_{\sigma_R}$.

Completing the Proof of Theorem 4. Suppose now that we have an algorithm for #OV that runs in $N^{2-\Omega(1/\log^{1-\epsilon} c)}$ time for some $0 < \epsilon < 1$ when given an input of N vectors from $\{0, 1\}^{c \log N}$. Take $N = 2^{n/2-1}$ and observe that $\log N = n/2 - 1$.

Let $K > 1$ be a constant that will depend on ϵ and the constant hidden by the $\Omega(\cdot)$ notation in the running time of the #OV algorithm. Take $g = \lfloor K^{-1/\epsilon} n (\log p)^{1-2/\epsilon} \rfloor$ and recall that the prime p is in the range $2 \leq p \leq n \ln n$. To compute the parameters $t_{\sigma, r}$ using the reduction in the previous section, for each prime p we need $4p^g$ invocations of the #OV algorithm on an input of N vectors of dimension $d = gp^{n/g}$. Thus, for all large enough n , since $\frac{1}{2}K^{-1/\epsilon}n(\log p)^{1-2/\epsilon} \leq g$, we have $d = gp^{n/g} \leq n2^{2K^{1/\epsilon}(\log p)^{2/\epsilon}}$. Since clearly $d = c \log N = c(n/2 - 1)$ and $2/\epsilon > 2$, for all large enough n , we have $\log c \leq 1 + 2K^{1/\epsilon}(\log p)^{2/\epsilon} \leq 3K^{1/\epsilon}(\log p)^{2/\epsilon}$, where the last inequality depends on choosing a large enough K so that the inequality is true for $p = 2$. Thus, $-(\log c)^{\epsilon-1} \leq -3^{\epsilon-1}K^{1-1/\epsilon}(\log p)^{2-2/\epsilon}$. One invocation of the #OV algorithm thus runs in $N^{2-\Omega(\log^{\epsilon-1} c)} = 2^{n-\Omega(n3^{\epsilon-1}K^{1-1/\epsilon}(\log p)^{2-2/\epsilon})}$ time. For each prime $2 \leq p \leq n \ln n$, we need $4p^g \leq 2^{2+K^{-1/\epsilon}n(\log p)^{2-2/\epsilon}}$ invocations of the #OV algorithm. Thus, the running time of all the invocations for the prime p is bounded by $4p^g N^{2-\Omega(\log^{\epsilon-1} c)} \leq 2^{n-\Omega(n3^{\epsilon-1}K^{1-1/\epsilon}(\log p)^{2-2/\epsilon})+2+K^{-1/\epsilon}n(\log p)^{2-2/\epsilon}}$. By choosing a large enough K to dominate the constant hidden by the $\Omega(\cdot)$ notation in the running time of the #OV algorithm, we thus have, for all large enough n ,

$$\begin{aligned}
4p^g N^{2-\Omega(\log^{\epsilon-1} c)} &\leq 2^{n-\Omega(n3^{\epsilon-1}K^{-1/\epsilon}(\log p)^{2-2/\epsilon})} \\
&\leq 2^{n-\Omega(n3^{\epsilon-1}K^{-1/\epsilon}(\log n + \log \ln n)^{2-2/\epsilon})} \\
&\leq 2^{n-\Omega(n(\log n)^{2-2/\epsilon})}.
\end{aligned}$$

Since there are at most $n \ln n$ primes p to consider, the total running time to compute per M is bounded by $2^{n-\Omega(n/\log^{2/\epsilon-2} n)}$. This completes the proof of Theorem 4. \blacktriangleleft

7 Further Applications

Counting Satisfying Assignments to a $\text{Sym}\circ\text{And}$ circuit via #InnerProduct. We describe how to embed a $\text{SYM}\circ\text{AND}$ circuit, i.e., a circuit of s AND gates working on n Boolean inputs, connected by a top gate that is an arbitrary symmetric gate, in a #INNERPRODUCT instance

of size $N = 2^{n/2}$ and $d = s$. Assuming n even, we divide the n inputs in two equal halves L and R . We let \mathcal{A} have one vector u for each assignment to the inputs in L , with one coordinate in u for each AND gate, representing the truth value of that gate restricted to the inputs in L . Likewise, we let \mathcal{B} have one vector v for each assignment to the inputs in R , with each coordinate set to the truth value of the represented gate restricted to the inputs in R . It is readily verified that $\langle u, v \rangle$ counts the number of AND gates that are satisfied by the assignment represented by (u, v) . Hence, knowing the number of assignments that satisfy exactly t of the AND gates, for $t = 0, 1, \dots, s$, which is what the solution to the #INNERPRODUCT gives us, we can count the total number of assignments that also satisfies the top symmetric gate.

Variations where the circuit instead is a SYM◦OR or a SYM◦XOR, are also possible.

Computing the Weight Enumerator Polynomial via #InnerProduct. A binary linear code of length n and rank k is a linear subspace C with dimension k of the vector space \mathbb{F}_2^n . The *weight enumerator polynomial* is $W(C; x, y) = \sum_{w=0}^n A_w x^w y^{n-w}$, where $A_w = |\{c \in C : \langle c, c \rangle = w\}|$ for $w = 0, 1, \dots, n$ is the *weight distribution*; that is, A_w equals the number of codewords of C having exactly w ones.

We will reduce the computation of the weight distribution, and hence the weight enumerator polynomial, to $(k/2 + 1)^2$ instances of #INNERPRODUCT with $N \leq 2^{k/2}$ and $d = 2(n - k)$ when k is even.

Let the $k \times n$ matrix G be the generating matrix of the code; that is, the codewords of C are exactly the row-span of G . We can assume without loss of generality that the generator matrix has the standard form $G = [I_k | P]$, where I_k is the $k \times k$ identity matrix. For each $s_A = 0, 1, \dots, k/2$ and $s_B = 0, 1, \dots, k/2$, we make one instance of #INNERPRODUCT.

We let the set \mathcal{A} have one vector u for each code c obtained as the linear combination of exactly s_A of the first $k/2$ rows. Each of the $n - k$ last coordinates in the code word c is described by a block of two coordinates in u . If $c_i = 0$ we encode this as 01 in u , and if $c_i = 1$ we encode this as 10 in u . We concatenate all $n - k$ encoded blocks to obtain u . Likewise, we let the set \mathcal{B} have one vector v for each code c obtained as a linear combination of s_B of the last $k/2$ rows. Again, each of the $n - k$ last coordinates in the code word c is described by a block of two coordinates in v , but the encoding is opposite the one for \mathcal{A} : If $c_i = 0$ we encode this as 10 in v , and if $c_i = 1$ we encode this as 01 in v . We again concatenate all $n - k$ encoded blocks to obtain v . With this design, it is readily verified that for $(u, v) \in \mathcal{A} \times \mathcal{B}$, the inner product $\langle u, v \rangle$ is equal to the number of ones in the last $n - k$ coordinates in the code word obtained as the sum of the code word represented by u and the code word represented by v . Also, by design the number of ones in the first k coordinates equals $s_A + s_B$. Hence, by summing over all pairs that have the same inner product t , and aggregating over all s_A and s_B , we can compute the weight distribution.

References

- 1 Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230. SIAM, 2015. doi:10.1137/1.9781611973730.17.
- 2 Josh Alman. An illuminating algorithm for the light bulb problem. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*, pages 2:1–2:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/OASICS.SOSA.2019.2.

- 3 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.57.
- 4 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Faster deterministic and Las Vegas algorithms for offline approximate nearest neighbors in high dimensions. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 637–649. SIAM, 2020. doi:10.1137/1.9781611975994.39.
- 5 Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1034–1055. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00067.
- 6 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.18.
- 7 Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory. Vol. 1. Foundations of Computing Series*. MIT Press, Cambridge, MA, 1996. Efficient algorithms.
- 8 Eric T. Bax and Joel Franklin. A permanent algorithm with $\exp[\Omega(N^{1/3}/2 \ln N)]$ expected speedup for 0-1 matrices. *Algorithmica*, 32(1):157–162, 2002. doi:10.1007/s00453-001-0072-0.
- 9 Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994. doi:10.1007/BF01263423.
- 10 Andreas Björklund. Below all subsets for some permutational counting problems. In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPICs*, pages 17:1–17:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.SWAT.2016.17.
- 11 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.
- 12 Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya sets for polynomial evaluation and faster computation of fermionants. *Algorithmica*, 81(10):4010–4028, 2019. doi:10.1007/s00453-018-0513-7.
- 13 Andreas Björklund and Ryan Williams. Computing permanents and counting hamiltonian cycles by listing dissimilar vectors. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, volume 132 of *LIPICs*, pages 25:1–25:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.25.
- 14 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016. doi:10.1137/1.9781611974331.ch87.
- 15 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 21–40. SIAM, 2019. doi:10.1137/1.9781611975482.2.
- 16 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

- 17 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
- 18 Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Trans. Algorithms*, 14(3):31:1–31:26, 2018. doi:10.1145/3174804.
- 19 Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, 1998.
- 20 Herbert Robbins. A remark on Stirling’s formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.
- 21 J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64–94, 1962.
- 22 Herbert John Ryser. *Combinatorial Mathematics*. The Carus Mathematical Monographs, No. 14. Published by The Mathematical Association of America; distributed by John Wiley and Sons, Inc., New York, 1963.
- 23 Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015. doi:10.1145/2728167.
- 24 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 25 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, third edition, 2013. doi:10.1017/CB09781139856065.
- 26 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.