Generalizations of Matrix Multiplication can solve the Light Bulb Problem

Josh Alman* Hengjie Zhang†

November 6, 2023

Abstract

In the light bulb problem, one is given as input vectors $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{-1, 1\}^d$ which are all uniformly random. They are all chosen independently except for a planted pair (x_{i^*}, y_{j^*}) which is chosen to have correlation ρ for some constant $\rho > 0$. The goal is to find the planted pair. The light bulb problem was introduced over 30 years ago by L. Valiant, and is known to have many applications in data analysis, statistics, and learning theory.

The naive algorithm runs in $\Omega(n^2)$ time, and algorithms based on Locality-Sensitive Hashing approach quadratic time as $\rho \to 0$. In 2012, G. Valiant gave a breakthrough algorithm running in time $O(n^{(5-\omega)/(4-\omega)}) < O(n^{1.615})$, no matter how small $\rho > 0$ is, by making use of fast matrix multiplication. This was subsequently refined by Karppa, Kaski, and Kohonen in 2016 to running time $O(n^{2\omega/3}) < O(n^{1.582})$, but is essentially the only known approach for this important problem.

In this paper, we propose a new approach based on replacing fast matrix multiplication with other *variants* and *generalizations* of matrix multiplication, which can be computed faster than matrix multiplication, but which may omit some terms one is supposed to compute, and include additional error terms. Our new approach can make use of a wide class of tensors which previously had no known algorithmic applications, including tensors which arise naturally as intermediate steps in border rank methods and in the Laser method.

We further show that our approach can be combined with locality-sensitive hashing to design an algorithm whose running time improves as ρ gets larger. To our knowledge, this is the first algorithm which combines fast matrix multiplication with hashing for the light bulb problem or any closest pair problem, and it leads to faster algorithms for small $\rho > 0$.

We then focus on tensors for "multiplying" 2×2 matrices; using such small tensors is typically required for practical algorithms. In this setting, the best prior algorithm, using Strassen's algorithm for matrix multiplication, yields a running time of only $O(n^{1.872})$. We introduce a new such low-rank tensor we call T_{2112}, which has omissions and errors compared to matrix multiplication, and using it, we design a new algorithm for the light bulb problem which runs in time $O(n^{1.797})$. We also explain why we are optimistic that this approach could yield asymptotically faster algorithms for the light bulb problem.

*josh@cs.columbia.edu. Columbia University. Supported in part by NSF Grant CCF-2238221 and a grant from the Simons Foundation (Grant Number 825870 JA).

†hengjie.z@columbia.edu. Columbia University. Supported in part by NSF grant CCF2008733 and ONR grant N00014-22-1-2713.

1 Introduction

We've known since the work of Strassen [Str73] that designing algebraic algorithms for matrix multiplication is equivalent to bounding the ranks of matrix multiplication tensors. Since then, an enormous amount of work has gone into bounding the ranks of these tensors in various regimes, combining techniques from algebra, combinatorics, algorithm design, and computer search. One big reason that so much effort has gone into this problem is that matrix multiplication has many algorithmic applications; algorithmic problems from nearly every domain of computation have been reduced to matrix multiplication.

A goal of this paper is to show that tensors other than matrix multiplication can also have algorithmic applications. In this way, the same techniques which have been developed for matrix multiplication could be repurposed to lead to new algorithms. Tensors whose support is a subset of the support of matrix multiplication have been applied to Boolean matrix multiplication [CU13, KK19, Har21] and even directly to matrix multiplication [Sch81], but we're unaware of applications of any tensors whose support is incomparable with matrix multiplication (other than a small handful of special problems like polynomial multiplication).

We focus here on the light bulb problem, a fundamental problem from learning theory which currently has two best algorithms depending on the parameter regime: one using fast matrix multiplication, and one using locality-sensitive hashing. Somewhat surprisingly, there are no known parameter regimes where combining the two approaches leads to an improved algorithm. Using tensors other than matrix multiplication, we achieve two main results

1. Any tensor can replace matrix multiplication to solve the light bulb problem. The efficiency of this algorithm comes from a trade-off between the tensor's rank and how similar it is to matrix multiplication. We find that, restricted to small tensors, there are better tensors than matrix multiplication that lead to improved algorithms.

2. Tensors other than matrix multiplication can be combined with locality-sensitive hashing to yield improved algorithms. We find that the symmetry of matrix multiplication prevents one from combining it with hashing, but that sufficiently asymmetric tensors can be improved with hashing.

1.1 The Light Bulb Problem

In the light bulb problem for n vectors of dimension d and correlation $\rho > 0$, we are given as input vectors $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{-1, 1\}^d$ which are all picked uniformly at random, and all picked independently except for a 'planted pair' (x_{i^*}, y_{j^*}) which is chosen to have correlation $\geq \rho$ (i.e., so that $\langle x_{i^*}, y_{j^*} \rangle \geq \rho \cdot d$). The indices i^* and j^* of the planted pair are unknown to us, and our goal is to find them.[1]

L. Valiant introduced the light bulb problem over 30 years ago [Val88] as a basic primitive which captures the fundamental task of detecting correlations among n random variables. It can be seen as a special case of a multitude of other problems in data analysis, statistics, and learning theory, and for many of these problems, the fastest known algorithm comes from a reduction to the light bulb problem. For instance:

- If one would like to detect correlations among random variables with a range R other than just $\{-1, 1\}$, one can typically reduce to the light bulb problem by making use of a locality-sensitive hash function for R. For example, if R is the Euclidean sphere, then one can map points in R to $\{-1, 1\}$ by determining which side of a random hyperplane they lie on, which only slightly decreases the correlation ρ, by a constant factor [Cha02].
- The light bulb problem is a special case of many learning problems, including learning sparse parities with noise, and learning k-Juntas with and without noise, and the fastest known algorithms for these problems come from reducing the general case to the light bulb problem [Val12].

[1] The light bulb problem is often stated as a 'monochromatic' problem, where we are not told which are 'x' or 'y' vectors, but this has a simple reduction to the 'bichromatic' version we define here.

Suppose $d = \Theta(\log n)$. The straightforward algorithm for this problem simply compares each pair of vectors and runs in time $\tilde{O}(n^2)$.[2] Techniques for nearest neighbor search like locality-sensitive hashing have been applied to the problem, culminating in Dubiner's algorithm [Dub10] which runs in time $n^{2/(\rho+1)+o(1)}$. This is the fastest known algorithm for larger ρ, but its running time becomes quadratic as $\rho \to 0$. In 2012, G. Valiant [Val12] gave a breakthrough algorithm running in time $O(n^{1.615})$ no matter how small the constant $\rho > 0$ is. Thereafter, Karppa, Kaski, and Kohonen [KKK18] improved the running time to $O(n^{1.582})$. This is faster than Dubiner's algorithm for all $0 < \rho < 0.264$. To emphasize, these algorithms work for any constant $\rho > 0$, but give essentially the same running time no matter how large ρ is.

The key ideas behind these latter two algorithms focus on the dimension d, which is sometimes called the 'sample complexity'. One would typically like to keep d low while still solving the problem quickly. It is information-theoretically necessary to pick $d = \Omega(\log n)$ (since if $d = o(\log n)$, then by the pigeonhole principle, two of the uncorrelated vectors will be *equal to* each other and indistinguishable from the correlated pair).

Interestingly, G. Valiant [Val12] introduced a 'XOR/Tensor Embedding' technique, and Karppa, Kaski, and Kohonen [KKK18] gave a more efficient 'compressed matrices' implementation, which (roughly) allows one to efficiently 'expand' lower-dimensional vectors, and convert $d = \Theta(\log n)$ to a much larger $d = n^{\Theta(1)}$ with only a negligible decrease in ρ. This allows one to focus on the task of designing faster algorithms for detecting correlations without worrying about d. More precisely, these prior algorithms consist of two phases solving two different problems: a 'vector aggregation' problem of converting groups of shorter vectors into single longer vectors, and a 'light bulb computation' problem of actually detecting the correlations among these vectors. The final running time of [Val12] trades off between the running times of these two problems, and the later work [KKK18] showed how to make the running time of vector aggregation negligible compared to the running time of light bulb computation. Both prior algorithms ultimately solve the light bulb computation problem using fast matrix multiplication, and we focus in this paper on faster algorithms for this problem. (See footnote 10 in Section 3 below for more details.)

Despite the importance of the light bulb problem, no approach beyond locality-sensitive hashing or 'expand then use fast matrix multiplication' has been proposed since the breakthrough almost 10 years ago [Val12], and these known approaches seem to have hit their limits [KKK18, Alm18]. Furthermore, although hashing approaches and matrix multiplication approaches have been known for both the light bulb problem as well as many other closest pair problems for some time (see, for instance, the survey [AIR18]), there are no known algorithms for any of these problems which truly combine the two.

In this paper, we propose a new approach to designing faster algorithms for the light bulb problem by replacing fast matrix multiplication with other tensors which are *generalizations* of matrix multiplication, and which can be computed faster. We also show how hashing methods can be combined with our approach to design even faster algorithms: while previous matrix multiplication-based algorithms for the light bulb problem have the same running time regardless of how large $\rho > 0$ is, our new approach yields algorithms which are faster as ρ gets larger. Before getting into more detail, we introduce some necessary background.

Known Algorithms and Exponents. The exponent of matrix multiplication, ω, is the smallest real number such that for any $\varepsilon > 0$, one can multiply $n \times n$ matrices over a field using $O(n^{\omega+\varepsilon})$ field operations[3]. Since $n \times n$ matrices have n^2 entries one must read and write, it is known that $\omega \geq 2$, and the best known algorithms show $\omega < 2.37286$ [CW82, DS13, Wil12, LG14, AW21].

We similarly define the exponent of the light bulb problem, ω_ℓ, to be the smallest real number such that for any $\varepsilon > 0$, one can solve the light bulb problem with n vectors, for any constant $\rho > 0$, in time $O(n^{\omega_\ell+\varepsilon})$.

[2] We write $\tilde{O}(t)$ to suppress $\text{polylog}(t)$ factors.

[3] The 'running time' and 'number of field operations' are typically related by low-order terms unless one is working with very large numbers. In principle, ω might depend on the characteristic of the field, although all known bounds work equally well over any field, so we will abuse notation and simply refer to the same ω for all fields.

G. Valiant [Val12] showed that $\omega_\ell \leq (5-\omega)/(4-\omega) < 1.615$, and Karppa, Kaski, and Kohonen [KKK18] later improved this to the best known bound $\omega_\ell \leq 2\omega/3 < 1.582$. Since the input size is only $\tilde{O}(n)$, the corresponding lower bound is $\omega_\ell \geq 1$. However, even showing $\omega = 2$ would only imply that $\omega_\ell \leq 4/3$ using the known algorithms [Val12, KKK18, Alm18].

We will also discuss the *Boolean* matrix multiplication problem, where we're given as input matrices $A, B \in \{0,1\}^{n\times n}$, and we need to compute the matrix product $C = A \times B$ over the Boolean semiring, i.e., the matrix $C \in \{0,1\}^{n\times n}$ given by $C[i,j] = \bigvee_{k=1}^{n}(A[i,k] \wedge B[k,j])$. Let ω_B denote the smallest real number such that for any $\varepsilon > 0$, one can solve this problem in time $O(n^{\omega_B+\varepsilon})$. It is known that $2 \leq \omega_B \leq \omega$, and there are no known algorithms for Boolean matrix multiplication that are asymptotically faster than the best known matrix multiplication algorithms (see, e.g., [KK19, Section 1]).

1.2 Bilinear Problems

A key technique in this paper will be designing and making use of algorithms for bilinear problems, wherein one would like to evaluate a prescribed set of bilinear polynomials when its variables are set to input numbers. Matrix multiplication is a prominent example, and we will focus particularly on bilinear problems like this where the inputs and outputs are naturally formatted as matrices. Bilinear problems which take as input a $q_i \times q_k$ matrix X and a $q_j \times q_k$ matrix Y, and output a $q_i \times q_j$ matrix Z, can be written as a (three-dimensional) tensor

$$T = \sum_{i,i'\in[q_i],j,j'\in[q_j],k,k'\in[q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}) \cdot \mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'},$$

where $T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}) \in \mathbb{R}$ is the coefficient of $\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}$ in the bilinear polynomial we output in entry $\mathsf{Z}_{i',j'}$. One can imagine plugging in values for each of the X and Y variables, and then the goal is to compute the coefficient of each Z variable. For example, for the $q \times q$ matrix multiplication problem $T_q = \langle q, q, q\rangle$,

$$T_q(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}) = \begin{cases} 1 & \text{if } i = i', j = j', \text{ and } k = k', \\ 0 & \text{otherwise.} \end{cases}$$

1.3 Main result when ρ is close to 0

Our main result, which we will state below, gives a way to use an algorithm for almost any bilinear problem T to solve the light bulb problem, even if T only computes some of the terms of matrix multiplication, and if T also computes other 'noise' terms. To state our result, we need to define two relevant properties of T. The first property, the *rank* of T, is a standard measure of how complicated T is, while the another property, *efficacy*, is the property we introduce for measuring how useful T is for solving the light bulb problem.

Rank. A tensor T has rank 1 if it can be written in the form

$$T = \left(\sum_{i\in[q_i],k\in[q_k]} \alpha_{i,k}\mathsf{X}_{i,k}\right)\left(\sum_{j\in[q_j],k\in[q_k]} \beta_{j,k}\mathsf{Y}_{j,k}\right)\left(\sum_{i\in[q_i],j\in[q_j]} \gamma_{i,j}\mathsf{Z}_{i,j}\right)$$

for coefficients $\alpha_{i,k}, \beta_{j,k}, \gamma_{i,j} \in \mathbb{R}$. More generally, the rank of T, denoted $\mathrm{rank}(T)$, is the minimum nonnegative integer k such that there are rank 1 tensors $T_1, \ldots, T_k$ with $T = T_1 + \cdots + T_k$. Rank is the most prominent measure of the complexity of a tensor, and rank upper bounds for tensors yield algorithms for applying that tensor to matrices. For instance, Strassen [Str69] famously showed that the rank of the tensor $\langle 2, 2, 2\rangle$ for multiplying two 2×2 matrices is at most 7, and hence that one can multiply $n \times n$ matrices in time $O(n^{\log_2 7})$.

Efficacy. The second property of T, which is a new property we introduce, is its *efficacy*[4]. For $i \in [q_i]$ and $j \in [q_j]$, the efficacy of T at (i, j) is given by:

$$\underset{i,j}{\mathrm{eff}}(T) := \frac{\sum_{k\in[q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k}\mathsf{Z}_{i,j})}{\sqrt{\sum_{i'\in[q_i],j'\in[q_j],k,k'\in[q_k]} T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j})^2}}.$$

The numerator of $\mathrm{eff}_{i,j}(T)$ is the sum of the coefficients of all the entries which are supposed to be included in $Z_{i,j}$ in regular matrix multiplication. The denominator is the ℓ_2 norm of the vector of coefficients of *all* the terms which are included in $Z_{i,j}$ in T. Hence, one can think of $\mathrm{eff}_{i,j}(T)$ as a ratio of the 'signal' and the 'noise' of T for computing the (i, j) output entry of matrix multiplication.

Then, the efficacy of the whole tensor T is the ℓ_2 norm of the efficacies of all its output entries:

$$\mathrm{eff}(T) := \sqrt{\sum_{i\in[q_i],j\in[q_j]} \left(\underset{i,j}{\mathrm{eff}}(T)\right)^2}.$$

We will see that $\mathrm{eff}(T)$ measures how useful T is for solving the light bulb problem from our main result, which shows how one could improve on the current best exponent $2\omega/3$:

Theorem 1.1. *For any tensor T, if*

$$\frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))} < \frac{2\omega}{3},$$

then

$$\omega_\ell < \frac{2\omega}{3}.$$

Moreover, if T has negligible aggregation time (see Section 1.4 below), then

$$\omega_\ell < \frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))}.$$

Hence, as long as T is easy to compute ($\mathrm{rank}(T)$ is small) and it has a high ratio of signal to noise for computing matrix multiplication ($\mathrm{eff}(T)$ is large), one can use it to design a fast algorithm for the light bulb computation problem. (Again, this algorithm works with this exponent for any constant $\rho > 0$, no matter how small.) We will see shortly that the algorithm consists of applying T to pairs of carefully-chosen (but simple to construct) matrices, and then doing a simple analysis of the result. In other words, the algorithm itself is fairly simple, but the proof of correctness is quite involved.

1.4 Aggregation time

The aggregation time assumption in Theorem 1.1 relates to the initial aggregation step that appears in all prior matrix multiplication-based light bulb algorithms including ours [Val12, KKK18, Alm18]. In [Val12], aggregation took a significant amount of time which needed to be "traded off" against later steps of the algorithm. [KKK18] substantially improved aggregation to take a negligible amount of time compared to the rest of the algorithm.

The same technique of [KKK18] applies in our setting as well, which makes aggregation negligible for all the tensors we study. We believe this technique makes the aggregation time negligible for all possible tensors T, although we're unable to prove this[5].

[4] We were inspired to pick this name by the 'luminous efficacy' of a light bulb, which measures the ratio of how much light is produced and how much power is consumed. It bears similarity to other known statistical ratios like the 'standardized second moment' and the 'Fano factor'.

[5] Tensors with nonnegligible aggregation time would have very low efficacy, so that very long vectors are needed in our algorithm, but extremely low rank so that the algorithm may still be fast; see Appendix A below for more details.

Nonetheless, we prove in Theorem 1.1 that in order to improve the current best exponent $2\omega/3$, it suffices to find any tensor T with $\frac{\log(\text{rank}(T))}{\log(\text{eff}(T))} < \frac{2\omega}{3}$, ignoring the aggregation time condition. To prove this, we show in Appendix A below that for any tensor T with nonnegligible aggregation time, if the quantity $\frac{\log(\text{rank}(T))}{\log(\text{eff}(T))}$ is less than the current best exponent $2\omega/3$, then one can slightly modify T to get a new tensor T' with negligible aggregation time which still has $\frac{\log(\text{rank}(T'))}{\log(\text{eff}(T'))} < 2\omega/3$.

1.5 Applying to Matrix Multiplication Generalizations

To demonstrate the promise of Theorem 1.1, we focus on tensors for 2×2 input matrices. (Using the notation above, we focus on $q_i = q_j = q_k = 2$.) This is the size of the tensor for Strassen's algorithm. As we discuss shortly, we introduce a new tensor with rank only 5 which is able to to achieve a better exponent than Strassen's algorithm.

We focus on this case for three reasons. First, using such a small tensor is typically necessary to design a practical algorithm (see, e.g., the introductions of [HSHVDG16, KK19, Pan18, FBH+22] where practicality concerns are discussed). Second, using such small tensors lets us more concretely see how more general tensors can be used, especially in conjunction with locality-sensitive hashing later. Third, small tensors are typically a good test bed for further improvements based on larger tensors. (We discuss this in more detail shortly, in Section 1.7 below.)

Three Matrix Multiplication Generalizations. We apply Theorem 1.1 to three tensors of interest. We will see that applying it when T is a matrix multiplication tensor recovers the best known bound $\omega_\ell \leq \frac{2}{3}\omega$ of [KKK18], but that other tensors can yield even faster algorithms, including a new tensor we introduce. See Figure 1 for descriptions of the three tensors; their precise definitions and rank expressions are given in Section 5 below.

First is the tensor for 2×2 matrix multiplication (denoted $\langle 2,2,2\rangle$). We can calculate that $\text{eff}(\langle 2,2,2\rangle) = \sqrt{8}$, and hence, using Strassen's bound $\text{rank}(\langle 2,2,2\rangle) \leq 7$, that $\omega_\ell \leq 1.872$. Prior to this work, this was the smallest known exponent for the light bulb problem based on a $q = 2$ tensor. Note that more generally, $\text{eff}(\langle n,n,n\rangle) = n^{3/2}$, so applying Theorem 1.1 to the $n \times n$ matrix multiplication tensor (denoted $\langle n,n,n\rangle$) for large n yields $\omega_\ell \leq \log(R(\langle n,n,n\rangle))/\log(n^{1.5}) \leq \log(n^{\omega+o(1)})/\log(n^{1.5}) \to \frac{2}{3}\omega$, which recovers the best known exponent for the light bulb problem [KKK18].

Second is the tensor SW, which consists of 7 of the 8 terms of 2×2 matrix multiplication, and which has rank 6 via an identity by Winograd [Win71]. Recent work by Karppa and Kaski [KK19] showed how to apply any tensor which consists of a subset of the terms of matrix multiplication to design a Boolean matrix multiplication algorithm. Follow-up work by Harris [Har21] improved their analysis specifically for the tensor SW to design a practical (since it is based on a small tensor) algorithm for Boolean matrix multiplication with exponent $\omega_B < 2.763$. It was previously unclear how to use SW, or any such 'subset of matrix multiplication' tensor, to design an algorithm for the light bulb problem, or any problem which does not have a known reduction to Boolean matrix multiplication. Applying our Theorem 1.1 to SW yields an algorithm with exponent $\omega_\ell < 1.842$, improving on Strassen's algorithm. More generally, for 'subset of matrix multiplication' tensors, our bound on ω_ℓ is strictly better than $\frac{2}{3}$ times the bound on ω_B which Karppa and Kaski [KK19] achieves, and equal to $\frac{2}{3}$ times the bound on ω_B which Harris [Har21] achieves (although [Har21] only applies to some such tensors[7]).

Third is a new rank-5 tensor T_{2112} that we design and give a rank bound for in this paper. In terms of a parameter $\varepsilon > 0$, T_{2112} is a sum of 6 of the 8 terms terms of 2×2 matrix multiplication, plus 7 additional terms with coefficients $O(\varepsilon)$, which can be made arbitrarily small by suitably picking ε. (Note that one

[7] It seems difficult to extend the approach of [Har21] to 'subset of matrix multiplication' tensors with very skewed patterns of terms, whereas [KK19] applies to all such tensors. In this paper we use a technique to 'regularize' the pattern of errors of a tensor (see Section 3.3 below) which seems at a first glance like it could apply to that setting as well, but unfortunately, the 'space of errors' in that settings is 3-dimensional, whereas we critically use the fact that it is 2-dimensional here.

Tensor Name	Rank	Tensor	Table of $\text{eff}_{i,j}$[6] value of eff, and resulting ω_ℓ bound
$\langle 2,2,2\rangle$ (Strassen's algorithm [Str69])	7	$(\mathsf{X}_{1,1}\mathsf{Y}_{1,1}+\mathsf{X}_{1,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,1}$ $+(\mathsf{X}_{1,1}\mathsf{Y}_{1,2}+\mathsf{X}_{1,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,1}$ $+(\mathsf{X}_{2,1}\mathsf{Y}_{1,1}+\mathsf{X}_{2,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,2}$ $+(\mathsf{X}_{2,1}\mathsf{Y}_{1,2}+\mathsf{X}_{2,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,2}$	$i\backslash j$ | **1** | **2** **1** | $\sqrt{2}$ | $\sqrt{2}$ **2** | $\sqrt{2}$ | $\sqrt{2}$ $\text{eff}(\langle 2,2,2\rangle)=\sqrt{8}$ $\omega_\ell \le \frac{\log(7)}{\log(\sqrt{8})} < 1.872$
SW (Strassen-Winograd identity [Win71])	6	$(\mathsf{X}_{1,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,1}$ $+(\mathsf{X}_{1,1}\mathsf{Y}_{1,2}+\mathsf{X}_{1,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,1}$ $+(\mathsf{X}_{2,1}\mathsf{Y}_{1,1}+\mathsf{X}_{2,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,2}$ $+(\mathsf{X}_{2,1}\mathsf{Y}_{1,2}+\mathsf{X}_{2,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,2}$	$i\backslash j$ | **1** | **2** **1** | 1 | $\sqrt{2}$ **2** | $\sqrt{2}$ | $\sqrt{2}$ $\text{eff}(SW)=\sqrt{7}$ $\omega_\ell \le \frac{\log(6)}{\log(\sqrt{7})} < 1.842$
T_{2112} (new tensor)	5	$(\mathsf{X}_{1,1}\mathsf{Y}_{1,1}+\mathsf{X}_{1,2}\mathsf{Y}_{2,1}+O(\varepsilon))\mathsf{Z}_{1,1}$ $+(\mathsf{X}_{1,2}\mathsf{Y}_{2,2}+O(\varepsilon))\mathsf{Z}_{2,1}$ $+(\mathsf{X}_{2,1}\mathsf{Y}_{1,1}+O(\varepsilon))\mathsf{Z}_{1,2}+$ $(\mathsf{X}_{2,1}\mathsf{Y}_{1,2}+\mathsf{X}_{2,2}\mathsf{Y}_{2,2}+O(\varepsilon))\mathsf{Z}_{2,2}$ ($O(\varepsilon)$ hides arbitrarily small positive coefficients in terms of a parameter $\varepsilon>0$)	$i\backslash j$ | **1** | **2** **1** | $\sqrt{2}-O(\varepsilon^2)$ | $1-O(\varepsilon^2)$ **2** | $1-O(\varepsilon^2)$ | $\sqrt{2}-O(\varepsilon^2)$ $\text{eff}(T_{2112})=\sqrt{6}-O(\varepsilon^2)$ $\omega_\ell \le \frac{\log(5)}{\log(\sqrt{6}-O(\varepsilon^2))} \to 1.797$

Figure 1: Three tensors with $q=2$ which we use in our algorithm, along with the resulting bounds on ω_ℓ from using them in Theorem 1.1. See Section 5 below where we define these tensors exactly (without hiding terms in '$O(\varepsilon)$') and give their rank expressions. The tensors $\langle 2,2,2\rangle$ and SW come from classical work on optimizing Strassen's algorithm, while T_{2112} and its rank upper bound are both new.

cannot eliminate these terms by setting $\varepsilon=0$, since we divide by ε when showing that T_{2112} has rank 5; see Section 5 for more details.)

Prior work would have concluded by taking the limit $\varepsilon\to 0$ in T_{2112} that the 'border rank' of 6 of the 8 terms of 2×2 matrix multiplication is 5. Border rank bounds could be used instead of rank in conjunction with our approach by using the technique of Bini [Bin80a]. One advantage of Theorem 1.1 is that it allows one to plug constants $\varepsilon>0$ into border rank expressions and avoid the complications of border rank. (Border rank identities are typically harder to find using numerical methods, and lead to less practical algorithms.) Setting just $\varepsilon=0.025$ suffices to get the best possible exponent using $q=2$:

Theorem 1.2. *There is a tensor T_{2112} with $q=2$ which achieves the exponent $\omega_\ell<1.797$.*

Before moving on, we note that a prior identity of Bini [Bin80b] already gave a different tensor B with (a different) 6 of the 8 terms of 2×2 matrix multiplication, and border rank 5. However, our tensor T_{2112} has one advantage over B. That is, the pattern of $\text{eff}_{i,j}(T_{2112})$, with larger entries along the diagonal, will allow us to use it in conjunction with hashing methods in our second result.

1.6 Main result when ρ is bounded away from 0, using locality-sensitive hashing

At a high level, our algorithm for Theorem 1.1 works by first mapping each of the n different x inputs into one of q_i independently random buckets, and each of the n different y inputs into one of q_j independently random buckets. (Recall that q_i, q_j are two of the parameters defining the size of the tensor T; think of them as n^c for a constant $0 < c < 1$, for instance by first taking an appropriate 'Kronecker power' of T.) If the correlated pair were mapped into buckets $i \in [q_i]$ and $j \in [q_j]$, then our algorithm will succeed as long as $\mathrm{eff}_{i,j}(T)$ is large enough. The proof of Theorem 1.1 requires carefully balancing the parameters so that, when i and j are picked uniformly randomly, then this becomes fairly likely.

Our second main result shows how to improve Theorem 1.1 by combining it with one of the most prevalent techniques in nearest neighbor search: locality-sensitive hashing. The main idea to improve on this is to place the inputs into buckets using (a variation on) bit sampling locality-sensitive hashing, instead of uniformly random hashing. In this way, thinking of the buckets as $\{-1, 1\}$ bit strings, we know that if the planted pair has correlation $\rho > 0$, then they are likely put into buckets $i \in \{-1, 1\}^{\log_2(q_i)}$ and $j \in \{-1, 1\}^{\log_2(q_j)}$ which also have correlation close to ρ. If such buckets have larger $\mathrm{eff}_{i,j}(T)$ than uniformly random buckets, then we can speed up our algorithm.

By construction, our new tensor T_{2112} has exactly this property! By renaming variables[8] and taking the limit $\varepsilon \to 0$ for notational simplicity, we see that it has $\mathrm{eff}_{1,1}(T_{2112}) = \mathrm{eff}_{-1,-1}(T_{2112}) = \sqrt{2}$ and $\mathrm{eff}_{1,-1}(T_{2112}) = \mathrm{eff}_{-1,1}(T_{2112}) = 1$. More generally, once we've taken a Kronecker power so that $q_i = q_j = q$ is larger than 2, the resulting tensor will have the property that, for buckets $i, j \in \{-1, 1\}^{\log_2(q)}$ with correlation ρ, we have $\mathrm{eff}_{i,j}(T_{2112}^{\otimes \log_2(q)}) = 2^{(1+\rho)(\log_2(q)/2)}$, whereas the median pair i, j of buckets has only $\mathrm{eff}_{i,j}(T_{2112}^{\otimes \log_2(q)}) = 2^{\log_2(q)/2}$. Thus, in a sense, the efficacy of our tensor T_{2112} is increasing with ρ, resulting in a faster algorithm. (We briefly note that although our analysis makes use of Kronecker powers of tensors, our algorithm itself does not, and only applies the tensor itself to input matrices.)

However, the formal statement of our result is more complicated than this because of a key detail behind Theorem 1.1 that we have thus far swept under the rug. Rather than map each input point into a single bucket, it actually makes many copies of each input point and independently maps them into buckets. This way, in order to solve the light bulb problem, it suffices for any one pair of copies of the correlated pair to map into buckets with large efficacy. Typically a locality-sensitive hashing scheme would map all the different copies of the same vector to the same bucket, and lose these savings. Nonetheless, we find a way to hash inputs into multiple buckets, so that the correlated pair is still hashed to correlated buckets, but the different pairs of buckets are 'sufficiently independent' of each other so that whether or not each succeeds isn't too correlated. Applying this to T_{2112}, we achieve:

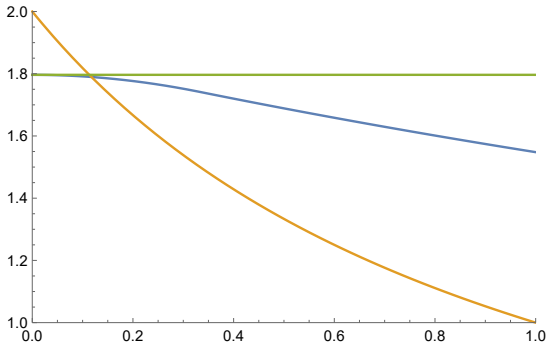
Theorem 1.3. *For the tensor T_{2112}, the bound of Theorem 1.1 can be improved to*

$$\omega_\ell \leq \begin{cases} \frac{2\log 5}{\log\left(6(1-\rho)^{-\rho/2}(1+\rho)^{\rho/2}(1-\rho^2)^{1/2}\right)} & \text{when } \rho < 1/3, \\ \frac{4\log 5}{(5+\rho)\log 2} & \text{when } 1/3 \leq \rho \leq 1. \end{cases}$$

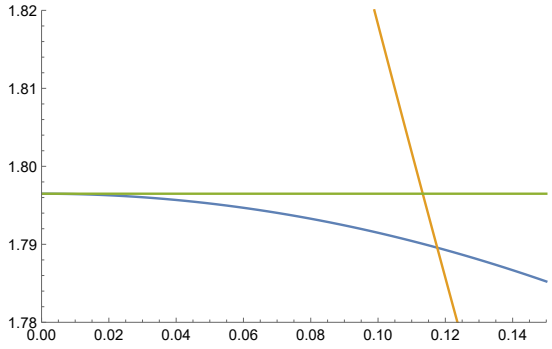
The resulting plot of w_ℓ with respect to ρ from can be found in Figure 2 (in blue). Our Theorem 1.1, as well as prior matrix multiplication-based algorithms for the light bulb problem, give the same running time exponent no matter how large $\rho > 0$ is, whereas Theorem 1.3 uses hashing to improve with ρ. We also show Dubiner's algorithm, which is purely based on hashing, and is worse for small $\rho > 0$, but better for larger ρ.

Custom-tailored hash functions for other tensors. In order to prove Theorem 1.3, we observed that bit sampling locality-sensitive hashing is likely to put the correlated pair of vectors into buckets i, j where i and j are correlated, and hence have a higher-than-average value of $\mathrm{eff}_{i,j}(T_{2112}^{\otimes \log_2(q)})$. What if we are working

[8] Whenever i or j was 2, we now call it -1.



(a) The running time exponent of three different methods in terms of $\rho \in [0, 1]$.



(b) The same plot as in the figure to the left, but focused on the range $\rho \in [0, 0.2]$.

Figure 2: The running time exponents (y-axis) of three algorithms in terms of ρ (x-axis): Theorem 1.3 (blue line), Theorem 1.1 using T_{2112} (green line), and Dubiner's algorithm [Dub10] (orange line).

with a different tensor T for which the value of $\mathrm{eff}_{i,j}(T)$ does not increase as i and j are more correlated? Bit sampling locality-sensitive hashing won't give an improvement, but this is only one possible hash function.

In fact, we can generalize Theorem 1.3 to almost any tensor. We show that for any T whose efficacy matrix $[(\mathrm{eff}_{i,j}(T))^2]_{i,j}$ is not 'degenerate' in some sense, one can custom-tailor hash functions for T which result in an improved running time as ρ grows. The formal statement of this result is somewhat complicated; we defer the details to Section 4 below. However, for one simple and important example, we show:

Theorem 1.4. *Suppose T is a $\langle q, q, q_k\rangle$-sized tensor which consists of a subset of the terms of a matrix multiplication tensor, and the matrix $[(\mathrm{eff}_{i,j}(T))^2]_{i\in[q],j\in[q]}$ has full rank. Let ω'_ℓ be the exponent one would get from T from applying Theorem 1.1. Then, for every $\rho > 0$, there is an $f(T, \rho) > 0$ such that the light bulb computation problem with correlation ρ can be solved with the improved exponent $\omega'_\ell - f(T, \rho)$.*

Theorem 1.4 shows that hashing can improve the algorithm based on almost any 'subset of matrix multiplication' tensor. These are the same tensors used by Karppa and Kaski [KK19] to solve Boolean matrix multiplication (they showed that bounds on their ranks give bounds on the 'probabilistic rank' of matrix multiplication) and include the tensors other than $\langle 2, 2, 2\rangle$ that we discussed above in Section 1.5.

Intuitively, we require the matrix $[(\mathrm{eff}_{i,j}(T))^2]_{i,j}$ to have full rank in Theorem 1.4 so that there are regions of buckets with higher efficacy that we could hope to hash the correlated pair to. For instance, if T is a matrix multiplication tensor, then hashing cannot move the correlated pair to a better bucket since all buckets have the same efficacy, and indeed, the efficacy matrix has rank 1 since all its entries are equal.

1.7 Comparison with Prior Work on Tensor and Nearest Neighbor Search Algorithms

Other Variants on Matrix Multiplication. As mentioned at the beginning of Section 1, prior work has solved Boolean matrix multiplication using tensors whose support is (a subset of) the support of matrix multiplication [CU13, KK19, Har21]. To our knowledge, we are the first to use tensors whose support may not be a subset of the support of matrix multiplication, and the first to use variants of matrix multiplication on a problem that is not (reducible to) Boolean matrix multiplication. This access to a larger class of tensors is what allows us to design a faster practical algorithm for the light bulb problem than the analogous fastest practical algorithm for Boolean matrix multiplication; our tensor T_{2112} cannot be applied in the other settings (for fixed $\varepsilon > 0$). We hope our techniques could be used to apply these tensors to other problems in the future, particularly problems which are currently solved with exact matrix multiplication but which may only need approximate matrix multiplication.

Improving the asymptotic exponent We are optimistic that Theorem 1.1 can be used to improve the best known exponent for the light bulb problem by using larger tensors. Finding improvements based on larger tensors has historically been very difficult compared to finding improvements based on small tensors; for instance, it took almost 10 years after Strassen's algorithm based on a 2×2 identity before Pan [Pan78] gave an improved exponent based on a larger tensor. Moreover, decades of work have gone into designing matrix multiplication algorithms for larger q which we need to catch up to for the light bulb problem. Many of these techniques can be directly repurposed to the light bulb problem (for instance, it is not hard to prove a version of the asymptotic sum inequality [Sch81] in this setting), but the centerpiece of fast matrix multiplication algorithms, the Coppersmith-Winograd tensor, seems particularly designed for exact matrix multiplication, and it is not clear how to improve it for our light bulb setting. (More generally, it is a major open challenge to understand the effectiveness of the Coppersmith-Winograd tensor or find any useful variants on it [HJMS22, BL16, CGLV19, CHL22].) Nonetheless, our computations suggest that this trend continues: our approach gives better bounds on $\frac{3}{2}\omega_\ell$ than prior approaches do for ω_B when restricted to certain small classes of tensors, such as tensors on the variable set of $\langle 3,3,3\rangle$, or tensors on the variable set of $\langle 2,2,2\rangle$ of rank at most 4 (but none of these beats the bound of T_{2112}). We are optimistic improvements are possible for larger q as well.

Exponent Comparison. Combining our result with [KK19, Har21] shows that, when restricted to rank bounds on small tensors over the same variable set as $\langle 2,2,2\rangle$, the best known upper bounds have $\frac{3}{2}\omega_\ell < \omega_B < \omega$. By comparison, the asymptotically best known upper bounds have $\frac{3}{2}\omega_\ell = \omega_B = \omega$. It would be exciting to determine the relationship between these exponents in the asymptotic setting, perhaps using fine-grained reduction techniques. Indeed, although we know $\omega_B \leq \omega$, it's not clear in general what the relationship between ω_ℓ and ω_B should be. Neither problem is known to be reducible to the other. Moreover, even our 'efficacy' approach for the light bulb problem is incomparable to the 'support rank' [CU13] and 'probabilistic rank' [KK19] approaches: our approach applies to a wider class of tensors, but the bound in Theorem 1.1 becomes worse when T has large or negative coefficients, whereas 'support rank' and 'probabilistic rank' aren't impacted by what the coefficients are. Nonetheless, from the small tensor regime, it appears plausible that $\omega_\ell < \omega_B$; it may be worth investigating whether other problems which are known to be reducible to matrix multiplication can actually be reduced to the light bulb problem instead!

Another advantage of our new algorithm is that it is not necessarily restricted to give exponents which are $\geq 4/3$. Recall that using the known bound $\omega_\ell \leq \frac{2}{3}\omega$, even if $\omega = 2$, one could only prove $\omega_\ell \leq 4/3$. Achieving an exponent less than $4/3$ requires another approach, and our Theorem 1.1 appears promising since it doesn't seem to have any such restrictions.

Tensors with Undesirable Terms. One motivation for this work is to use tensors with 'undesirable' terms which, in other contexts, make them unusable. Typically one would expend rank to remove those terms, but one could design faster algorithms by allowing them to contribute to the efficacy of the tensor instead. We've already discussed the case of tensors from border rank upper bounds via our example T_{2112}. Tensors with undesirable terms also arise in the Laser method [Str87], the tool used to design the best known upper bounds on ω. A key step at the end of the Laser method, which was improved but not entirely removed in recent work of Alman and Vassilevska Williams [AW21], removes such undesirable terms. Leaving them in to contribute to the efficacy could lead to asymptotically faster algorithms.

Further Generalizations of the Light Bulb Problem. In Section 4 below, we show that our algorithm can also solve a generalization of the light bulb problem where each group of coordinates of the 'correlated pair' are sampled from *any* non-uniform joint distribution. Other generalizations have also been previously considered [Val12, KKK18], including a variant with many correlated pairs to find, and an 'outlier correlation detection' variant where we are promised that the correlated pair has correlation ρ, and all other pairs have correlation at most τ, for parameters $0 < \tau < \rho < 1$. Our approach can also solve these generalizations, by using the same techniques from prior work (which essentially amplify the differences in correlations by

taking large Kronecker powers of the input vectors), since we focus on finding the correlated vectors after this amplification step. The details, which are essentially the same as in the past work, are omitted here. As discussed earlier, the best known algorithms for generalizations to other learning problems such as learning sparse parities or Juntas with noise also come from reductions to the light bulb problem [Val12].

Other Closest Pair Problems. The light bulb problem is an average case version of the bichromatic $(1+\varepsilon)$-approximate closest pair problem, where one is given as input two sets X, Y of n points from a metric space, and one wants to find $x^* \in X$ and $y^* \in Y$ satisfying $\mathrm{dist}(x^*, y^*) \leq (1+\varepsilon) \cdot \min_{x \in X, y \in Y} \mathrm{dist}(x, y)$. Similar to the previous state of the art for the light bulb problem, for many popular metric spaces, there are two known approaches for solving this problem: one based on matrix multiplication which is faster when $\varepsilon > 0$ is small [AW15, ACW16, ACW20], and one based on locality-sensitive hashing which is faster when ε is larger [AR15, ALRW17]; see also [AIR18]. To our knowledge, our hashing-based algorithm is the first to successfully combine matrix multiplication and hashing methods for any such problem. It is not hard to see that more straightforward ways to combine the two, such as hashing into smaller buckets and then using matrix multiplication within each bucket, cannot be faster than just using one of the two techniques on its own; we get around this by carefully choosing a hash function which correlates well with our chosen tensor. It would be exciting to apply a similar technique to other nearest neighbor search problems.

1.8 Algorithm Overview

Although Theorem 1.1 has a simple form (perhaps reminiscent of the bound $\omega \leq \log(\mathrm{rank}(\langle q, q, q \rangle))/\log(q)$ which follows from the simple recursive argument), the algorithm itself involves a number of subtle steps in the case when T is not 'symmetric enough', and the proof of correctness is ultimately quite involved. At a high level, the elaborate probabilistic analyses which arise in prior works on the light bulb problem [Val12, KKK18], wherein one needs to prove tail bounds on sums of correlated events, return in full force when combined with errors which arise from using the tensor T instead of matrix multiplication. We end up applying a simple variant on the Laser method to the tensor T to 'regularize' it without changing $\mathrm{eff}(T)$ too much, to help with the analysis.[9]

We focus here on describing the algorithm for Theorem 1.1 in the case when the tensor T is sufficiently 'symmetric' (as is the case for the three tensors described in Figure 1). Afterwards we will briefly discuss how we deal with asymmetric tensors, and how we extend our result using hashing to Theorems 1.3 and 1.4.

The main algorithm is given in Algorithm 1. There are two key results we need to prove its correctness. First, because of how $\mathrm{eff}_{i,j}$ is defined, if $|X_i| \cdot |Y_j| \leq \mathrm{eff}_{i,j}^2(T^{\otimes N})$ but $C[i,j] \geq \mathrm{eff}_{i,j}^2(T^{\otimes N})$, this means the correlated pair is likely to be in X_i and Y_j. Roughly, we prove the fact that $\mathrm{eff}_{i,j}(T^{\otimes N})$ is so large *means that* random noise cannot explain $C_k[i,j]$ being so large for too many k. Second, there is a decent probability that the bucketing used by the algorithm will result in copies of the correlated pair being put into X_i and Y_j for which $\mathrm{eff}_{i,j}(T^{\otimes N})$ is large enough.

This second result requires some work since whether or not the planted pair has been put into the pair of groups (X_i, Y_j) is *not* independent of whether it has been put into other pairs of groups. Moreover, it becomes more complicated in the case when the set $\{(i,j) \in [q]^N : \mathrm{eff}_{i,j} \geq g^2\}$ is 'skewed' and mostly consists of pairs in a small number of rows or columns. In this case, we modify our algorithm by alternatingly applying either T or its (appropriately defined) transpose. After this transformation, the large efficacies are 'balanced enough' that a second moment method can be used to imply our second property.

Finally, as discussed in Section 1.6 above, the idea behind Theorems 1.3 and 1.4 is to modify Line 8 of the algorithm to sample the indices $i_1, \ldots, i_t$ according to a locality-sensitive hash function. This further complicates the analysis: not only are the pairs (X_i, Y_j) which the planted pair has been put into not independent of each other, but even the buckets $X_{i_1}, \ldots, X_{i_t}$ which a single one of the planted vectors has

[9] Alman [Alm18] recently simplified some steps in prior algorithms for the light bulb problem using the polynomial method, but using our tensors in Alman's approach doesn't seem to work since Alman creates matrices with large entries to multiply.

Algorithm 1 Light bulb algorithm for Theorem 1.1

procedure LIGHTBULB($x_1 \cdots x_n, y_1 \cdots y_n \in \{-1,1\}^d, T$)
▷ $x_1, \cdots, x_n, y_1, \cdots, y_n$ are input vectors; T is a tensor with $q_i = q_j = q_k = q$
Let N be such that $n^{\omega_\ell} = \mathrm{rank}(T)^N$. ▷ $\omega_\ell = \frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))}$
Calculate $\mathrm{eff}_{i,j}(T^{\otimes N})$ for all $i, j \in [q]^N$. ▷ Can be done in negligible $\tilde{O}(q^{2N})$ time
Let g be such that $g^2 \cdot |\{i, j \in [q]^N : \mathrm{eff}_{i,j}(T^{\otimes N}) \geq g^2\}|$ is maximized, and set $t \leftarrow q^N g/n$.
$X_1, \cdots, X_{q^N}, Y_1, \cdots, Y_{q^N} \leftarrow \emptyset$.
for $i \in [n]$ **do**
Uniformly independently at random pick $i_1, \cdots, i_t$ and $j_1, \cdots, j_t$ from $[q^N]$.
Add i to all the sets $X_{i_1}, \cdots, X_{i_t}, Y_{j_1}, \cdots, Y_{j_t}$.
end for
For all $i \in [q]^N$, let $a_i := \sum_{j \in X_i} x_j$ and $b_i := \sum_{j \in Y_i} y_j$.
Let $A = [a_1^\top, \cdots, a_{q^N}^\top]$ be $q^N \times d$ matrix of all $a_i^\top$ vectors.
Let $B = [b_1^\top, \cdots, b_{q^N}^\top]$ be the $q^N \times d$ matrix of all $b_i^\top$ vectors. ▷ Assume $d = q_k^N$.
Random multiply each row of A and B by -1 or 1.
Recursively apply T to 'multiply' A and $B^\top$ and get their 'product' C. ▷ Take time $\tilde{O}(\mathrm{rank}(T)^N)$.
Do the multiplication $100 \log n$ times to get $C_1, \cdots, C_{100 \log n}$. ▷ Each time redo from beginning using fresh inputs.
Find (i, j) such that, there are at least $20 \log n$ different $k \in [100 \log n]$ that, $C_k[i, j] \geq 10\, \mathrm{eff}_{i,j}^2(T^{\otimes N}) \geq 10 g^2$.
end procedure

been put into are not independent. We address this by independently perturbing t copies of each input vector before applying a locality-sensitive hash function to them so that the different buckets are 'sufficiently independent'. The key behind applying our approach to tensors T other than T_{2112} in Theorem 1.4 is to do this perturbation in a biased way which correlates with the efficacy matrix of T. Fortunately, although the analysis requires these probabilistic analyses and explicitly analyzing the Kronecker power $T^{\otimes N}$, the algorithm itself is simple and only applies T in the usual recursive way to pairs of matrices.

1.9 Outline

The remainder of our paper is organized as follows. After the preliminaries in Section 2, we prove Theorem 1.1 in Section 3, then we prove Theorem 1.3 in Section 4. In Section 5 we define and give the rank expressions for the tensors in Figure 1, including introducing our new tensor T_{2112}. In Section 6 we prove Theorem 1.4 that hashing can be used to improve the algorithm from most tensors. Finally, in Appendix A we discuss techniques from prior work for vector aggregation.

2 Preliminaries

Notation For positive integer q, write $[q] := \{1, 2, 3, \ldots, q\}$.

For an event F, write $[F]_\mathbf{1}$ to be 1 if F happens, and 0 if F does not happen.

For vectors $v \in \mathbb{R}^d$, and $\ell \in [d]$, we write $v[\ell]$ to denote entry ℓ of v. Similarly, for matrices $M \in \mathbb{R}^{d_1 \times d_2}$, and $\ell_1 \in [d_1], \ell_2 \in [d_2]$, we write $M[\ell_1, \ell_2]$ to denote the corresponding entry of M.

Multinomial Coefficients If $a_1, a_2, \ldots, a_k \in [0, 1]$ and $N \in \mathbb{N}$ are such that $\sum_{i=1}^k a_i = 1$, and $a_i \cdot N$ is an integer for all i, then we write the multinomial coefficient:

$$\binom{N}{\{a_i \cdot N\}_{i \in [k]}} := \prod_{i=1}^k \binom{N \cdot (1 - \sum_{j=1}^{i-1} a_j)}{a_i \cdot N}.$$

Standard bounds show that as $N \to \infty$, we have

$$\binom{N}{\{a_i \cdot N\}_{i\in[k]}} = \left(\prod_{i=1}^{k} a_i^{-a_i}\right)^{N-o(N)}.$$

Chebyshev's inequality Chebyshev's inequality says that if $U \in \mathbb{R}$ is a random variable with finite mean and finite non-zero variance, then for any real $k > 0$ we have

$$\Pr\left[|U - \mathbb{E}[U]| \geq k \cdot \sqrt{\mathrm{var}[U]}\right] \leq \frac{1}{k^2}.$$

Second Moment Method The second moment method says that if $U \in \mathbb{R}$ is a random variable such that U is always nonnegative, and $\mathrm{var}[U]$ is finite, then

$$Pr[U > 0] \geq \frac{(\mathbb{E}[U])^2}{\mathbb{E}[U^2]}.$$

Tensors For positive integer q, q_k, let $\mathsf{X} = \{\mathsf{X}_{i,k}\}_{i\in[q],k\in[q_k]}$, $\mathsf{Y} = \{\mathsf{Y}_{j,k}\}_{j\in[q],k\in[q_k]}$, and $\mathsf{Z} = \{\mathsf{Z}_{i,j}\}_{i,j\in[q]}$. Most of the tensors in this paper will be over these sets, and we call a tensor over these sets a $\langle q, q, q_k\rangle$-sized tensor.

A tensor T over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ is a trilinear form in $\mathbb{R}^{|\mathsf{X}|\times|\mathsf{Y}|\times|\mathsf{Z}|}$. For $i, i', j, j' \in [q]$ and $k, k' \in [q_k]$ we write $T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'})$ for the coefficient of the term $\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}$ in T. In other words, we can write:

$$T = \sum_{i,i',j,j'\in[q],k,k'\in[q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}) \cdot \mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}.$$

We say X are the x-variables of T, Y are the y-variables of T, and Z are the z-variables of T.

The matrix multiplication tensor $\langle q, q, q_k\rangle$ is a tensor over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ given by

$$\langle q, q, q_k\rangle := \sum_{i,j\in[q],k\in[q_k]} \mathsf{X}_{i,k}\mathsf{Y}_{j,k}\mathsf{Z}_{i,j}.$$

Tensor Rank A tensor T over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ has rank 1 if it can be written in the form

$$T = \left(\sum_{i\in[q],k\in[q_k]} a_{i,k}\mathsf{X}_{i,k}\right)\left(\sum_{j\in[q],k\in[q_k]} b_{j,k}\mathsf{Y}_{j,k}\right)\left(\sum_{i,j\in[q]} c_{i,j}\mathsf{Z}_{i,j}\right)$$

for coefficients $a_{i,k}, b_{j,k}, c_{i,j} \in \mathbb{R}$. More generally, $\mathrm{rank}(T)$ is the minimum number of rank 1 tensors whose sum is T.

Kronecker Product If $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathsf{X}', \mathsf{Y}', \mathsf{Z}'$ are sets of variables, T is a tensor over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$, and T' is a tensor over $\mathsf{X}', \mathsf{Y}', \mathsf{Z}'$, then the Kronecker product $T \otimes T'$ is a tensor over $\mathsf{X}\times\mathsf{X}', \mathsf{Y}\times\mathsf{Y}', \mathsf{Z}\times\mathsf{Z}'$ given by, for $x \in \mathsf{X}, x' \in \mathsf{X}', y \in \mathsf{Y}, y' \in \mathsf{Y}', z \in \mathsf{Z}, z' \in \mathsf{Z}'$,

$$T \otimes T'((x, x')(y, y')(z, z')) = T(xyz) \cdot T'(x'y'z').$$

Notice in particular that for positive integers q, q', q_k, q'_k we have $\langle q, q, q_k\rangle \otimes \langle q', q', q'_k\rangle = \langle qq', qq', q_kq'_k\rangle$. (Here, we say two tensors are equal if they are the same up to renaming variables.) We can view $\langle qq', qq', q_kq'_k\rangle$ as a tensor whose X-variables are either $\{\mathsf{X}_{i,k}\}_{i\in[q\cdot q'],k\in[q_k\cdot q'_k]}$ or $\{\mathsf{X}_{(i,i'),(k,k')}\}_{i\in[q],i'\in[q'],k\in[q_k],k'\in[q'_k]}$. These are the same up to a natural bijection, and we will use both notations interchangeably.

For a tensor T over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ and positive integer k, we define the Kronecker power $T^{\otimes k}$ to be the Kronecker product of k copies of T. It is a tensor over $\mathsf{X}^k, \mathsf{Y}^k, \mathsf{Z}^k$, and its coefficients are all the products of k coefficients of T.

Applying a Tensor to Matrices If T is a tensor over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$, and $A, B \in \mathbb{R}^{q \times q_k}$ are matrices, then the result of applying T to A and B is a matrix $C \in \mathbb{R}^{q \times q}$ given by

$$C[i,j] = \sum_{i',j' \in [q], k,k' \in [q_k]} T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j}) \cdot A[i',k] \cdot B[j',k'].$$

The usual recursive algorithm (similar to Strassen's algorithm) shows that, for positive integers N, the tensor $T^{\otimes N}$ can be applied using only $\tilde{O}(\mathrm{rank}(T)^N)$ field operations, or the improved bound $O(\mathrm{rank}(T)^N)$ when $\mathrm{rank}(T) > q^2$.

Tensor Reflection For a tensor T over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$, its reflection $T^\top$ is another tensor over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ given by, for $i, i', j, j' \in [q]$ and $k, k' \in [q_k]$,

$$T^\top(\mathsf{X}_{i,k}\mathsf{Y}_{j,k'}\mathsf{Z}_{i',j'}) = T(\mathsf{X}_{j,k'}\mathsf{Y}_{i,k}\mathsf{Z}_{i',j'}).$$

This swaps the roles of the X and Y variables.

Kronecker Products of Matrices and Vectors If $A \in \mathbb{R}^{n_a \times m_a}$ and $B \in \mathbb{R}^{n_b \times m_b}$ are matrices, one can analogously define their Kronecker product $A \otimes B \in \mathbb{R}^{n_a n_b \times m_a m_b}$ by, for $i \in [n_a], i' \in [n_b], j \in [m_a], j' \in [m_b]$, $A \otimes B[(i,i'),(j,j')] = A[i,j] \cdot B[i',j']$. Similarly, for vectors $u \in \mathbb{R}^{n_a}, v \in \mathbb{R}^{n_b}$, one can define $u \otimes v \in \mathbb{R}^{n_a n_b}$ by $u \otimes v[(i,i')] = u[i] \cdot v[i']$.

Suppose P is a property of vectors which is preserved under Kronecker product, i.e., if u, v have the property, then so does $u \otimes v$. One example is the property of whether $\|v\|_2 \geq 1$. For a tensor T over $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$, let $S_P(T) \in \mathbb{R}^{q \times q}$ denote the matrix such that $S_P(T)[i,j] = 1$ if the vector $(T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j}))_{i',j' \in [q], k,k' \in [q_k]}$ has property P, and $S_P(T)[i,j] = 0$ otherwise. Then, we can see that $S_P(T^{\otimes N}) = S_P(T)^{\otimes N}$. This will be particularly helpful to us in the case when P is the property that $\mathrm{eff}_{i,j}(T) \geq f$ for some threshold f. (See Definition 3.1 below for the the definition of eff.)

3 Algorithm for the light bulb problem

Definition 3.1 (Efficacy)**.** Given any $\langle q, q, q_k \rangle$-sized tensor T, for $i, j \in [q]$, we define the (i,j)-*efficacy of* T as:

$$\underset{i,j}{\mathrm{eff}}(T) := \frac{\sum_{k \in [q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k}\mathsf{Z}_{i,j})}{\sqrt{\sum_{i',j' \in [q], k,k' \in [q_k]} T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j})^2}}.$$

We further define the *efficacy of* T as:

$$\mathrm{eff}(T) := \sqrt{\sum_{i \in [q]} \sum_{j \in [q]} \left(\underset{i,j}{\mathrm{eff}}(T)\right)^2}.$$

Note that if T, T' are two tensors, for $(i,i'),(j,j') \in [q]^2$, we have $\mathrm{eff}_{(i,i'),(j,j')}(T \otimes T') = \mathrm{eff}_{i,j}(T) \cdot \mathrm{eff}_{i',j'}(T')$.

We now begin giving our algorithm for the light bulb problem. Our goal is to analyze Algorithm 1 in order to prove Theorem 1.1. In particular, we assume throughout this section that T is such that the aggregation step (lines 11, 12, , 13) take negligible time compared to the rest of the algorithm; in Appendix A below, we show how to modify T, if necessary, so that this is the case.

Theorem 3.2. *Suppose T is a $\langle q, q, q_k \rangle$-sized tensor. For any $f \geq 1$, and any set $S_f \subseteq [q]^2$ such that $\mathrm{eff}_{i,j}(T) \geq f$ for all $(i,j) \in S_f$, we have $\omega_\ell \leq \log(\mathrm{rank}(T) \cdot q^2/|S_f|)/\log(f \cdot q)$.*

Proof. Suppose we are given as input $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{-1,1\}^d$ which are all generated independently and uniformly at random except for an unknown $(i^*, j^*) \in [n]^2$ with $\langle x_{i^*}, y_{j^*} \rangle \geq \rho \cdot d$. Permute the inputs at random so that (i^*, j^*) is a uniformly random pair in $[n]^2$.

We can solve the light bulb problem using $O(\log n)$ calls of its decision version – we randomly take half x and half y, and for the decision problem, we need to distinguish between two cases 1). all inputs are uniformly at random in $\{-1,1\}^d$, and 2). there exists one correlated (i^*, j^*) pair. From now on, we will only consider the decision version.

Let $m = (\frac{20n}{\rho})^{\frac{1}{1+\log_q(f)}}$ and let $g = n/m$. Partition $x_1, \ldots, x_n$ into m groups $X_1, \ldots, X_m$ of size g each, and partition $y_1, \ldots, y_n$ into m groups $Y_1, \ldots, Y_m$ of size g each. For each $i \in [m]$, create vectors $a_i, b_i \in \mathbb{R}^d$ given by $a_i = \sum_{u \in X_i} u$ and $b_i = \sum_{v \in Y_i} v$. (These vectors aggregate all the data points which were put into the same group; we will see soon that if groups i and j contain the correlated pair, then a_i and b_j are still somewhat correlated.) Let $s_a, s_b \in \{-1,1\}^m$ be two vectors whose entries are independently uniformly sampled from $\{-1,1\}$. Finally, we form the matrices $A, B \in \mathbb{R}^{m \times d}$ whose rows are $s_a[1] \cdot a_1, \ldots, s_a[m] \cdot a_m$ and $s_b[1] \cdot b_1, \ldots, s_b[m] \cdot b_m$, respectively.

For simplicity, let us assume that $d = m^{\log(q_k)/\log(q)}$ so that $T^{\otimes c}$ is a $\langle m, m, d \rangle$-sized tensor that can be used on A and B, where $c = \log(m)/\log(q)$. As discussed in the introduction, if one would like to remove this requirement, then using the 'compressed matrices' method introduced in [KKK18, Section 4.2], one can 'expand' lower-dimensional vectors, and hence relax this assumption to only require $d \geq \Omega(\log n)$ while only decreasing ρ by a negligible factor[10]; see Appendix A below for more details.

We now apply the usual recursive algorithm using the tensor T to the matrices A and B^T, resulting in the matrix $C \in \mathbb{R}^{m \times m}$. The running time is $\tilde{O}(m^{\log(\mathrm{rank}(T))/\log(q)}) = \tilde{O}(n^{\log(\mathrm{rank}(T))/\log(qf)})$. The output C is the result of applying the tensor $T^{\otimes c}$ to the matrices $X = A$ and $Y = B^T$, so that each entry $C[i,j]$ is the sum:

$$C[i,j] = \sum_{i_a, j_b \in [q^c], k_a, k_b \in [q_k^c]} T^{\otimes c}(X_{i_a,k_a} Y_{j_b,k_b} Z_{i,j}) \cdot A[i_a, k_a] \cdot B[j_b, k_b]. \tag{1}$$

Consider the product of two terms $A[i,k_a] \cdot B[j,k_b]$. These are distributed as follows:

- If there's a planted pair (x_{i^*}, y_{j^*}) and $k_a = k_b$, $x_{i^*} \in X_i$, and $y_{j^*} \in Y_j$, then this is the sum of g^2 random $\{-1,1\}$ variables which are pairwise-independent from each other. They all have mean 0 except one of them has mean ρ, so the entire variable $A[i,k_a] \cdot B[j,k_b]$ has mean ρ and variance g^2.
- Otherwise, it is the sum of g^2 uniformly random pairwise-independent $\{-1,1\}$ variables, which has mean 0 and variance g^2.

Let's compute the variance of $C[i,j]$. We showed earlier that each term $A[i_a,k_a] \cdot B[j_b,k_b]$ has variance g^2, so $T^{\otimes c}(X_{i_a,k_a} Y_{j_b,k_b} Z_{i,j}) \cdot A[i_a,k_a] \cdot B[j_b,k_b]$ has variance $T^{\otimes c}(X_{i_a,k_a} Y_{j_b,k_b} Z_{i,j})^2 \cdot g^2$. Since we use s_a, s_b entry-wise independently sampled from $\{-1,1\}$, the terms in the sum (1) are pairwise-independent of each other. It follows that regardless of whether there's a planted pair, every $C[i,j]$ has variance:

$$\mathrm{var}[C[i,j]] = g^2 \cdot \sum_{i_a, j_b \in [q^c], k_a, k_b \in [q_k^c]} T^{\otimes c}(X_{i_a,k_a} Y_{j_b,k_b} Z_{i,j})^2.$$

[10] In fact, the result of the compressed matrices method gives that if x_i, y_j are not the correlated pair, the the entries of the entry-wise product $x_i \circ y_j$ are only pairwise-independent of each other, and not fully independent. (This is because they are products of different entries of the original vectors.) As we will see below, this pairwise-independence suffices for our algorithm.

The compressed matrices technique particularly speeds up the time to compute the aggregated vectors a_i, b_j so that it is negligible compared to the remaining running time of the algorithm, and one can confirm that it remains negligible here. We refer the reader to [KKK18, Section 4.4] for more details.

Next let's compute the mean of $C[i,j]$. If there's no planted pair, every $C[i,j]$ has mean 0. If the planted pair (x_{i^*}, y_{j^*}) exists, let's assume $x_{i^*} \in X_i$ and $y_{j^*} \in Y_j$ and only consider the mean of $C[i,j]$. Recall that $A[i_a,k_a]\cdot B[j_b,k_b]$ has mean nonzero only if $k_a = k_b$, $i_a = i$ and $j_b = j$. It follows by linearity of expectation that

$$\begin{aligned}\mathbb{E}[C[i,j]] &= \sum_{k\in[q_k^c]} T^{\otimes c}(X_{i,k}Y_{j,k}Z_{i,j})\cdot \mathbb{E}[A[i,k]\cdot B[j,k]] \\ &= \rho\cdot s_a[i]\cdot s_b[j]\cdot \sum_{k\in[q_k^c]} T^{\otimes c}(X_{i,k}Y_{j,k}Z_{i,j}).\end{aligned}$$

To summarize: when there's no planted pair, every $C[i,j]$ has mean 0 and variance

$$g^2\cdot \sum_{i_a,j_b\in[q^c],k_a,k_b\in[q_k^c]} T^{\otimes c}(X_{i_a,k_a}Y_{j_b,k_b}Z_{i,j})^2.$$

When the planted pair exists, and x_{i^*} is in X_i, y_{j^*} is in Y_j, and the entry (i,j) is 'good' (We say (i,j) is 'good' if $\mathrm{eff}_{i,j}(T^{\otimes c}) \geq f^c$), the ratio of its mean and standard deviation is at least:

$$\begin{aligned}&\left|\frac{\rho\cdot s_a[i]\cdot s_b[j]\cdot\sum_{k\in[q_k^c]} T^{\otimes c}(X_{i,k}Y_{j,k}Z_{i,j})}{g^2\cdot\sum_{i_a,j_b\in[q^c],k_a,k_b\in[q_k^c]} T^{\otimes c}(X_{i_a,k_a}Y_{j_b,k_b}Z_{i,j})^2}\right| \\ =&\ \frac{\rho}{g}\cdot \operatorname*{eff}_{i,j}(T^{\otimes c}) \\ \geq&\ \frac{\rho}{g}\cdot f^c \\ =&\ \frac{\rho\cdot m}{n}\cdot f^c \\ \geq&\ \frac{\rho\cdot m}{n}\cdot f^{\log(m)/\log(k)} \\ =&\ \frac{\rho}{n}\cdot m^{1+\log(f)/\log(k)} \\ =&\ \frac{\rho}{n}\cdot\frac{20\cdot n}{\rho} \\ =&\ 20.\end{aligned}$$

Therefore, follows by Chebyshev's inequality, when there's no planted pair, $C[i,j] \leq 10\,\mathrm{var}[C[i,j]]$ with probability ≥ 0.99 for all $(i,j)\in[m]^2$, and when there exists a planted pair in a 'good' entry (i^*,j^*), then $C[i^*,j^*] \geq 10\,\mathrm{var}[C[i^*,j^*]]$ with probability ≥ 0.99. Thus, we can independently repeating $O(\log n)$ times to distinguish the two cases with polynomially-low error.

Let $|S_f| = q^\alpha$. Since $\mathrm{eff}_{(i,i'),(j,j')}(T\otimes T') = \mathrm{eff}_{i,j}(T)\cdot\mathrm{eff}_{i',j'}(T')$, there are at least $|S_f|^c$ pairs of $(i,j)\in[m]^2$ that $\mathrm{eff}_{i,j}(T^{\otimes c}) \geq f^c$. So the planted pair has $|S_f|^c/q^{2c} = q^{-(2-\alpha)c}$ probability going to a 'good' entry. We repeat $q^{(2-\alpha)c}\log n$ times to make sure the planted pair goes to a 'good' entry at least once with high probability, therefore we can distinguish the planted-pair case and the non planted-pair case.

Running time: Each run cost $\tilde{O}(n^{\log(\mathrm{rank}(T))/\log(qf)})$ time. We repeat the whole procedure $O(n^{(2-\alpha)\log(q)/\log(qf)}\cdot\log^2 n)$ times to succeed with high probability.

The total running time is $\tilde{O}(n^{\log(\mathrm{rank}(T))/\log(qf)}\cdot n^{(2-\alpha)\log(q)/\log(qf)}) = \tilde{O}(n^{\log(\mathrm{rank}(T)q^{2-\alpha})/\log(qf)}) = \tilde{O}(n^{\log(\mathrm{rank}(T)q^2/|S_f|)/\log(qf)})$, as desired. □

3.1 Improvement when S_f is not 'skewed'

We next show that in the special case when S_f is not too 'skewed', we can improve the bound of Theorem 3.2.

Recall that for $f \geq 1$, we chose a subset $S_f \subseteq [q]^2$ consisting of pairs $(i,j) \in [q]^2$ for which $\text{eff}_{i,j}(T) \geq f$. Let's define the following measurement of how a set S_f is closed to 'skewed'.

Definition 3.3 ($V_x(S)$ and $V_y(S)$)**.** For any set $S \subseteq [q]^2$, let's define $V_x(S) := \sum_{i \in [q]} |\{j \in [q] \mid (i,j) \in S\}|^2$, and similarly define $V_y(S) := \sum_{j \in [q]} |\{i \in [q] \mid (i,j) \in S\}|^2$.

Theorem 3.4. *Suppose T is a $\langle q, q, q_k \rangle$-sized tensor. For any $f \geq 1$, and any set $S_f \subseteq [q]^2$ such that $\text{eff}_{i,j}(T) \geq f$ for all $(i,j) \in S_f$, if $V_x(S_f), V_y(S_f) \leq |S_f|^{1.5}$, then, $\omega_\ell \leq \log(\text{rank}(T))/\log(f \cdot \sqrt{|S_f|})$.*

Proof. Let $E_1 = \log(\text{rank}(T))/\log(qf)$ and $E_2 = \log(q^2/|S_f|)/\log(qf)$.

Recall that in Theorem 3.2, we randomly partitioned the inputs $x_1, \ldots, x_n$ into sets $X_1, \ldots, X_m$, and the inputs $y_1, \ldots, y_n$ into sets $Y_1, \ldots, Y_m$, then we ran an algorithm which takes time $\tilde{O}(n^{E_1})$, and which will (with high probability) distinguish the all-random case and the planted-pair case, if x_{i^*} was put into set X_i and y_{j^*} was put into set Y_j such that $(i,j) \in S_f^{\otimes c}$. Call such (i,j) 'good'. The probability (i,j) is good is $(|S_f|^c/q^{2c}) = n^{-E_2}$.

In Theorem 3.2, we then repeated $\tilde{O}(n^{E_2})$ times, resulting in a final running time of $\tilde{O}(n^{E_1+E_2})$, but we will now instead do something more clever.

Let $t = n^{E_2/(2-E_2)}$. We will make t copies of each x_i and y_j vector, and then run the above algorithm on this new instance with $n \cdot t$ vectors, with the caveat that we never put two copies of the same x vector in the same group X_i, or two copies of the same y vector in the same group Y_j. The running time of this is $\tilde{O}((nt)^{E_1}) = \tilde{O}(n^{2 \cdot E_1/(2-E_2)})$, and the probability that a particular correlated pair will be put in a good pair of groups is now $(nt)^{-E_2}$. Since we made t copies of x_{i^*} and t copies of y_{j^*}, there are now t^2 correlated pairs, so the expected number of correlated pairs in a good pair of groups is

$$t^2 \cdot (nt)^{-E_2} = t^{2-E_2} \cdot n^{-E_2} = n^{\frac{E_2}{2-E_2} \cdot (2-E_2) - E_2} = n^{E_2 - E_2} = 1.$$

With more work, and using our bounds on $V_x(S_f)$ and $V_y(S_f)$, we can show that there is a positive constant probability that a correlated pair was put in a good pair of groups. See Lemma 3.7 below for the details. Hence, if we repeat $O(\log n)$ times, a correlated pair will be put in a good pair of groups with polynomially low error.

The proof that our algorithm can distinguish the planted-pair case with the non planted-pair case is almost identical to the proof in Theorem 3.2, except the following. In the old proof, when the planted pair is in X_i and Y_j and $\text{eff}_{i,j}(T^{\otimes c}) \geq f^c$, there's ≥ 0.99 probability that $C[i,j] \geq 10 \operatorname{var}[C[i,j]]$. Now, we can only prove the probability is ≥ 0.24 because duplicated vectors created correlation. Nonetheless, we can distinguish from the non-planted-pair case, in which every $C[i,j] \geq 10 \operatorname{var}[C[i,j]]$ with probability ≤ 0.01. We show probability ≥ 0.24 as follows.

Let

$$P[i_a, j_b] := \sum_{k_a, k_b \in [d]} T^{\otimes c}(X_{i_a,k_a} Y_{j_b,k_b} Z_{i,j}) \cdot a_{i_a,k_a} \cdot b_{j_b,k_b},$$

so that $C[i,j]$ can be written as

$$C[i,j] = \sum_{i_a, j_b \in [m]} P[i_a, j_b] \cdot s_a[i_a] \cdot s_b[j_b], \tag{2}$$

where $s_a, s_b \in \{-1,1\}^m$ has i.i.d. $\{-1,1\}$ entries.

Use Lemma 3.5 on $C[i,j]$, we conclude that with $\geq 1/4$ probability over random choice of s_a, s_b, $|C[i,j]| \geq |P[i,j]|$.

$\mathbb{E}[P[i,j]] = \rho \cdot \sum_{k \in [q_k^c]} T^{\otimes c}(X_{i,k} Y_{j,k} Z_{i,j})$ and $\operatorname{var}[P[i,j]] \leq \operatorname{var}[C[i,j]]$. By the same analysis in Theorem 3.2, with $\geq 1/4 - 0.01$ probability, $|C[i,j]| \geq |\mathbb{E}[P[i,j]]| - 10 \operatorname{var}[P[i,j]] \geq 10 \operatorname{var}[C[i,j]]$.

The resulting exponent is hence:

$$2 \cdot E_1/(2 - E_2) = 2 \cdot E_1/(\log(f^2 \cdot |S_f|)/\log(qf)) = \log(\operatorname{rank}(T))/\log(f \cdot \sqrt{|S_f|}).$$

□

Lemma 3.5. *Let $P \in \mathbb{R}^{n \times n}$ be a matrix. Let $a, b \in \{-1,1\}^n$ be entry-wise i.i.d. uniformly sampled from $\{-1,1\}$. Then with probability $\geq 1/4$,*

$$\left| \sum_{i,j \in [n]} a[i] \cdot b[j] \cdot P[i,j] \right| \geq |P[1,1]| \,. \tag{3}$$

Proof. Define $\overline{T} = \sum_{j \geq 2} P[1,j] \cdot b[j]$, and $T = \overline{T} + P[1,1] \cdot b[1]$. Regardless of how large $\overline{T}$ is, with $\geq 1/2$ probability over $b[1]$, $|T| \geq |P[1,1]|$.

Let S be the left side of Eq. (3). Notice that

$$S = \left| \sum_{i \geq 2} a[i] (\sum_{j \in [n]} P[i,j] \cdot b[j]) + T \cdot a[1] \right|,$$

use the same argument, we have $|S| \geq |T| \geq |P[1,1]|$ with $\geq 1/4$ probability. □

3.2 Probabilistic lemma for when S_f is not 'skewed'

Lemma 3.6. *Suppose q is a positive integer and $S \subseteq [q]^2$ is a nonempty subset. Let $V_x(S) := \sum_{i \in [q]} |\{j \in [q] \mid (i,j) \in S\}|^2$ and $V_y(S) := \sum_{j \in [q]} |\{i \in [q] \mid (i,j) \in S\}|^2$, and suppose that $V_x(S) \leq |S|^{1.5}$ and $V_y(S) \leq |S|^{1.5}$.*

Suppose we pick $S_x, S_y \subseteq [q]$ of size $|S_x|, |S_y| \geq q/\sqrt{|S|}$ independently and uniformly at random. Then, the probability that $|(S_x \times S_y) \cap S| > 0$ is at least $1/4$.

Proof. Let U denote the random variable $|(S_x \times S_y) \cap S|$. We will use the second moment method, which says that

$$\Pr[U > 0] \geq \frac{(\mathbb{E}[U])^2}{\mathbb{E}[U^2]}.$$

First, by linearity of expectation, we compute that $\mathbb{E}[U] = |S_x| \cdot |S_y| \cdot (|S|/q^2) = 1$.

Next, again by linearity of expectation, we compute:

$$
\begin{aligned}
\mathbb{E}[U^2] &= \sum_{(i,j)\in S} \sum_{(i',j')\in S} \Pr[i,i' \in S_x \text{ and } j,j' \in S_y] \\
&= \sum_{(i,j)\in S} \left(\left(\frac{1}{\sqrt{|S|}}\right)^2 + \left(\frac{1}{\sqrt{|S|}}\right)^3 \cdot |\{(i',j') \in S \mid i = i' \text{ xor } j = j'\}| \right. \\
&\quad \left. + \left(\frac{1}{\sqrt{|S|}}\right)^4 \cdot |\{(i',j') \in S \mid i \neq i' \text{ and } j \neq j'\}| \right) \\
&= \left(\frac{1}{\sqrt{|S|}}\right)^2 \cdot |S| + \left(\frac{1}{\sqrt{|S|}}\right)^3 \cdot (V_x(S) + V_y(S) - 2|S|) \\
&\quad + \left(\frac{1}{\sqrt{|S|}}\right)^4 \cdot (|S|^2 - V_x(S) - V_y(S) + |S|) \\
&= 2 + \frac{1}{|S|} - \frac{2}{|S|^{0.5}} + (V_x(S) + V_y(S)) \cdot \left(\frac{1}{|S|^{1.5}} - \frac{1}{|S|^2}\right) \\
&\leq 4 + \frac{1}{|S|} - \frac{4}{|S|^{0.5}} \\
&< 4,
\end{aligned}
$$

where the last step follows since $\frac{1}{s} - \frac{4}{\sqrt{s}} < 0$ for all $s \geq 1$.

In total, we get as desired that

$$\Pr[U > 0] \geq \frac{(\mathbb{E}[U])^2}{\mathbb{E}[U^2]} = \frac{1}{\mathbb{E}[U^2]} > \frac{1}{4}.$$

□

Lemma 3.7. *Suppose q is a positive integer and $S \subseteq [q]^2$ is a nonempty subset. Let $V_x(S) := \sum_{i\in[q]} |\{j \in [q] \mid (i,j) \in S\}|^2$ and $V_y(S) := \sum_{j\in[q]} |\{i \in [q] \mid (i,j) \in S\}|^2$, and suppose that $V_x(S) + V_y(S) \leq |S|^{1.5}$. Let c be a positive integer.*

Suppose we pick $S_x, S_y \subseteq [q^c]$ of size $|S_x|, |S_y| \geq q^c/\sqrt{|S|^c}$ independently and uniformly at random. Then, the probability that $|(S_x \times S_y) \cap S^{\otimes c}| > 0$ is at least $1/4$.

Proof. Apply Lemma 3.6 to $S^{\otimes c} \subseteq [q^c]^2$. Note that $V_x(S^{\otimes c}) = (V_x(S))^c$ and $V_y(S^{\otimes c}) = (V_y(S))^c$, so the conditions are still satisfied after taking the c^{th} Kronecker power. □

3.3 Symmetrizing a tensor to avoid skew

Definition 3.8. For a positive integer q, we say a set $S \subseteq [q]^2$ is *regular* if there are positive integers a and b such that, for all $i \in [q]$, $|\{j \in [q] \mid (i,j) \in S\}|$ is either equal to a or equal to 0, and similarly for all $j \in [q]$, $|\{i \in [q] \mid (i,j) \in S\}|$ is either equal to b or equal to 0.

Lemma 3.9. *Suppose q is a positive integer and $S \subseteq [q]^2$ is regular. Let $V_x(S) := \sum_{i\in[q]} |\{j \in [q] \mid (i,j) \in S\}|^2$ and $V_y(S) := \sum_{j\in[q]} |\{i \in [q] \mid (i,j) \in S\}|^2$. Then, $V_x(S) \cdot V_y(S) \leq |S|^3$.*

Proof. If S is empty, then the result holds since $V_x(S) = V_y(S) = |S| = 0$. Otherwise, assume without loss of generality that $(1,1) \in S$.

Let $a = |\{j \in [q] \mid (1,j) \in S\}|$, and $b = |\{i \in [q] \mid (i,1) \in S\}|$. Since S is regular, there are $|S|/a$ choices of $i \in [q]$ for which $|\{j \in [q] \mid (i,j) \in S\}| = a$, and so $V_x(S) = \frac{|S|}{a} \cdot a^2 = a \cdot |S|$. Similarly, $V_y(S) = b \cdot |S|$, which means $V_x(S) \cdot V_y(S) = a \cdot b \cdot |S|^2$.

Let $W := \{j \in [q] \mid (1,j) \in S\}$, so $|W| = a$. Next, for each $j \in W$, let $W_j := \{(i,j) \mid i \in [q] \text{ and } (i,j) \in S\} \subseteq S$. Let $W' := \bigcup_{j \in W} W_j \subseteq S$, and note that the W_j sets are disjoint, so $|W'| = \sum_{j \in W} |W_j|$. By definition of W, we know that W_j is nonempty for each $j \in W$, and so $|W_j| = b$. It follows that $|S| \geq |W'| = |W| \cdot b = a \cdot b$.

We thus get as desired that $V_x(S) \cdot V_y(S) = a \cdot b \cdot |S|^2 \leq |S|^3$. □

Theorem 3.10. *Suppose T is a $\langle q, q, q_k \rangle$-sized tensor. For any $f \geq 1$, and any regular set $S_f \subseteq [q]^2$ such that $\mathrm{eff}_{i,j}(T) \geq f$ for all $(i,j) \in S_f$, we have $\omega_\ell \leq \log(\mathrm{rank}(T))/\log(f \cdot \sqrt{|S_f|})$.*

Proof. Define $T' = T \otimes T^\top$ and $S'_f = S_f \otimes S_f^\top$. We can see that $V_x(S'_f) = V_y(S'_f) = V_x(S_f) \cdot V_y(S_f)$. By Lemma 3.9, this is at most $|S_f|^3 = |S'_f|^{1.5}$. Furthermore, for any $i, i', j, j' \in [q]$ such that $((i,j'),(j,i')) \in S'_f$, we have that $\mathrm{eff}_{(i,j'),(j,i')}(T') = (\mathrm{eff}_{i,j}(T)) \cdot (\mathrm{eff}_{j',i'}(T^\top)) \geq f^2$. We may thus apply Theorem 3.4 to T' and S'_f to yield the desired result. □

Theorem 3.11 (Restatement of Theorem 1.1)**.** *Suppose T is a $\langle q, q, q_k \rangle$-sized tensor, then*

$$\omega_\ell \leq \frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))}.$$

Proof. Let N be a sufficiently large positive integer. We will partition the set $[q]^N \times [q]^N$ into many regular sets in the following way. For any $(I,J) = ((i_1, \cdots, i_N), (j_1, \cdots, j_N)) \in ([q]^N)^2$, let $p \in [q]^2 \to \mathbb{N}$ be the counter, such that $p(u,v)$ counts the number of pairs (i_ℓ, j_ℓ) that equal to (u,v), and let $S_p \subseteq [q]^{2N}$ be the set including all such pairs (I,J) whose counter is p. We have that $\{S_p\}_{\{p\}}$ form a partition of $[q]^N \times [q]^N$. We can also see that for every p, S_p is a regular set (since its definition does not depend on the order of the N indices). Let $f_p = \Pi_{i,j \in [q]} \mathrm{eff}_{i,j}(T)^{p(i,j)}$, every pair $(I,J) \in S_p$ has $\mathrm{eff}_{I,J}(T^{\otimes N}) = f_p$. By Theorem 3.10, we have

$$w_\ell \leq \frac{\log(\mathrm{rank}(T^{\otimes N}))}{\log(\sqrt{f_p^2 \cdot |S_p|})}.$$

The next step is to choose the best p that maximize $f_p^2 \cdot |S_p|$. Note that the number of different p is upper bounded by N^{q^2}. And thus

$$\max_p f_p^2 \cdot |S_p| \geq \frac{1}{N^{q^2}} \sum_p f_p^2 |S_p| = \frac{1}{N^{q^2}} \sum_p \sum_{(I,J) \in S_p} \mathrm{eff}_{I,J}(T^{\otimes N})^2 = \frac{1}{N^{q^2}} \mathrm{eff}(T^{\otimes N})^2,$$

where the last step is because $\{S_p\}_{\{p\}}$ is the partition of $[q]^N \times [q]^N$ and the definition of $\mathrm{eff}(T)$.

Therefore, we have

$$w_\ell \leq \frac{\log(\mathrm{rank}(T^{\otimes N}))}{\log(\sqrt{f_p^2 \cdot |S_p|})} \leq \frac{\log(\mathrm{rank}(T^{\otimes N}))}{\log\left(\sqrt{\frac{1}{N^{q^2}} \mathrm{eff}(T^{\otimes N})^2}\right)} = \frac{N \log(\mathrm{rank}(T))}{\log(\sqrt{\frac{1}{N^{q^2}}}) + N \log(\mathrm{eff}(T))} \leq \frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))} + o(1),$$

and the result follows from taking $N \to \infty$. □

4 Solving the P-light bulb problem with locality-sensitive hashing

General Faster Algorithm. The general statement of Theorem 1.3 which applies to any tensor needs a few definitions. Let $q \geq 2$ be an integer, and $P \in \mathbb{R}^{q \times q}$ be a matrix of nonnegative real numbers whose entries sum to 1, but whose entries are *not* all equal to $1/q^2$. We say that two vectors $x, y \in [q]^d$ are *jointly sampled according to* P if, for each $\ell \in [d]$, the coordinates $x[\ell], y[\ell]$ are sampled independently of all other coordinates, and $(x[\ell], y[\ell]) = (i, j)$ with probability $P[i, j]$ for all $(i, j) \in [q]^2$.

We focus on a generalization of the light bulb problem which our algorithm is naturally able to solve. In the *P-light bulb problem*, one is given as input vectors $x_1, \ldots, x_n, y_1, \ldots, y_n \in [q]^d$ which are all independent and uniformly random except for a planted pair which has been jointly sampled according to P, and the goal is to find the planted pair. The light bulb problem with correlation ρ is a special case of this problem with $q = 2$, $P[0, 0] = P[1, 1] = (1 + \rho)/4$, $P[0, 1] = P[1, 0] = (1 - \rho)/4$. It could alternatively be viewed as a special case of this problem for any q which is a power of 2, along with the appropriately defined P.

4.1 Overview of the proof

Let x^*, y^* be the correlated pair. Since each bit of (x_i^*, y_i^*) is sampled according to the joint distribution P, the number of coordinates l such that $(x_l^*, y_l^*) = (i, j)$ will be proportional to $P[i, j]$ in expectation. In fact, we will assume that the number is equal to its expectation for all (i, j); this happens with decent probability, and will simplify our analysis. The assumption below that (x^*, y^*) falls into the set V_N (defined in Eq. (4) below) captures this property.

Given a tensor T of size $\langle q, q, q \rangle$ along with its eff matrix, our hope is that (x^*, y^*) falls into a bucket $(i, j) \in [q]^2$ with high $\mathrm{eff}_{i,j}$. However, depending on how the eff matrix correlates with the P distribution matrix, this may not be the case. To address this, we will choose a pair of stochastic matrices Q_x, Q_y (Def. 4.1 below) which we use to process vectors after they have been sampled, so that P after this transformation will be correlated with eff.

More precisely, we use Q_x and Q_y to decide which bucket every vector goes into as follows. Take a vector x as example. For every coordinate l, if $x_l = i$, we switch x_l to j with probability $Q_x[i, j]$. The final x after this transformation is the bucket we put this vector into. Vectors y are transformed in a similar way, but with the matrix Q_y. Note that different buckets may have different numbers of points since the matrices Q_x, Q_y are not necessarily doubly-stochastic. This needs to be taken into account since (x_i^*, y_i^*) wil only be detected it they are put into a bucket where $\mathrm{eff}_{i,j}^2$ is larger than the number of pairs of points. Below we will rescale eff by ∂_x and ∂_y (Def. 4.1 below) to "normalize" this effect.

Ultimately we need to optimize over choices of Q_x, Q_y to achieve the best running time. The number γ_{Q_x, Q_y} (Def. 4.1 below) indicates the performance of particular matrices Q_x, Q_y. The higher this number is, the better the choice of Q_x, Q_y.

Ultimately we will find using properties of the Kronecker power that many buckets shares the same property: they have the same chance that of containing the correlated pair (x^*, y^*), and they have the same eff value. We cluster these buckets into many groups, each specified by a mapping τ (Def. 4.6 below). We find the best cluster S_τ in Lemma 4.8 below, and our algorithm will only consider the buckets inside this cluster to find (x^*, y^*). (We will calculate that other clusters give a negligible additional probability of finding (x^*, y^*).) Similar to Theorem 3.4 above, we copy x^* and y^* multiple times to guarantee that there is a constant probability that at least one copy falls into a bucket in that cluster.

Similar to before, we will aim to use Lemma 3.6, and toward this goal, we need to ensure our cluster S_τ is not too "skewed". Similar to Section 3.3 above, we consider the Kronecker power of S_τ with its transpose $S_\tau^\top$ to "symmetrize" the cluster and avoid this issue. Section 4.4.2 below is devoted to dealing with this issue.

See Algorithm 2 below for the full algorithm description.

4.2 Preliminaries

Given any tensor T, we now define its P-$\operatorname{eff}(T)$, a generalization of $\operatorname{eff}(T)$. We start by defining some useful functions.

Definition 4.1 (effQ and γ)**.** Suppose T is a $\langle q, q, q_k\rangle$-sized tensor. Given two stochastic matrices $Q_x, Q_y \in \mathbb{R}^{q\times q}$, define $\partial_x(i) := \sum_{j\in[q]} Q_x[i,j]$ for $i \in [q]$, and same for ∂_y. we define the Q version of $\operatorname{eff}(T)$ as follows. For $i, j \in [q]$,

$$\underset{i,j}{\operatorname{effQ}}(Q_x, Q_y, T) := \frac{\sum_{k\in[q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k}\mathsf{Z}_{i,j})}{\sqrt{\sum_{i',j'\in[q],k,k'\in[q_k]} T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j})^2\partial_x(i')\partial_y(j')}}.$$

Given a joint probability matrix $P \in \mathbb{R}^{p\times p}$, we further define the performance of Q_x, Q_y as

$$\gamma_{Q_x,Q_y} := \prod_{i,j\in[q]} \left(\sum_{u,v\in[q]} Q_x[i,u]Q_y[j,v](\underset{u,v}{\operatorname{effQ}}(Q_x, Q_y, T))^2 \right)^{P[i,j]}.$$

Let γ be the best γ_{Q_x,Q_y} over all stochastic matrices Q_x, Q_y.

$$\gamma := \max_{Q_x,Q_y} \gamma_{Q_x,Q_y},$$

Note that effQ is multiplicative: suppose T, T' both have size $\langle q, q, q_k\rangle$, and Q_x, Q'_x, Q_y, Q'_y are all stochastic matrices, then for $(i,i'), (j,j') \in [q]^2$, we have $\operatorname{effQ}_{(i,i'),(j,j')}(Q_x \otimes Q'_x, Q_y \otimes Q'_y, T \otimes T') = \operatorname{effQ}_{i,j}(Q_x, Q_y, T) \cdot \operatorname{effQ}_{i',j'}(Q'_x, Q'_y, T')$.

Remark 4.2. *By choosing matrix Q_x, Q_y to be the matrix where each entry is $1/q$, $\operatorname{effQ}(T)$ is the same as $\operatorname{eff}(T)$ and thus $\gamma \geq \operatorname{eff}(T)^2/q^2$.*

We finally define:

$$P\text{-}\operatorname{eff}(T) := \sqrt{\gamma \cdot q^2}.$$

Here's our main theorem of this section.

Theorem 4.3. *Suppose T is a $\langle q, q, q_k\rangle$-sized tensor. Suppose there are n vectors uniformly independently sampled from $[q]^d$, with a planted pair (x^*, y^*) where each bit of them is sampled from a symmetric joint probability matrix $P \in \mathbb{R}^{q\times q}$.*

Let γ be as defined in Definition 4.1, then (x^*, y^*) can be found in $O(n^{\omega_P+o(1)})$ time, where

$$\omega_P \leq \frac{\log \operatorname{rank}(T)}{\log(q\gamma^{1/2})}.$$

Theorem 1.3 follows from Theorem 4.3 by finding the optimal Q for the tensor T_{2112} (see Section 5 for the definition of T_{2112}) and the matrix P arising from ρ in the light bulb problem; see the Example 4.4 below for more details.

Example 4.4. *Consider our standard light-bulb problem where the correlated pair has ρ-correlation, i.e.,*

$$P_\rho = \begin{pmatrix} \frac{1+\rho}{4} & \frac{1-\rho}{4} \\ \frac{1-\rho}{4} & \frac{1+\rho}{4} \end{pmatrix}.$$

Let $a \in [0,1]$ be some parameter, let both Q_x and Q_y be

$$Q_x = Q_y = \begin{pmatrix} 1-a & a \\ a & 1-a \end{pmatrix}.$$

Given our T_{2112} tensor, by definition of γ_{Q_x,Q_y} (Definition 4.1), we can calculate

$$\gamma_{Q_x,Q_y} = \left(2\cdot\left((1-a)^2+a^2\right)+1\cdot\left(2a(1-a)\right)\right)^{\frac{1+\rho}{2}}\left(2\cdot\left(2a(1-a)\right)+1\cdot\left((1-a)^2+a^2\right)\right)^{\frac{1-\rho}{2}}.$$

The optimal a is given as $a = \max\{0,(1-\sqrt{3\rho})/2\}$. Write ω_ρ to be the exponent ω_P in Theorem 4.3 given our $P = P_\rho$. Then we get

$$w_\rho = \begin{cases} \frac{2\log 5}{\log\left(6(1-\rho)^{-\rho/2}(1+\rho)^{\rho/2}(1-\rho^2)^{1/2}\right)} & \text{,when } \rho < 1/3 \\ \frac{4\log 5}{(5+\rho)\log 2} & \text{,when } 1/3 \le \rho \le 1. \end{cases}$$

In the remainder of this section, we prove Theorem 4.3.

4.3 Preparation before symmetrization

We will show that for every stochastic Q_x, Q_y, our algorithm gives an exponent $\omega_P \le 2\frac{\log \mathrm{rank}(T)}{\log(\gamma_{Q_x,Q_y})}$ and therefore, the theorem follows by choosing the best Q_x and Q_y. In the following, we assume Q_x, Q_y are fixed stochastic matrices.

Fix a joint probability matrix P. Let N be a sufficiently large positive integer so that $P[i,j]N$ are all integers.[11] Define the set V_N as all pairs of (x,y) where there are exactly $P[i,j]\cdot N$ number of coordinates l such that $(x[l],y[l]) = (i,j)$,

$$V_N := \{(x,y)\in[q]^N : \forall i,j\in[q],\ |\{l \mid (x[l],y[l]) = (i,j)\}| = P[i,j]\cdot N\}. \tag{4}$$

If the planted pair (x^*,y^*) is drawn from the joint distribution P, then there's a descent chance that $(x^*,y^*)\in V_N$.

Definition 4.5 (Distribution $\mathcal{D}^x$ and $\mathcal{D}^y$)**.** Let $i\in[q]$ and $Q_x\in\mathbb{R}^{q\times q}$ be a stochastic matrix, we define the distribution $\mathcal{D}^i_{Q_x}\in\mathbb{R}^q$ as

$$\mathcal{D}^i_{Q_x}(j) := Q_x[i,j],\ \forall j\in[q].$$

In another words, the distribution $\mathcal{D}^i_{Q_x}$ is generated by transforming i to j with probability $Q_x[i,j]$.

Let $x\in[q]^N$ be any vector, we define the distribution $\mathcal{D}^x_{Q_x^{\otimes N}}\in\mathbb{R}^{q^N}$ as

$$\mathcal{D}^x_{Q_x^{\otimes N}} := \otimes_{l=1}^N \mathcal{D}^{x[l]}_{Q_x}.$$

In another words, the distribution $\mathcal{D}^x_{Q_x^{\otimes N}}$ is generated by transforming independently each entry $x[\ell]$ to $x'[\ell]$ with probability $Q_x[x[\ell],x'[\ell]]$.

Similarly, we define

$$\mathcal{D}^y_{Q_y^{\otimes N}} := \otimes_{l=1}^N \mathcal{D}^{y[l]}_{Q_y}\in\mathbb{R}^{q^N}.$$

For simplicity, if Q_x and Q_y are clear from the context, we write $\mathcal{D}^x_{Q_x^{\otimes N}}$ as $\mathcal{D}^x$ and $\mathcal{D}^y_{Q_y^{\otimes N}}$ as $\mathcal{D}^y$.

We define $\mathcal{D}^{x,y} := \mathcal{D}^x\otimes\mathcal{D}^y$ to be their joint distribution. In particular, for $x',y'\in[q]^n$, we will write $\mathcal{D}^{x,y}(x',y') = \mathcal{D}^x(x')\cdot\mathcal{D}^y(y')$ to denote the probability that $\mathcal{D}^{x,y}$ outputs (x',y').

[11] We only need this property when constructing the set V_N. There is a negligible change in our algorithm if we round $P[i,j]\cdot N$ to be the integer closest to $P[i,j]\cdot N$ in the construction of V_N for large N.

Next, similar to the proof of Theorem 3.11, we will partition the entire space $[q]^N \times [q]^N$ into several regular sets.

Definition 4.6 (τ-partition)**.** Call a mapping $\tau : [q]^4 \to \{0, 1, \cdots, N\}$ *valid* if for all $i, j \in [q]$, we have $\sum_{u,v\in[q]} \tau(i, j, u, v) = N \cdot P[i, j]$. Fix a pair $(x^*, y^*) \in V_N$. Every pair $(x, y) \in [q]^N \times [q]^N$ *corresponds to* one valid mapping τ defined by $\tau(i, j, u, v) = |\{l : (x^*[l], y^*[l]) = (i, j) \text{ and } (x[l], y[l]) = (u, v)\}|$. For a valid mapping τ, let $S_\tau^{x^*,y^*} \subseteq [q]^{2N}$ be the set of all pairs (x, y) that correspond to τ. When x^*, y^* is clear from the context, we simply write S_τ instead of $S_\tau^{x^*,y^*}$.

We next make some key observations about valid mappings τ.

Fact 4.7. *Fix $(x^*, y^*) \in V_N$, then*

1. *$\{S_\tau\}_{\{\text{valid } \tau\}}$ is a partition of $[q]^N \times [q]^N$;*
2. *S_τ is regular for all valid τ;*
3. *Every $(x, y) \in S_\tau$ has the same $\mathcal{D}^{x^*,y^*}(x, y)$, since $\mathcal{D}^{x^*,y^*}(x, y) = \prod_{i,j,u,v}(Q_x[i, u]Q_y[j, v])^{\tau(i,j,u,v)}$ only depends on τ. For simplicity, we denote them as $\mathcal{D}_\tau$;*
4. *Every $(x, y) \in S_\tau$ has the same $\mathrm{effQ}_{x,y}(T^{\otimes N})$, since*

$$\underset{x,y}{\mathrm{effQ}}(Q_x^{\otimes N}, Q_y^{\otimes N}, T^{\otimes N}) = \prod_{i,j,u,v} \underset{u,v}{\mathrm{effQ}}(Q_x, Q_y, T)^{\tau(i,j,u,v)}$$

only depends on τ. For simplicity, we denote them as effQ_τ.

Proof. 1. This is because each pair (x, y) corresponds to exactly one valid τ.

2. For $i, j \in [q]$, let $Z_{i,j} := \{l \mid x^*[l] = i \text{ and } y^*[l] = j\} \subseteq [N]$, and let $(S_\tau)_{i,j} \subseteq [q]^{Z_{i,j}}$ be the set of pairs $(x, y) \in [q]^{Z_{i,j}}$ for which, for all $(u, v) \in [q]^2$, we have $|\{\ell \in Z_{i,j} \mid (x[\ell], y[\ell]) = (u, v)\}| = \tau(i, j, u, v)$. We can see that $(S_\tau)_{i,j}$ is regular since it is defined independently of the order of the indices. Thus, $S_\tau = \otimes_{i,j}(S_\tau)_{i,j}$, which is a Kronecker product of regular sets $(S_\tau)_{i,j}$, is also regular.

3 and 4. Proved in the statement. □

Lemma 4.8. *Let N be a sufficient large integer. Let $(x^*, y^*) \in V_N$ be any pair. Then there exists a valid mapping τ such that*

$$\mathcal{D}_\tau \cdot |S_\tau| \cdot \mathrm{effQ}_\tau^2 \geq \frac{1}{(N+1)^{q^4}} \gamma_{Q_x,Q_y}^N,$$

where γ_{Q_x,Q_y} is defined in Definition 4.1.

Proof.

$$\begin{aligned}
&\max_\tau \mathcal{D}_\tau \cdot |S_\tau| \cdot \mathrm{effQ}_\tau^2 \\
\geq\ &\frac{1}{(N+1)^{q^4}} \sum_\tau \mathcal{D}_\tau \cdot |S_\tau| \cdot \mathrm{effQ}_\tau^2 \\
=\ &\frac{1}{(N+1)^{q^4}} \sum_\tau \mathbb{E}_{(x,y)\sim\mathcal{D}^{x^*,y^*}}[\mathrm{effQ}_{x,y}^2(T^{\otimes N}) \cdot \left[(x, y) \in S_\tau\right]_1] \\
=\ &\frac{1}{(N+1)^{q^4}} \mathbb{E}_{(x,y)\sim\mathcal{D}^{x^*,y^*}}[\mathrm{effQ}_{x,y}^2(T^{\otimes N})],
\end{aligned}$$

where the first step follows because there are at most $(N+1)^{q^4}$ different valid τ, the second step follows from part 3 and part 4 of Fact 4.7, the third step follows from part 1 of Fact 4.7. We conclude by computing that

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim\mathcal{D}_{x^*,y^*}}[\mathrm{effQ}^2_{x,y}(T^{\otimes N})]\\
=&\mathbb{E}_{\substack{(x_1,y_1)\sim\mathcal{D}^{x_1^*,y_1^*}\\ \vdots \\ (x_N,y_N)\sim\mathcal{D}^{x_N^*,y_N^*}}}[\Pi_{i=1}^N \mathrm{effQ}^2_{x_i,y_i}(T)]\\
=&\prod_{i=1}^N \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}^{x_i^*,y_i^*}}[\mathrm{effQ}^2_{x_i,y_i}(T)]\\
=&\prod_{i,j\in[q]}\left(\sum_{u,v\in[q]} Q_x[i,u]Q_y[j,v]\mathrm{effQ}^2_{u,v}(T)\right)^{P_{i,j}N}\\
=&\gamma_{Q_x,Q_y}.
\end{aligned}$$

□

We also give this lemma for later use.

Lemma 4.9. *Suppose q is even, and $P \in \mathbb{R}^{q\times q}$ is a joint probability matrix. There exist two mappings $h, g : [q] \to \{-1, 1\}$ and a constant $\rho > 0$ such that:*

- *If b_0, b_1 are sampled independently from $[q]$, and at least one of them is sampled uniformly, then $\mathbb{E}[g(b_0)\cdot h(b_1)] = 0$, but*
- *If b_0, b_1 are sampled from $[q]$ according to P (so $(b_0, b_1) = (i, j)$ with probability $P[i, j]$), then $\mathbb{E}[g(b_0)\cdot h(b_1)] = \rho$.*

Proof. We construct g, h in a greedy fashion. Pick any row of P that is not uniform. Such a row exists since we assume P is not the uniform matrix. Fix g to map the column indices of the $q/2$ largest entries in that row to 1, and the others to -1. Let $v := P \cdot g \in \mathbb{R}^q$. Fix h to map the indices of the $q/2$ largest entries of v to 1 and the others to -1.

If b_0 or b_1 is sampled uniformly from $[q]$, then $g(b_0)$ or $h(b_1)$, respectively, is uniformly chosen from $\{-1, 1\}$ since h and g each map half of $[q]$ to 1 and the other half to -1. Hence, in this case, if b_0 and b_1 are sampled independently, then $\mathbb{E}[g(b_0)\cdot h(b_1)] = 0$.

Meanwhile, by our construction of h,

$$\mathbb{E}_{(b_0,b_1)\sim P}[g(b_0)\cdot h(b_1)] = \langle v, h\rangle > 0,$$

which is strictly larger than 0 because v is non-zero. We may thus pick $\rho = \langle v, h\rangle$. □

4.4 Proof of Theorem 4.3

In Lemma 4.8 from the previous section, we found the best mapping τ, and we aimed to use S_τ to detect the correlated pair. However, although S_τ is a regular set, it still can be "skewed" (in the sense of Section 3.2). In this section, at a high level, we are going to symmetrize S_τ using a Kronecker product with its transpose $S_\tau^\top$ to avoid skew. We will prove Theorem 4.3 in four steps.

Algorithm 2

1: **Input**
2: Let T be a $\langle q, q, q_k \rangle$-sized tensor.
3: Let $x_1, \cdots, x_n, y_1, \cdots, y_n \in [q]^d$ be $2n$ vectors with one planted pair (x^*, y^*).
4: Let $P \in \mathbb{R}^{q \times q}$ be joint-probability symmetric matrix for planted pair, that every bit $(x^*[l], y^*[l])$ is sampled according to P.
5: **Algorithm**
6: Let γ and its corresponding stochastic matrices $Q_x, Q_y \in \mathbb{R}^{q \times q}$ be from Definition 4.1.
7: Let N be such that $q^{2N}\gamma^N = 20n \cdot (N+1)^{q^4}$
8: $T' \leftarrow (T \otimes T^\top)^{\otimes N}$
9: **for** g a power of 2 from $1, 2, 4 \cdots, \max_{i,j} \mathrm{eff}_{i,j}(T')$ **do**
10: Prepare q^{2N} sets indexed by vector in $[q]^{2N}$ for both x and y, $X_1, \cdots, X_{q^{2N}}, Y_1, \cdots, Y_{q^{2N}}$.
11: $c \leftarrow q^{2N} \cdot g/n$
12: for each x_i, independently generate c indices $i_1, \cdots, i_c$ from $\mathcal{D}^{x_i}_{Q_x^{\otimes N} \otimes Q_y^{\otimes N}}$ and add $\{i\}$ to every X_{i_j}.
13: for each y_i, independently generate c indices $i_1, \cdots, i_c$ from $\mathcal{D}^{y_i}_{Q_y^{\otimes N} \otimes Q_x^{\otimes N}}$ and add $\{i\}$ to every Y_{i_j}.
14: Find correlated pair under such groupings. ▷ See details in Section 4.4.4
15: **end for**

4.4.1 Step 1. General start

We may assume $\gamma > 1/q$, since otherwise, the bound on ω_P we claim in Theorem 4.3 is worse than the trivial exponent of 2. We first fix N such that

$$q^{2N} = 20n \left(\frac{1}{\gamma}\right)^N (N+1)^{q^4}. \tag{5}$$

Note that $N = \Theta_q(\log n)$.

For simplicity, let us assume that the input vectors are long enough; as discussed in the introduction and early proofs, that one can use the 'compressed matrices' method introduced in [KKK18, Section 4.2] to 'expand' lower-dimensional vectors without losing too much correlations on correlated pair.

Let the planted pair be (x^*, y^*). We abuse notation here and also write $x^*, y^* \in [q]^{2N}$ to denote the first $2N$ coordinates of the planted pair. We write $x^* = x_1^* \circ x_2^*$, where $x_1^*, x_2^* \in [q]^N$, and write $y^* = y_1^* \circ y_2^*$ in the same way. (Here $\circ$ denotes vector concatenation.)

We will use Algorithm 2 to solve the problem. As suggested by line 9 to line 15 of the algorithm, our goal here is to copy every vector c times and partition them into q^{2N} groups, with each group containing roughly g vectors (so $nc/g \approx q^{2N}$). Our algorithm enumerates over g from 1 to $(\max_{i,j} \mathrm{eff}_{i,j})^{2N}$ by doubling each time. We are going to prove that we will successfully find the correlated pair for one of these choices of g.

We set aside the first $2N$ entries of each input vector which we will use to decide the grouping. We will later use fresh entries from the input vectors in later parts of the algorithm (when we perform matrix multiplication) so that there is no correlation between the independent random vectors which are placed in the same group.

With probability $1/(q^N)^{o(1)}$, we have both pairs $(x_1^*, y_1^*), (x_2^*, y_2^*) \in V_N$. We will assume this happens in the later analysis, since it only cost a $1/(q^N)^{o(1)}$ overhead on the running time by repeating the algorithm using fresh bits.

4.4.2 Step 2. Symmetrizing S_τ

Recall that $Q_x, Q_y \in \mathbb{R}^{q \times q}$ are the given stochastic matrices. We apply Q_x to x_1^* and Q_y to y_1^*, and let τ_1 be the best τ chosen from Lemma 4.8 with respect to x_1^*, y_1^*, Q_x, Q_y. For ease of presentation, we will still use

notation S_τ, $\mathcal{D}_\tau$, effQ_τ in the Lemma 4.8 and note that they are with respect to τ_1 and x_1^*, y_1^*, Q_x, Q_y, i.e., we have

$$\begin{aligned} S_\tau &= S_{\tau_1}^{x_1^*,y_1^*}, \quad \text{and} \\ \forall (x,y) \in S_\tau,\ \mathcal{D}_\tau &= \mathcal{D}^{x_1^*,y_1^*}_{Q_x^{\otimes N},Q_y^{\otimes N}}(x,y), \\ \forall (x,y) \in S_\tau,\ \underset{\tau}{\text{effQ}} &= \underset{x,y}{\text{effQ}}(Q_x^{\otimes N}, Q_y^{\otimes N}, T^{\otimes N}). \end{aligned} \tag{6}$$

Now, consider symmetrizing S_τ as follows. Let τ_2 be the ("transposed") mapping such that, for all $i,j,u,v \in [q]$, we have $\tau_2(i,j,u,v) := \tau_1(j,i,v,u)$. Since P is a symmetric matrix, we know that τ_2 is also *valid*. Define $S := S_{\tau_1}^{x_1^*,y_1^*} \otimes S_{\tau_2}^{x_2^*,y_2^*} \subseteq [q]^{2N}$, and we will show in Claim 4.10, $S = S_\tau \otimes S_\tau^\top$. For the purpose of symmetricity, we will further apply Q_y to x_2 and Q_x to y_2. We are able to do so because P is symmetric and thus, (y_2^*, x_2^*) is also in V_N.

We will prove that, with a careful choice of c (the number of copies we make of each vector), there will be a decent probability that a copy of the planted pair falls into some bucket in S. To do this, we first need to prove S is not skewed. The following are some useful fact about S.

Claim 4.10. $(S_{\tau_1}^{x_1^*,y_1^*})^\top = S_{\tau_2}^{y_1^*,x_1^*}$.

Proof.

$$\begin{aligned} &(S_{\tau_1}^{x_1^*,y_1^*})^\top \\ &= \{(x,y) \in [q]^{2N} : \forall i,j,u,v \in [q]\ \ |\{\ell \in [N] : x_1^*[\ell] = i \text{ and } y_1^*[\ell] = j \text{ and } x[\ell] = v \text{ and } y[\ell] = u\}| = \tau_1(i,j,u,v)\} \\ &= \{(x,y) \in [q]^{2N} : \forall i,j,u,v \in [q]\ \ |\{\ell \in [N] : x_1^*[\ell] = i \text{ and } y_1^*[\ell] = j \text{ and } x[\ell] = v \text{ and } y[\ell] = u\}| = \tau_2(j,i,v,u)\} \\ &= \{(x,y) \in [q]^{2N} : \forall i,j,u,v \in [q]\ \ |\{\ell \in [N] : x_1^*[\ell] = j \text{ and } y_1^*[\ell] = i \text{ and } x[\ell] = u \text{ and } y[\ell] = v\}| = \tau_2(i,j,u,v)\} \\ &= S_{\tau_2}^{y_1^*,x_1^*}, \end{aligned}$$

where the first step is by definition, the second step replaces τ_1 with τ_2, and the third step is by switching variables. □

Definition 4.11. For a ground set U, we say two sets $S_1, S_2 \subseteq U^2$ are *isomorphic* if they are equal up to permuting first and second coordinates, i.e., there are permutations $\pi_1, \pi_2 : U \to U$ such that, for all $(a,b) \in U^2$, $(a,b) \in S_1$ if and only if $(\pi_1(a), \pi_2(b)) \in S_2$.

Claim 4.12. *1. $V_x(S), V_y(S) \le |S|^{1.5}$;* *($V_x(S)$, $V_y(S)$ is defined in Definition 3.3)*

2. $|S| = |S_\tau|^2$;

3. $\forall (x,y) \in S$, $\mathcal{D}^{x^*}_{Q_x^{\otimes N} \otimes Q_y^{\otimes N}}(x) = \mathcal{D}^{y^*}_{Q_y^{\otimes N} \otimes Q_x^{\otimes N}}(y) = \mathcal{D}_\tau$;

4. Let $\widetilde{T} := T^{\otimes N}$. For all $(x,y) \in S$, we have $\text{effQ}_{x,y}(Q_x^{\otimes N} \otimes Q_y^{\otimes N}, Q_y^{\otimes N} \otimes Q_x^{\otimes N}, \widetilde{T} \otimes \widetilde{T}^\top) = \text{effQ}_\tau^2$.

Proof. **Part 1 and 2.**

For any valid τ and $(x_1,y_1),(x_2,y_2) \in V_N$, the sets $S_\tau^{x_1,y_1}$ and $S_\tau^{x_2,y_2}$ are isomorphic, since the definition of S_τ does not depend on the order of the N indices. Therefore, because both (y_1^*, x_1^*) and (x_2^*, y_2^*) are in V_N, we know that $S_{\tau_2}^{y_1^*,x_1^*}$ is isomorphic to $S_{\tau_2}^{x_2^*,y_2^*}$. By Claim 4.10, $(S_{\tau_2}^{y_1^*,x_1^*})^\top = S_{\tau_1}^{x_1^*,y_1^*}$, so

$$|S| = |S_{\tau_1}^{x_1^*,y_1^*}| \cdot |S_{\tau_2}^{x_2^*,y_2^*}| = |S_{\tau_1}^{x_1^*,y_1^*}| \cdot |S_{\tau_2}^{y_1^*,x_1^*}| = |S_{\tau_1}^{x_1^*,y_1^*}|^2,$$

and

$$V_x(S) = V_x(S_{\tau_1}^{x_1^*,y_1^*}) \cdot V_x(S_{\tau_2}^{x_2^*,y_2^*}) = V_x(S_{\tau_1}^{x_1^*,y_1^*}) \cdot V_y(S_{\tau_1}^{x_1^*,y_1^*}) \le |S_{\tau_1}^{x_1^*,y_1^*}|^3 = |S|^{1.5},$$

where the third step follows because $S_{\tau_1}^{x_1^*,y_1^*}$ is regular (by part 2 of Fact 4.7) and by Lemma 3.9.

$V_y(S) \le |S|^{1.5}$ follows for the same reason.

Part 3. For all $(x, y) \in S$,

$$\mathcal{D}^{x^*}_{Q_x^{\otimes N}\otimes Q_y^{\otimes N}}(x) = \prod_{i,j,u,v} (Q_x)_{i,u}^{\tau_1(i,j,u,v)} \prod_{i,j,u,v} (Q_y)_{i,u}^{\tau_2(i,j,u,v)} = \prod_{i,j,u,v} (Q_x)_{i,u}^{\tau_1(i,j,u,v)} \prod_{i,j,u,v} (Q_y)_{j,v}^{\tau_1(i,j,u,v)} = \mathcal{D}_\tau,$$

where the second step is by the symmetry of τ_1 and τ_2, and the last step is by the definition of $\mathcal{D}_\tau$ (part 3 of Fact 4.7). Similarly,

$$\mathcal{D}^{y^*}_{Q_y^{\otimes N}\otimes Q_x^{\otimes N}}(y) = \prod_{i,j,u,v} (Q_y)_{j,v}^{\tau_1(i,j,u,v)} \prod_{i,j,u,v} (Q_x)_{j,v}^{\tau_2(i,j,u,v)} = \prod_{i,j,u,v} (Q_y)_{j,v}^{\tau_1(i,j,u,v)} \prod_{i,j,u,v} (Q_x)_{i,u}^{\tau_1(i,j,u,v)} = \mathcal{D}_\tau.$$

Part 4. For all $x, y \in S$, we write $x = x_1 \circ x_2$ and $y = y_1 \circ y_2$ to partition them into two halves. Then,

$$\mathrm{effQ}_{x,y}(Q_x^{\otimes N} \otimes Q_y^{\otimes N}, Q_y^{\otimes N} \otimes Q_x^{\otimes N}, \widetilde{T} \otimes \widetilde{T}^\top) = \mathrm{effQ}_{x_1,y_1}(Q_x^{\otimes N}, Q_y^{\otimes N}, \widetilde{T}) \cdot \mathrm{effQ}_{x_2,y_2}(Q_y^{\otimes N}, Q_x^{\otimes N}, \widetilde{T}^\top).$$

We can calculate that

$$\begin{aligned}
\mathrm{effQ}_{x_2,y_2}(Q_y^{\otimes N}, Q_x^{\otimes N}, \widetilde{T}^\top) &= \prod_{i,j,u,v} \mathrm{effQ}_{u,v}(Q_y, Q_x, T^\top)^{\tau_2(i,j,u,v)} = \prod_{i,j,u,v} \mathrm{effQ}_{v,u}(Q_x, Q_y, T)^{\tau_2(i,j,u,v)} \\
&= \prod_{i,j,u,v} \mathrm{effQ}_{u,v}(Q_x, Q_y, T)^{\tau_1(j,i,u,v)} = \prod_{u,v} \mathrm{effQ}_{u,v}(Q_x, Q_y, T)^{\sum_{i,j} \tau_1(j,i,u,v)} \\
&= \mathrm{effQ}_{x_1,y_1}(Q_x, Q_y, T) = \mathrm{effQ}_\tau,
\end{aligned}$$

where the second step is because $\mathrm{effQ}_{u,v}(Q_y, Q_x, T^\top) = \mathrm{effQ}_{v,u}(Q_x, Q_y, T)$, the third step is because $\tau_2(i,j,u,v) = \tau_1(j,i,v,u)$ and switching u and v, and the last step is by definition of eff_τ (part 4 of Fact 4.7).

Therefore, the statement holds. □

4.4.3 Step 3. Detect (x^*, y^*) using S

To simplify notation, we write the distribution $\mathcal{D}^x_{Q_x^{\otimes N}\otimes Q_y^{\otimes N}}$ as $\mathcal{D}^x$, omitting the matrix $Q_x^{\otimes N} \otimes Q_y^{\otimes N}$ acting on x. Similarly, we write $\mathcal{D}^{y^*}_{Q_y^{\otimes N}\otimes Q_x^{\otimes N}}$ as $\mathcal{D}^y$.

We let $S_x := \{x \mid \mathcal{D}^{x^*}(x) = \mathcal{D}_\tau\}$ and $S_y := \{y \mid \mathcal{D}^{y^*}(y) = \mathcal{D}_\tau\}$. By Claim 4.12, $S \subseteq S_x \otimes S_y$, and also $|S_x| = |S_y|$ follows by symmetry.

In line 12, 13 of our Algorithm 2, for each $x_i \in [q]^{2N}$ (also y_i), we make c copies $i_1, \dots, i_c$ independently drawn from $\mathcal{D}^{x_i}$, and put x_i into the buckets $X_{i_1}, \dots, X_{i_c}$. (If two copies turned out to be identical, we only put x_i into that bucket once.) Let g^* be the least power of 2 that is larger than $\frac{n}{q^{2N} \cdot |S_\tau| \cdot D_\tau}$ (this number is roughly effQ_τ^2; see Eq. (8)). Since our algorithm iterates over all g which are powers of 2, there's an iteration where $g = g^*$. In the remaining analysis, we focus on this case. Note that

$$c = q^{2N} g^* / n \ge \frac{1}{|S_\tau| \cdot \mathcal{D}_\tau} \ge 1. \tag{7}$$

Claim 4.13. *Suppose $c \ge \frac{1}{|S_\tau| \cdot \mathcal{D}_\tau}$, then with $\ge 1/4$ probability, there's one copy x' of x^* and one copy y' of y^* such that $(x', y') \in S$.*

Proof. The copies of the planted vector x^* are drawn independently according to $\mathcal{D}^{x^*}$. Hence, the number of copies which fall into S_x follows a binomial distribution with mean $c \cdot \mathcal{D}_\tau \cdot |S_x| \geq 1$. It follows that, with constant probability, there are at least $c \cdot \mathcal{D}_\tau \cdot |S_x|$ many copies of x^* which fall into S_x. For the same reason, there is a constant probability that $c \cdot \mathcal{D}_\tau \cdot |S_y|$ many copies of y^* fall into S_y.

Furthermore, since for all $x \in S_x, y \in S_y$, $\mathcal{D}^{x^*}(x) = \mathcal{D}^{y^*}(y) = \mathcal{D}_\tau$, it follows that: conditioned on some particular copies of x^* falling into S_x, those copies will be independently uniformly random elements of S_x (and similarly for S_y). Also, since $|S_x| = |S_y|$ and $c \cdot \mathcal{D}_\tau \cdot |S_x| \geq |S_x|/\sqrt{|S|}$ (by the choice of our c and $|S| = |S_\tau|^2$) and $V_x(S), V_y(S) \leq |S|^{1.5}$ (Claim 4.12), it follows by Lemma 3.6 that with probability at least $1/4$, there is a pair of one copy of x^* and one copy of y^* which falls into S. □

Lemma 4.14. *Given $i, j \in [q]^{2N}$, let $|X_i|$ and $|Y_j|$ be the number of input vectors that placed a copy into X_i and Y_j, respectively. Then, over the randomness of the first $2N$ coordinates of the input vectors and the process of making random copies,*

$$\begin{aligned}\mathbb{E}[|X_i|] &= g^* \cdot \Pi_{l=1}^{N} \partial_x(i_l) \cdot \Pi_{l=N+1}^{2N} \partial_y(i_l), \quad \text{and} \\ \mathbb{E}[|Y_i|] &= g^* \cdot \Pi_{l=1}^{N} \partial_y(j_l) \cdot \Pi_{l=N+1}^{2N} \partial_x(j_l).\end{aligned}$$

Proof. For every input vector x other than the planted pair, x is uniformly sampled from $[q]^{2N}$. For a copy x' of x drawn from the distribution $\mathcal{D}^x_{Q_x^{\otimes N} \otimes Q_y^{\otimes N}}$, we have

$$\Pr[x' \in X_i] = \frac{1}{q^{2N}} \Pi_{l=1}^{N} \partial_x(i_l) \cdot \Pi_{l=N+1}^{2N} \partial_y(i_l).$$

By linearity of expectation over all input vectors and all copies, we have

$$\mathbb{E}[|X_i|] = nc \cdot \Pr[x' \in X_i].$$

Similarly, on the y side, we have

$$\mathbb{E}[|Y_j|] = \frac{nc}{q^{2N}} \Pi_{l=1}^{N} \partial_y(j_l) \cdot \Pi_{l=N+1}^{2N} \partial_x(j_l).$$

Thus, the claim follows since $g^* = \frac{nc}{q^{2N}}$. □

The following gives an upper bound on g^*,

$$g^* \leq \frac{2n}{q^{2N}|S_\tau|\mathcal{D}_\tau} = \frac{n\gamma^N}{10(N+1)^{q^4} \cdot |S_\tau|\mathcal{D}_\tau n} \leq \frac{N^{q^4}\mathcal{D}_\tau |S_\tau| \text{effQ}_\tau^2}{10(N+1)^{q^4} \cdot |S_\tau|\mathcal{D}_\tau} = \text{effQ}_\tau^2/10, \tag{8}$$

where the second step is by our choice of N from Equation (5) and c from Equation (7), and the third step is by Lemma 4.8.

4.4.4 Step 4. Matrix multiplication

Our vectors currently come from $[q]^N$, but we would like to map them to vectors in $\{-1, 1\}^N$ so that the independent uniformly random vectors are still independent uniformly random, and the planted pair is correlated. If q is even, we use the mappings g, h from Lemma 4.9. If q is odd, we first map each bit of vectors from $[q]$ to $[2q]$ by adding a uniform bit in $\{0, 1\}$, so that the planted pair still has non-zero correlation, then we use the mappings g, h.

As we discussed earlier, we use fresh bits (different from the ones used in the bucketing process above) for each matrix multiplication. Sample q_k^{2N} coordinates, and apply the mapping g to x, and h to y, bit-wise. Here we abuse notation and still write $x_i, y_i \in \{-1, 1\}^{q_k^{2N}}$ to denote the mapped input vectors x_i, y_i. The

result is that the mapped vectors x_i and y_i are independently uniformly chosen from $\{-1,1\}^{q_k^{2N}}$, except the correlated pair x^*, y^* has

$$\langle x^*, y^* \rangle = \Omega(q_k^{2N}).$$

For each $i \in [q]^{2N}$, create vectors $a_i, b_i \in \mathbb{R}^m$ given by $a_i = \sum_{j \in X_i} x_j$ and $b_i = \sum_{j \in Y_i} y_j$. Let $s_a, s_b \in \{-1,1\}^{q^{2N}}$ be random vectors whose entries are i.i.d. uniformly sampled from $\{-1,1\}$. Form the matrices $A, B \in \mathbb{R}^{q^{2N} \times m}$ whose rows are $s_a[1] \cdot a_1, \ldots, s_a[q^{2N}] \cdot a_{q^{2N}}$ and $s_b[1] \cdot b_1, \ldots, s_b[q^{2N}] \cdot b_{q^{2N}}$, respectively.

We now apply the tensor T' to the matrices A and $B^\top$, resulting in the matrix $C \in \mathbb{R}^{q^{2N} \times q^{2N}}$. By Claim 4.13, with $\geq 1/4$ probability, one copy of x^* and one copy of y^* fall into S. Denote the index by $(i,j) \in S$.

Using the same variance-based analysis from Theorem 3.2 and Theorem 3.4, we have

$$\begin{aligned}
\mathbb{E}[C[i,j]] &= \Omega(1) \cdot \sum_k T'(X_{i,k} Y_{j,k} Z_{i,j}) \\
\mathrm{var}[C[i,j]] &= \sum_{i',j'k,k'} T'(X_{i',k} Y_{j',k'} Z_{i,j})^2 \cdot \mathbb{E}[|X'_i|] \cdot \mathbb{E}[|Y'_j|].
\end{aligned}$$

$$\begin{aligned}
&\frac{\mathbb{E}[C[i,j]]}{\mathrm{var}[C[i,j]]^{1/2}} \\
&= \Omega(1) \cdot \frac{\sum_k T'(X_{i,k} Y_{j,k} Z_{i,j})}{\mathrm{effQ}_\tau^2 \cdot \sqrt{\sum_{i',j'k,k'} T'(X_{i',k} Y_{j',k'} Z_{i,j})^2 \cdot \Pi_{l=1}^N \partial_x(i'_l) \cdot \Pi_{l=N+1}^{2N} \partial_y(i'_l) \cdot \Pi_{l=1}^N \partial_y(j'_l) \cdot \Pi_{l=N+1}^{2N} \partial_x(j'_l)}} \\
&= \Omega(1) \cdot \frac{\mathrm{effQ}_{i,j}(Q_x^{\otimes N} \otimes Q_y^{\otimes N}, Q_y^{\otimes N} \otimes Q_x^{\otimes N}, T')}{\mathrm{effQ}_\tau^2} \\
&= \Omega(1),
\end{aligned}$$

where the first step is by replacing $\mathbb{E}[|X'_i|], \mathbb{E}[|Y'_j|]$ from Lemma 4.14 and Eq. (8), the second step is by definition of eff_Q (Def.4.1), the third step is because $\mathrm{effQ}_\tau = \mathrm{effQ}_{x,y}(Q_x^{\otimes N}, Q_y^{\otimes N}, T^{\otimes N})$ for any $(x,y) \in S_\tau$ from Eq.6 and the fact that $(i,j) \in S = S_\tau \otimes S_\tau^\top$.

As before, because the expectation exceeds the square root of variance, we can detect x^* and y^* by repeatedly running T'-matrix multiplication $O(\log n)$ times, which takes total running time $\tilde{O}(\mathrm{rank}(T'))$.

Overall, by repeating $(q^{2N})^{o(1)}$ times, we can boost the success probability to nearly 1. Once we can detect if the planted pair (x^*, y^*) exists, we can do binary search to find them with comparably negligible time overhead.

The exponent ω_P we get is

$$\omega_P = \frac{\log\left((q^{2N})^{o(1)} \cdot \mathrm{rank}(T)^{2N}\right)}{\log n} = \frac{\log\left((q^{2N})^{o(1)} \cdot \mathrm{rank}(T)^{2N}\right)}{\log\left(q^{2N} \gamma^N N^{-q^4}\right)} = \frac{\log \mathrm{rank}(T)}{\log(q\gamma^{1/2})} + o(1),$$

where the second step is by (5).

5 New Tensor Construction

In this section, we formally give the tensors summarized in Figure 1 from the introduction. (In Figure Figure 1, the bounds on ω_ℓ from each of these tensors is calculated.) We begin with our new tensor T_{2112}.

For any $\varepsilon > 0$, define the rank-5 tensor T_{2112} as the sum of the following five rank-1 tensors:

$$
\begin{aligned}
&(\mathsf{X}_{0,0}+\mathsf{X}_{1,0}/\varepsilon+\mathsf{X}_{0,1}/\varepsilon^3+\mathsf{X}_{1,1})(\mathsf{Y}_{0,0}/\varepsilon^3+\mathsf{Y}_{1,0}+\mathsf{Y}_{0,1}+\mathsf{Y}_{1,1}/\varepsilon)(\varepsilon^3\mathsf{Z}_{0,0}+\varepsilon^4\mathsf{Z}_{1,0}+\varepsilon^4\mathsf{Z}_{0,1}+\varepsilon\mathsf{Z}_{1,1})/4\\
&+(\mathsf{X}_{0,0}+\mathsf{X}_{1,0}/\varepsilon-\mathsf{X}_{0,1}/\varepsilon^3-\mathsf{X}_{1,1})(\mathsf{Y}_{0,0}/\varepsilon^3-\mathsf{Y}_{1,0}-\mathsf{Y}_{0,1}+\mathsf{Y}_{1,1}/\varepsilon)(\varepsilon^3\mathsf{Z}_{0,0}+\varepsilon^4\mathsf{Z}_{1,0}-\varepsilon^4\mathsf{Z}_{0,1}-\varepsilon\mathsf{Z}_{1,1})/4\\
&+(\mathsf{X}_{0,0}-\mathsf{X}_{1,0}/\varepsilon-\mathsf{X}_{0,1}/\varepsilon^3+\mathsf{X}_{1,1})(\mathsf{Y}_{0,0}/\varepsilon^3-\mathsf{Y}_{1,0}+\mathsf{Y}_{0,1}-\mathsf{Y}_{1,1}/\varepsilon)(\varepsilon^3\mathsf{Z}_{0,0}-\varepsilon^4\mathsf{Z}_{1,0}+\varepsilon^4\mathsf{Z}_{0,1}-\varepsilon\mathsf{Z}_{1,1})/4\\
&+(\mathsf{X}_{0,0}-\mathsf{X}_{1,0}/\varepsilon+\mathsf{X}_{0,1}/\varepsilon^3-\mathsf{X}_{1,1})(\mathsf{Y}_{0,0}/\varepsilon^3+\mathsf{Y}_{1,0}-\mathsf{Y}_{0,1}-\mathsf{Y}_{1,1}/\varepsilon)(\varepsilon^3\mathsf{Z}_{0,0}-\varepsilon^4\mathsf{Z}_{1,0}-\varepsilon^4\mathsf{Z}_{0,1}+\varepsilon\mathsf{Z}_{1,1})/4\\
&-\mathsf{X}_{0,1}\mathsf{Y}_{0,0}\mathsf{Z}_{1,1}/\varepsilon^5\\
&=(\mathsf{X}_{0,0}\mathsf{Y}_{0,0}+\mathsf{X}_{0,1}\mathsf{Y}_{1,0}+\varepsilon^3\mathsf{X}_{1,1}\mathsf{Y}_{0,1}+\varepsilon\mathsf{X}_{1,0}\mathsf{Y}_{1,1})\mathsf{Z}_{0,0}\\
&+(\varepsilon^4\mathsf{X}_{0,0}\mathsf{Y}_{0,1}+\mathsf{X}_{0,1}\mathsf{Y}_{1,1}+\varepsilon\mathsf{X}_{1,1}\mathsf{Y}_{0,0}+\varepsilon^3\mathsf{X}_{1,0}\mathsf{Y}_{1,0})\mathsf{Z}_{0,1}\\
&+(\mathsf{X}_{1,0}\mathsf{Y}_{0,0}+\varepsilon^4\mathsf{X}_{1,1}\mathsf{Y}_{1,0}+\varepsilon\mathsf{X}_{0,1}\mathsf{Y}_{0,1}+\varepsilon^3\mathsf{X}_{0,0}\mathsf{Y}_{1,1})\mathsf{Z}_{1,0}\\
&+(\mathsf{X}_{1,0}\mathsf{Y}_{0,1}+\mathsf{X}_{1,1}\mathsf{Y}_{1,1}+\varepsilon\mathsf{X}_{0,0}\mathsf{Y}_{1,0})\mathsf{Z}_{1,1}.
\end{aligned}
$$

(As discussed earlier, one might normally interpret this as a border rank expression, but here we substitute fixed values of $\varepsilon > 0$ and view it as a rank expression instead.) We can see that for any $\varepsilon > 0$, it has efficacies:

$$
\begin{aligned}
\underset{0,0}{\mathrm{eff}}(T) &= \frac{1+1}{\sqrt{1+1+\varepsilon^6+\varepsilon^2}} = \sqrt{2}-O(\varepsilon^2),\\
\underset{0,1}{\mathrm{eff}}(T) &= \frac{\varepsilon^4+1}{\sqrt{\varepsilon^8+1+\varepsilon^2+\varepsilon^6}} = 1-O(\varepsilon^2),\\
\underset{1,0}{\mathrm{eff}}(T) &= \frac{1+\varepsilon^4}{\sqrt{1+\varepsilon^8+\varepsilon^2+\varepsilon^6}} = 1-O(\varepsilon^2),\\
\underset{1,1}{\mathrm{eff}}(T) &= \frac{1+1}{\sqrt{1+1+\varepsilon^2}} = \sqrt{2}-O(\varepsilon^2).
\end{aligned}
$$

Hence,

$$
\mathrm{eff}(T) = \sqrt{(\sqrt{2}-O(\varepsilon^2))^2+(1-O(\varepsilon^2))^2+(1-O(\varepsilon^2))^2+(\sqrt{2}-O(\varepsilon^2))^2} = \sqrt{6}-O(\varepsilon^2).
$$

5.1 Derivation of T_{2112}

Although the rank expression above for T_{2112} suffices for our algorithm, we give an alternate, fairly simple way to see why T_{2112} has rank 5; this is how we first found this tensor. Our rank expression for T_{2112} was derived by a modification of the structural tensor $T_{(\mathbb{Z}/2)^2}$ of the group $(\mathbb{Z}/2)^2$ in the following way. $T_{(\mathbb{Z}/2)^2}$ is defined as

$$
T_{(\mathbb{Z}/2)^2} = \sum_{a,b\in(\mathbb{Z}/2)^2} \mathsf{X}_a\mathsf{Y}_b\mathsf{Z}_{a+b},
$$

and it has rank 4 since it is the structural tensor of an Abelian group. We can expand its terms:

$$
\begin{aligned}
&X_{0,0}Y_{0,0}Z_{0,0}+X_{0,0}Y_{0,1}Z_{0,1}+X_{0,0}Y_{1,0}Z_{1,0}+X_{0,0}Y_{1,1}Z_{1,1}\\
&+X_{0,1}Y_{0,0}Z_{0,1}+X_{0,1}Y_{0,1}Z_{0,0}+X_{0,1}Y_{1,0}Z_{1,1}+X_{0,1}Y_{1,1}Z_{1,0}\\
&+X_{1,0}Y_{0,0}Z_{1,0}+X_{1,0}Y_{0,1}Z_{1,1}+X_{1,0}Y_{1,0}Z_{0,0}+X_{1,0}Y_{1,1}Z_{0,1}\\
&+X_{1,1}Y_{0,0}Z_{1,1}+X_{1,1}Y_{0,1}Z_{1,0}+X_{1,1}Y_{1,0}Z_{0,1}+X_{1,1}Y_{1,1}Z_{0,0}
\end{aligned}
$$

First, we rename some variables, swapping the names of $\mathsf{X}_{0,1} \leftrightarrow \mathsf{X}_{1,1}$ and the names of $\mathsf{Y}_{1,0} \leftrightarrow \mathsf{Y}_{1,1}$ to yield

$$\begin{aligned}
&X_{0,0}Y_{0,0}Z_{0,0} + X_{0,0}Y_{0,1}Z_{0,1} + X_{0,0}Y_{1,1}Z_{1,0} + X_{0,0}Y_{1,0}Z_{1,1} \\
&+X_{1,1}Y_{0,0}Z_{0,1} + X_{1,1}Y_{0,1}Z_{0,0} + X_{1,1}Y_{1,1}Z_{1,1} + X_{1,1}Y_{1,0}Z_{1,0} \\
&+X_{1,0}Y_{0,0}Z_{1,0} + X_{1,0}Y_{0,1}Z_{1,1} + X_{1,0}Y_{1,1}Z_{0,0} + X_{1,0}Y_{1,0}Z_{0,1} \\
&+X_{0,1}Y_{0,0}Z_{1,1} + X_{0,1}Y_{0,1}Z_{1,0} + X_{0,1}Y_{1,1}Z_{0,1} + X_{0,1}Y_{1,0}Z_{0,0}
\end{aligned}$$

Since we just renamed variables, this tensor still has rank 4.

Next, we multiply some variables by powers of ε. We multiply $X_{1,0}$ by $1/\varepsilon$, multiply $X_{0,1}$ by $1/\varepsilon^3$, multiply $Y_{1,1}$ by $1/\varepsilon$, multiply $Y_{0,0}$ by $1/\varepsilon^3$, multiply $Z_{0,0}$ by ε^3, multiply $Z_{0,1}$ by ε^4, multiply $Z_{1,0}$ by ε^4, and multiply $Z_{1,1}$ by ε, to yield

$$\begin{aligned}
&X_{0,0}Y_{0,0}Z_{0,0} + \varepsilon^4 X_{0,0}Y_{0,1}Z_{0,1} + \varepsilon^3 X_{0,0}Y_{1,1}Z_{1,0} + \varepsilon X_{0,0}Y_{1,0}Z_{1,1} \\
&+\varepsilon X_{1,1}Y_{0,0}Z_{0,1} + \varepsilon^3 X_{1,1}Y_{0,1}Z_{0,0} + X_{1,1}Y_{1,1}Z_{1,1} + \varepsilon^4 X_{1,1}Y_{1,0}Z_{1,0} \\
&+X_{1,0}Y_{0,0}Z_{1,0} + X_{1,0}Y_{0,1}Z_{1,1} + \varepsilon X_{1,0}Y_{1,1}Z_{0,0} + \varepsilon^3 X_{1,0}Y_{1,0}Z_{0,1} \\
&+\varepsilon^{-5} X_{0,1}Y_{0,0}Z_{1,1} + \varepsilon X_{0,1}Y_{0,1}Z_{1,0} + X_{0,1}Y_{1,1}Z_{0,1} + X_{0,1}Y_{1,0}Z_{0,0}
\end{aligned}$$

Since we just multiplied variables by scalars, this did not change the rank, so this tensor still has rank 4. (This is similar to an operation called a "monomial degeneration" or "toric degeneration" in the literature, although here we are thinking of ε as a fixed, small positive value rather than a formal variable.)

Finally, we delete the term $\varepsilon^{-5} X_{0,1}Y_{0,0}Z_{1,1}$, yielding

$$\begin{aligned}
&X_{0,0}Y_{0,0}Z_{0,0} + \varepsilon^4 X_{0,0}Y_{0,1}Z_{0,1} + \varepsilon^3 X_{0,0}Y_{1,1}Z_{1,0} + \varepsilon X_{0,0}Y_{1,0}Z_{1,1} \\
&+\varepsilon X_{1,1}Y_{0,0}Z_{0,1} + \varepsilon^3 X_{1,1}Y_{0,1}Z_{0,0} + X_{1,1}Y_{1,1}Z_{1,1} + \varepsilon^4 X_{1,1}Y_{1,0}Z_{1,0} \\
&+X_{1,0}Y_{0,0}Z_{1,0} + X_{1,0}Y_{0,1}Z_{1,1} + \varepsilon X_{1,0}Y_{1,1}Z_{0,0} + \varepsilon^3 X_{1,0}Y_{1,0}Z_{0,1} \\
&+\varepsilon X_{0,1}Y_{0,1}Z_{1,0} + X_{0,1}Y_{1,1}Z_{0,1} + X_{0,1}Y_{1,0}Z_{0,0}
\end{aligned}$$

Since a single term has rank 1, this is a rank-1 update to our tensor, so this new tensor has rank at most 5. This is exactly our desired tensor T_{2112}.

We note that, since as $\varepsilon \to 0$, T_{2112} becomes 6 of the 8 terms of $\langle 2,2,2 \rangle$, one could add in the remaining two terms to give a relatively simple proof that the border rank of $\langle 2,2,2 \rangle$ is at most $5+2=7$. The fact that these 6 terms of $\langle 2,2,2 \rangle$ have border rank 6 has also been independently observed by Vrana, although with a different border rank identity (and hence not yielding T_{2112} specifically) [CV22].

5.2 Other Tensor Rank Bounds

We give the other tensor rank bounds mentioned in the introduction.

Strassen [Str69] showed that $\langle 2,2,2\rangle$ has rank at most 7 via the following expression:

$$\begin{aligned}
&(\mathsf{X}_{1,1}+\mathsf{X}_{2,2})(\mathsf{Y}_{1,1}+\mathsf{Y}_{2,2})(\mathsf{Z}_{1,1}+\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{2,1}+\mathsf{X}_{2,2})(\mathsf{Y}_{1,1})(\mathsf{Z}_{2,1}-\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,1})(\mathsf{Y}_{1,2}-\mathsf{Y}_{2,2})(\mathsf{Z}_{1,2}+\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{2,2})(\mathsf{Y}_{2,1}-\mathsf{Y}_{1,1})(\mathsf{Z}_{1,1}+\mathsf{Z}_{2,1})\\
+&(\mathsf{X}_{1,1}+\mathsf{X}_{1,2})(\mathsf{Y}_{2,2})(-\mathsf{Z}_{1,1}+\mathsf{Z}_{1,2})\\
+&(\mathsf{X}_{2,1}-\mathsf{X}_{1,1})(\mathsf{Y}_{1,1}+\mathsf{Y}_{1,2})(\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,2}-\mathsf{X}_{2,2})(\mathsf{Y}_{2,1}+\mathsf{Y}_{2,2})(\mathsf{Z}_{1,1})\\
=&(\mathsf{X}_{1,1}\mathsf{Y}_{1,1}+\mathsf{X}_{1,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,1}\\
+&(\mathsf{X}_{1,1}\mathsf{Y}_{1,2}+\mathsf{X}_{1,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,1}\\
+&(\mathsf{X}_{2,1}\mathsf{Y}_{1,1}+\mathsf{X}_{2,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,2}\\
+&(\mathsf{X}_{2,1}\mathsf{Y}_{1,2}+\mathsf{X}_{2,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,2}
\end{aligned}$$

Winograd [Win71] showed via the Strassen-Winograd identity that the tensor SW, which consists of 7 out of the 8 terms of $\langle 2,2,2\rangle$, has rank at most 6 as follows:

$$\begin{aligned}
&(\mathsf{X}_{2,1}+\mathsf{X}_{2,2})(\mathsf{Y}_{2,1}+\mathsf{Y}_{2,2})(-\mathsf{Z}_{1,2}+\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,2})(\mathsf{Y}_{2,1})(\mathsf{Z}_{1,1}-\mathsf{Z}_{1,2}-\mathsf{Z}_{2,1}+\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,2}+\mathsf{X}_{2,2})(\mathsf{Y}_{1,2}-\mathsf{Y}_{2,2})(\mathsf{Z}_{2,1}-\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,2}+\mathsf{X}_{2,1}+\mathsf{X}_{2,2})(-\mathsf{Y}_{1,2}+\mathsf{Y}_{2,1}+\mathsf{Y}_{2,2})(\mathsf{Z}_{1,2}+\mathsf{Z}_{2,1}-\mathsf{Z}_{2,2})\\
+&(\mathsf{X}_{1,1}+\mathsf{X}_{1,2}+\mathsf{X}_{2,1}+\mathsf{X}_{2,2})(\mathsf{Y}_{1,2})(\mathsf{Z}_{1,2})\\
+&(\mathsf{X}_{2,1})(\mathsf{Y}_{1,1}+\mathsf{Y}_{1,2}-\mathsf{Y}_{2,1}-\mathsf{Y}_{2,2})(\mathsf{Z}_{2,1})\\
=&(\mathsf{X}_{1,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,1}\\
+&(\mathsf{X}_{1,1}\mathsf{Y}_{1,2}+\mathsf{X}_{1,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,1}\\
+&(\mathsf{X}_{2,1}\mathsf{Y}_{1,1}+\mathsf{X}_{2,2}\mathsf{Y}_{2,1})\mathsf{Z}_{1,2}\\
+&(\mathsf{X}_{2,1}\mathsf{Y}_{1,2}+\mathsf{X}_{2,2}\mathsf{Y}_{2,2})\mathsf{Z}_{2,2}
\end{aligned}$$

6 Hashing gives an improvement for almost any tensor

The goal of this section is to prove the following Theorem 6.1.

Theorem 6.1 (Restatement of Theorem 1.4)**.** *Suppose T is a $\langle q,q,q_k\rangle$-sized tensor which consists of a subset of the terms of a matrix multiplication tensor, and the matrix $[(\mathrm{eff}_{i,j}(T))^2]_{i,j}$ has full rank. Let $\omega'_\ell := \frac{\log(\mathrm{rank}(T))}{\log(\mathrm{eff}(T))}$ be the exponent one would get from T from applying Theorem 1.1. Then, there is a non-decreasing, positive function $f_T : (0,1) \to \mathbb{R}_{>0}$ such that the bound of Theorem 1.1 can be improved to*

$$\omega_\ell \le \omega'_\ell - f_T(\rho).$$

When T is composed of a subset of the terms of a matrix multiplication tensor, $T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j}) \neq 0$ only if $i'=i$, $j'=j$ and $k'=k$. Thus, we can rewrite the effQ (Def. 4.1) as

$$\underset{i,j}{\mathrm{effQ}}(Q_x, Q_y, T) = \frac{1}{\partial_x(i)\partial_y(j)} \frac{\sum_{k\in[q_k]} T(\mathsf{X}_{i,k}\mathsf{Y}_{j,k}\mathsf{Z}_{i,j})}{\sqrt{\sum_{i',j'\in[q],k,k'\in[q_k]} T(\mathsf{X}_{i',k}\mathsf{Y}_{j',k'}\mathsf{Z}_{i,j})^2}}$$
$$= \frac{\mathrm{eff}_{i,j}(T)}{\partial_x(i)\partial_y(j)}$$

And γ_{Q_x,Q_y} can be also rewritten as

$$\gamma_{Q_x,Q_y} = \prod_{i,j\in[q]} \left(\sum_{u,v\in[q]} Q_x[i,u]Q_y[j,v](\underset{u,v}{\mathrm{effQ}}(Q_x,Q_y,T))^2 \right)^{P[i,j]}$$
$$= \prod_{i,j\in[q]} \left(\sum_{u,v\in[q]} \underbrace{\frac{Q_x[i,u]}{\partial_x(u)}}_{N_x[i,u]} \underbrace{\frac{Q_y[j,v]}{\partial_y(v)}}_{N_y[j,v]} \underset{u,v}{\mathrm{eff}}(T)^2 \right)^{P[i,j]},$$

where we define $N_x, N_y \in \mathbb{R}^{q\times q}$ as above. In other words, we normalize every column of Q_x, Q_y to get N_x, N_y.

We begin with the key lemma behind our proof of Theorem 6.1, which shows how we will pick the matrices Q_x, Q_y for our hashing scheme.

Lemma 6.2. *Let T be the tensor having the same property as that in Theorem 6.1. There exist stochastic matrices Q_x, Q_y such that $\gamma_{Q_x,Q_y} > \frac{1}{q^2}\sum_{i,j} \mathrm{eff}_{i,j}^2(T)$.*

Proof. Let $\varepsilon > 0$ be a small constant. Let $\mathrm{avg} = \frac{1}{q^2}\sum_{i,j} \mathrm{eff}_{i,j}^2(T)$. Let $A = \mathrm{eff}^2(T) \in \mathbb{R}^{q\times q}$ be the matrix given by $A_{i,j} = \mathrm{eff}_{i,j}^2(T)$.

Let $C \in \mathbb{R}^{q\times q}$ be the matrix defined by

$$C_{i,j} = \begin{cases} \mathrm{avg} + \varepsilon, & \text{if } i = j = 1; \\ \mathrm{avg} - \varepsilon/(q^2-1), & \text{otherwise.} \end{cases}$$

Suppose we can design N_x, N_y such that

$$C = N_x \cdot A \cdot N_y^\top,$$

then $\gamma_{Q_x,Q_y} = \prod_{i,j\in[q]} C_{i,j}^{P[i,j]}$. According to Lemma 6.3, $\gamma_{Q_x,Q_y} > \mathrm{avg}$ and therefore we conclude the lemma.

In the following, we are going to prove that such N_x, N_y exist in two steps. For $j \in [q]$, let $c_j = \frac{1}{q}\sum_{i\in[q]} \mathrm{eff}_{i,j}^2(T)$ be the average of j-th column of $\mathrm{eff}^2(T)$. Define $B := \mathbf{1}_q^\top \cdot (c_1, \cdots, c_q) + \Delta \in \mathbb{R}^{q\times q}$. Here, $\mathbf{1}_q$ is an all-one vector of length q, and Δ is defined as follows, where $\delta := \frac{q}{q+1}\varepsilon$:

$$\Delta_{i,j} := \begin{cases} 0, & \text{if } j \geq 2; \\ \delta, & \text{if } j = 1 \text{ and } i = 1; \\ -\delta/(q-1), & \text{if } j = 1 \text{ and } i \neq 1. \end{cases}$$

The first step is to design Q_x such that $N_x \cdot \mathrm{eff}^2(T) = B$, and the second step is to design Q_y such that $B \cdot N_y^\top = C$.

Step 1. Design Q_x We first design N_x as follows. Let $N_x := \frac{1}{q} \cdot \mathbf{1}_{q\times q} + N'_x$, where $\mathbf{1}_{q\times q}$ is an all-one matrix of size $q \times q$. We will note that every entry in N'_x is of order $O(\varepsilon)$.

By $N_x \cdot A = B$, we have

$$(\frac{1}{q} \cdot \mathbf{1}_{q\times q} + N'_x) \cdot A = \mathbf{1}_q^\top \cdot (c_1, \cdots, c_q) + \Delta,$$

therefore,

$$\begin{aligned} N'_x &= \Delta \cdot A^{-1}. \\ (N'_x)_{i,j} &= \begin{cases} \delta \cdot (A^{-1})_{1,j}, & \text{if } i = 1\ ; \\ -\frac{1}{q-1}\delta \cdot (A^{-1})_{1,j}, & \text{if } i \geq 2. \end{cases} \end{aligned}$$

We design Q_x as follows. For $j \in [q]$, let z_j be variables. Let $(Q_x)_{i,j} := (N_x)_{i,j} \cdot z_j$. We will set z_j so that every row of Q_x sums up to 1. Since all but the first row are all the same, we only need to care about the first row and the second row:

$$\begin{cases} \sum_j (\frac{1}{q} + \delta \cdot (A^{-1})_{1,j}) \cdot z_j = 1; \\ \sum_j (\frac{1}{q} - \frac{\delta}{q-1} \cdot (A^{-1})_{1,j}) \cdot z_j = 1. \end{cases} \tag{9}$$

We solve this pair of equations case by case.

Case 1: When A is a diagonal matrix. In this special case, one solution to the Eq. 9 is $z_1 = 0$ and $z_2 = z_3 = \cdots = z_q = \frac{q}{q-1}$. In this case, all the entries in Q_x are in $[0, 1]$ and thus Q_x is valid.

Case 2: When A is not diagonal. By Lemma 6.5, there are two indices $j_1, j_2 \in [q]$ that $(A^{-1})_{1,j_1} > 0$ and $(A^{-1})_{1,j_2} < 0$. Let $x = \delta \cdot (A^{-1})_{1,j_1} > 0$ and $y = \delta \cdot (A^{-1})_{1,j_2} < 0$. Then we let $z_{j_1} := \frac{q|y|}{|x|+|y|}$, $z_{j_2} := \frac{q|x|}{|x|+|y|}$, and all $z_j := 0$ for all other js. One can verify that this satisfies the Eq. (9).

Since all entries of $(N_x)_{i,j} = 1/q + \Theta(\epsilon)$ and $0 < z_{j_1}, z_{j_2} < q$, for small enough ε, we have every entry of Q_x are in the range $(0, 1)$, so that Q_x is valid.

Step 2. Design Q_y We first show how to construct N_y so that $B \cdot N_y^\top = C$.

Let $b \in \mathbb{R}$ be a parameter. Define N_y to be

$$(N_y^\top)_{i,j} = \begin{cases} 1, & \text{if } i = j = 1; \\ 0, & \text{if } i = 1,\ j \geq 2; \\ b, & \text{if } i \geq 2,\ j = 1; \\ \frac{1-b}{q-1}, & \text{if } i \geq 2 \text{ and } j \geq 2. \end{cases}$$

Under this design of N_y, the equation $B \cdot N_y^\top = C$ will become three small equations.

$$\begin{cases} c_1 + \delta + b \cdot (c_2 + \cdots + c_q) = \text{avg} + \epsilon \\ c_1 - \frac{\delta}{q-1} + b \cdot (c_2 + \cdots + c_q) = \text{avg} - \frac{\varepsilon}{q^2-1} \\ \frac{1-b}{q-1}(c_2 + \cdots, c_q) = \text{avg} - \frac{\varepsilon}{q^2-1}. \end{cases} \tag{10}$$

Set $\delta = \frac{q}{q+1}\epsilon$, by solving Eq. 10, we get

$$b = \frac{1}{q} - \frac{c_1(1-1/q) - \varepsilon/(q+1)}{c_2 + \cdots + c_q} \leq \frac{1}{q}.$$

Using the same method as in Step 1, we let $(Q_x)_{i,j} := (N_x)_{i,j} \cdot z_j$. We need to find $z_1, \cdots, z_q$ so that every row of Q_x sums up to 1. By setting $z_2 = z_3 = \cdots = z_q$, this reduces to two equations.

$$\begin{cases} z_1 + (q-1)b \cdot z_2 = 1; \\ \frac{1-b}{q-1} \cdot z_2 = 1. \end{cases}$$

By solving these equations, we get

$$z_1 = 1 - \frac{b(q-1)}{1-b}, \quad z_2 = 1/(1-b).$$

Since $b \le 1/q$, one can verify that every entry of Q_x is in $[0, 1]$, and Q_x is a stochastic matrix. □

We now prove the helper lemmas for the above result:

Lemma 6.3. *Let q be a positive integer, and suppose $a, \rho > 0$ and $1 \ge p > 1/q^2$. Then, for all sufficiently small $\varepsilon > 0$ we have*

$$(a+\varepsilon)^p \cdot \left(a - \frac{\varepsilon}{q^2-1}\right)^{1-p} > a.$$

Proof. Define $f(\varepsilon) := (a+\varepsilon)^p \cdot (a - \frac{\varepsilon}{q^2-1})^{1-p}$. Since $f(0) = a$, it suffices to prove that $f'(0) > 0$. Let $m = q^2 - 1$ so that $f(\varepsilon) := (a+\varepsilon)^p \cdot (a - \frac{\varepsilon}{m})^{1-p}$. Note that $(m+1) \cdot p > 1$ by definition of p. We have:

$$\begin{aligned} f'(\varepsilon) &= p \cdot (a+\varepsilon)^{p-1} \cdot (a - \frac{\varepsilon}{m})^{1-p} + (a+\varepsilon)^p \cdot \frac{p-1}{m}(a - \frac{\varepsilon}{m})^{-p} \\ &= \frac{(amp + ap - a - \varepsilon)(a+\varepsilon)^{p-1}(a - \varepsilon/m)^{-p}}{m}. \end{aligned}$$

Hence, as desired,

$$\begin{aligned} f'(0) &= \frac{(amp + ap - a)(a)^{p-1}(a)^{-p}}{m} \\ &= \frac{mp + p - 1}{m} > 0. \end{aligned}$$

□

Lemma 6.4. *Suppose $A \in \mathbb{R}_{\ge 0}^{q \times q}$ for $q \ge 2$ is a full-rank matrix with nonnegative entries such that at least one of its rows has at least two nonzero entries. Then, one can permute the columns of A so that it has the following property: For every row of A, if its first entry is nonzero, then another one of its entries is also nonzero.*

Proof. It suffices to prove that there is a column j of A such that: for every i with $A[i, j] \ne 0$, there exists an $i' \ne i$ with $A[i', j] \ne 0$. We can then permute the columns of A so that j becomes the first column as desired.

Assume to the contrary that there were no such j. Since A has full rank, we know every column of A has a nonzero entry. It follows that for every j, there is a row with a nonzero entry in column j but no other column. Since A has the same number of rows and columns, this means every row and every column of A has exactly one nonzero entry. This contradicts our assumption that A has a row with at least two nonzero entries. □

Lemma 6.5. *Suppose $A \in \mathbb{R}_{\geq 0}^{q \times q}$ for $q \geq 2$ is a full-rank matrix with nonnegative entries such that at least one of its rows has at least two nonzero entries. Then, one can permute the columns of A so that it has the following property: In the top row of the matrix A^{-1}, there is at least one positive entry and at least one negative entry.*

Proof. Applying Lemma 6.4, we may assume that for every row i of A, if $A[i,1] \neq 0$, then there is an $j \neq 1$ such that $A[i,j] \neq 0$.

We claim first that the top row of A^{-1} must have at least two nonzero entries. Assume to the contrary that this is not the case. It must have at least one nonzero entry since A^{-1} has full rank, so it has exactly one nonzero entry. Suppose it is in column j, so $A^{-1}[1,j] \neq 0$ and $A^{-1}[1,j'] = 0$ for all $j' \neq j$. We know that the top-right entry of the product $A^{-1}A$ is 1, so it follows that $A[j,1] = 1/A^{-1}[1,j] \neq 0$. By the property of the previous paragraph, there is a $i \neq 1$ such that $A[j,i] \neq 0$. It follows that entry $(1,i)$ of the product $A^{-1}A$ is equal to $\sum_k A^{-1}[1,k] \cdot A[k,i] = A^{-1}[1,j] \cdot A[j,i] \neq 0$, contradicting the fact that $A^{-1}A$ is the identity matrix whose $(1,i)$ entry is 0. This proves the claim.

Now, we know the top row of A^{-1} has at least two nonzero entries. We claim that the nonzero entries of the top row of A^{-1} cannot all be positive or all be negative, which will complete the proof. Assume to the contrary that they are all positive (the all negative case is identical), and as before, suppose $A^{-1}[1,j] > 0$ is one of the nonzero entries of the first row of A^{-1}. Since the first row has at least two nonzero entries, we may assume $j \neq 1$. Since A has full rank, there is an i such that $A[j,i] \neq 0$, and since A has nonnegative entries, we further have $A[j,i] > 0$ and $A[j',i] \geq 0$ for all j'. It follows that entry $(1,j)$ of the product $A^{-1}A$ is $\sum_k A^{-1}[1,k]A[k,i] \geq A^{-1}[1,j]A[j,i] > 0$, contradicting again that it must equal 0. This completes the proof. □

Finally we conclude the main proof:

Proof of Theorem 6.1. By Lemma 6.2, we can construct stochastic matrices Q_x, Q_Y such that $\gamma_{Q_x,Q_y} > \frac{1}{q^2}\sum_{i,j} \mathrm{eff}_{i,j}^2(T)$. By Theorem 4.3, we have

$$\omega_P \leq \frac{\log \mathrm{rank}(T)}{\log(P\text{-}\,\mathrm{eff}(T))},$$

where $P\text{-}\,\mathrm{eff}(T) \geq q\gamma_{Q_x,Q_y}^{1/2} > \mathrm{eff}(T)$. It's clear from the proof that the difference between $P\text{-}\,\mathrm{eff}(T)$ and $\mathrm{eff}(T)$ is a function of ρ. The function $f_T(\rho)$ is an non-decreasing function since we can always reduce larger ρ to smaller ρ. □

Appendix

A Aggregation time

In the algorithms throughout this paper, we assumed that the input vectors $x_1, \ldots, x_n, y_1, \ldots, y_n$ have long enough length d which is polynomial in n, i.e., $d = q^N$ (q and N are defined in Algorithm 1), whereas we would like our algorithm to work for the information-theoretically minimum $d = O(\log n/\rho^2)$ (recall that $\rho \in (0,1)$ is the correlation of the planted pair). Furthermore, we assumed that the aggregation step of the algorithm (lines 12, 13 of Algorithm 1) takes negligible time compared to the rest of the algorithm. (See footnote 10 above.) However, if implemented naively, the aggregation step can actually take time $q^N \cdot d$, which can potentially be the slowest step of the algorithm. In this section, we show how the "compressed matrix" technique of [KKK18] can be used to require only the smaller $d = O(\log n/\rho^2)$, and simultaneously decrease the aggregation time.

Suppose the given vectors $x \in \{-1,1\}^d$ have short length $d = O(\log n/\rho^2)$ and we want a long enough vector $x' \in \{-1,1\}^m$ to be used in our algorithm, for some $m = \mathrm{poly}(n)$. To "prolong" the vector, we first

pick $r \le d$ such that $\binom{d}{r} = m$, and define for every subset $S \subseteq [d]$ with $|S| = r$, the entry

$$x'_S = \prod_{j \in S} x_j. \tag{11}$$

This new vector x' will be *implicitly* used as the true input vector in our algorithm; in fact, we will never compute x', but rather the aggregation of all such "prolonged" vectors defined as follows.

Definition A.1 (Aggregation problem)**.** Given $x_1, \dots, x_g \in \{-1,1\}^d$ and $r \le d$. Let $m = \binom{d}{r}$ and $x'_1, \dots, x'_g$ be defined as in Eq. (11). The goal is to compute, for every $j \in [m]$, the value $X_j = \sum_{i \in [g]} x'_i[j]$. (I.e., the goal is to compute the vector $\sum_{i \in [g]} x'_i$.)

Note that the aggregation vectors a_i, b_i in line 11 of Algorithm 1 can be computed by solving this aggregation problem $O(q^N)$ times, and hence we construct our matrices A and B from lines 12 and 13.

Lemma A.2 ([KKK18, Alm18])**.** *The Aggregation problem defined above (Def. A.1) can be solved in* $\boldsymbol{MM}(m^{1/2+o(1)}, g, m^{1/2+o(1)})$ *time, where* $\boldsymbol{MM}(a,b,c)$ *is the time to multiply a matrix of size* $a \times b$ *with another matrix of size* $b \times c$*.*

Proof. The proof is identical to the aggregation algorithm used by prior light bulb algorithms, such as [Alm18, page 6, second and third paragraphs]. □

Lemma A.3. *While running Algorithm 1 with input vectors* $x_1, \dots, x_n, y_1, \dots, y_n \in \{-1,1\}^d$*, and* $\langle q,q,q \rangle$*-sized tensor* T*, the matrix* A *and* B *defined in line 12 and line 13 can be computed in* $n^{\frac{1+\omega/2}{\log_q \operatorname{eff}(T)} + o(1)}$ *time.*

Proof. Since A and B are constructed in the same way, we only analyze A.

Let g be such that $g^2 \cdot |\{i,j \in [q^N] : \operatorname{eff}_{i,j}(T^{\otimes N}) \ge g^2\}|$ is maximized, as defined in line 5 of the algorithm. Note that $\operatorname{eff}_{i,j}(T)$ cannot exceed q by the Cauchy-Schwarz inequality (it is maximized when $T = \langle q,q,q \rangle$), and so $g \le \sqrt{\max_{i,j} \operatorname{eff}_{i,j}(T^{\otimes N})} \le \sqrt{q^N}$.

In line 11, each a_i aggregates together $|X_i|$ vectors. We have $\mathbb{E}[|X_i|] = nt/q^N = g$ since we set $t = q^N g/n$. By a Chernoff bound, $|X_i| = O(g)$ for each i with high probability.

Let $m = q^N$. Since the tensor T has size $q_k = q$, the desired length of input vectors d is also m. By Lemma A.2, calculating all the a_i can be done in time

$$m \cdot \mathbf{MM}(\sqrt{d}, g, \sqrt{d}) = m \cdot \mathbf{MM}(\sqrt{m}, g, \sqrt{m}) \le m \cdot g^\omega \cdot (\sqrt{m}/g)^2 = m^2 g^{\omega-2} \le m^{2+\frac{\omega-2}{2}}.$$

(Here we used that, since $\sqrt{m} \ge g$, we have $\mathbf{MM}(\sqrt{m}, g, \sqrt{m}) \le (\sqrt{m}/g)^2 \cdot \mathbf{MM}(g,g,g)$.)

Since $\operatorname{rank}(T)^N = n^{\frac{\log \operatorname{rank}(T)}{\log \operatorname{eff}(T)}}$, we have $N = \log n / \log \operatorname{eff}(T)$. Thus, this is the desired running time since $m = q^N = n^{\log q / \log \operatorname{eff}(T)}$. □

Remark A.4. *When* T *is a matrix multiplication tensor,* $\log_q \operatorname{eff}(T) = 1.5$*, so the aggregation time is* $n^{\frac{1+\omega/2}{\log_q \operatorname{eff}(T)} + o(1)} = O(n^{\frac{2+\omega}{3}}) < O(n^{\frac{2\omega}{3}})$*. The aggregation time exponent* $\frac{2+\omega}{3}$ *is less than* $\frac{2\omega}{3}$*, so aggregation takes negligible time compared to the remainder of the algorithm.*

Lemma A.5. *If there is a* $\langle q,q,q \rangle$*-sized tensor* T *with*

$$\frac{\log \operatorname{rank}(T)}{\log \operatorname{eff}(T)} < \frac{2\omega}{3},$$

then there is another tensor T' *that can solve light bulb problem in time* $n^{\frac{2\omega}{3} - \varepsilon}$ *for some* $\varepsilon > 0$*.*

Proof. Let N be a large enough constant and we let $T' = T^{\otimes \delta N} \otimes \langle q,q,q\rangle^{\otimes (1-\delta)N}$ for some $\delta \in (0,1)$ to be determined. So

$$\log \operatorname{rank}(T') = N \cdot (\delta \log \operatorname{rank}(T) + (1-\delta) \log \operatorname{rank}(\langle q,q,q\rangle))$$

and

$$\log \operatorname{eff}(T') = N \cdot (\delta \log \operatorname{eff}(T) + (1-\delta) \log \operatorname{eff}(\langle q,q,q\rangle)).$$

Since $\frac{\log \operatorname{rank}(\langle q,q,q\rangle)}{\log \operatorname{eff}(\langle q,q,q\rangle)} = \frac{2\omega}{3}$, choosing any $\delta > 0$ results in $\frac{\log \operatorname{rank}(T')}{\log \operatorname{eff}(T')} < \frac{2\omega}{3}$.

By Lemma A.3, the aggregation time of T' is $n^{\frac{1+\omega/2}{\log_q \operatorname{eff}(T')}+o(1)} \leq n^{\frac{1+\omega/2}{(1-\delta)1.5}+o(1)}$. We can choose a small enough δ so that $\frac{1}{1-\delta} \cdot \frac{1+\omega/2}{1.5} < \frac{2\omega}{3}$. Thus, the running time for both the main procedure and the aggregation part while using tensor T' is small. □

References

[ACW16] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016. 10

[ACW20] Josh Alman, Timothy M Chan, and Ryan Williams. Faster deterministic and las vegas algorithms for offline approximate nearest neighbors in high dimensions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 637–649. SIAM, 2020. 10

[AIR18] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3287–3318. World Scientific, 2018. 2, 10

[Alm18] Josh Alman. An illuminating algorithm for the light bulb problem. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 2, 3, 4, 10, 37

[ALRW17] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–66. SIAM, 2017. 10

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801, 2015. 10

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–150. IEEE, 2015. 10

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021. 2, 9

[Bin80a] Dario Bini. Border rank of ap× q× 2 tensor and the optimal approximation of a pair of bilinear forms. In *International Colloquium on Automata, Languages, and Programming*, pages 98–108. Springer, 1980. 6

[Bin80b] Dario Bini. Relations between exact and approximate bilinear algorithms. applications. *Calcolo*, 17(1):87–97, 1980. 6

[BL16] Markus Bläser and Vladimir Lysikov. On degeneration of tensors and algebras. *arXiv preprint arXiv:1606.04253*, 2016. 9

[CGLV19] Austin Conner, Fulvio Gesmundo, Joseph M Landsberg, and Emanuele Ventura. Tensors with maximal symmetries. *arXiv preprint arXiv:1909.09518*, 2019. 9

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002. 1

[CHL22] Austin Conner, Hang Huang, and JM Landsberg. Bad and good news for strassen's laser method: Border rank of perm 3 and strict submultiplicativity. *Foundations of Computational Mathematics*, pages 1–39, 2022. 9

[CU13] Henry Cohn and Christopher Umans. Fast matrix multiplication using coherent configurations. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1074–1086. Society for Industrial and Applied Mathematics, 2013. 1, 8, 9

[CV22] Matthias Christandl and Péter Vrana. personal communication, 2022. 31

[CW82] Don Coppersmith and Shmuel Winograd. On the asymptotic complexity of matrix multiplication. *SIAM J. Comput.*, 11(3):472–492, 1982. 2

[DS13] A.M. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh, Section: A Mathematics*, 143:351–369, 4 2013. 2

[Dub10] Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010. 2, 8

[FBH+22] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. 5

[Har21] David G Harris. Improved algorithms for boolean matrix multiplication via opportunistic matrix multiplication. *arXiv preprint arXiv:2109.13335*, 2021. 1, 5, 8, 9

[HJMS22] Roser Homs, Joachim Jelisiejew, Mateusz Michałek, and Tim Seynnaeve. Bounds on complexity of matrix multiplication away from coppersmith–winograd tensors. *Journal of Pure and Applied Algebra*, 226(12):107142, 2022. 9

[HSHVDG16] Jianyu Huang, Tyler M Smith, Greg M Henry, and Robert A Van De Geijn. Strassen's algorithm reloaded. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 690–701. IEEE, 2016. 5

[KK19] Matti Karppa and Petteri Kaski. Probabilistic tensors and opportunistic boolean matrix multiplication. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 496–515. SIAM, 2019. 1, 3, 5, 8, 9

[KKK18] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Transactions on Algorithms (TALG)*, 14(3):1–26, 2018. 2, 3, 4, 5, 9, 10, 14, 25, 36, 37

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC*, pages 296–303, 2014. 2

[Pan78] V Ya Pan. Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 166–176. IEEE, 1978. 9

[Pan18] Victor Y Pan. Fast feasible and unfeasible matrix multiplication. *arXiv preprint arXiv:1804.04102*, 2018. 5

[Sch81] Arnold Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3):434–455, 1981. 1, 9

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969. 3, 6, 32

[Str73] Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. 1

[Str87] V. Strassen. Relative bilinear complexity and matrix multiplication. *J. reine angew. Math. (Crelles Journal)*, 375–376:406–443, 1987. 9

[Val88] Leslie G Valiant. Functionality in neural nets. In *AAAI*, 1988. 1

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2012. 1, 2, 3, 4, 9, 10

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *STOC*, pages 887–898, 2012. 2

[Win71] Shmuel Winograd. On multiplication of 2×2 matrices. *Linear algebra and its applications*, 4(4):381–388, 1971. 5, 6, 32