A Faster Algorithm for Finding Closest Pairs in Hamming Metric

Andre Esser ✉
Cryptography Research Center, Technology Innovation Institute, Abu Dhabi, UAE

Robert Kübler ✉
Metro AG, Düsseldorf, Germany

Floyd Zweydinger ✉
Ruhr Universität Bochum, Germany

── Abstract ─────────

We study the Closest Pair Problem in Hamming metric, which asks to find the pair with the smallest Hamming distance in a collection of binary vectors. We give a new randomized algorithm for the problem on uniformly random input outperforming previous approaches whenever the dimension of input points is small compared to the dataset size. For moderate to large dimensions, our algorithm matches the time complexity of the previously best-known locality sensitive hashing based algorithms. Technically our algorithm follows similar design principles as Dubiner (IEEE Trans. Inf. Theory 2010) and May-Ozerov (Eurocrypt 2015). Besides improving the time complexity in the aforementioned areas, we significantly simplify the analysis of these previous works. We give a modular analysis, which allows us to investigate the performance of the algorithm also on non-uniform input distributions. Furthermore, we give a proof of concept implementation of our algorithm which performs well in comparison to a quadratic search baseline. This is the first step towards answering an open question raised by May and Ozerov regarding the practicability of algorithms following these design principles.

1 Introduction

Finding closest pairs in a given dataset of binary points is a fundamental problem in theoretical computer sciences with numerous applications in data science, machine learning, computer vision, cryptography, and many others.

Image data for example is often represented via compact binary codes to allow for efficient closest pair search in applications like similarity search in images or facial recognition systems [8, 15, 20]. The usage of binary codes also allows decoding the represented data to common codewords. Here, the most efficient algorithms known for decoding such random binary linear codes also heavily benefit from improved algorithms for the Closest Pair Problem [6, 17]. Another common application lies in the field of bioinformatics, where the analysis of genomes involves closest pair search on large datasets to identify the most correlated genetic markers [16, 19].

To be more precise, the Closest Pair Problem asks to find the pair of vectors with the minimal Hamming distance among n given binary vectors. While the general version of this problem does not make any restrictions on the distribution of input points, several settings imply a uniform distribution of dataset elements [6, 16, 17, 19]. Usually, in such settings, there is a planted pair, which attains relative distance $\gamma \in [0, \frac{1}{2}]$, which has to be found. This uniform version is also known as the *light bulb problem* [22]. The problem can be solved in

time linearly in the dataset size[1] as long as the dimension of vectors is constant [5, 14]. As soon as the dimension is non-constant an effect occurs known as *curse of dimensionality*, which lets the problem become much harder.

The most common framework to assess the problem is based on *locality-sensitive hashing* (LSH), whose research was initiated in the pioneering work of Indyk and Motwani [12]. Roughly speaking, a locality-sensitive hash function is more likely to hash points that are close to each other to the same value, rather than points that are far apart. To solve the Closest Pair Problem leveraging an LSH family one chooses a random hash function of the family and computes the hash value of all points in the dataset. In a next step, one computes the pairwise distance only for those pairs, hashing to the same value. This process is then repeated for different hash functions until the closest pair is found. The initial algorithm by Indyk-Motwani achieves a time complexity of $n^{\log_2(\frac{2}{1-\gamma})}$. In general, a time lower bound of $n^{\frac{1}{1-\gamma}}$ is known for LSH based algorithms [9, 18]. In [9] Dubiner also gives an abstract idea of an algorithm achieving this lower bound.[2] Later May and Ozerov [17] gave the first concrete algorithmic description following similar design principles, also achieving the mentioned lower bound. Additionally, current data-dependent hashing schemes [2], where the hash function depends also on the actual points in the dataset, improve on the initial idea by Indyk-Motwani and also match the time lower bound of [9, 18].

In the uniform setting Valiant [21] was able to circumvent the lower bound by leveraging fast matrix multiplication and hence breaking out of the LSH framework to give an algorithm that runs in time $n^{1.63}\text{poly}(d)$. Remarkably, the complexity exponent of Valiant's algorithm does not depend on the relative distance γ at all. Later this bound was improved to $n^{1.58}\text{poly}(d)$ by Karpa et al. [13] and simplified in an elegant algorithm by Alman [1] achieving the same complexity.

All mentioned algorithms have in common, that they assume a dimension of $d = c(n)\log(n)$, where $c(n)$ is at least a big constant. The explicit size of those constants is usually not stated, instead an asymptotic argument yields their existence. Moreover, the results by [2, 9, 21] for instance assume even larger dimensions where $\frac{1}{c(n)} = o(1)$. Here, the algorithm by May-Ozerov forms an exception by being applicable for any $c(n) \geq \frac{1}{1-H(\frac{\gamma}{2})}$, where $H(\cdot)$ denotes the binary entropy function. Nevertheless, the mentioned lower bound is only achieved for $c(n)$ approaching infinity. Recently, Xie, Xu and Xu [23] proposed a new algorithm based on decoding the points of the data set according to some random code, exploiting that close vectors are more likely to be decoded to the same word. Their algorithm is also applicable for any $c(n)$ that allows to bound the number of pairs attaining relative distance γ to a constant number with high probability. The authors are able to derandomize their approach and, thus, obtain the fastest known deterministic algorithm for small constants $c(n)$. However, if one also considers probabilistic procedures, their method is inferior to the one by May-Ozerov.

1.1   Our Contribution

We design a randomized algorithm, which achieves the best-known running time for solving the Closest Pair Problem on uniformly random input, when the dimension d is small, which means $c(n)$ being a small constant. Additionally, our algorithm matches the running time of

---

[1] here we ignore polylogarithmic factors in the dataset size
[2] A precise algorithmic description and a proof of the running time in the case where the vector length is restricted (referred to as *limited amount of data case* in his work) is missing.

(a) Dimension $d = 4\log_2(n)$.   **(b)** Dimension $d = 2\log_2(n)$.   **(c)** Dimension $d = 1.2\log_2(n)$.
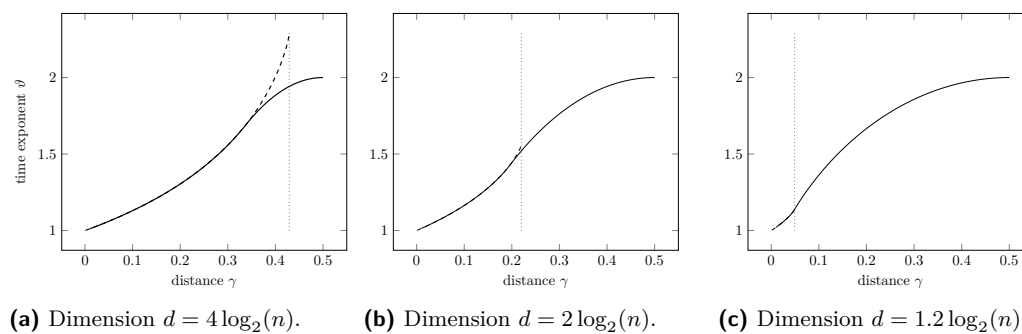
Figure 1 Time complexity exponent ϑ as a function of the relative distance γ of the closest pair for different dimensions. The running time is of the form $n^{\vartheta} \cdot \text{poly}(d)$, where the dashed line represents May-Ozerov's algorithm and the solid line depicts the exponent of our new algorithm. The dotted line gives the maximal γ for which the algorithm by May-Ozerov is still applicable.

the best known LSH algorithms for larger values of $c(n)$ and still matches the time lower bound for LSH based schemes if $\frac{1}{c(n)} = o(1)$. To quantify we give in Figure 1 the achieved runtime exponent for $c(n) \in \{1.2, 2, 4\}$ of our algorithm in comparison to May-Ozerov. As indicated by the graphics, our algorithm can be seen as a natural extension of the May-Ozerov algorithm to higher distances. Moreover, we show that for large distances our algorithm is indeed optimal. Note that apart from the May-Ozerov algorithm *none* of the previously mentioned algorithms is applicable for those choices of $c(n)$. A detailed comparison to the result of May and Ozerov is given right after Theorem 3.

Our improvements over previous work lie in the high density regime, which implies multiple solutions to the Closest Vector Problem. Since the distance alone does not allow to distinguish the planted pair in such cases at least a non-negligible fraction of those pairs needs to be reported, to find the planted pair. The relevance of this setting is mostly given by cryptographic [4, 17] and coding-theoretic [7, 10, 11] applications, precisely the decoding of linear codes. Here, the searched error-vector has known weight and is usually constructed in a tree-wise meet-in-the-middle fashion. Even though the error-vector is usually unique the tree-wise decomposition of the problem introduces multiple solution candidates, such that the lists in the tree can even hold exponentially many pairs with relative distance smaller than γ. However, in such settings, only the elements attaining relative distance γ can possibly sum to the searched error-vector. In the algorithm of Both and May [7], which is the fastest known for decoding random binary linear codes, the authors had to define naive fallback routines for the high density case, for which the May-Ozerov algorithm is not applicable. Here our result allows at least for a unified analysis of the algorithm without the need of fallback routines and in the best case leads to runtime improvements. Also, the generalization of the May-Ozerov nearest neighbor algorithm to $\mathbb{F}_q$ by Hirose [11] suffers similar limitations regarding the high density regime, while also forming the basis for the fastest known decoding algorithm for random linear codes over $\mathbb{F}_q$ [10].

Technically our algorithm follows similar design principles as [9, 17]. At its core, these algorithms group the elements of the given datasets recursively into buckets according to some criterion, which fulfills properties that are similar to those of locality-sensitive hash functions. As the buckets in the recursion are decreasing in size, at the end of the recursion they become small enough to compute the pairwise distance of all contained elements naively.

In contrast to previous works, we exchange the used bucket criteria, which allows us to significantly simplify the algorithms' analysis as well as improve for the mentioned parameter regimes. Also, our approach is applicable for any $c(n)$, thus we are able to remove the restriction $c(n) \geq \frac{1}{1 - H(\frac{\gamma}{2})}$.

Following May-Ozerov and Dubiner, we study the bichromatic version of the Closest Pair Problem, which takes as input two datasets rather than one and the goal is to find the closest pair between those given datasets. Obviously, there exists a randomized reduction between the Closest Pair Problem and its bichromatic version, but our algorithm can also be easily adapted to the single dataset case. However, May and Ozerov require the elements within each dataset to be pairwise independent of each other, as a minor contribution we get rid of this restriction, too.

Also, we investigate the algorithms' performance on different input distributions. Therefore we give a modular analysis, which allows for an easy exchange of dataset distribution as well as the choice of bucketing criterion. We also give numerical upper bounds for the algorithm's complexity exponent on some exemplary input distributions. These examples suggest that the chosen criterion is well suited as long as the distance between input elements concentrates around $\frac{d}{2}$ (as in the case of random input lists), while being non-optimal as soon as the expected distance decreases.

We also address an open research question regarding the practical applicability of algorithms following the design of [9,17] raised by May and Ozerov. As their algorithm inherits a huge polynomial overhead in time and space, they left it as an open problem to give a more practical algorithm following a similar design. While our analysis first suggests an equally high overhead, we are able to give an efficient implementation of our algorithm, which requires in addition to the input dataset only constant space. Also, our practical experiments show that most of the overhead of our algorithm is an artifact of the analysis and can be circumvented in practice so that our algorithm performs well compared to a quadratic search baseline.

2   Preliminaries

2.1   Notation

For $a, b \in \mathbb{N}$, $a \leq b$ we denote $[a, b] := \{a, a + 1, \ldots, b - 1, b\}$. In particular, let $[b] := [1, b]$. For a vector $\mathbf{v} \in \mathbb{F}_2^d$ and $I \in [d]$ let $\mathbf{v}_I$ be the projection of $\mathbf{v}$ onto the coordinates indexed by I, i.e. for $\mathbf{v} = (v_1, v_2, \ldots, v_d)$ and $I = \{i_1, i_2, \ldots, i_k\}$ we have $\mathbf{v}_I = (v_{i_1}, \ldots, v_{i_k}) \in \mathbb{F}_2^k$. We denote the uniform distribution on $\mathbb{F}_2^d$ as $\mathcal{U}\left(\mathbb{F}_2^d\right)$. We define $f(n) = \tilde{\mathcal{O}}\left(g(n)\right) :\Leftrightarrow \exists i \in \mathbb{N}: f(n) = \mathcal{O}\left(g(n) \cdot \log^i(g(n))\right)$, i.e. the tilde additionally suppresses polylogarithmic factors in comparison to the standard Landau notation $\mathcal{O}$.

Furthermore, we consider all logarithms having base 2. Define the binary entropy function as $H(x) = -x \log(x) - (1 - x) \log(1 - x)$ for $x \in (0, 1)$, and additionally $H(0) = H(1) := 0$. Using this together with Stirling's formula $n! = \Theta\left(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n\right)$ we obtain $\binom{n}{\gamma n} = \tilde{\Theta}\left(2^{H(\gamma)n}\right)$. We additionally define $H^{-1}: [0, 1] \to [0, \frac{1}{2}]$ to be the inverse of the left branch of H.

2.2   Closest Pair Definition

In this work, we consider the Bichromatic Closest Pair Problem in Hamming metric. Here, the inputs are two lists of equal size containing elements drawn uniformly at random from $\mathbb{F}_2^d$ plus a planted pair, whose Hamming distance is γd for some known γ. More formally, we state the problem in the following definition. To allow for easy comparison to the result of May-Ozerov, we follow their notation using the dimension as the primary difficulty parameter. Thus we let the list sizes be $n := 2^{\lambda d}$, which means $\lambda = \frac{1}{c(n)}$, where $d = c(n) \log n$.

▶ **Definition 1** (Bichromatic Closest Pair Problem). *Let $d \in \mathbb{N}$, $\gamma \in \left[0, \frac{1}{2}\right]$ and $\lambda \in (0, 1]$. Let $L_1 = (\mathbf{v}_i)_{i \in [2^{\lambda d}]}, L_2 = (\mathbf{w}_i)_{i \in [2^{\lambda d}]} \in \left(\mathbb{F}_2^d\right)^{2^{\lambda d}}$ be two lists containing elements uniformly drawn at random, together with a distinguished pair $(\mathbf{x}, \mathbf{y}) \in L_1 \times L_2$ with $\mathrm{wt}(\mathbf{x} + \mathbf{y}) = \gamma d$. We further assume that for each i, j the vectors $\mathbf{v}_i$ and $\mathbf{w}_j$ are pairwise stochastically independent. The Closest Pair Problem $\mathcal{CP}_{d,\lambda,\gamma}$ asks to find this closest pair $(\mathbf{x}, \mathbf{y})$ given L_1, L_2 and the weight parameter γ. We call $(\mathbf{x}, \mathbf{y})$ the solution of the $\mathcal{CP}_{d,\lambda,\gamma}$ problem.*

First, note that $\lambda \leq 1$ is not a real restriction since for $\lambda > 1$ the lists must contain duplicates, which can be safely removed, giving us a problem instance with $\lambda \leq 1$. We also consider the Closest Pair Problem on input lists whose elements are distributed according to some distribution $\mathcal{D}$ different from the uniform one used in Definition 1. To indicate this, we refer to the $\mathcal{CP}_{d,\lambda,\gamma}$ *over distribution* $\mathcal{D}$. Note that in this case, the meaningful upper bound for λ is the entropy of $\mathcal{D}$.

Technically speaking, it is also not necessary to know the value of γ, as the time complexity of appropriate algorithms to solve the $\mathcal{CP}_{d,\lambda,\gamma}$ problem is solely increasing in γ. Thus if γ is unknown, one would apply the algorithm for each $\gamma d = 0, 1, 2, \ldots$ until the solution is found, which results at most in an overhead polynomial in d.

It is well known, that any LSH based algorithm solving the problem of Definition 1 with non-negligible probability needs at least time complexity $|L_1|^{\frac{1}{1-\gamma}} = 2^{\frac{\lambda d}{1-\gamma}}$ [9,18]. However, this lower bound assumes the promised pair to be uniquely distinguishable from all other pairs in $L_1 \times L_2$. Obviously, if the relation of γ and λ lets us expect more than the promised pair of distance γd in the input lists, an algorithm solving the Closest Pair Problem needs to find all (or at least a non-negligible fraction) of these closest pairs.[3] Hence, if the input lists contain E closest pairs the time complexity of any algorithm solving the problem is lower bounded by

$$\tilde{\Omega}\left(\max(2^{\frac{\lambda d}{1-\gamma}},\ E)\right)$$

Let $(\mathbf{v}, \mathbf{w}) \in L_1 \times L_2 \setminus \{(\mathbf{x}, \mathbf{y})\}$ be arbitrary list elements. If the elements are chosen independently and uniformly at random, as stated in Definition 1 we expect E to be of size

$$\mathbb{E}[E] = (|L_1 \times L_2| - 1) \cdot \Pr\left[\mathrm{wt}(\mathbf{v} + \mathbf{w}) = \gamma d\right] + \underbrace{1}_{\text{from } (\mathbf{x},\mathbf{y})}$$

$$= \left(2^{2\lambda d} - 1\right) \cdot \frac{\binom{d}{\gamma d}}{2^d} + 1$$

$$= \tilde{\Theta}\left(2^{(2\lambda + H(\gamma) - 1)d}\right) \ ,$$

and, thus, the time complexity to solve the $\mathcal{CP}_{d,\lambda,\gamma}$ is lower bounded by

$$T_{\text{opt}} = \tilde{\Omega}\left(\max\left(2^{\frac{\lambda d}{1-\gamma}},\ 2^{(2\lambda + H(\gamma) - 1)d}\right)\right) \ . \tag{1}$$

3 Our new Algorithm

Our algorithm groups the input elements according to some criterion into several buckets, each one representing a new closest pair instance with smaller list size. We then apply this bucketing procedure recursively until the buckets contain few enough elements to eventually solve the Closest Pair Problem represented by them via a naive quadratic search algorithm, the exhaustive search.

---

[3] Note that in such a scenario the searched $(\mathbf{x}, \mathbf{y})$ is probably not the pair with the smallest Hamming distance, however, we still refer to elements attaining Hamming distance γd as *closest pairs*.
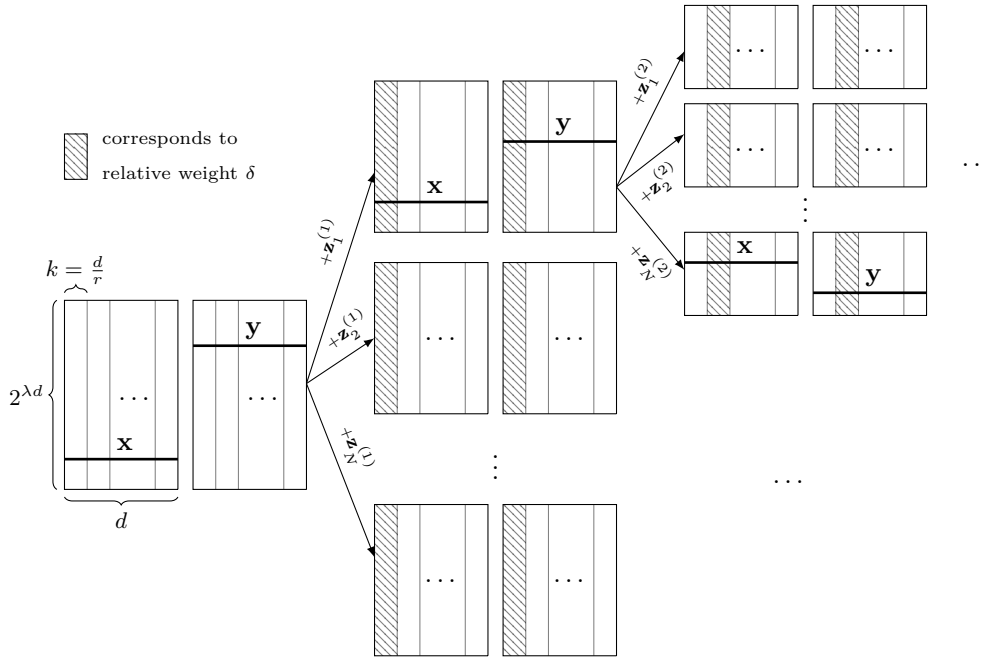
Figure 2 We start off on the left side of the illustration with the two input lists L_1, L_2 containing the closest pair $(\mathbf{x}, \mathbf{y})$. Going right, in each iteration of the algorithm, N different $\mathbf{z}_i^{(j)}$ are randomly chosen and all of the list elements are tested if they fulfill the bucketing criterion. The crosshatched pattern indicates the parts where the bucket criterion is fulfilled, i.e. the list vectors differ from $\mathbf{z}_i^{(j)}$ in δk positions.

As a bucketing criterion, we choose the weight of the vectors after adding a randomly drawn vector $\mathbf{z}$ from $\mathbb{F}_2^d$. Thus, each bucket is represented by a vector $\mathbf{z}$ and only those elements $\mathbf{v}$ are added to the bucket, which satisfy $\mathrm{wt}(\mathbf{v} + \mathbf{z}) = \delta d$, where δ is determined later.

More precisely in each recursive iteration, our algorithm works only on equally large blocks of the input vectors and not on the full d coordinates, i.e. the weight condition is only checked on the current block. This is a technical necessity to obtain independence of vectors in the same bucket on fresh blocks. Let us formally define the notion of blocks.

▶ **Definition 2** (Block). *Let $d, r \in \mathbb{N}$ with $r \mid d$ and $i \in [r]$. Then we denote the i-th* block *of* $[d]$ *as*

$$B_{i,r}^d := \left[(i-1)\frac{d}{r} + 1, \ i\frac{d}{r} \right] \ .$$

Note that $[d] = \bigcup_{i \in [r]} B_{i,r}^d$ and $\left| B_{i,r}^d \right| = \frac{d}{r}$ for each $i \in [r]$. Furthermore, the blocks are disjoint. For a leaner notation and since the role of d does not change in the course of this paper, we omit the index d in the following, thus we write $B_{i,r} := B_{i,r}^d$.

Note that May and Ozerov choose the weight of the vectors on random projections as a criterion. In comparison to our variant, their approach involves more parameters and requires extensive re-randomizations of the instance to achieve good success probabilities which together complicates analysis considerably. We cannot rule out the possibility that a different analysis of the May-Ozerov algorithm would allow for an application in the high density regime. However, the complexities of this hypothetical variant are unclear, while our version allows for easy analysis and yields provably optimal complexities in this regime.

In each iteration, we choose the number N of buckets in such a way that with overwhelming probability the closest pair lands in at least one of the buckets. Hence, our algorithm creates a tree with branching factor N with the distinguished pair being contained in one of the leaves. The deeper we get into the tree, the smaller and, hence, the easier the closest pair instances get. An algorithmic description of the whole procedure is given in pseudocode in Algorithm 1. For convenience, a summary of all parameter choices made in line 1 of the algorithm can be found in Equation (8) at the end of Section 3.

▪ **Algorithm 1** CLOSEST-PAIR(L_1, L_2, γ).

Input: lists $L_1, L_2 \in \left(\mathbb{F}_2^d\right)^{2^{\lambda d}}$, weight parameter $\gamma \in \left[0, \frac{1}{2}\right]$
Output: list L containing the solution $(\mathbf{x}, \mathbf{y}) \in L_1 \times L_2$ to the $\mathcal{CP}_{d,\lambda,\gamma}$
 1: Set $r, P, N \in \mathbb{N}$, $\delta \in \left[0, \frac{1}{2}\right]$ properly and define $k := \frac{d}{r}$
 2: **for** P permutations π **do**   ▷ permutation on the bit positions
 3:     Stack $S := [(\pi(L_1), \pi(L_2), 0)]$
 4:     $L \leftarrow \emptyset$
 5:     **while** S is not empty **do**
 6:         $(A, B, i) \leftarrow S.\text{pop}()$
 7:         **if** $i < r$ **then**
 8:             **for** N randomly chosen $\mathbf{z} \in \mathbb{F}_2^k$ **do**
 9:                 $A' \leftarrow (\mathbf{v} \in A \mid \text{wt}\big((\mathbf{v} + \mathbf{z})_{B_{i+1,r}}\big) = \delta k)$
10:                 $B' \leftarrow (\mathbf{w} \in B \mid \text{wt}\big((\mathbf{w} + \mathbf{z})_{B_{i+1,r}}\big) = \delta k)$
11:                 $S.\text{push}((A', B', i+1))$
12:         **else**
13:             **for** $\mathbf{v} \in A, \mathbf{w} \in B$ **do**   ▷ Naive search
14:                 **if** $\text{wt}(\mathbf{v} + \mathbf{w}) = \gamma d$ **then**
15:                     $L \leftarrow L \cup \{(\mathbf{v}, \mathbf{w})\}$
16:     **return** L

The following theorem gives the time complexity of our algorithm to solve the $\mathcal{CP}_{d,\lambda,\gamma}$.

▶ **Theorem 3.** *Let $\gamma \in \left[0, \frac{1}{2}\right]$ and $\lambda \in [0,1]$. Then Algorithm 1 solves the $\mathcal{CP}_{d,\lambda,\gamma}$ problem with overwhelming success probability in expected time $2^{\vartheta d(1+o(1))}$, where*

$$
\vartheta = \begin{cases} (1-\gamma)\left(1 - H\left(\frac{\delta^\star - \frac{\gamma}{2}}{1-\gamma}\right)\right) & \text{for } \gamma \leq \gamma^\star \\ 2\lambda + H(\gamma) - 1 & \text{for } \gamma > \gamma^\star \ , \end{cases}
$$

with $\delta^\star := H^{-1}(1 - \lambda)$ and $\gamma^\star := 2\delta^\star(1 - \delta^\star)$.

Note that the case distinction marks the transition to the high density regime. Precisely, the transition happens when the amount of closest pairs becomes larger than the running time in the first case. In this first case, where $\gamma \leq \gamma^\star$ our algorithm exactly matches the running time of the May-Ozerov algorithm, which itself is shown to match the lower bound for LSH based approaches whenever λ approaches zero [17] (see also Lemma 8). In the second case, where $\gamma > \gamma^\star$ the running time of our algorithm becomes linear in the number of closest pairs, hence it matches the lower bound from Equation (1), while the running time of May-Ozerov stays as in the first case. Our algorithm hence optimally extends the May-Ozerov algorithm to the high density regime.

We establish the proof of Theorem 3 in a series of lemmata and theorems. Note that any bucketing algorithm heavily depends on two probabilities specific to the chosen bucketing criterion. First, the probability that any element falls into a bucket, which we call p in the

remainder of this work. This probability is mainly responsible for the lists' sizes throughout the algorithm. The second relevant probability, which we call q describes the event of both, $\mathbf{x}$ and $\mathbf{y}$, falling into the same bucket, where $(\mathbf{x}, \mathbf{y})$ is the solution to the $\mathcal{CP}_{d,\lambda,\gamma}$ problem. This is the probability of $(\mathbf{x}, \mathbf{y})$ *surviving* one iteration meaning that q determines the success probability of the algorithm. In summary, for our choice of bucketing criteria, we get

$$p := \Pr_{\mathbf{z}} \left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}}) = \delta k \right] \text{ for any } \mathbf{v} \in \mathbb{F}_2^k \text{ and}$$

$$q := \Pr_{\mathbf{z}} \left[\mathrm{wt}((\mathbf{x} + \mathbf{z})_{B_{i,r}}) = \mathrm{wt}((\mathbf{y} + \mathbf{z})_{B_{i,r}}) = \delta k \right] \quad, \tag{2}$$

where $k = \frac{d}{r}$ is the block width. If we assume that the γd differing coordinates of $\mathbf{x}$ and $\mathbf{y}$ distribute evenly into the r blocks, i.e. $\mathrm{wt}((\mathbf{x} + \mathbf{y})_{B_{i,r}}) = \gamma k$ for each i, these probabilities are independent of i for δk fixed. This property is ensured for at least one of the P permutations in Algorithm 1 with overwhelming probability, as we will see in the proof of Theorem 4.

We determine the exact form of p and q later. First, we are going to prove the following statement about the expected running time of Algorithm 1 in dependence on both probabilities.

▶ **Theorem 4.** *Let p and q be as defined in Equation (2), $\gamma \in \left[0, \frac{1}{2}\right]$, $\lambda \in [0, 1]$ and $r = \frac{\lambda d}{\log^2 d}$. Then Algorithm 1 solves the $\mathcal{CP}_{d,\lambda,\gamma}$ problem in expected time*

$$\max \left(q^{-r}, \frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{\left(2^{\lambda d} \cdot p^r\right)^2}{q^r} \right)^{1+o(1)}$$

with a success probability overwhelming in d.

Proof. First, we are going to prove the statement about the time complexity.

The algorithm maintains a stack, containing list pairs together with an associated counter. In every iteration of the loop in line 5, one element is removed from the stack and if the counter i associated with this element is smaller than r, N additional elements $(A', B', i+1)$ are pushed to the stack in line 11. Let us consider the elements on the stack as nodes in a tree of depth r, where all elements with associated counter i are siblings on level i of the tree. Also, depict the elements pushed to the stack in line 11 as child nodes of the currently processed node (A, B, i). Then the total number of elements with associated counter i pushed to the stack is bounded by the number of nodes on level i in a tree with branching factor N, which is N^i.

Next, let us determine the lists' sizes on level i of that tree. Therefore, let the expected size of lists on level i be $\mathcal{L}_i$. As these lists are constructed from the lists of the previous level by testing the weight condition in line 9 and 10, it holds that

$$\mathcal{L}_i = \mathcal{L}_{i-1} \cdot \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}})) = \delta k \right] := \mathcal{L}_{i-1} \cdot p \ ,$$

where $i > 0$ and by construction $\mathcal{L}_0 = |L_1|$. By substitution we get

$$\mathcal{L}_i = |L_1| \cdot p^i \ , \text{ for } i = 0, \ldots, r.$$

Now, we are able to compute the time needed to create the nodes on level i of the tree. Observe that for the creation of a level-i node we need to linearly scan through the larger lists of a node on level $i-1$ to check the weight conditions. Thus, to construct all N^i nodes of level i we need a total time of

$$T_i = \tilde{\mathcal{O}}\left(\mathcal{L}_{i-1} \cdot N^i\right) = \tilde{\mathcal{O}}\left(|L_1| \cdot p^{i-1} \cdot N^i\right) \ ,$$

for each $0 < i \leq r$. Eventually, the list pairs on level r are matched by a naive search with quadratic runtime resulting in

$$T_{r+1} = \tilde{\mathcal{O}}\left(N^r \cdot \mathbb{E}[|A_r| \cdot |B_r|]\right) \ ,$$

where A_r, B_r describe the lists of a level-r node.

The expected value of the product, now, depends on the chosen input distribution. We next argue that for the given input distribution we have

$$\mathbb{E}[|A_r| \cdot |B_r|] = \mathcal{O}\left(\mathbb{E}[|A_r|] \cdot \mathbb{E}[|B_r|]\right) = \mathcal{O}(\mathcal{L}_r^2) \ .$$

To see this, first note that for $\mathbf{v}, \mathbf{w}, \mathbf{z}$ independent and uniform, $\mathbf{v} + \mathbf{z}$ and $\mathbf{w} + \mathbf{z}$ are also independent and uniform. This in turn implies

$$\begin{aligned}
&\Pr\left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}})) = \delta k, \mathrm{wt}((\mathbf{w} + \mathbf{z})_{B_{i,r}})) = \delta k\right] \\
&= \Pr\left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}})) = \delta k\right] \cdot \Pr\left[\mathrm{wt}((\mathbf{w} + \mathbf{z})_{B_{i,r}})) = \delta k\right] \\
&= p^2
\end{aligned}$$

since deterministic functions of independent random variables are still independent. This also works for either $\mathbf{v} = \mathbf{x}$ or $\mathbf{w} = \mathbf{y}$, but not for $(\mathbf{v}, \mathbf{w}) = (\mathbf{x}, \mathbf{y})$. In this case, however, we have $\Pr\left[\mathrm{wt}((\mathbf{x} + \mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{y} + \mathbf{z})_{B_{i,r}})) = \delta k\right] = q$ by definition. With this insight, we can express $\mathbb{E}[|A_i| \cdot |B_i|]$ in terms of $\mathbb{E}[|A_{i-1}| \cdot |B_{i-1}|]$ for each i via

$$\begin{aligned}
\mathbb{E}[|A_i| \cdot |B_i| \mid A_{i-1}, B_{i-1}] &= \sum_{\substack{\mathbf{v} \in A_{i-1}, \ \mathbf{w} \in B_{i-1} \\ (\mathbf{v}, \mathbf{w}) \neq (\mathbf{x}, \mathbf{y})}} \Pr\left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w} + \mathbf{z})_{B_{i,r}}) = \delta k\right] \\
&\quad + \Pr\left[\mathrm{wt}((\mathbf{x} + \mathbf{z})_{B_{i,r}})) = \delta k, \mathrm{wt}((\mathbf{y} + \mathbf{z})_{B_{i,r}})) = \delta k\right] \\
&= (|A_{i-1}| \cdot |B_{i-1}| - 1)p^2 + q \\
&\leq |A_{i-1}| \cdot |B_{i-1}| \cdot p^2 + 1 \ ,
\end{aligned}$$

Applying the Law of total Expectation we obtain

$$\mathbb{E}[|A_i| \cdot |B_i|] = \mathbb{E}[\mathbb{E}[|A_i| \cdot |B_i| \mid A_{i-1}, B_{i-1}]] \leq \mathbb{E}[|A_{i-1}| \cdot |B_{i-1}|] \cdot p^2 + 1 \qquad (3)$$

Successive application of Equation (3) yields

$$\mathbb{E}[|A_r| \cdot |B_r|] \leq \mathbb{E}[|L_1| \cdot |L_2|] \cdot p^{2r} + r = 2^{2\lambda d} p^{2r} + r = \mathcal{O}(\mathcal{L}_r^2) \qquad (4)$$

Finally, the algorithm is repeated for P different permutations on the bit positions of elements in L_1, L_2. In summary, the expected time complexity to build all list becomes the sum of the T_i multiplied by P, thus, by choosing $N := \frac{d}{q}$ and $P = (d+1)^{r+1}$ we get

$$\begin{aligned}
T' = P \cdot \sum_{i=1}^{r+1} T_i &\leq (d+1)^{r+1} \cdot \left(\sum_{i=1}^r N^i \cdot |L_1| \cdot p^{i-1} + (|L_1| \cdot p^r)^2 \cdot N^r \right) \\
&= (d+1)^{r+1} \cdot \left(\sum_{i=1}^r \frac{|L_1| \cdot d^i}{q} \cdot \left(\frac{p}{q}\right)^{i-1} + \frac{(|L_1| \cdot p^r)^2 \cdot d^r}{q^r} \right) \\
&\leq (d+1)^{2r+1} \cdot \left(\frac{r \cdot |L_1| \cdot p^{r-1}}{q^r} + \frac{(|L_1| \cdot p^r)^2}{q^r} \right) \\
&= \max\left(\frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{\left(2^{\lambda d} \cdot p^r\right)^2}{q^r} \right)^{1+o(1)} \ ,
\end{aligned}$$

where the inequality follows from the fact that $\frac{p}{q} \geq 1$ since

$$
\begin{aligned}
q &= \Pr\left[\text{wt}((\mathbf{x}+\mathbf{z})_{B_{i,r}}) = \text{wt}((\mathbf{y}+\mathbf{z})_{B_{i,r}}) = \delta k\right] \\
&\leq \Pr\left[\text{wt}((\mathbf{x}+\mathbf{z})_{B_{i,r}}) = \delta k\right] \\
&= p \ ,
\end{aligned}
$$

and the final equality stems from the fact that $|L_1| = 2^{\lambda d}$ and $r = o(\frac{\lambda d}{\log d})$ as given in the theorem.

Note that T' disregards the fact that no matter how small the lists in the tree become, the algorithm needs to traverse all

$$
T'' = \tilde{\mathcal{O}}\left(N^r\right) = \tilde{\mathcal{O}}\left(\left(\frac{d}{q}\right)^r\right)
$$

nodes of the tree. Hence, the expected time complexity of the whole algorithm is

$$
T = \max(T', T'') \ ,
$$

which proves the claim.

Let us now consider the success probability of the algorithm. Therefore, we assume that the chosen permutation distributes the weight on $\mathbf{x}+\mathbf{y}$ such that in every block of length r the weight is equal to $\frac{\gamma d}{r}$, which we describe as a *good* permutation. The probability of a random permutation π distributing the weight in such a way is

$$
\Pr\left[\text{good } \pi\right] = \Pr\left[\text{wt}\left(\pi(\mathbf{x}+\mathbf{y})_{B_{i,r}}\right) = \frac{\gamma d}{r}, \text{ for } i = 1, \ldots, r\right] = \frac{\left(\frac{\frac{d}{r}}{\frac{\gamma d}{r}}\right)^r}{\binom{d}{\gamma}} \geq \left(\frac{d}{r}+1\right)^{-r} \ .
$$

Thus, the probability of at least one out of $(d+1)^{r+1}$ chosen permutations being good is

$$
p_1 := \Pr\left[\text{at least one good } \pi\right]
$$

$$
= 1 - (1 - \Pr\left[\text{good } \pi\right])^{(d+1)^{r+1}} = 1 - \left(1 - \left(\frac{d}{r}+1\right)^{-r}\right)^{(d+1)^{r+1}} \geq 1 - e^{-d} \ .
$$

The algorithm succeeds, whenever there exists a leaf node in the tree, containing the distinguished pair $(\mathbf{x}, \mathbf{y})$. As every node in the tree is constructed based on its parent, it follows that all nodes on the path from the root to that leaf need to contain $(\mathbf{x}, \mathbf{y})$. By definition the probability of $\mathbf{x}$ and $\mathbf{y}$ satisfying the bucket criterion at the same time (thus for the same $\mathbf{z}$) is q and since we condition on a good permutation, q is equal for every considered block. Let us define indicator variables X_j for the first level, where $X_j = 1$ iff the j-th node contains $(\mathbf{x}, \mathbf{y})$. Observe that the X_j for independent choices of $\mathbf{z}$ are independent. Thus, clearly the number of trials until $(\mathbf{x}, \mathbf{y})$ is contained in any node on level one is distributed geometrically with parameter q. Hence, the probability of the solution being contained in at least one node on the first level is

$$
p_2 := \Pr\left[\exists (A, B, 1) \in S : (\mathbf{x}, \mathbf{y}) \in A \times B\right]
$$

$$
= 1 - (1-q)^N = 1 - (1-q)^{\frac{d}{q}} \geq 1 - e^{-d} \ .
$$

Now, imagine the pair being contained in some level-i node. Considering that node, we have with the same probability p_2 again that at least one child contains the solution, and the same argument holds until we reach the leaves. Also, by the independent choices of $\mathbf{z}$ the events remain independent which implies that the probability of $(\mathbf{x}, \mathbf{y})$ being contained in a level-r list is p_2^r. In summary, the success probability is

$$\Pr[\text{success}] = p_1 \cdot p_2^r \geq (1 - e^{-d})^{r+1} \geq 1 - \frac{r+1}{e^d} \geq 1 - \frac{d}{e^d} \ . \qquad \blacktriangleleft$$

The proof of Theorem 4 already shows, how different distributions may affect the complexity of the algorithm by changing the expected value $\mathbb{E}[|A_r| \cdot |B_r|]$. This influence on the algorithms complexity by different input distributions is further investigated in Section 4.

In the next two lemmata, we will determine the exact forms of p and q to conduct the run time analysis.

▶ **Lemma 5.** *Let* $k \in \mathbb{N}$, $\delta \in [0, 1]$. *If* $\mathbf{x} \in \mathbb{F}_2^k$ *and* $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^k)$ *then*

$$\Pr_{\mathbf{z}}[\text{wt}(\mathbf{x} + \mathbf{z}) = \delta k] = \binom{k}{\delta k} \left(\frac{1}{2}\right)^k .$$

Proof. Since $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^k)$, the probability is

$$\frac{\left|\{\mathbf{z} \in \mathbb{F}_2^k \mid \text{wt}(\mathbf{x} + \mathbf{z}) = \delta k\}\right|}{\left|\mathbb{F}_2^k\right|} .$$

To compute the numerator, note that $\text{wt}(\mathbf{x} + \mathbf{z}) = \delta k$ means that $\mathbf{x}$ and $\mathbf{z}$ differ in δk out of k coordinates, for which there are $\binom{k}{\delta k}$ possibilities. Using $\left|\mathbb{F}_2^k\right| = 2^k$, the lemma follows. ◀

Before we continue, let us make a small definition.

▶ **Definition 6.** *Let* $k \in \mathbb{N}$ *and* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^k$. *Then we define* $D(\mathbf{x}, \mathbf{y}) \subseteq [k]$ *to be the set of coordinates where* $\mathbf{x}$ *and* $\mathbf{y}$ *differ, i.e.*

$$D(\mathbf{x}, \mathbf{y}) := \{i \in [k] \mid \mathbf{x}_i \neq \mathbf{y}_i\}.$$

Furthermore, let $S(\mathbf{x}, \mathbf{y}) := [k] \setminus D(\mathbf{x}, \mathbf{y})$ *be the set of coordinates where they are the same.*

Now we derive the exact form of the probability q of a pair with difference γk falling into the same bucket.

▶ **Lemma 7.** *Let* $k \in \mathbb{N}$, $\delta \in [0, 1]$. *If* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^k$ *with* $\text{wt}(\mathbf{x} + \mathbf{y}) = \gamma k$ *and* $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^k)$. *Then*

$$\Pr_{\mathbf{z}}[\text{wt}(\mathbf{x} + \mathbf{z}) = \text{wt}(\mathbf{y} + \mathbf{z}) = \delta k] = \binom{\gamma k}{\frac{1}{2}\gamma k}\binom{(1-\gamma)k}{(\delta - \frac{\gamma}{2})k}\left(\frac{1}{2}\right)^k .$$

Proof. Let

$$A := \{\mathbf{z} \in \mathbb{F}_2^k \mid \text{wt}(\mathbf{x} + \mathbf{z}) = \text{wt}(\mathbf{y} + \mathbf{z}) = \delta k\}.$$

In analogy to Lemma 5, the probability we search for is $\frac{|A|}{|\mathbb{F}_2^k|} = |A| \cdot \left(\frac{1}{2}\right)^k$.

In the following, let $\gamma_{\mathbf{x}} := \text{wt}(\mathbf{x} + \mathbf{z})$ and analogously $\gamma_{\mathbf{y}} := \text{wt}(\mathbf{y} + \mathbf{z})$. Now observe that every coordinate z_i of $\mathbf{z}$ with $i \in S(\mathbf{x}, \mathbf{y})$, so belonging to the set of equal coordinates between $\mathbf{x}$ and $\mathbf{y}$, either contributes to both $\gamma_{\mathbf{x}}$ *and* $\gamma_{\mathbf{y}}$ with one or does not affect either one of them. Let us define the amount of the z_i's with $i \in S(\mathbf{x}, \mathbf{y})$ that contribute to the weight as $a := |S(\mathbf{x}, \mathbf{y}) \cap D(\mathbf{x}, \mathbf{z})|$.

Now consider the z_i's with $i \in D(\mathbf{x}, \mathbf{y})$. Clearly, any such z_i contributes *either* to $\gamma_\mathbf{x}$ or to $\gamma_\mathbf{y}$. Thus, let us define the number of those z_i with $i \in D(\mathbf{x}, \mathbf{y})$ that contribute to $\gamma_\mathbf{x}$ as $b_\mathbf{x} := |D(\mathbf{x}, \mathbf{y}) \cap D(\mathbf{x}, \mathbf{z})|$ and analogously those which contribute to $\gamma_\mathbf{y}$ as $b_\mathbf{y} := |D(\mathbf{x}, \mathbf{y}) \cap D(\mathbf{y}, \mathbf{z})|$. Obviously we have

$$b_\mathbf{x} + b_\mathbf{y} = |D(\mathbf{x}, \mathbf{y})| = \gamma k \tag{5}$$

On the other hand we are only interested in those $\mathbf{z}$ for which $\gamma_\mathbf{x} = \gamma_\mathbf{y} = \delta k$, which yields the two equations

$$\gamma_\mathbf{x} = a + b_\mathbf{x} = \delta k \tag{6}$$
$$\gamma_\mathbf{y} = a + b_\mathbf{y} = \delta k \tag{7}$$

All three equations together yield the unique solution

$$b_\mathbf{x} = b_\mathbf{y} = \frac{\gamma k}{2} \text{ and } a = \left(\delta - \frac{\gamma}{2}\right) k \ .$$

This shows the following: If $\mathbf{z} \in A$, it is necessary that $\mathbf{z}$ differs from $\mathbf{x}$ (analogously $\mathbf{y}$) in exactly
- $\frac{\gamma}{2}k$ out of γk coordinates of $D(\mathbf{x}, \mathbf{y})$ and
- $\left(\delta - \frac{\gamma}{2}\right) k$ out of $(1 - \gamma)k$ coordinates of $S(\mathbf{x}, \mathbf{y})$.

Thus, because we can freely combine both conditions, in total there are

$$|A| = \binom{\gamma k}{\frac{\gamma}{2} k} \binom{(1 - \gamma)k}{\left(\delta - \frac{\gamma}{2}\right) k}$$

different values for $\mathbf{z}$, finishing the proof. ◀

Now we are ready to prove Theorem 3 about the time complexity of Algorithm 1 for solving the $\mathcal{CP}_{d,\lambda,\gamma}$ problem. For convenience, we restate the theorem here.

▶ **Theorem 3.** *Let $\gamma \in \left[0, \frac{1}{2}\right]$ and $\lambda \in [0, 1]$. Then Algorithm 1 solves the $\mathcal{CP}_{d,\lambda,\gamma}$ problem with overwhelming success probability in expected time $2^{\vartheta d(1+o(1))}$, where*

$$\vartheta = \begin{cases} (1 - \gamma)\left(1 - H\left(\frac{\delta^\star - \frac{\gamma}{2}}{1 - \gamma}\right)\right) & \text{for } \gamma \leq \gamma^\star \\ 2\lambda + H(\gamma) - 1 & \text{for } \gamma > \gamma^\star \ , \end{cases}$$

with $\delta^\star := H^{-1}(1 - \lambda)$ and $\gamma^\star := 2\delta^\star(1 - \delta^\star)$.

Proof. First let us give the exact form of $\log p$ and $\log q$ using Stirling's formula to approximate the binomial coefficients in Lemma 5 and 7. By setting the block width $k = \frac{d}{r}$ we get

$$\log q \ = (1 - \gamma)\left(H\left(\frac{\delta - \frac{\gamma}{2}}{1 - \gamma}\right) - 1\right)\frac{d}{r}(1 + o(1)), \quad \log p = \left(H(\delta) - 1\right)\frac{d}{r}(1 + o(1)) \ .$$

Now, let us reconsider the running time given in Theorem 4 as

$$T = \max\left(\underbrace{\frac{1}{q^r}}_{(a)}, \underbrace{\frac{2^{\lambda d} \cdot p^{r-1}}{q^r}}_{(b)}, \underbrace{\frac{\left(2^{\lambda d} \cdot p^r\right)^2}{q^r}}_{(c)}\right)^{1 + o(1)} \ ,$$

where $r = \frac{\lambda d}{\log^2 d}$.

We now show that the running time for all values of $\delta \geq \delta^\star := H^{-1}(1 - \lambda)$ is solely dominated by (c). Observe that we have $(c) \geq (b)$, whenever

$$2^{\lambda d} \cdot p^{2r} \geq p^{r-1}$$

$$\Leftrightarrow \quad H(\delta) \geq 1 - \frac{\lambda r}{r+1}$$

$$\Leftrightarrow \quad \delta \geq H^{-1}\Big(1 - \frac{\lambda}{1 + \frac{1}{r}}\Big) \rightarrow H^{-1}(1 - \lambda) = \delta^\star ,$$

since $\frac{1}{r} = o(1)$. Also we have $(c) \geq (a)$ for the same choice of delta, as

$$2^{2\lambda d} \cdot p^{2r} \geq 1$$

$$\Leftrightarrow \quad \delta \geq H^{-1}(1 - \lambda) = \delta^\star .$$

Thus, for all choices of $\delta \geq \delta^\star$ the running time is $(T_\delta)^{(1+o(1))}$ with

$$\vartheta^\star(\delta) := \frac{\log T_\delta}{d} = 2(\lambda + H(\delta) - 1) + (1 - \gamma)\Big(1 - H\Big(\frac{\delta - \frac{\gamma}{2}}{1 - \gamma}\Big)\Big) .$$

Now, minimizing $\vartheta^\star$ yields a global minimum at $\delta_{\min} = \frac{1}{2}(1 - \sqrt{1 - 2\gamma})$ attaining a value of

$$\vartheta^\star(\delta_{\min}) = 2\lambda + H(\gamma) - 1 .$$

As we are restricted to values for δ which are larger than $\delta^\star$ solving $\delta_{\min} \geq \delta^\star$ for γ yields

$$\delta_{\min} \geq \delta^\star$$

$$\Leftrightarrow \quad \gamma \geq 2\delta^\star(1 - \delta^\star) = \gamma^\star .$$

This proves the claim of the theorem whenever $\gamma > \gamma^\star$. For all other values of γ we simply choose $\delta = \delta^\star$, which yields

$$\vartheta = \vartheta^\star(\delta^\star) = (1 - \gamma)\Big(1 - H\Big(\frac{\delta^\star - \frac{\gamma}{2}}{1 - \gamma}\Big)\Big) \text{ for } \gamma \leq \gamma^\star$$

as claimed.

Now to boost the expected running time $2^{\vartheta d(1+o(1))}$ of the algorithm to actually being obtained with overwhelming probability we use a standard Markov argument. Let X denote the random variable describing the running time of the algorithm. Then the probability that the algorithm needs more time than $2^{\sqrt{d}}E[X]$ to finish is

$$\Pr\Big[X \geq 2^{\sqrt{d}} \cdot E[X]\Big] \leq \frac{E[X]}{2^{\sqrt{d}} \cdot E[X]} = 2^{-\sqrt{d}} ,$$

or equivalently the algorithm finishes in less time than $2^{\sqrt{d}}E[X] = 2^{\vartheta d(1+o(1))}$ with overwhelming probability. Also, a standard application of the union bound yields that the intersection of the algorithm finishing within the claimed time and the algorithm having success in finding the solution is still overwhelming. ◀

The theorem shows that whenever $\gamma > \gamma^*$ our algorithm obtains the optimal time complexity for uniformly random lists as given in Equation (1). Additionally, our algorithm reaches the time lower bound for locality-sensitive hashing based algorithms for all values of γ, whenever the input list sizes are subexponential in the dimension d, which is shown in the following lemma.

▶ **Lemma 8.** *Let $\gamma \in \left[0, \frac{1}{2}\right]$, and ϑ as defined in Theorem 3. Then we have*

$$\lim_{\lambda \to 0} \frac{\vartheta}{\lambda} = \frac{1}{1 - \gamma} \ .$$

Proof. Note that for λ converging zero, $\delta^\star = H^{-1}(1 - \lambda)$ approaches $\frac{1}{2}$. This implies $\gamma^\star := 2\delta^\star(1 - \delta^\star) = \frac{1}{2}$ and hence for all choices of γ we have

$$\vartheta = (1 - \gamma)\left(1 - H\left(\frac{\delta - \frac{\gamma}{2}}{1 - \gamma}\right)\right) \ .$$

Now, for this choice of ϑ, May and Ozerov [17, Corollary 1] already showed the statement of this lemma, by applying L'Hopital's rule twice. ◀

For convenience, we restate all parameter choices of Algorithm 1 for solving the $\mathcal{CP}_{d,\lambda,\gamma}$ in the following overview:

$$r = \frac{d}{\log^2 d}, \ P = (d + 1)^{r+1}, \ k = \frac{d}{r}$$

$$N = \frac{d}{q}, \ \text{where } q = \binom{\gamma k}{\frac{1}{2}\gamma k}\binom{(1 - \gamma)k}{\left(\delta - \frac{\gamma}{2}\right)k}\left(\frac{1}{2}\right)^k$$

$$\delta = \begin{cases} \delta^\star & \text{for } \gamma \leq 2\delta^\star(1 - \delta^\star) \\ \frac{1}{2}\left(1 - \sqrt{1 - 2\gamma}\right) & \text{else} \end{cases} \ , \ \text{with } \delta^\star := H^{-1}(1 - \lambda) \tag{8}$$

4 Different Input Distributions

In this section, we show how to adapt the analysis of Algorithm 1 to variable input distributions. Therefore, we first reformulate Theorem 4 in Corollary 9 for the case of considering the $\mathcal{CP}_{d,\lambda,\gamma}$ over an arbitrary distribution $\mathcal{D}$. As already indicated in the proof of Theorem 4, this reformulation depends on the expected value $\mathcal{E}$ of the cost of the naive search at the bottom of the computation tree, which is highly influenced by the distribution $\mathcal{D}$. Then, we show how to compute $\mathcal{E}$ and how to upper bound it effectively. Finally, we give upper bounds for the time complexity of the algorithm to solve the $\mathcal{CP}_{d,\lambda,\gamma}$ over some generic distributions. These examples suggest that the algorithm is best suited for distributions $\mathcal{D}$, where the weight of the sum $\mathbf{v} + \mathbf{w}$ of elements $\mathbf{v}, \mathbf{w} \sim \mathcal{D}$ concentrates at $\frac{d}{2}$.[4]

Let us start with the reformulation of the theorem.

▶ **Corollary 9.** *Let $\mathcal{D}$ be some distribution over $\mathbb{F}_2^d$, q and p be as defined in Equation (2), $\gamma \in \left[0, \frac{1}{2}\right]$, $\lambda \in [0, 1]$ and $r = \frac{\lambda d}{\log^2 d}$. Also let $\mathcal{E} = \mathbb{E}[|A| \cdot |B|]$ for A and B in line 13 of Algorithm 1 (where the expectation is taken over the distribution of input lists and the random choices of the algorithm). Then Algorithm 1 solves the $\mathcal{CP}_{d,\lambda,\gamma}$ problem over $\mathcal{D}$ in time*

$$\max\left(q^{-r}, \frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{\mathcal{E}}{q^r}\right)^{1+o(1)}$$

with success probability overwhelming in d.

---

[4] This behavior seems quite natural as in this case, the solution is most distinguishable from random input pairs.

Proof. The proof follows along the lines of the proof of Theorem 4, by observing that $T_{r+1} = N^r \cdot \mathcal{E}$ and the expected time complexity is again amplified to being obtained with overwhelming probability by using a Markov argument similar to the proof of Theorem 3. ◄

In the next lemma, we show how to upper bound the value of $\mathcal{E}$.

▶ **Lemma 10** (Expectation of Naive Search). *Let $\mathcal{D}$ be some distribution over $\mathbb{F}_2^d$, $\gamma \in \left[0, \frac{1}{2}\right]$, $\lambda \in [0,1]$ and $r = \frac{\lambda d}{\log^2 d}$. Also let $\mathcal{E} = \mathbb{E}[|A| \cdot |B|]$ for A and B in line 13 of Algorithm 1 when solving some instance of the $\mathcal{CP}_{d,\lambda,\gamma}$ over $\mathcal{D}$ (where the expectation is taken over the distribution of input lists and the random choices of the algorithm). Then we have*

$$\mathcal{E} \leq 2^{2\lambda d} \prod_{i=1}^{r} \alpha_i + 4r \cdot 2^{\lambda d} \cdot p^r$$

where $\alpha_i := \Pr_{\mathbf{v},\mathbf{w} \sim \mathcal{D}} \left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w} + \mathbf{z})_{B_{i,r}}) = \delta k \right].$

Proof. See Appendix A.1. ◄

While Lemma 10 gives an upper bound on the required expectation, it is not very handy. In the next lemma, we show how to further bound this expectation and how it affects the running time of the algorithm.

▶ **Lemma 11** (Complexity for Arbitrary Distributions). *Let $\mathcal{D}$ be some distribution over $\mathbb{F}_2^d$, $r := \frac{\lambda d}{\log^2 d}$, $\gamma \in \left[0, \frac{1}{2}\right]$ and $\lambda \in [0,1]$. Also let $\mathcal{E} = \mathbb{E}[|A| \cdot |B|]$ for A and B in line 13 of Algorithm 1 when solving some instance of the $\mathcal{CP}_{d,\lambda,\gamma}$ over $\mathcal{D}$ (where the expectation is taken over the distribution of input lists and the random choices of the algorithm). Then Algorithm 1 solves the $\mathcal{CP}_{d,\lambda,\gamma}$ over $\mathcal{D}$ in time*

$$\max\left(q^{-r}, \frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{2^{\varepsilon d}}{q^r} \right)^{1+o(1)} ,$$

where

$$\varepsilon = 2\lambda - \min_{\substack{i \in [r] \\ \eta \in [0,1]}} (1-\eta)\left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1-\eta}\right)\right) - \frac{r \cdot \log p_{i,\eta k}}{d}$$

with $p_{i,\eta k} := \Pr\left[\mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = \eta k\right].$

Proof. See Appendix A.2. ◄

Note that if it further holds that for $\mathbf{v} \sim \mathcal{D}$ each of the r blocks of $\mathbf{v}$ is identically distributed we can further simplify the term of ε from Lemma 11. In this case, we have $p_{i,\eta k}^r \leq \Pr\left[\mathrm{wt}(\mathbf{v} + \mathbf{w}) = \eta d\right] := p_{\eta d}$, thus we get

$$\varepsilon = 2\lambda - \min_{\eta \in [0,1]} (1-\eta)\left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1-\eta}\right)\right) - \frac{\log p_{\eta d}}{d} .$$

Now if we are given an arbitrary distribution $\mathcal{D}$ we can maximize ε according to η. Then we can similar to the proof of Theorem 3 derive a value for δ minimizing the overall time complexity.
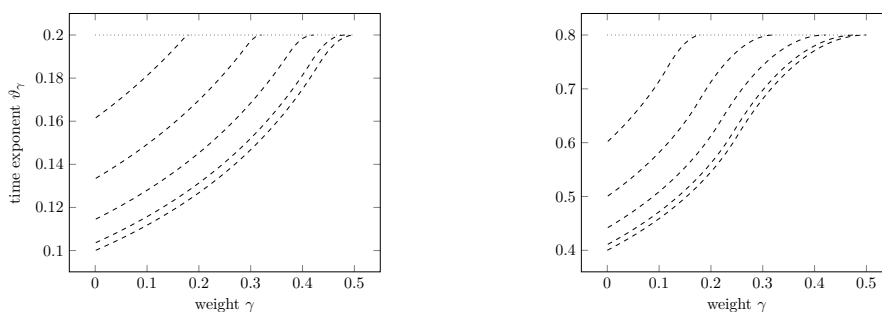
(a) List sizes $|L_1| = |L_2| = 2^{0.1d}$.

(b) List sizes $|L_1| = |L_2| = 2^{0.4d}$.

Figure 3 Time complexity exponents as a function of the weight of the closest pair for different input list distributions, where the expected weight of input elements is equal to $0.1d$, $0.2d$, $0.3d$, $0.4d$, $0.5d$ from left to right.

We performed this maximization and optimization numerically for some generic input distributions. We considered distributions, where the weight of input vectors is distributed *binomially*, chosen according to a *Poisson* distribution or *fixed* to a specific value. This means, first a weight is sampled according to the chosen distribution and then a vector of that weight is selected uniformly among all vectors of that weight.

The running time of Algorithm 1 for solving the $\mathcal{CP}_{d,\lambda,\gamma}$ over the considered distributions seems to be only dependent on the expected weight of vectors contained in the input lists. That means the time complexity for input lists containing random vectors whose weight is either fixed to ηd or binomially or Poisson distributed with expectation ηd is equal. This can possibly be explained by the low variance of all these distributions, which implies a high concentration around this expected weight.

We see in Figure 3, that the value for γ, from where on the complexity becomes quadratic in the lists sizes shifts to the left. This behavior stems from the fact, that the expected weight of a sum of elements is no longer $\frac{d}{2}$, but roughly $2\eta(1-\eta)d$. What also stands out is, that the complexity for $\gamma = 0$ is no longer linear in the lists sizes. The reason for this is that the probability of random pairs falling into the same bucket and the probability of the closest pair falling into the same bucket converge for decreasing weight of input list elements. This indicates that for input distributions with smaller expected weight a different bucketing criterion might be beneficial. We pose this as an open question for further research.

References

1   Josh Alman. An illuminating algorithm for the light bulb problem. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*, pages 2:1–2:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/OASIcs.SOSA.2019.2`.

2   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.

3   Yoshinori Aono, Phong Q Nguyen, Takenobu Seito, and Junji Shikata. Lower bounds on lattice enumeration with extreme pruning. In *Annual International Cryptology Conference*, pages 608–637. Springer, 2018.

4   Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24, Arlington, VA, USA, January 10–12 2016. ACM-SIAM. `doi:10.1137/1.9781611974331.ch2`.

5 Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.

6 Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 25–46, Fort Lauderdale, Florida, United States, April 9–11 2018. Springer, Heidelberg, Germany. `doi:10.1007/978-3-319-79063-3_2`.

7 Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for lpn security. In *International Conference on Post-Quantum Cryptography*, pages 25–46. Springer, 2018.

8 Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

9 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.

10 Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Shoichi Hirose. Generalization of bjmm-isd using may-ozerov nearest neighbor algorithm over an arbitrary finite field $\mathbb{F}_q$. In *International Conference on Codes, Cryptology, and Information Security*, pages 96–109. Springer, 2017.

11 Shoichi Hirose. May-ozerov algorithm for nearest-neighbor problem over $\mathbb{F}_q$ and its application to information set decoding. In *International Conference for Information Technology and Communications*, pages 115–126. Springer, 2016.

12 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, Dallas, TX, USA, May 23–26, 1998. ACM Press. `doi:10.1145/276698.276876`.

13 Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1288–1305, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM. `doi:10.1137/1.9781611974331.ch90`.

14 Samir Khuller and Yossi Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, 1995.

15 Jiwen Lu, Venice Erin Liong, Xiuzhuang Zhou, and Jie Zhou. Learning compact binary face descriptor for face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2041–2056, 2015.

16 Jonathan Marchini, Peter Donnelly, and Lon R Cardon. Genome-wide strategies for detecting multiple loci that influence complex diseases. *Nature genetics*, 37(4):413–417, 2005.

17 Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. `doi:10.1007/978-3-662-46800-5_9`.

18 Rajeev Motwani, Assaf Naor, and Rina Panigrahi. Lower bounds on locality sensitive hashing. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 154–157, 2006.

19 Solomon K Musani, Daniel Shriner, Nianjun Liu, Rui Feng, Christopher S Coffey, Nengjun Yi, Hemant K Tiwari, and David B Allison. Detection of gene× gene interactions in genome-wide association studies of human population data. *Human heredity*, 63(2):67–84, 2007.

20 Christoph Strecha, Alex Bronstein, Michael Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 34(1):66–78, 2011.

21 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.

22 Leslie G Valiant. Functionality in neural nets. In *COLT*, volume 88, pages 28–39, 1988.

23 Ning Xie, Shuai Xu, and Yekun Xu. A new coding-based algorithm for finding closest pair of vectors. *Theoretical Computer Science*, 782:129–144, 2019.

A   Proofs Regarding Different Input Distributions

In this section, we give the proofs for the lemmata regarding the performance and correctness of our algorithm applied to different input distributions that were omitted in the main body.

A.1   Proof of Lemma 10

Similar to the proof of Theorem 4, let us bound $\mathbb{E}[|A_i| \cdot |B_i|]$ in terms of $\mathbb{E}[|A_{i-1}| \cdot |B_{i-1}|]$, $\mathbb{E}[|A_{i-1}|]$ and $\mathbb{E}[|B_{i-1}|]$ for each i.

$$
\begin{aligned}
\mathbb{E}[|A_i| \cdot |B_i| \mid A_{i-1}, B_{i-1}] = &\sum_{\substack{\mathbf{v} \in A_{i-1} \backslash \{\mathbf{x}\} \\ \mathbf{w} \in B_{i-1} \backslash \{\mathbf{y}\}}} \underbrace{\Pr\left[\mathrm{wt}((\mathbf{v}+\mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w}+\mathbf{z})_{B_{i,r}}) = \delta k\right]}_{=:\alpha_i} \\
&+ \sum_{\mathbf{v} \in A_{i-1}} \underbrace{\Pr\left[\mathrm{wt}((\mathbf{v}+\mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{y}+\mathbf{z})_{B_{i,r}}) = \delta k\right]}_{\leq p} \\
&+ \sum_{\mathbf{w} \in B_{i-1}} \underbrace{\Pr\left[\mathrm{wt}((\mathbf{x}+\mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w}+\mathbf{z})_{B_{i,r}}) = \delta k\right]}_{\leq p} \\
&+ \Pr\left[\mathrm{wt}((\mathbf{x}+\mathbf{z})_{B_{i,r}})) = \delta k, \mathrm{wt}((\mathbf{y}+\mathbf{z})_{B_{i,r}})) = \delta k\right] \\
\leq\ & \alpha_i \cdot |A_{i-1}| \cdot |B_{i-1}| + p \cdot (|A_{i-1}| + |B_{i-1}| + 1)
\end{aligned}
$$

and hence $\mathbb{E}[|A_i| \cdot |B_i|] \leq \alpha_i \cdot \mathbb{E}[|A_{i-1}| \cdot |B_{i-1}|] + p \cdot (\mathbb{E}[|A_{i-1}|] + \mathbb{E}[|B_{i-1}|] + 1)$. Again, applying this equation successively, we obtain

$$
\mathcal{E} = \mathbb{E}[|A_r| \cdot |B_r|] \leq 2^{2\lambda d} \prod_{i=1}^{r} \alpha_i + 4 \cdot 2^{\lambda d} \cdot \sum_{i=1}^{r} \left(\prod_{j=0}^{i-2} \alpha_{r-j}\right) p^{r-i+1} \leq 2^{2\lambda d} \prod_{i=1}^{r} \alpha_i + 4r \cdot 2^{\lambda d} \cdot p^r \ . \quad \blacktriangleleft
$$

A.2   Proof of Lemma 11

Taking the result for $\mathcal{E}$ from Lemma 10 and plugging into the run time formula from Corollary 9 we get that the $\mathcal{CP}_{d,\lambda,\gamma}$ problem over $\mathcal{D}$ can be solved with probability overwhelming in d in time

$$
\max\left(q^{-r}, \frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{\mathcal{E}}{q^r}\right)^{1+o(1)} \leq \max\left(q^{-r}, \frac{2^{\lambda d} \cdot p^{r-1}}{q^r}, \frac{2^{2\lambda d} \prod_{i=1}^{r} \alpha_i}{q^r}\right)^{1+o(1)}
$$

since the right summand $\frac{4r \cdot 2^{\lambda d} \cdot p^r}{q^r}$ of $\frac{\mathcal{E}}{q^r}$ is asymptotically smaller than the second entry in the max, i.e. $\frac{2^{\lambda d} \cdot p^{r-1}}{q^r}$. Thus, is suffices to find an easier upper bound for the first summand $S := 2^{2\lambda d} \prod_{i=1}^{r} \alpha_i$. Remembering $\alpha_i = \Pr\left[\mathrm{wt}((\mathbf{v}+\mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w}+\mathbf{z})_{B_{i,r}}) = \delta k\right]$ we receive

$$S \leq 2^{2\lambda d} \cdot \left(\max_{i \in [r]} \alpha_i \right)^r$$

$$= 2^{2\lambda d} \cdot \left(\max_{i \in [r]} \sum_{j=0}^{k} q_{i,j} \cdot \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = j \right] \right)^r$$

$$\leq 2^{2\lambda d + o(d)} \cdot \left(\max_{i \in [r],\, j \in [k] \cup \{0\}} q_{i,j} \cdot \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = j \right] \right)^r$$

$$= 2^{2\lambda d + o(d)} \cdot \left(\max_{i \in [r],\, \eta \in [0,1]} q_{i,\eta k} \cdot \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = \eta k \right] \right)^r \ ,$$

where $q_{i,\eta k} = \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{z})_{B_{i,r}}) = \delta k, \mathrm{wt}((\mathbf{w} + \mathbf{z})_{B_{i,r}}) = \delta k \mid \mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = \eta k \right]$. Lemma 7 lets us rewrite this probability as

$$q_{i,\eta k} = \binom{\eta k}{\frac{1}{2}\eta k} \binom{(1-\eta)k}{\left(\delta - \frac{\eta}{2}\right)k} \left(\frac{1}{2} \right)^k \leq 2^{-\left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1 - \eta} \right)\right)(1-\eta)k} \ .$$

We end up with

$$S \leq 2^{2\lambda d + r \cdot \max\limits_{i \in [r],\, \eta \in [0,1]} -\left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1-\eta} \right)\right)(1-\eta)k + \log p_{i,\eta k} + o(d)}$$

$$= 2^{\left(2\lambda + \max\limits_{i \in [r],\, \eta \in [0,1]} -\left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1-\eta} \right)\right)(1-\eta) + \frac{r}{d} \cdot \log p_{i,\eta k} \right) d + o(d)}$$

$$= 2^{\left(2\lambda - \min\limits_{i \in [r],\, \eta \in [0,1]} \left(1 - H\left(\frac{\delta - \frac{\eta}{2}}{1-\eta} \right)\right)(1-\eta) - \frac{r}{d} \cdot \log p_{i,\eta k} \right) d + o(d)}$$

with $p_{i,\eta k} := \Pr \left[\mathrm{wt}((\mathbf{v} + \mathbf{w})_{B_{i,r}}) = \eta k \right]$, which proves the claim. ◀

B Practical Experiments

In this section, we give experimental results of the performance of a proof of concept implementation of our new algorithm. These experiments verify the performance gain of our algorithm over a naive quadratic search approach. We also verify the numerical estimates of the algorithm's performance on different input distributions from the previous section and give some practical related improvements to our algorithm. Our implementation is publicly available at `https://github.com/FloydZ/NNAlgorithm`.

Before discussing the benchmark results let us first briefly describe some of the practical improvements we introduced in our implementation, which differ from the description in Section 3. We implemented a true depth-first search rather than the iterative description given previously. The iterative description just allowed for a more convenient analysis. Thus, our algorithm needs to store only the lists of a single path from the root to a leaf node at any time. Also, as all lists of subsequent levels are subsets of previous ones, we do not create r different lists. We rather rearrange the elements of the input list such that elements belonging to the list of the subsequent level are consecutive, making it sufficient to just memorize the range of elements that belong to the next level list. This way, we only need to store the input list plus two integer markers for each level. Also, it turns out that in practice often a small depth of the tree (not exceeding 8 in our experiments) is already sufficient to achieve good runtime results. Regarding the branching factor N of the tree, we achieve
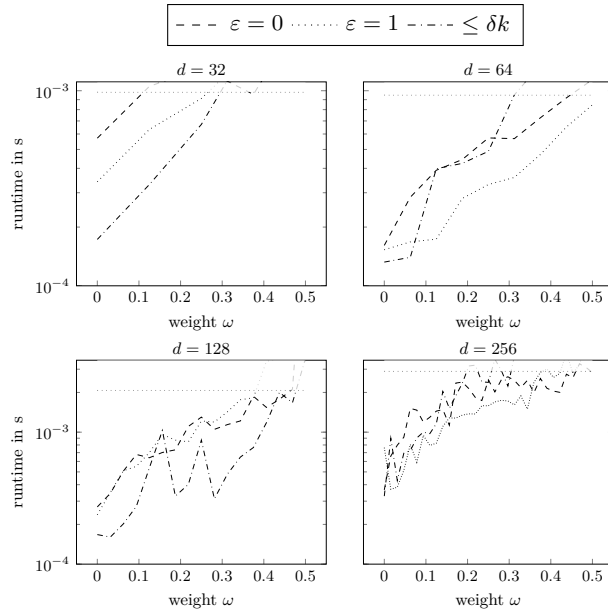
Figure 4 Runtime in seconds in logarithmic scale (y-axis) as a function of distance ω of the closest pair (x-axis) on random lists of size 2^{10}. Dotted, dashed and dash-dotted lines indicate results for different bucketing, straight horizontal line is the time used by a naive quadratic search.

optimal results either for values close to its expectation $\frac{1}{q}$ as given by the analysis or values being significantly smaller. The case of using a very small branching factor can be seen as a pruning strategy, similar to the one used in lattice enumeration algorithms for shortest vector search [3]. Additionally, we benchmarked three different strategies for the weight criteria:

1. Strictly enforcing a weight of δk in each block, as described in our algorithm.
2. Allowing for a small deviation $\pm\varepsilon$ around δk.
3. Allowing for weights of *at most δk*.

Further, we introduced a threshold for the size of the lists in the tree, below which the computation of further leaves is aborted and naive search is used instead.

Figure 4 shows the runtime results for the different bucket criteria on small input lists of size 2^{10} containing random elements. Here, each data point was averaged over 50 measurements. The experimental results clearly indicate a significant gain over the quadratic search approach. The less significant gain for small dimension d is due to the reduced amount of possible blocks or equivalently the low depth of the computation tree, which lets the algorithm not reach its full potential. In the case of small input lists, we observe that a bucketing strategy that allows a deviation of $\varepsilon = 1$ from δk is beneficial for most values of d.

Figure 5 shows the same experiments performed on larger input lists of size 2^{15}. Besides a more significant improvement over the naive search, we can observe that the bucketing criterion that uses δk as an upper bound becomes more beneficial for nearly all values of γ and d.

Eventually, Figure 6 shows the experimental runtime results on input lists, whose elements are drawn from a different input distribution, analyzed in Section 4. Here the distribution is the uniformly random distribution over vectors of weight ηd. One can observe that for growing d the shape of the graph resembles the theoretical results from Figure 3.
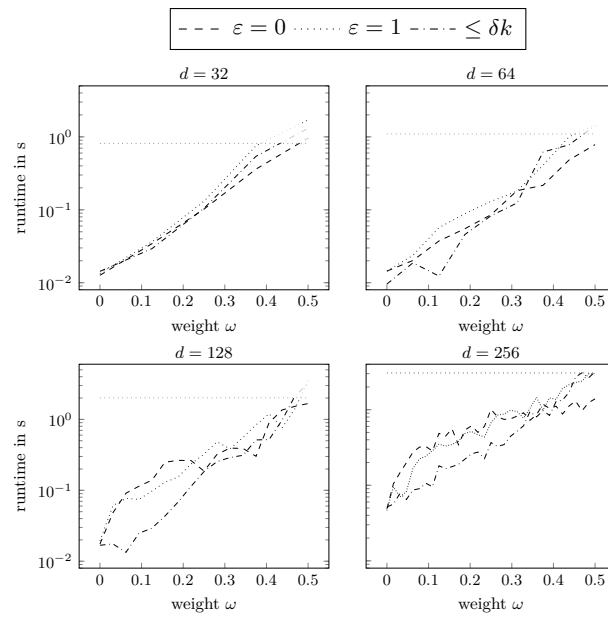
Figure 5 Runtime in seconds in logarithmic scale (y-axis) as a function of distance ω of the closest pair (x-axis) on random lists of size 2^{15}. Dotted, dashed and dash-dotted lines indicate results for different bucketing strategies, straight horizontal line is time used by a naive quadratic search.
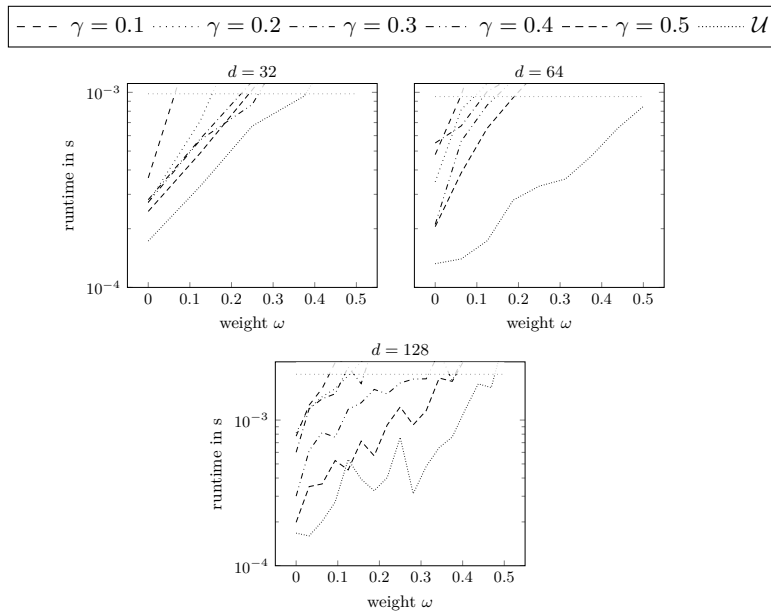
Figure 6 Runtime in seconds in logarithmic scale (y-axis) as a function of distance ω of the closest pair (x-axis) on lists of size 2^{10} containing random elements of weight γd. The densely dashed line ($\mathcal{U}$) indicates the runtime on uniformly random lists.