









The May-Ozerov Algorithm for Syndrome Decoding is “Galactic”

Charles Bouillaguet¹  , Claire Delaplace²   and
Mickaël Hamdad^{1,2}  

¹ Sorbonne Université, CNRS, LIP6, Paris, France

² Laboratoire MIS, Université de Picardie Jules Verne, Amiens, France

Abstract. In 2015, May and Ozerov proposed a new method to solve the *Nearest Neighbor* Problem. They also observed that it could be used as a subroutine in various Information Set Decoding (ISD) algorithms for arbitrary linear codes. This led to an asymptotic improvement in their complexity. However, the proposed improvement has been widely perceived as impractical because of the huge hidden factors in its asymptotic complexity. The main contribution of this article is to provide a sound foundation for this claim. We show that it is indeed “galactic”, namely that it only improves upon much simpler methods when instances are so large that they fill the whole universe.

More precisely, we argue that for the May-Ozerov ISD algorithm to require less operations than a technique based on the Stern ISD algorithm, the length of the code has to be greater than 1, 874, 400, with a number of operation beyond 2^{63489} , making it practically useless.

Keywords: Code-based Cryptography · Decoding algorithm · Concrete security

1 Introduction

A “galactic” algorithm is

an algorithm that is wonderful in its asymptotic behavior, but is never used to actually compute anything. [LR13]

There are many well-known examples, the most famous one probably being the notorious square matrix multiplication algorithm of Coppersmith and Winograd from 1990 [CW90], with complexity $\mathcal{O}(n^{2.3755})$, where n is the size of the matrices.

The analysis of algorithms is often asymptotic, relying on “big O” notations. These may hide potentially huge constants. In addition, they mean that the complexity is below the announced value “when the instance is large enough”, but the actual threshold is usually not known or stated. All of this is well-known.

Cryptanalysts sometimes come up with galactic algorithms. The case of the (binary) syndrome decoding problem is interesting from this point of view. The computational hardness of this problem underlies the security of several code-based cryptographic constructions, including notably the McEliece cryptosystem [McE78]. The problem consists in finding sparse solutions to under-constrained linear systems over \mathbb{F}_2 . More precisely, given an $(n - k)$ -by- n boolean matrix H , a vector s and an integer w , find a vector e such that $He = s$ and the Hamming weight of e is less than w .

E-mail: charles.bouillaguet@lip6.fr (Charles Bouillaguet), claire.delaplace@u-picardie.fr (Claire Delaplace), mickael.hamdad@lip6.fr (Mickaël Hamdad)



Table 1: Some algorithms for the Syndrome Decoding problem. It is assumed that the number of errors to decode is less than half the minimum distance of the code (“maximum likelihood decoding”). The algorithms run in time $\tilde{O}(2^{\tau n})$ using space $\tilde{O}(2^{\mu n})$.

Algorithm	τ	μ
Lee-Brickell [LB88]	0.0575	0
Stern [Ste88]	0.0556	0.0135
Ball-collision [BLP11]	0.0556	0.0148
MMT [MMT11]	0.0536	0.0216
BJMM [BJMM12]	0.0493	0.0331
May-Ozerov [MO15]	0.0473	0.0346

The problem is well-known to be NP-complete [BMVT78]. Over the years, several algorithms have been developed to solve it. Table 1 summarizes some of them. All of them have exponential complexity, and much effort has been devoted to minimize the exponent in the worst case. Their complexity is usually given using the “big O tilde” notation that hides unknown constants and polynomial factors.

To the best of our knowledge, the last algorithm, with the best exponent, has not been used to solve any serious instance of the problem. On the contrary, the Stern, MMT and BJMM algorithms have been used to obtain computational records in [BLP08], [EMZ22] and [NUO⁺24] respectively. Table 1 shows that the reduction in the complexity exponent comes at the expense of an exponential increase in memory consumption. Whereas the Stern and MMT algorithm have $M \leq \sqrt{T}$, this is no longer the case for the most recent algorithms — this is a serious hint that they are impractical.

Galactic algorithms are clearly impractical, but an algorithm can be impractical for reasons other than its sheer number of operations. For instance, breaking the double-DES with a simple meet-in-the-middle attack requires about 2^{57} DES evaluations and enough memory to store 2^{56} 64-bit blocks. The number of operations is not a problem (exhaustive search on the DES has been done in practice), however the memory requirement is way above the capacity of largest computing centers at the time of this writing (they have roughly 2^{55} bits of RAM). This meet-in-the-middle attack is thus quite impractical.

In this article, we revisit the complexity analysis of the algorithm (DECODE) presented by May and Ozerov in 2015 to solve the syndrome decoding problem [MO15]. We focus on its number of operations and disregard any other practical consideration. We show that the algorithm is “galactic”.

It relies on a clever procedure that solves the (*Hamming distance*) *Nearest Neighbor* problem (also called the *bichromatic closest pair* or the *light bulb* problem): given two lists of bit vectors $L, R \subseteq \mathbb{F}_2^m$ of size $N = 2^{\lambda m}$, find $x \in L$ and $y \in R$ such that the Hamming distance between x and y is less than γm (if it exists).

Related work The most common subquadratic algorithm to solve the Nearest Neighbor Problem is based on what is now called locality-sensitive hashing. It is sometimes attributed to an article of 1998 by Indyk and Motwani [IM98] (cf. for instance [EB22]). We believe that the actual algorithm goes way back. At the very least, it is discussed in a 1995 article by Karp, Waarts and Zweig [KWZ95]; its main idea is contained in the ISD algorithm of Stern from 1988 [Ste88]; a article by Rivest from 1974 [Riv74] attributes it to Peter Elias, following a technical report from 1971 by Welch [Wel71]. In this article, we call it the “Projection method” (because the locality-sensitive hash function that we use is just a projection).

In [MO15], May and Ozerov proposed an improved procedure to solve the nearest neighbor problem. We refer to it as the MO-NN algorithm here. In [MO15] the authors

prove the following

Theorem (Theorem 1 in [MO15]). *Let H be the binary entropy function. For any constant $\epsilon > 0$, any (γ, λ) such that $0 < \gamma < \frac{1}{2}$ and $0 < \lambda < 1 - H\left(\frac{\gamma}{2}\right)$, the MO-NN algorithm solves the Nearest Neighbor problem with overwhelming probability in time $\tilde{O}\left(2^{(y(\gamma, \lambda) + \epsilon)m}\right)$, with*

$$y(\gamma, \lambda) = (1 - \gamma) \left(1 - H\left(\frac{H^{-1}(1 - \lambda) - \frac{\gamma}{2}}{1 - \gamma}\right) \right).$$

We note that the “big O tilde” is not strictly necessary and that a simple “big O” would have been sufficient: an arbitrarily small increase of ϵ is sufficient to absorb any hidden polynomial factor.

The algorithm is asymptotically better than Projection Method for solving the problem when $m = c(N) \log(N)$ where $c(N) \geq \frac{1}{1 - H(\frac{\gamma}{2})}$. The exponent y in the complexity is close to optimal [Dub10].

However, the algorithm has been suspected of being galactic for a while. Already in [MO15], the authors state that due to hidden factor, their algorithm is not practical. In [EB22], Esser and Bellini provide an estimator to evaluate the hardness of concrete instances of the syndrome decoding problem. They explicitly disregard the MO-NN algorithm, arguing that

“While the algorithm achieves theoretically close to optimal complexities, it inherits a huge polynomial overhead limiting its practicality, despite recent efforts to reduce that overhead. As one of the major goals of this work is to provide precise practical estimates we do not consider the algorithm by May-Ozerov.” [EB22]

Indeed, there are several reasons why this algorithm could be galactic:

- The success probability could be very small until n becomes very large.
- The polynomial factor hidden in the “big O tilde” could be very large, so that the small exponent only brings a benefit for extremely large values of n .
- The smaller the value of ϵ , the larger the hidden polynomial factor becomes. In addition, the threshold at which the actual complexity is less than the function inside the “big O” may shift towards $+\infty$.

Our main contribution is to show that the MO-NN algorithm is indeed “galactic”, and compare its complexity to the aforementioned Projection Method. Our new refined complexity analysis allows us to give an estimation of the threshold value of n for which the MO-NN algorithm outperform the Projection Method. For instance, if $\lambda = 0.025$ and $\gamma = 0.1$, this threshold is beyond $m = 336,017$ using $t = 20$ (hence $\epsilon = 0.000755\dots$). The size of the input lists at the crossover point is then 2^{8400} . The actual number of operations is at least 2^{9697} . This is a relevant setting, as argued later.

We then consider what happens when the algorithm is used for syndrome decoding, in particular to attack the McEliece cryptosystem. This yields instances of the Nearest Neighbor problem where the May-Ozerov ISD algorithm with the MO-NN algorithm is even more galactic. For example, for $R = 0.8$ and $D = 0.03$, the value of n for which the May-Ozerov ISD algorithm (DECODE) with the MO-NN algorithm is better than with the Projection Method is beyond $n = 1,874,400$. This implies a list size greater than 2^{22282} in input of the MO-NN algorithm. The estimator code used to produce these numbers is available as Supplementary Material.

Organization of this article. In Section 2, we present the problem and important notations we use in the rest of the paper. We describe the Projection Method and the MO-NN algorithm in Section 3. In Section 4, we present our main results, namely a lower bound on the number of operations performed by the the MO-NN algorithm. This allows us to find a “crossing point” with the folklore Projection Method, that is a value m_0 for which we know for sure that, for all $m < m_0$, the Projection Method will perform better than the MO-NN algorithm. We give numerical estimation of this m_0 value (depending on various parameters) in Section 5, and show that this value is indeed “galactic”. Finally, in Section 6, we present a concrete application. Given the May-Ozerov ISD algorithm (DECODE) we estimate the minimum length n a linear code must have for the MO-NN algorithm to outperform the Projection Method. For a random binary linear code of rate $1/2$, n must be larger than 512,000.

2 Preliminaries

Useful notations. An element u of \mathbb{F}_2^m is an m -bit vector. We denote by $wt(u)$ the Hamming weight of u . $d(u, v)$ represents the Hamming distance between u and v (the number of bits that differ in the two vectors).

For $x > 0$ we denote by $\log_2(x)$ the logarithm in base 2 of x . For $0 < x < 1$, H represents the binary entropy function:

$$H(x) = -x \log_2(x) - (1-x) \log_2(1-x).$$

We also use the following bounds:

$$\sqrt{\frac{n}{8k(n-k)}} 2^{nH(\frac{k}{n})} \leq \binom{n}{k} \leq \sqrt{\frac{n}{2\pi k(n-k)}} 2^{nH(\frac{k}{n})}, \quad (1)$$

$$H(x) \geq 4x(1-x). \quad (2)$$

and for all x and for all $i \in \{0, \dots, x\}$

$$\binom{i}{\frac{i}{2}} \binom{x-i}{\frac{x-i}{2}} \leq \binom{x}{\frac{x}{2}} \quad (3)$$

(1) can be obtained from the Stirling formula. A proof of (2) can be found in [Top01]. (3) is established by a combinatorial argument: the right-hand side of the inequality is the number of ways to choose half of the elements of the set $\{1, \dots, x\}$. The left hand size is the number of ways to choose half of the elements of the same set but with a restriction — $\frac{i}{2}$ elements in the subset $\{1, 2, \dots, i\}$. It is therefore smaller.

2.1 The nearest neighbor problem

Definition 1 (*(m, γ, λ) -Nearest Neighbor Problem*). Let $m \in \mathbb{N}$, $0 < \gamma < \frac{1}{2}$ and $0 < \lambda < 1$. Given γ and two lists $L, R \subset \mathbb{F}_2^m$ of equal size $N = 2^{\lambda m}$ with uniform and independent vectors, the (m, γ, λ) -Nearest Neighbor Problem consists in exhibiting a pair (u^*, v^*) from $L \times R$, such that $d(u^*, v^*) \leq \gamma m$, or returning \perp if none exists.

In this paper, we refer to all pairs $(u, v) \in L \times R$ such that $d(u, v) \leq \gamma m$ as *good pairs*. We say that an algorithm succeeds in solving the *Nearest Neighbor Problem* if it returns a good pair whenever the input lists contain any. We work under the assumption that the input lists contain a *single* good pair (u^*, v^*) ; the algorithms succeed when they return this specific target pair.

Computational model. We consider m -bit words to fit our problem instance. All basic instructions such as reading, writing, comparing, as well as usual binary operations (e.g. exclusive-OR, AND, negation) and arithmetic ones (e.g. sum, multiplication) on m -bit words are assumed to be “elementary operations”. We measure the time complexity by counting the number of such operations.

3 Algorithms for the nearest neighbor problem

A naive “brute-force” technique solves the *Nearest Neighbor Problem* by simply computing the Hamming distance of each $(u, v) \in L \times R$ in quadratic time $2^{2\lambda m}$. This yields the QUADRATICNN function below.

Algorithm 1 QUADRATICNN

```

1: function QUADRATICNN( $L, R, \ell = \gamma m$ )
2:   for all  $(u, v) \in L \times R$  do
3:     if  $d(u, v) \leq \ell$  then
4:       return  $(u, v)$ 
5:   return  $\perp$ 

```

3.1 The projection method

Let u_J be the $|J|$ -bit vector formed by the coefficients of u whose indices are in J . The basic idea consists in guessing a subset J of the indices and making the assumption that the target solution (u^*, v^*) is such that $u_J^* = v_J^*$.

This procedure heavily relies upon the PARTITION routine. Given a list L of m -bit vectors and a subset J of $\{1, \dots, m\}$ such that $|J| = k$, partition L into 2^k sublists: all vectors u inside each sublist coincide on u_J .

This can clearly be implemented in linear time, for instance using a bucket sort that uses $3N$ operations, where N is the size of the input list. Put another way, the vectors are bucketized using a locality-sensitive hash function that returns the k bits designated by J .

Algorithm 2 THE PROJECTION METHOD

```

1: function PROJECTIONMTD( $\ell, N, L, R, k$ )  $\triangleright N = |L| = |R|, \ell = \gamma m$ 
2:    $p \leftarrow \binom{m}{k} / \binom{m-\ell}{k}$   $\triangleright$  Success probability of a single iteration
3:   for  $1/p$  times do
4:      $J \leftarrow$  random set of  $k$  distinct integers from  $\{1, \dots, m\}$ 
5:      $L_1, \dots, L_{2^k} \leftarrow$  PARTITION( $L, J$ )
6:      $R_1, \dots, R_{2^k} \leftarrow$  PARTITION( $R, J$ )
7:     for  $1 \leq i \leq 2^k$  do
8:        $x \leftarrow$  QUADRATICNN( $L_i, R_i, \ell$ )
9:       if  $x \neq \perp$  then return  $x$ 
10:  return  $\perp$ 

```

3.2 May-Ozerov’s algorithm

We present an overview of the the MO-NN in Algorithm 3. This procedure takes as input the two lists of m -bit vectors L and R , of size $|L| = |R| = 2^{\lambda m}$, the target value γ , as well as parameters t and ϵ , discussed later. The core idea of this algorithm is to partition each list into t strips of indices, and solve the problem recursively on each of these strips.

The j -th strip is made up of indices whose index belongs to the interval I_j such that $I_1 = [1 : \alpha_1 m]$ and for all $j \geq 2$, $I_j = [(\sum_{x=1}^{j-1} \alpha_x m) + 1 : \sum_{x=1}^j \alpha_x m]$. We denote by u_{I_j} the $|I_j|$ -bit vector formed by the coefficients of u whose indices are in I_j . For the sake of simplicity, we will denote by u_j (resp. v_j) the vector u_{I_j} (resp. v_{I_j}).

In order to make the MO-NN algorithm work, May and Ozerov first rerandomizes the input so that the targeted solution (u^*, v^*) satisfies the following conditions with some probability. For all $1 \leq j \leq t$:

$$i) \quad d(u_j^*, v_j^*) = \gamma |I_j| = \gamma \alpha_j m;$$

$$ii) \quad wt(u_j^*) = wt(v_j^*) = \frac{\alpha_j m}{2}.$$

The whole procedure is then repeated sufficiently many times in order to guarantee that the conditions are satisfied in at least one iteration of this ‘‘double rerandomization’’ with overwhelming probability (lines 6 and 7 of Algorithm 3). The first loop randomly permutes the indices while the second one XORs a random bit vector r of well chosen Hamming weight onto each element of the lists. As shown in [MO15], this ensures that if a solution (u^*, v^*) exists, it will be found with overwhelming probability after restarting the procedure a polynomial number of times (in m). In section 4.2, we revisit their proof to present more precise bounds; in particular we give lower-bounds on $f_1(m)$ and $f_2(m)$.

Algorithm 3 The MO-NN algorithm

```

1: function MO-NN( $L, R, \lambda, \gamma, m, t, \epsilon$ )
2:    $y(\gamma, \lambda) \leftarrow (1 - \gamma) \left( 1 - H \left( \frac{H^{-1}(1-\lambda) - \gamma/2}{1-\gamma} \right) \right)$ 
3:    $\alpha_1 \leftarrow (y(\gamma, \lambda) - \lambda + \epsilon/2) / y(\gamma, \lambda)$ 
4:   for  $2 \leq j \leq t$  do
5:      $\alpha_j \leftarrow \frac{\lambda}{y(\gamma, \lambda)} \alpha_{j-1}$  ▷ Divide the lists into  $t$  strips of  $\alpha_j m$  indices
6:   for  $f_1(m)$  uniformly random permutation  $\pi$  of  $\{1, \dots, m\}$  do
7:     for  $f_2(m)$  times do
8:        $r = (r_1, \dots, r_t) \leftarrow (\text{RANDOM}(\mathbb{F}_2^{\alpha_j m}))_{j=1}^t$  s.t.  $wt(r_j) = \alpha_j \frac{m}{2}$ 
9:        $\bar{L} \leftarrow \pi(L) + r$ 
10:       $\bar{R} \leftarrow \pi(R) + r$ 
11:      Remove from  $\bar{L}$  and  $\bar{R}$  all vectors that are not of weight  $\alpha_j \frac{m}{2}$  on the  $j$ -th strip
12:       $C \leftarrow \text{RECURSIVEMO}(\bar{L}, \bar{R}, m, t, \epsilon, \gamma, \lambda, (\alpha)_{j=1}^t, 1)$ 
13:      if  $C \neq \perp$  then return  $C$ 
14:   return  $\perp$ 

```

The main idea of the MO-NN algorithm lies in the RECURSIVEMO procedure invoked on line 12. It takes as input the two ‘‘rerandomized’’ list \bar{L} and \bar{R} , the parameters t, ϵ and γ , as well as $(\alpha_1, \dots, \alpha_t)$ that give the size of each strip, and an index j (initialized to the value 1) corresponding to the index of the current strip. As it is assumed that all vectors of L and R are of Hamming weight $\alpha_j m/2$ on each strip I_j , the list are filtered beforehand to remove all vectors that do not satisfy this condition.

The recursive procedure is described in Algorithm 4. Its main idea is to compute exponentially many sublists of L and R of small size and try to solve the problem recursively. This creates a recursion tree, where each node represents a pair of sublists (L', R') of (L, R) . The root of the tree corresponds to original (filtered) pair (\bar{L}, \bar{R}) . May and Ozerov show that, at a given step j , if each node has ms_j children, then the procedure succeeds in finding (u^*, v^*) with overwhelming probability, assuming that $(u^*, v^*) \in (\bar{L}, \bar{R})$.

At depth one, we consider I_1 , the first of the t strips. Each node is built as follows. A subset A of I_1 , such that $|A| = |I_1|/2 = \alpha_1 m/2$ is picked uniformly at random. Then, the

Algorithm 4 RECURSIVEMO

```

1: function RECURSIVEMO( $L, R, m, t, \epsilon, \lambda, \gamma, (\alpha)_1^t, j$ )
2:   if  $j = t + 1$  then
3:     return QUADRATICNN( $L, R, \gamma m$ )
4:    $C \leftarrow \perp$ 
5:    $s_j \leftarrow \frac{\binom{\alpha_j m}{\frac{1}{2}\alpha_j m}}{\binom{(1-\gamma)\frac{\alpha_j m}{2}}{(1-h-\frac{\gamma}{2})\frac{\alpha_j m}{2}} \binom{(1-\gamma)\frac{\alpha_j m}{2}}{(h-\frac{\gamma}{2})\frac{\alpha_j m}{2}} \left(\frac{\gamma\frac{\alpha_j m}{2}}{\frac{\gamma}{2}\frac{\alpha_j m}{2}}\right)^2}$ 
6:   for  $ms_j$  times do
7:     Pick a random subset  $A$  of  $\frac{\alpha_j m}{2}$  indices inside strip  $j$ 
8:      $L' \leftarrow \{u \in L \text{ s.t. } wt(u_A) = H^{-1}(1-\lambda)\frac{\alpha_j m}{2}\}$ 
9:      $R' \leftarrow \{v \in R \text{ s.t. } wt(v_A) = H^{-1}(1-\lambda)\frac{\alpha_j m}{2}\}$ 
10:    if  $|L'|$  and  $|R'|$  are not too big then
11:       $x \leftarrow$  RECURSIVEMO( $L', R', m, t, \epsilon, \lambda, \gamma, (\alpha)_1^t, j + 1$ )
12:      if  $x \neq \perp$  then
13:        return  $x$ 
14:  return  $C$ 

```

lists L and R are filtered according to the following criteria: only the vectors of Hamming weight equal to $H^{-1}(1-\lambda)\frac{\alpha_j m}{2}$ on subset A are kept. If the resulting two lists L' and R' are not too big the node is kept, otherwise it is discarded (the precise meaning of “not too big” is discussed below).

The procedure then works recursively, by moving to the next strip as we go down the recursion tree. The recursion stops when the (constant) depth $t + 1$ is reached. At this stage, the resulting lists should be small enough and an exhaustive search (QuadraticNN) is performed to find a solution, assuming that it survived the filtering steps on this branch.

The recursion tree is schematized in Figure 1. A node that contains a good pair (u^*, v^*) is represented by (L^*, R^*) . At each level, the lists are smaller and smaller.

May and Ozerov also discuss the case where the size of resulting lists $L^{(j)}$ and $R^{(j)}$ are “too big”. Given a node at depth j , the recursive procedure stops if the size of the two lists deviates from their expected size by a factor of $(1 + 2^{\frac{5}{2}})$. Using Chebychev bounds this only happens with probability less than $2^{-\epsilon m}$. So the probability that the procedure stops at a path of length t while containing the good pair (u^*, v^*) is $2t2^{-\epsilon m}$. However this adds further constraints on the choice of ϵ , as discussed in section 5.

4 Analysis of both algorithms

The objective of this section is to establish matching bounds on the runtime of both of the aforementioned methods. More precisely we give an upper bound on the expected time complexity the Projection Method and a *lower bound* on that of the MO-NN algorithm. For fixed values of λ and γ , these bounds allow us to estimate the values of m for which we know *for sure* that the Projection Method requires less operations than the MO-NN algorithm. Indeed, denote by $T_{proj}(m)$ the expected time complexity of the Projection Method and by $T_{MO}(m)$ the expected time complexity of the MO-NN algorithm; assume that there are bounds $B_1(m)$ and $B_2(m)$ such that $T_{proj}(m) \leq B_1(m)$ and $B_2(m) \leq T_{MO}(m)$; then obviously $B_1(m) \leq B_2(m)$ implies that $T_{proj}(m) \leq T_{MO}(m)$.

In Section 5 and 6, we give some experimental estimation of values of m for fixed γ and λ .

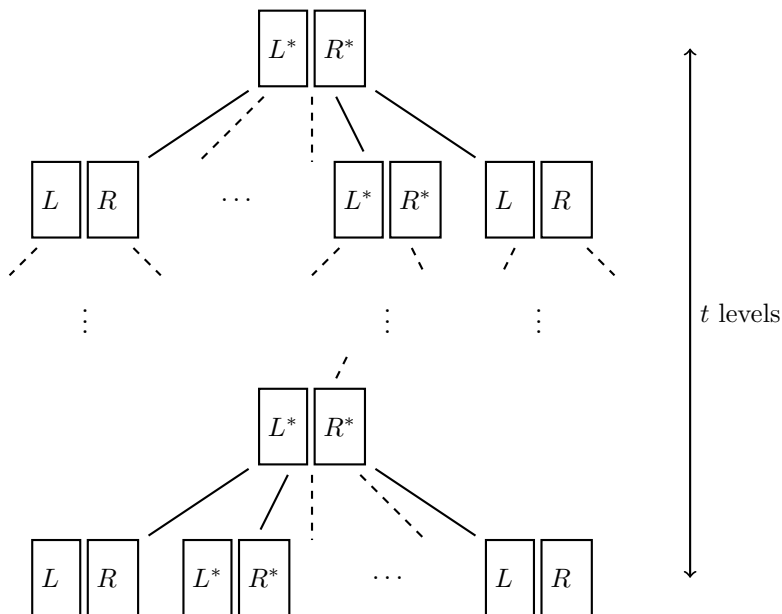


Figure 1: May-Ozerov recursion tree

4.1 Upper bound on the complexity of the projection method

Recall that N is the size of the input lists and k is the number of indices picked at line 4 of Algorithm 2. We recall the well-known analysis of the Projection Method.

Theorem 1. *Let $k = \lambda m = \log_2 N$, the Projection Method succeeds with probability greater than $1 - \frac{1}{e}$, in expected runtime T_{proj} . We have:*

$$\log_2(T_{proj}) \leq g(\gamma, \lambda)m + c(\gamma, \lambda)$$

with $c(\gamma, \lambda) = \log_2 \left(7 \sqrt{\frac{4(1-\gamma-\lambda)}{\pi(1-\lambda)(1-\gamma)}} \right)$ and $g(\gamma, \lambda) = \lambda + H(\lambda) - (1-\gamma)H\left(\frac{\lambda}{1-\gamma}\right)$. The success probability depends only on the random coins chosen by the algorithm, while the expected running time depends on the randomness of the input.

Proof. Using a bucket sort, both lists can be partitioned in less than $2 \times 3N$ operations. The total expected number of vector comparisons done in all the invocations of QUADRATICNN is $\frac{N^2}{2^k}$. Indeed, let Q denote the number of vector comparisons. If two vectors coincide on the k indices drawn in J , they belong to the same category. There are 2^k possible categories. Let A_i (resp. B_i) denote the size of L_i (resp. R_i), in other words, the number of vectors in L (resp R) of category i . Then,

$$Q = \sum_{i=1}^{2^k} A_i B_i$$

For each i , A_i and B_i are independent random variables. Thus,

$$\mathbb{E}[Q] = \mathbb{E} \left[\sum_{i=1}^{2^k} A_i B_i \right] = \sum_{i=1}^{2^k} \mathbb{E}[A_i B_i] = \sum_{i=1}^{2^k} \mathbb{E}[A_i] \mathbb{E}[B_i] = \sum_{i=1}^{2^k} \binom{N}{2^k} \binom{N}{2^k} = \frac{N^2}{2^k}.$$

It follows that one iteration of the algorithm requires an expected number of vector operations given by $T_{it} \leq 6N + \frac{N^2}{2^k}$. An iteration succeeds if and only if the target pair

(u^*, v^*) coincides on the k chosen indices. This happens with probability

$$p := \frac{\binom{m-\ell}{k}}{\binom{m}{k}}$$

The probability that this event happens in less than $N_{it} = \frac{1}{p}$ iterations is

$$\mathbb{P}[\text{success}] = 1 - \mathbb{P}[\text{fail at each iteration}] = 1 - (1 - p)^{\frac{1}{p}}$$

This is always greater than $1 - \frac{1}{e} \approx 0.63$ (reached when p gets close to zero). Thus, the Projection Method succeeds with probability greater than $1 - 1/e$ in expected time

$$T_{proj} \leq \left(6N + \frac{N^2}{2^k}\right) \frac{1}{p} = \left(6N + \frac{N^2}{2^k}\right) \frac{\binom{m}{k}}{\binom{m-\ell}{k}}$$

A well-known tradeoff for k is $\lambda m = \log_2 N$. Since we are choosing this particular value here, then we know that the optimal choice is at least as good. Then,

$$\begin{aligned} T_{proj} &\leq 7 \frac{\binom{m}{\lambda m}}{\binom{m-\ell}{\lambda m}} 2^{\lambda m} \\ &\leq 7 \sqrt{\frac{4(1-\gamma-\lambda)}{\pi(1-\lambda)(1-\gamma)}} 2^{m(\lambda+H(\lambda)-(1-\gamma)H(\frac{\lambda}{1-\gamma}))} \quad \text{using (1)} \end{aligned}$$

□

4.2 Analysis of the MO-NN algorithm

The main technical result of this paper is the following theorem which gives a lower bound on the expected number of operations of the MO-NN algorithm. In the following, $h = H^{-1}(1 - \lambda)$.

Theorem 2. *Let $\gamma < \frac{1}{2}$, $\ell = \gamma m$, $\lambda < \frac{1}{2}$. For choices of ϵ and t that guarantee a probability of success greater than $\frac{1}{8}$, the MO-NN algorithm terminates in expected time T_{MO} with*

$$\log_2(T_{MO}) \geq \left(y(\gamma, \lambda) + \frac{\epsilon}{2}\right) m + \left(\frac{t}{2} + 2\right) \log_2 m + f(\gamma, \lambda, \epsilon, t)$$

where

$$\begin{aligned} f(\gamma, \lambda, \epsilon, t) &= \log_2 \left(K \frac{(1-h-\frac{\gamma}{2})(h-\frac{\gamma}{2})\gamma^2}{(\gamma(1-\gamma))^{\frac{3}{2}-t}} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)}\right)^{\frac{1}{2}(t+3)} \left(\frac{\lambda}{y(\gamma, \lambda)}\right)^{\frac{t(t-1)}{4}} \right) \\ K &= \left(\frac{\pi^{3/2}}{2\sqrt{2}}\right)^t \times \frac{\pi^2}{32}. \end{aligned}$$

The best choices of t and ϵ are discussed in Section 5.

We first consider the “double rerandomization”, namely the two nested loops on lines 6 and 7 of Algorithm 3. Recall that its purpose is to ensure that the target pair (u^*, v^*) has a “good shape” in at least one iteration, in other terms that the two conditions stated in section 3.2 are satisfied at least once.

We first give a lower bound on the number of iteration of this double rerandomization that are required if we want to achieve of probability of success of at least $\frac{1}{4}$.

Lemma 1. *Let N_{it} the number of iterations of the double rerandomization. The solution (u^*, v^*) satisfies Conditions *i*) and *ii*) in at least one of the iterations with probability greater than $1/4$ only if*

$$N_{it} \geq \frac{1}{8\sqrt{2}} \left(\frac{\pi^{\frac{3}{2}}}{2} \right)^t m^{t-\frac{1}{2}} (\gamma(1-\gamma))^{t-\frac{1}{2}} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)} \right)^t \left(\frac{\lambda}{y(\gamma, \lambda)} \right)^{\frac{t(t-1)}{2}}.$$

It must be noted that performing more than this number of iterations does not imply that the probability of success will be greater than $1/4$ as we do not know how tight this bound is. We rather show that performing less iterations implies that the probability of success would be even smaller than $1/4$. This is far from the overwhelming probability claimed in [MO15] Lemma 1. There, May and Ozerov argued that the probability of success of one iteration of each randomization is $1/\text{poly}(m)$, and thus the expected number of iteration of the whole procedure in order to find the solution should be $\text{poly}(m)$. Restarting again $\text{poly}(m)$ times should be enough to ensure that the whole procedure succeed with overwhelming probability. Since the product of two polynomials in m is still a polynomial in m even though the degree may be huge, they ignored it in the asymptotic analysis of the runtime.

In order to reach the same ratio of success, we would have to restart the rerandomization process even more times.

Proof. Let p_1 be the probability that one iteration of the first rerandomization succeeds, that is (u^*, v^*) satisfies $d(u_j^*, v_j^*) = \gamma\alpha_j m$ for all $1 \leq j \leq t$ (condition *i*). As already discussed in [MO15] Lemma 1,

$$p_1 := \frac{\binom{\alpha_1 m}{\gamma\alpha_1 m} \dots \binom{\alpha_t m}{\gamma\alpha_t m}}{\binom{m}{\gamma m}}.$$

This means that if we make $f_1(m)$ independent attempts, the probability of succeeding in at least one of them is upper bounded by $f_1(m)p_1$ by a union bound. So if $f_1(m) < \frac{1}{2p_1}$, the probability that (u^*, v^*) satisfies Condition *i*) in at least one of them satisfies

$$\mathbb{P}[\textit{i} \text{ is satisfied in less than } f_1(m) \text{ attempts}] \leq f_1(m)p_1 \leq \frac{1}{2}$$

Now, assuming that a given attempt succeeded, we still need to have the second rerandomization to succeed as well. We denote by p_2 the probability that $wt(u_j^*) = wt(v_j^*) = \alpha_j m/2$ after one iteration of the second rerandomization (Condition *ii*), assuming Condition *i*) is satisfied.

Consider a fixed value $j \in \{1, \dots, t\}$. Let c_{01} denote the number of indices such that u_j^* has a coordinate value of 0 and v_j^* has a coordinate value of 1. Define c_{00}, c_{10} and c_{11} analogously. Our goal here is to lower-bound the required number of iterations by getting rid of the unknowns c_{xy} . We choose a uniformly random $r = (r_1, \dots, r_t) \in \mathbb{F}_2^{\alpha_1 m} \times \dots \times \mathbb{F}_2^{\alpha_t m}$ with weight $\frac{1}{2}\alpha_j m$ on each of the t strips and add it to all elements in L and R . Consider the target pair (u_j^*, v_j^*) on strip j . If r_j has exactly $\frac{1}{2}c_{xy}$ ones in all four parts xy then on u_j^* and v_j^* half the bits of each of the 4 ‘‘ xy parts’’ will be inverted after summing r_j . So the hamming weight of $u_j^* + r_j$ and $v_j^* + r_j$ will be $\frac{1}{2}c_{xy}$ on each of the 4 parts. Hence the hamming weight of $u_j^* + r_j$ and $v_j^* + r_j$ will be $\frac{1}{2}\alpha_j m$ because $c_{00} + c_{01} + c_{10} + c_{11} = \alpha_j m$. We say that r_j is good on the j -th strip if it has exactly $\frac{1}{2}c_{xy}$ ones in all four parts xy . The probability that this happens is

$$p_{2j} := \binom{c_{00}}{\frac{1}{2}c_{00}} \binom{c_{11}}{\frac{1}{2}c_{11}} \binom{c_{01}}{\frac{1}{2}c_{01}} \binom{c_{10}}{\frac{1}{2}c_{10}} / \binom{\alpha_j m}{\frac{1}{2}\alpha_j m}$$

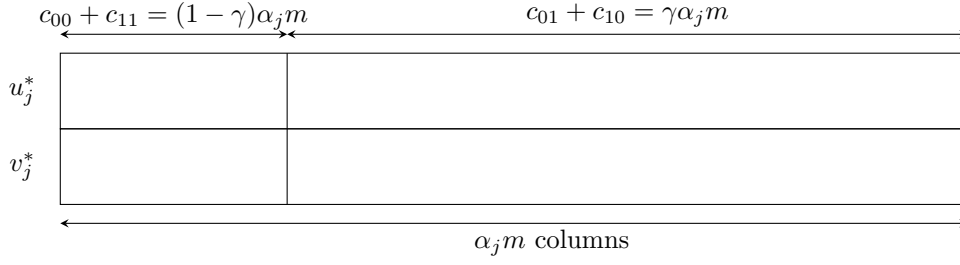


Figure 2: Bits distribution in May-Ozerov double randomization

In Figure 2, the first part of the vectors corresponds to the indices where (u^*, v^*) coincide and the second to the one where they do not.

We can therefore rewrite p_{2j} as follows:

$$p_{2j} = \binom{c_{00}}{\frac{1}{2}c_{00}} \binom{(1-\gamma)\alpha_j m - c_{00}}{\frac{(1-\gamma)\alpha_j m - c_{00}}{2}} \binom{c_{01}}{\frac{1}{2}c_{01}} \binom{\gamma\alpha_j m - c_{01}}{\frac{\gamma\alpha_j m - c_{01}}{2}} / \binom{\alpha_j m}{\frac{1}{2}\alpha_j m}$$

Using (3) we have that:

$$p_{2j} \leq \binom{(1-\gamma)\alpha_j m}{\frac{(1-\gamma)\alpha_j m}{2}} \binom{\gamma\alpha_j m}{\frac{\gamma\alpha_j m}{2}} / \binom{\alpha_j m}{\frac{1}{2}\alpha_j m}.$$

Moreover, the events are independent on each strip j , therefore if we note p_2 the probability to find the right r , we have :

$$p_2 = \prod_{j=1}^t p_{2j}$$

With the same argument than the one we used for the first randomization, if $f_2(m) \leq \frac{1}{2p_2}$, the probability that the second randomization succeed in less than $f_2(m)$ attempts is less than $1/2$.

Denoting by E_1 the event “ (i) in less than $f_1(m)$ attempts” and E_2 “ (ii) in less than $f_2(m)$ attempts” and combining the two, we obtain:

$$\mathbb{P}[\text{success}] = \mathbb{P}[E_1 \text{ and } E_2] = \mathbb{P}[E_1]\mathbb{P}[E_2|E_1] = p_1 p_2 < 1/4.$$

To summarize, we have proved the following: “If $N_{it} = f_1(m)f_2(m) < 1/(4p_1p_2)$ then the probability of success is lower than $1/4$ ”, taking the reverse statement, it means: “If we want a probability of success greater than $1/4$, then N_{it} has to be greater than $1/(4p_1p_2)$.”

We still have to give a lower bound on $1/(4p_1p_2)$. Recall that

$$\frac{1}{p_1} = \frac{\binom{m}{\gamma m}}{\binom{\alpha_1 m}{\gamma \alpha_1 m} \dots \binom{\alpha_t m}{\gamma \alpha_t m}} \geq \frac{\sqrt{2\pi\gamma(1-\gamma)}^t}{\sqrt{8\gamma(1-\gamma)}} m^{\frac{t-1}{2}} \prod_{i=1}^t \sqrt{\alpha_i} \quad \text{using (1) and simplifying.}$$

Then

$$\frac{1}{2p_1} \geq \frac{(2\pi)^{\frac{t}{2}}}{4\sqrt{2}} (\gamma(1-\gamma))^{\frac{t-1}{2}} m^{\frac{t-1}{2}} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)} \right)^{\frac{t}{2}} \left(\frac{\lambda}{y(\gamma, \lambda)} \right)^{\frac{t(t-1)}{4}}.$$

Similarly

$$\frac{1}{p_2} = \frac{1}{\prod_{j=1}^t p_{2j}} \geq \prod_{j=1}^t \frac{\pi}{2\sqrt{2}} \sqrt{\gamma(1-\gamma)} \sqrt{\alpha_j m}^{\frac{1}{2}} \quad \text{using (1) and simplifying.}$$

Then

$$\frac{1}{2p_2} \geq \frac{1}{2} \left(\frac{\pi}{2\sqrt{2}} \right)^t m^{\frac{t}{2}} (\gamma(1-\gamma))^{\frac{t}{2}} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)} \right)^{\frac{t}{2}} \left(\frac{\lambda}{y(\gamma, \lambda)} \right)^{\frac{t(t-1)}{4}}$$

Thus if we want the double re-randomization to succeed with probability greater than $\frac{1}{4}$ during at least one of the iterations, the number of iterations N_{it} needs to satisfy

$$N_{it} = f_1(m)f_2(m) \geq \frac{1}{8\sqrt{2}} \left(\frac{\pi^{\frac{3}{2}}}{2} \right)^t m^{t-\frac{1}{2}} (\gamma(1-\gamma))^{t-\frac{1}{2}} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)} \right)^t \left(\frac{\lambda}{y(\gamma, \lambda)} \right)^{\frac{t(t-1)}{2}}$$

□

In the May-Ozerov algorithm, they choose $f_1(m)$ and $f_2(m)$ such that that $d(u_j^*, v_j^*) = \gamma\alpha_j m$ and $wt(u_j^*) = wt(v_j^*) = \frac{1}{2}\alpha_j m$ in at least one of the rerandomized input lists with overwhelming probability. Thus $f_1(m)f_2(m) \geq N_{it}$. In order to prove Theorem 2, we now need to give a lower bound on the expected runtime of Algorithm 4, assuming that a solution (u^*, v^*) exists in $L \times R$. Recall that this procedure forms a recursion tree where each node is labeled by a pair of lists, and which possess exponentially many children themselves labeled by pairs of smaller filtered lists.

In [MO15, proof of Lemma 2], May and Ozerov show that at a given step j , the expected number of iterations until the good pair is found with overwhelming probability is given by ms_j . Using the expression of s_j in (5) and (1) we obtain

$$ms_j \geq \frac{\pi^2 (1-h-\frac{\gamma}{2})(h-\frac{\gamma}{2})(\frac{\gamma}{2})^2}{\sqrt{2} (1-\gamma)\gamma} \alpha_j^{\frac{3}{2}} m^{\frac{5}{2}} 2^{\alpha_j m y(\gamma, \lambda)}$$

We are now able to prove theorem 2.

Proof of Theorem 2. We split this proof in two part. First, we give a lower bound on the runtime of the RECURSIVEMO procedure from the beginning ($j = 1$) to the end ($j = t+1$). Then we combine this result with Lemma 1, in order to have a lower bound on the expected runtime of the MO-NN algorithm.

Let us start with the complexity analysis of RECURSIVEMO. We have seen that at each call of RECURSIVEMO with a parameter $j \leq t$, the for-loop line 6 does ms_j iterations. In particular, for $j = 1$,

$$ms_1 \geq \frac{\pi^2 (1-h-\frac{\gamma}{2})(h-\frac{\gamma}{2})(\frac{\gamma}{2})^2}{\sqrt{2} (1-\gamma)\gamma} \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)} \right)^{\frac{3}{2}} m^{\frac{5}{2}} 2^{\alpha_1 m y(\gamma, \lambda)},$$

by replacing α_1 by its actual value as defined in Algorithm 3. Recall that both lists L and R are filtered during the MO-NN algorithm before entering the recursion tree (cf. Algorithm 3, line 11). This is to ensure that they consist only of vectors of Hamming weight $|I_j|/2$ on every strip $|I_j|$. As such, the input lists L and R at the root of the tree are both smaller than their original version.

Let N_1 be the expected size of \bar{L} and \bar{R} at the root of the tree, and let x_j , for $1 \leq j \leq t$ be a uniformly random vector of $\mathbb{F}_2^{\alpha_j m}$. We denote by W_j the random variable that gives the Hamming Weight of x_j . It follows a binomial distribution of parameters $(\alpha_j m, 1/2)$. As such

$$\mathbb{P} \left[W_j = \frac{1}{2} \alpha_j m \right] = \binom{\alpha_j m}{\frac{1}{2} \alpha_j m} 2^{-\alpha_j m}. \quad (4)$$

Considering the fact that each strip is independent from the others, the probability that an m -bit vector has Hamming weight $\frac{1}{2}\alpha_j m$ on each of t strips is

$$\begin{aligned} P_2 &:= \prod_{j=1}^t \frac{\binom{\alpha_j m}{\frac{1}{2}\alpha_j m}}{2^{\alpha_j m}} \geq \prod_{j=1}^t \frac{1}{\sqrt{2\alpha_j m}} \quad \text{using (1) and simplifying} \\ &\geq \left(\frac{1}{\sqrt{2}}\right)^t \left(\frac{y(\gamma, \lambda)}{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}\right)^{\frac{t}{2}} \left(\frac{y(\gamma, \lambda)}{\lambda}\right)^{\frac{t(t-1)}{4}} m^{-\frac{t}{2}} \end{aligned}$$

It follows that the expected size of \bar{L} and \bar{R} at the root of the tree is lower-bounded as follows:

$$N_1 \geq 2^{\lambda m} \left(\frac{1}{\sqrt{2}}\right)^t \left(\frac{y(\gamma, \lambda)}{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}\right)^{\frac{t}{2}} \left(\frac{y(\gamma, \lambda)}{\lambda}\right)^{\frac{t(t-1)}{4}} m^{-\frac{t}{2}}.$$

Let T_j the expected number of operations at depth j of the tree. Then, in particular $T_1 = 2ms_1N_1$ with the 2 factor coming from the fact that both lists are considered. Expanding the definitions gives:

$$T_1 \geq p(\lambda, \gamma) \left(\frac{y(\gamma, \lambda) - \lambda + \frac{\epsilon}{2}}{y(\gamma, \lambda)}\right)^{\frac{3}{2} - \frac{t}{2}} \left(\frac{1}{\sqrt{2}}\right)^t \left(\frac{y(\gamma, \lambda)}{\lambda}\right)^{\frac{t(t-1)}{4}} m^{\frac{5}{2} - \frac{t}{2}} 2^{(\alpha_1 y(\gamma, \lambda) + \lambda)m},$$

with $p(\lambda, \gamma)$ defined as

$$p(\lambda, \gamma) = \frac{\pi^2}{2\sqrt{2}} \frac{(1 - h - \frac{\gamma}{2})(h - \frac{\gamma}{2})\gamma}{1 - \gamma}.$$

This in fact gives us a lower bound on the running time of the whole recursive procedure, since at least this step will be performed. The time complexity T_{rec} of RECURSIVEMO therefore satisfies $T_{rec} \geq T_1$.

In addition, Lemma 1 gives us a lower bound of N_{it} , the number of iterations we have to repeat the whole process in order to ensure that a solution in “good shape” will be present at the beginning of the recursion with probability greater than $\frac{1}{4}$. This gives.

$$T_{MO} \geq N_{it} \left(\sum_{j=1}^{t+1} T_j \right) \geq N_{it} T_1$$

The probability that a list containing one of the components of the good pair is deleted is upper-bounded by $2t2^{-\epsilon m}$. Therefore, choosing ϵ greater than $\epsilon > \frac{\log_2(4t)}{m}$ ensures that a list containing the desired pair is not deleted with a probability greater than $\frac{1}{2}$. \square

5 Crossover point between the two algorithms

In this section, we detail a procedure that, given λ and γ finds a threshold T such that the Projection Method is always faster than the MO-NN algorithm on instances of size $m \leq T$. Because the values of the threshold will often be very large, this will officially qualify the MO-NN algorithm as “galactic”.

To compare the performance of the two algorithms, we have to choose the parameters t and ϵ in the MO-NN algorithm. We choose them in a way that minimizes the lower bound T_{MO} given by theorem 2.

Suppose there is a procedure that, given (λ, γ, m) , finds the parameters (t, ϵ) minimizing the lower-bound T_{MO} . To find the crossover point between the two algorithms, we proceed

as follows: starting from $m = 1$, repeatedly double m while $T_{proj}(m) \leq T_{MO}(m)$. Then find the actual threshold inside the interval $[m/2; m]$ using bisection search. The end result is a m such that $T_{proj}(m) \leq T_{MO}(m)$ and $T_{proj}(m + 1) > T_{MO}(m + 1)$. This is the crossover point.

It remains to describe the procedure that finds the optimal parameters for the May-Ozerov algorithm. We recall that, in [MO15], t is defined as follows:

$$t = \left\lceil \frac{\log(2(y(\gamma, \lambda) - \lambda)/\epsilon + 1)}{\log(y(\gamma, \lambda)/\lambda)} \right\rceil$$

Observe that t grows when ϵ goes to zero. However, ϵ can not take arbitrary small values for the following reasons:

1. It is required that $\gamma\alpha_t m \geq 1$. Otherwise, we could not apply the algorithm because at each node, on the current strip j , $d(u_j^*, v_j^*)$ must be at least one and the smallest strip is the t -th.
2. The probability that a list containing one of the components of the good pair is deleted is upper-bounded by $2t2^{-\epsilon m}$. We therefore consider that this probability must be smaller than $1/2$, *i.e.* that $\epsilon > \frac{\log_2(4t)}{m}$.
3. t must be at least 2, for otherwise the algorithm would have a complexity greater than $2^{2\lambda m}$.

When ϵ varies, t may jump from an integer value to the next. There are ranges of ϵ that leads to the same value of t . A quick examination of the formula for T_{MO} reveals that, as long as t does not change, T_{MO} is an increasing function of ϵ . Therefore, the optimal value of ϵ is necessarily one that triggers a jump in t . Because $t \leq m$, there is a finite number of values to test. To find the optimal parameters, we thus use the following procedure:

- Start from $t = 2$
- Compute the smallest possible ϵ that yields this value of t
- If ϵ satisfies the constraints given above, evaluate T_{MO} and record it along with t and ϵ
- Increase t and retry, until $t = m$
- Return the choice that yield the smallest possible value of T_{MO}

As an example, fix $\lambda = 0.025$ and $\gamma = 0.1$ (this specific choice of values will be justified below). Ignoring the polynomial and constant factors, we find that the Projection Method runs in time $\tilde{O}(2^{0.0289\dots m})$ while for any $\epsilon > 0$, the MO-NN algorithm runs in time $\tilde{O}(2^{(0.02819\dots + \epsilon)m})$. It wins asymptotically.

The procedure described above shows that the crossover point between the two algorithms is at $m = 336,017$ using $t = 20$ (hence $\epsilon = 0.000755\dots$). The May-Ozerov algorithm requires at least $11m^{12}2^{0.02819m}$ operations. The size of the input lists at the crossover point is then 2^{8400} . The actual number of operations is at least 2^{9697} . This clearly qualifies as “galactic”.

6 Application to the syndrome decoding problem

Consider a $[n, k, d]$ binary linear code: it has length n , dimension k and minimal distance d . It encodes k -bit vectors into n -bit vectors and up to $(d - 1)/2$ errors can be decoded at

Table 2: Proposed parameter sets for the McEliece cryptosystem. The star mean that the target security level is reached by injecting $t + 1$ or $t + 2$ errors; list-decoding and some redundancy in the plaintext are then required.

r	n	t	Code	R	D	Target security level	Origin
10	1024	50	[1024, 524, 101]	0.51	0.099	64	[McE78]
11	1632	33	[1632, 1269, 67]	0.78	0.041	80*	[BLP08]
11	2048	27	[2048, 1751, 55]	0.85	0.027	80	
11	2960	56	[2960, 2288, 113]	0.77	0.038	128*	
12	3488	64	[3488, 2720, 129]	0.78	0.037	128	[BCL ⁺ 17]
13	4608	96	[4608, 3360, 193]	0.73	0.042	192	
13	6688	128	[6688, 5024, 257]	0.75	0.038	256	
13	6960	119	[6960, 5413, 238]	0.78	0.034	256	
13	8192	128	[8192, 6528, 257]	0.80	0.031	256	

maximum likelihood. The rate of the code is $R = k/n$ and its relative distance is $D = d/n$. The code is entirely described by its parity matrix H , of size $(n - k) \times n$, which is such that $Hx = 0$ whenever x is a code word. When $\hat{x} = x + e$ is a noisy codeword (where e is an error vector), then $s := H\hat{x} = He$ is the *syndrome*, and it depends only on the error. Solving $He = s$ with $|e| \leq (d - 1)/2$ thus reveals the error vector (and indirectly the original input).

The first direct cryptographic application of decoding arbitrary linear codes consisted in attacking the McEliece public-key encryption scheme [McE78]. The parameters initially proposed by McEliece in 1978 are [1024, 524, 101]. They have been practically broken in [BLP08]. More generally, the binary Goppa code used in McEliece is defined by a square-free degree- t polynomial over \mathbb{F}_{2^r} and is usually $[n, n - tr, 2t + 1]$ (under the constraint that $n \leq 2^r$). Several sets of parameters have been proposed; they are collected in Table 2.

Random linear codes are also interesting from a cryptographic point of view. They approximately meet the Gilbert-Varshamov bound: we have $R \approx 1 - H(D)$ for a random linear code. In particular, we find that $H^{-1}(1 - R) \approx D$. The decodingchallenge.org website offers a challenge that consists in finding a lowest possible weight codeword inside length-1280 code with rate 1/2. The Gilbert-Varshamov bound suggests that it has minimum distance 144.

To the best of our knowledge, random linear codes have been first used in Stern’s identification protocol [Ste93]. The Stern-Fischer pseudorandom generator [FS96] is provably secure under the hardness of the syndrome decoding problem. The FSB hash function [AFS05] has been submitted to the NIST “SHA-3” competition. Its collision-resistance relied on the hardness of solving random instances of the (regular) syndrome decoding problem. More recently, candidate “post-quantum” signature schemes have been designed around proofs of knowledge of a solution to an instance of the syndrome decoding problem with a random linear code with rate 0.5 [FJR22]. In SDitH [MGF⁺23], this is also a random code of length 1280.

Most algorithms for the Syndrome Decoding problem rely on the idea of Information Set Decoding. This amounts to make an assumption on the location of the non-zero bits of the error vector e . Randomly permuting the coordinates of e (and thus the columns of H) may satisfy this assumption with some non-zero probability. If this assumption is satisfied, there is an efficient way to recover e entirely. The whole process has to be repeated until it succeeds.

More concretely, we describe below the DECODE algorithm given in [MO15]. It is a variant of the Leon/Stern/Dumer decoding algorithm that uses nearest neighbor search.

The columns of H are permuted (right-multiplication with a permutation matrix P), then the result is put into Reduced Row-Echelon Form (left-multiplication by an invertible

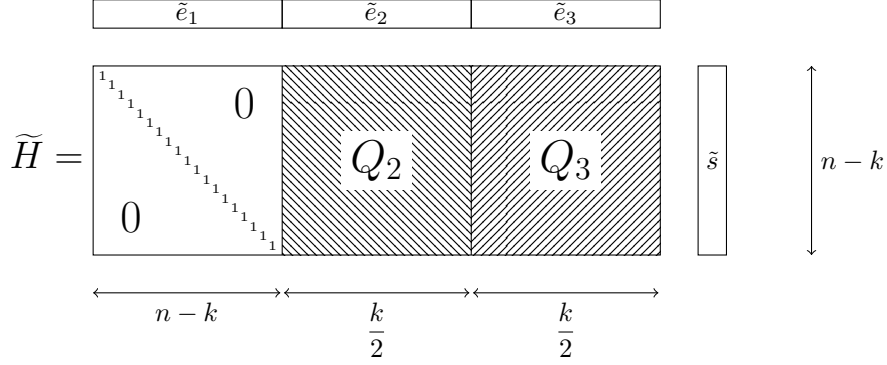


Figure 3: The technique of *Information Set Decoding* combined with a Stern-style meet-in-the-middle.

matrix T), as in Fig. 3. If the principal $(n-k) \times (n-k)$ submatrix is not invertible, abort. This yields $\tilde{H} = THP$, and the initial instance of the problem then becomes $\tilde{H}\tilde{e} = \tilde{s}$ with $\tilde{s} = Ts$, $P\tilde{e} = e$ and still $|\tilde{e}| \leq w$.

The permuted error vector \tilde{e} is then split in three parts: \tilde{e}_1 corresponds to the first $n-k$ coordinates, \tilde{e}_2 and \tilde{e}_3 correspond to the next and the last $k/2$ coordinates, respectively.

The main idea consists in choosing a parameter p (even) and making the assumption that $|e_1| \leq w-p$, $|e_2| = p/2$ and $|e_3| = p/2$ (of course this only holds true with some probability). The full procedure is given below:

1. Apply a random permutation P to the columns of Q
2. Compute an invertible matrix T such that $THP = (I|Q_2|Q_3)$. If no such T exists, go back to step 1.
3. Set $L \leftarrow \{Q_2u \in \mathbb{F}_2^{n-k}, |u| = p/2\}$
4. Set $R \leftarrow \{Q_3v + Ts \in \mathbb{F}_2^{n-k}, |v| = p/2\}$
5. If there is no $(x, y) \in L \times R$ such that $d(x, y) \leq w-p$, go back to step 1.
6. Otherwise, let $x = Q_2u$ and $y = Q_3v + \tilde{s}$ with $|x+y| \leq w-p$. Then $\tilde{e} = (x+y)uv$ satisfies $\tilde{H}\tilde{e} = \tilde{s}$ and $|\tilde{e}| \leq w$.

It is expected that the resolution of the instance of the nearest neighbor problem dominates all the other steps. Suppose that the input problem admits a solution e^* ; this procedure succeeds when the permutation P is ‘‘good’’, namely when e^* has $w-p$ bits in the first $n-k$ entries, $p/2$ bits in the next $k/2$ entries and again $p/2$ bits in the last $k/2$ entries. Define $\bar{p} = p/n$. The expected number of iterations is therefore

$$N = \binom{n}{w} / \left[\binom{n-k}{w-p} \binom{k/2}{p/2}^2 \right].$$

The lists L and R are made up of vectors in \mathbb{F}_2^{n-k} , they have size

$$\binom{k/2}{p/2} \approx 2^{(n-k) \frac{R}{2(1-R)} H\left(\frac{\bar{p}}{R}\right)}$$

Table 3: Best parameter choice and instance characteristics at the crossover point.

R	D	n	Projection method			MO-NN					$\log_2 T$
			p	λ	γ	p	λ	γ	t	$10^6 \epsilon$	
0.5	0.11	533502	1656	0.0272	0.1038	1894	0.0304	0.1029	20	877	29566
0.8	0.03	1874400	4324	0.0570	0.0635	4546	0.0594	0.0629	30	1276	63487

Table 4: Complexity of both algorithms when using MO-NN potentially becomes better than using the projection method for decoding. The projection method runs in time less than $\gamma 2^{\alpha m} m^\beta$ operations, while the MO-NN algorithm requires at least this many operations.

λ	γ	Projection method			MO-NN				
		α	β	γ	t	$10^6 \epsilon$	α	β	γ
0.0304	0.1029	0.03527	0	7.88	20	877	0.03441	12	24.5
0.0594	0.0629	0.06519	0	7.88	30	1276	0.06419	17	$2^{56.5}$

and the maximum allowed distance is $w - p \approx (n - k) \times \left(\frac{D}{2(1-R)} - \frac{\bar{p}}{1-R} \right)$. It follows that the corresponding instance of the Nearest Neighbor problem has

$$\lambda \approx \frac{R}{2(1-R)} H\left(\frac{\bar{p}}{R}\right), \quad \gamma \approx \frac{D}{2(1-R)} - \frac{\bar{p}}{1-R}$$

with $m = n - k$.

To ascertain how galactic the MO-NN algorithm is in this context, we do the following. We consider a family of codes (e.g. random linear codes or binary Goppa codes). For a given length, we exhaustively try all possible values of p ; each one yields an instance of the *Nearest Neighbor Problem*. We estimate the number of operations needed to solve it using either the Projection Method or the MO-NN. When the best option is to use the MO-NN, we stop. Otherwise, we increase the length of the code and retry. Tables 3 and 4 show the results.

A first setting involves random linear codes of rate $1/2$. Their minimum distance is estimated using the Gilbert-Varshamov bound ($D \approx 0.11003$). The smallest code length for which the MO-NN algorithm seems to bring an improvement is greater than $n = 533,500$. The estimated total number of operation of the decoding algorithm is way beyond 2^{29565} . In this case, the instances of the nearest neighbor problem that have to be solved have $\lambda \approx 0.03$ and $\gamma \approx 0.1$ (this is close to the values we selected earlier to compare the MO-NN algorithm with the projection method).

Another setting involves binary Goppa codes (as in McEliece). We assumed that they have rate $R = 0.8$ and relative distance $D = 0.03$. The smallest length at which the asymptotic advantage of the MO-NN algorithm arises is beyond $n = 1,874,400$. Again, the required number of operation for decoding is beyond 2^{63489} . This leads to instances of the nearest neighbor problem with $\lambda \approx 0.06$ and $\gamma \approx 0.063$.

Table 4 then shows the concrete complexities of both algorithms as a function of m when using MO-NN becomes better than using the projection method in the Decode algorithm. It shows that, at the crossover point, the MO-NN is penalized by huge polynomial and constant factors. This explains why extremely large instance sizes are required to benefit from the improved exponent in the asymptotic complexity.

Acknowledgements

We thank the anonymous reviewers for their comments. We acknowledge financial support from the French *Agence Nationale de la Recherche* under project ‘‘GORILLA’’ (ANR-20-

CE39-0002).

References

- [AFS05] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer, 2005. doi:[10.1007/11554868_6](https://doi.org/10.1007/11554868_6).
- [BCL⁺17] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang. Classic McEliece, 2017.
- [BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012. doi:[10.1007/978-3-642-29011-4_31](https://doi.org/10.1007/978-3-642-29011-4_31).
- [BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008. doi:[10.1007/978-3-540-88403-3_3](https://doi.org/10.1007/978-3-540-88403-3_3).
- [BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011. doi:[10.1007/978-3-642-22792-9_42](https://doi.org/10.1007/978-3-642-22792-9_42).
- [BMVT78] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. doi:[10.1109/TIT.1978.1055873](https://doi.org/10.1109/TIT.1978.1055873).
- [CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial. doi:[10.1016/S0747-7171\(08\)80013-2](https://doi.org/10.1016/S0747-7171(08)80013-2).
- [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. doi:[10.1109/TIT.2010.2050814](https://doi.org/10.1109/TIT.2010.2050814).
- [EB22] Andre Esser and Emanuele Bellini. Syndrome decoding estimator. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2022. doi:[10.1007/978-3-030-97121-2_5](https://doi.org/10.1007/978-3-030-97121-2_5).

- [EMZ22] Andre Esser, Alexander May, and Floyd Zweydinger. McEliece needs a break - solving mceliece-1284 and quasi-cyclic-2918 with modern ISD. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 433–457. Springer, 2022. doi:10.1007/978-3-031-07082-2_16.
- [FJR22] Thibault Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 541–572. Springer, 2022. doi:10.1007/978-3-031-15979-4_19.
- [FS96] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 245–255. Springer, 1996. doi:10.1007/3-540-68339-9_22.
- [IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613. ACM, 1998. doi:10.1145/276698.276876.
- [KWZ95] Richard M. Karp, Orli Waarts, and Geoffrey Zweig. The bit vector intersection problem (preliminary version). In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 621–630. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492663.
- [LB88] Pil Joong Lee and Ernest F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988. doi:10.1007/3-540-45961-8_25.
- [LR13] Richard J. Lipton and Kenneth W. Reagan. David Johnson: Galactic Algorithms. In *People, Problems, and Proofs: Essays from Gödel's Lost Letter: 2010*, pages 109–112. Springer, 2013. doi:10.1007/978-3-642-41422-0_20.
- [McE78] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.
- [MGF⁺23] Carlos Aguilar Melchor, Shay Gueron, Thibault Feneuil, James Howe, Edoardo Persichetti, David Joseph, Nicolas Gama, Antoine Joux, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. The syndrome decoding in the head (sd-in-the-head) signature scheme, May 2023.
- [MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference*

- on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011. doi:[10.1007/978-3-642-25385-0_6](https://doi.org/10.1007/978-3-642-25385-0_6).
- [MO15] 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 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228. Springer, 2015. doi:[10.1007/978-3-662-46800-5_9](https://doi.org/10.1007/978-3-662-46800-5_9).
- [NUO⁺24] Shintaro Narisada, Shusaku Uemura, Hiroki Okada, Hiroki Furue, Yusuke Aikawa, and Kazuhide Fukushima. Solving mceliece-1409 in one day - cryptanalysis with the improved BJMM algorithm. In Nicky Mouha and Nick Nikiforakis, editors, *Information Security - 27th International Conference, ISC 2024, Arlington, VA, USA, October 23-25, 2024, Proceedings, Part II*, volume 15258 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2024. doi:[10.1007/978-3-031-75764-8_1](https://doi.org/10.1007/978-3-031-75764-8_1).
- [Riv74] Ronald L. Rivest. On the optimality of elia’s algorithm for performing best-match searches. In Jack L. Rosenfeld, editor, *Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974*, pages 678–681. North-Holland, 1974.
- [Ste88] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988. doi:[10.1007/BFb0019850](https://doi.org/10.1007/BFb0019850).
- [Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993. doi:[10.1007/3-540-48329-2_2](https://doi.org/10.1007/3-540-48329-2_2).
- [Top01] Flemming Topsøe. Bounds for entropy and divergence for distributions over a two-element set. *JIPAM. Journal of Inequalities in Pure and Applied Mathematics*, 2(2):Paper No. 25, 13 p., 2001.
- [Wel71] Terry A. Welch. Bounds on information retrieval efficiency in static file structures, June 1971. Project MAC Report MAC-TR-88.