# Quantum Information Set Decoding Algorithms

Ghazal Kachigar[1] and Jean-Pierre Tillich[2]

[1] Institut de Mathématiques de Bordeaux
Université de Bordeaux
Talence Cedex F-33405, France
ghazal.kachigar@u-bordeaux.fr
[2] Inria, EPI SECRET
2 rue Simone Iff, Paris 75012, France
jean-pierre.tillich@inria.fr

**Abstract.** The security of code-based cryptosystems such as the McEliece cryptosystem relies primarily on the difficulty of decoding random linear codes. The best decoding algorithms are all improvements of an old algorithm due to Prange: they are known under the name of information set decoding techniques. It is also important to assess the security of such cryptosystems against a quantum computer. This research thread started in [22] and the best algorithm to date has been Bernstein's quantising [5] of the simplest information set decoding algorithm, namely Prange's algorithm. It consists in applying Grover's quantum search to obtain a quadratic speed-up of Prange's algorithm. In this paper, we quantise other information set decoding algorithms by using quantum walk techniques which were devised for the subset-sum problem in [6]. This results in improving the worst-case complexity of $2^{0.06035n}$ of Bernstein's algorithm to $2^{0.05869n}$ with the best algorithm presented here (where $n$ is the codelength).

**Keywords:** code-based cryptography, quantum cryptanalysis, decoding algorithm.

## 1 Introduction

As humanity's technological prowess improves, quantum computers have moved from the realm of theoretical constructs to that of objects whose consequences for our other technologies, such as cryptography, must be taken into account. Indeed, currently prevalent public-key cryptosystems such as RSA and ECDH are vulnerable to Shor's algorithm [26], which solves factorisation and the discrete logarithm problem in polynomial time. Thus, in order to find a suitable replacement, it has become necessary to study the impact of quantum computers on other candidate cryptosystems. Code-based cryptosystems such as the McEliece [20] and the Niederreiter [21] cryptosystems are such possible candidates.

Their security essentially relies on decoding a linear code. Recall that the decoding problem consists, when given a linear code $\mathcal{C}$ and a noisy codeword $c + e$, in recovering $c$, where $c$ is an unknown codeword of $\mathcal{C}$ and $e$ an unknown error of Hamming weight $w$. A (binary) linear code $\mathcal{C}$ of dimension $k$ and length $n$ is specified by a full rank binary matrix $H$ (i.e. a parity-check matrix) of size $(n-k) \times n$ as

$$\mathcal{C} = \{c \in \mathbb{F}_2^n : Hc^T = 0\}.$$

Since $H(c+e)^T = Hc^T + He^T = He^T$ the decoding problem can be rephrased as a syndome decoding problem

*Problem 1 (Syndrome Decoding Problem).* Given $H$ and $s^T = He^T$, where $|e| = w$, find $e$.

This problem has been studied since the Sixties and despite significant efforts on this issue [23, 27, 11, 2, 18, 7, 4, 19] the best algorithms for solving this problem [4, 19] are exponential in the number of errors that have to be corrected: correcting $w$ errors in a binary linear code of length $n$ and dimension $k$ has with the aforementioned algorithms a cost of $\tilde{O}(2^{\alpha(\frac{k}{n}, \frac{w}{n})n})$ where $\alpha(R, \omega)$ is

positive when $R$ and $\omega$ are both positive. All these algorithms use in a crucial way the original idea due to Prange [23] and are known under the name of Information Set Decoding (ISD) algorithms: they all take advantage of the fact that there might exist a rather large set of positions containing an information set of the code[3] that is almost error free.

All the efforts that have been spent on this problem have only managed to decrease slightly this exponent $\alpha(R,\omega)$. The following table gives an overview of the average time complexity of currently existing classical algorithms when $w$ is the Gilbert-Varshamov distance $d_{\mathrm{GV}}(n,k)$ of the code. This quantity is defined by $d_{\mathrm{GV}}(n,k) \stackrel{\triangle}{=} nH_2^{-1}\left(1-\frac{k}{n}\right)$ where $H_2$ is the binary entropy function $H_2(x) \stackrel{\triangle}{=} -x\log_2(x)-(1-x)\log_2(1-x)$ and $H_2^{-1}$ its inverse defined from $[0,1]$ to $[0,\frac{1}{2}]$. It corresponds to the largest distance for which we may still expect a unique solution to the decoding problem. If we want uniqueness of the solution, it can therefore be considered as the hardest instance of decoding. In the following table, $\omega_{\mathrm{GV}}$ is defined by the ratio $\omega_{\mathrm{GV}} \stackrel{\triangle}{=} d_{\mathrm{GV}}(n,k)/n$.

| **Author(s)** | Year | $\max_{0\le R\le 1} \alpha(R,\omega_{\mathrm{GV}})$ to 4 dec. places |
|---|---|---|
| Prange [23] | 1962 | 0.1207 |
| Dumer [11] | 1991 | 0.1164 |
| MMT [18] | 2011 | 0.1114 |
| BJMM [4] | 2012 | 0.1019 |
| MO [19] | 2015 | 0.0966 |

The question of using quantum algorithms to speed up ISD decoding algorithms was first put forward in [22]. However, the way Grover's algorithm was used in [22, Subsec. 3.5] to speed up decoding did not allow for significant improvements over classical ISD algorithms. Later on, it was shown by Bernstein in [5] that it is possible to obtain much better speedups with Grover's algorithm: by using it for finding an error-free information set, the exponent of Prange's algorithm can indeed be halved.

This paper builds upon this way of using Grover's search algorithm, as well as the quantum algorithms developped by Bernstein, Jeffery, Lange and Meurer in [6] to solve the subset sum problem more efficiently. The following table summarises the ingredients and average time complexity of the algorithm of [5] and the new quantum algorithms presented in this paper.

| **Author(s)** | Year | Ingredients | $\max_{0\le R\le 1} \alpha(R,\omega_{\mathrm{GV}})$ to 5 dec. places |
|---|---|---|---|
| Bernstein [5] | 2010 | Prange+Grover | 0.06035 |
| This paper | 2017 | Shamir-Schroeppel+Grover+QuantumWalk | 0.05970 |
| This paper | 2017 | MMT+"1+1=0"+Grover+QuantumWalk | 0.05869 |

A quick calculation shows that the complexity exponent of our best quantum algorithm, $MMTQW$, fulfils $\alpha_{\mathrm{MMTQW}} \approx \frac{\alpha_{\mathrm{Dumer}}}{2} + 4.9\times 10^{-4}$. Thus, our best quantum algorithm improves in a small but non-trivial way on [5]. Several reasons will be given throughout this paper on why it has been difficult to do better than this.

**Notation.** Throughout the paper, we denote by $|e|$ the Hamming weight of a vector $e$. We use the same notation for denoting the cardinality of a set, i.e. $|\mathscr{S}|$ denotes the cardinality of the set $\mathscr{S}$. The meaning of this notation will be clear from the context and we will use calligraphic letters to denote sets: $\mathscr{S}, \mathcal{I}, \mathcal{M}, \ldots$. We use the standard $O\,()$, $\Omega\,()$, $\Theta\,()$ notation and use the less standard $\tilde{O}\,()$, $\tilde{\Omega}\,()$, $\tilde{\Theta}\,()$ notation to mean "$O\,()$, $\Omega\,()$, $\Theta\,()$, when we ignore logarithmic factors". Here all the quantities we are interested in are functions of the codelength $n$ and we write $f(n) = \tilde{O}\,(g(n))$ for instance, when there exists a constant $k$ such such that $f(n) = O\left(g(n)\log^k(g(n))\right)$.

[3] An information set of a linear code $C$ of dimension $k$ is a set $\mathcal{I}$ of $k$ positions such that when given $\{c_i : i \in \mathcal{I}\}$ the codeword $c$ of $C$ is specified entirely.

## 2 Quantum search algorithms

### 2.1 Grover search

Grover's search algorithm [13, 14] is, along with its generalisation [8] which is used in this paper, an optimal algorithm for solving the following problem with a quadratic speed-up compared to the best-possible classical algorithm.

*Problem 2 (Unstructured search problem).* Given a set $\mathcal{E}$ and a function $f : \mathcal{E} \to \{0, 1\}$, find an $x \in \mathcal{E}$ such that $f(x) = 1$.

In other words, we need to find an element that fulfils a certain property, and $f$ is an oracle for deciding whether it does. Moreover, in the new results presented in this paper, $f$ will be a quantum algorithm. If we denote by $\varepsilon$ the proportion of elements $x$ of $\mathcal{E}$ such that $f(x) = 1$, Grover's algorithm solves the problem above using $O(\frac{1}{\sqrt{\varepsilon}})$ queries to $f$, whereas in the classical setting this cannot be done with less than $O(\frac{1}{\varepsilon})$ queries. Furthermore, if the algorithm $f$ executes in time $T_f$ on average, the average time complexity of Grover's algorithm will be $O(\frac{T_f}{\sqrt{\varepsilon}})$.

### 2.2 Quantum Walk

**Random Walk.** Unstructured search problems as well as search problems with slightly more but still minimal structure may be recast as graph search problems.

*Problem 3 (Graph search problem).* Given a graph $G = (\mathcal{V}, \mathcal{E})$ and a set of vertices $\mathcal{M} \subset \mathcal{V}$, called the set of *marked elements*, find an $x \in \mathcal{M}$.

The graph search problem may then be solved using random walks (discrete-time Markov chains) on the vertices of the graph. From now on, we will take the graph to be undirected, connected, and $d$-regular, i.e. such that each vertex has exactly $d$ neighbours.

*Markov chain.* A Markov chain is given by an initial probability distribution $v$ and a stochastic transition matrix $M$. The transition matrix of a random walk on a graph (as specified above) is obtained from the graph's adjacency matrix $A$ by $M = \frac{1}{d}A$.

*Eigenvalues and the spectral gap.* A closer look at the eigenvalues and the eigenvectors of $M$ is needed in order to analyse the complexity of a random walk on a graph. The eigenvalues will be noted $\lambda_i$ and the corresponding eigenvectors $v_i$. We will admit the following points (see [10]):
(i) all the eigenvalues lie in the interval $[-1, 1]$;
(ii) 1 is always an eigenvalue, the corresponding eigenspace is of dimension 1;
(iii) there is a corresponding eigenvector which is also a probability distribution (namely the uniform distribution $u$ over the vertices). It is the unique stationary distribution of the random walk.
We will suppose that the eigenvalues are ordered from largest to smallest, so that $\lambda_1 = 1$ and $v_1 = u$. An important value associated with the transition matrix of a Markov chain is its *spectral gap*, defined as $\delta \triangleq 1 - \max_{i=2,\ldots,d} |\lambda_i|$. Such a random walk on an undirected regular graph is always *reversible* and it is also *irreducible* because we have assumed that the graph is connected. The random walk is *aperiodic* in such a case if and only if the spectral gap $\delta$ is positive. In such a case, a long enough random walk in the graph converges to the uniform distribution since for all $\eta > 0$, we have $||M^k v - u|| < \eta$ for $k = \tilde{O}(1/\delta)$, where $v$ is the initial probability distribution.

Finding a marked element by running a Markov chain on the graph just consists in

**Algorithm 1:** *RandomWalk*

**Input**: $G = (\mathcal{E}, \mathcal{V})$, $\mathcal{M} \subset \mathcal{V}$, initial probability distribution $v$
**Output**: An element $e \in \mathcal{M}$

```
Setup : Sample a vertex x according to v and initialise the data structure.
repeat
  Check : if current vertex x is marked then
    return x
else
    repeat
      Update : Take one step of the random walk and update data structure accordingly.
    until x is sampled according to a distribution close enough to the uniform distribution
```

Let $T_s$ be the cost of Setup, $T_c$ be the cost of Check and $T_u$ be the cost of Update. It follows from the preceding considerations that $\tilde{O}(1/\delta)$ steps of the random walk are sufficient to sample $x$ according to the uniform distribution. Furthermore, if we note $\varepsilon := \frac{|\mathcal{M}|}{|\mathcal{V}|}$ the proportion of marked elements, it is readily seen that the algorithm ends after $O(1/\varepsilon)$ iterations of the outer loop. Thus the complexity of classical random walk is $T_s + \frac{1}{\varepsilon}\left(T_c + \frac{1}{\delta}T_u\right)$.

Several quantum versions of random walk algorithms have been proposed by many authors, notably Ambainis [1], Szegedy [28], and Magniez, Nayak, Roland and Santha [17]. A survey of these results can be found in [24]. We use here the following result

**Theorem 1 ([17]).** *Let $M$ be an aperiodic, irreducible and reversible Markov chain on a graph with spectral gap $\delta$, and $\varepsilon := \frac{|\mathcal{M}|}{|\mathcal{V}|}$ as above. Then there is a quantum walk algorithm that finds an element in $\mathcal{M}$ with cost*

$$\boxed{T_s + \frac{1}{\sqrt{\varepsilon}}\left(T_c + \frac{1}{\sqrt{\delta}}T_u\right)} \tag{1}$$

**Johnson graphs and product graphs.** With the exception of Grover's search algorithm seen as a quantum walk algorithm, to date an overwhelming majority of quantum walk algorithms are based on Johnson graphs or a variant thereof. The decoding algorithms which shall be presented in this paper rely on cartesian products of Johnson graphs. All of these objects are defined in this section and some important properties are mentioned.

**Definition 1 (Johnson graphs).** *A Johnson graph $J(n, r)$ is an undirected graph whose vertices are the subsets containing $r$ elements of a set of size $n$, with an edge between two vertices $S$ and $S'$ iff $|S \cap S'| = r - 1$. In other words, $S$ is adjacent to $S'$ if $S'$ can be obtained from $S$ by removing an element and adding a new element in its place.*

It is clear that $J(n, r)$ has $\binom{n}{r}$ vertices and is $r(n-r)$-regular. Its spectral gap is given by

$$\delta = \frac{n}{r(n-r)}. \tag{2}$$

**Definition 2 (Cartesian product of graphs).** *Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two graphs. Their cartesian product $G_1 \times G_2$ is the graph $G = (\mathcal{V}, \mathcal{E})$ where:*

1. $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$, *i.e.* $\mathcal{V} = \{v_1 v_2 \mid v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2\}$
2. $\mathcal{E} = \{(u_1 u_2, v_1 v_2) \mid (u_1 = v_1 \wedge (u_2, v_2) \in \mathcal{E}_2) \vee ((u_1, v_1) \in \mathcal{E}_1 \wedge u_2 = v_2)\}$

The spectral gap of products of Johnson graphs is given by

**Theorem 2 (Cartesian product of Johnson graphs).** *Let $J(n, r) = (\mathcal{V}, \mathcal{E})$, $m \in \mathbb{N}$ and $J^m(n, r) := \times_{i=1}^{m} J(n, r) = (\mathcal{V}_m, \mathcal{E}_m)$. Then:*

1. *$J^m(n, r)$ has $\binom{n}{r}^m$ vertices and is $md$-regular where $d = r(n-r)$.*

2. *We will write $\delta(J)$ resp. $\delta(J^m)$ for the spectral gaps of $J(n,r)$ resp. $J^m(n,r)$. Then: $\delta(J^m) \geq \frac{1}{m}\delta(J)$.*
3. *The random walk associated with $J^m(n,r)$ is aperiodic, irreducible and reversible for all positive $m$, $n$ and $r < n$.*

For a proof of this statement, see the appendix.

## 3 Generalities on classical and quantum decoding

We first recall how the simplest ISD algorithm [23] and its quantised version [5] work and then give a skeleton of the structure of more sophisticated classical and quantum versions.

### 3.1 Prange's algorithm and Bernstein's algorithm

Recall that the goal is to find $e$ of weight $w$ given $s^T = He^T$, where $H$ is an $(n-k) \times n$ matrix. In other words, the problem we aim to solve is finding a solution to an underdetermined linear system of $n-k$ equations in $n$ variables and the solution is unique owing to the weight condition. Prange's algorithm is based on the following observation: if it is known that $k$ given components of the error vector are zero, the error positions are among the $n-k$ remaining components. In other words, if we know for sure that the $k$ corresponding variables are not involved in the linear system, then the error vector can be found by solving the resulting linear system of $n-k$ equations in $n-k$ variables in polynomial time.

The hard part is finding a correct size-$k$ set (of indices of the components). Prange's algorithm samples such sets and solves the resulting linear equation until an error vector of weight $w$ is found. The probability for finding such a set is of order $\Omega\left(\frac{\binom{n-k}{w}}{\binom{n}{w}}\right)$ and therefore Prange's algorithm has complexity

$$O\left(\frac{\binom{n}{w}}{\binom{n-k}{w}}\right) = \tilde{O}\left(2^{\alpha_{\text{Prange}}(R,\omega)n}\right)$$

where

$$\alpha_{\text{Prange}}(R,\omega) = H_2(\omega) - (1-R)H_2\left(\frac{\omega}{1-R}\right)$$

by using the well known formula for binomials

$$\binom{n}{w} = \tilde{\Theta}\left(2^{H_2\left(\frac{w}{n}\right)n}\right).$$

Bernstein's algorithm consists in using Grover's algorithm to find a correct size-$k$ set. Indeed, an oracle for checking that a size-$k$ set is correct can be obtained by following the same steps as in Prange's algorithm, i.e. deriving and solving a linear system of $n-k$ equations in $n-k$ variables and returning 1 iff the resulting error vector has weight $w$. Thus the complexity of Bernstein's algorithm is the square root of that of Prange's algorithm, i.e. $\alpha_{\text{Bernstein}} = \frac{\alpha_{\text{Prange}}}{2}$.

### 3.2 Generalised ISD algorithms

More sophisticated classical ISD algorithms [27, 11, 12, 7, 18, 4, 19] generalise Prange's algorithm in the following way: they introduce a new parameter $p$ and allow $p$ error positions inside of the size-$k$ set (henceforth denoted by $\mathscr{S}$). Furthermore, from Dumer's algorithm onwards, a new parameter $\ell$ is introduced and the set $\mathscr{S}$ is taken to be of size $k+\ell$. This event happens with probability $P_{\ell,p} \triangleq \frac{\binom{k+\ell}{p}\binom{n-k-\ell}{w-p}}{\binom{n}{w}}$. The point is that

**Proposition 1.** *Assume that the restriction of $H$ to the columns belonging to the complement of $\mathscr{S}$ is a matrix of full rank, then*

*(i) the restriction $e'$ of the error to $\mathscr{S}$ is a solution to the syndrome decoding problem*

$$H' e'^T = s'^T. \tag{3}$$

*with $H'$ being an $\ell \times (k+\ell)$ binary matrix, $|e'| = p$ and $H'$, $s'$ that can be computed in polynomial time from $\mathscr{S}$, $H$ and $s$;*

*(ii) once we have such an $e'$, there is a unique $e$ whose restriction to $\mathscr{S}$ is equal to $e'$ and which satisfies $He^T = s^T$. Such an $e$ can be computed from $e'$ in polynomial time.*

*Remark:* The condition in this proposition is met with very large probability when $H$ is chosen uniformly at random: it fails to hold with probability which is only $O(2^{-\ell})$.

*Proof.* Without loss of generality assume that $\mathscr{S}$ is given by the $k+\ell$ first positions. By performing Gaussian elimination, we look for a square matrix $U$ such that

$$UH = \begin{pmatrix} H' & 0_\ell \\ H" & I_{n-k-\ell} \end{pmatrix}$$

That such a matrix exists is a consequence of the fact that $H$ restricted to the last $n-k-\ell$ positions is of full rank. Write now $e = (e', e")$ where $e'$ is the word formed by the $k+\ell$ first entries of $e$. Then

$$Us^T = UHe^T = \begin{pmatrix} H' e'^T \\ H" e'^T + e"^T \end{pmatrix}.$$

If we write $Us^T$ as $(s', s")^T$, where $s'^T$ is the vector formed by the $\ell$ first entries of $Us^T$, then we recover $e$ from $e'$ by using the fact that $H" e'^T + e"^T = s"^T$. □

From now on, we denote by $\Sigma$ and $h$ the functions that can be computed in polynomial time that are promised by this proposition, i.e.

$$s' = \Sigma(s, H, \mathscr{S})$$
$$e = h(e')$$

In other words, all these algorithms solve in a first step a new instance of the syndrome decoding problem with different parameters. The difference with the original problem is that if $\ell$ is small, which is the case in general, there is not a single solution anymore. However searching for all (or a large set of them) can be done more efficiently than just brute-forcing over all errors of weight $p$ on the set $\mathscr{S}$. Once a possible solution $e'$ to (3) is found, $e$ is recovered as explained before. The main idea which avoids brute forcing over all possible errors of weight $p$ on $\mathscr{S}$ is to obtain candidates $e'$ by solving an instance of a generalised $k$-sum problem that we define as follows.

*Problem 4 (generalised $k$-sum problem).* Consider an Abelian group $\mathcal{G}$, an arbitrary set $\mathcal{E}$, a map $f$ from $\mathcal{E}$ to $\mathcal{G}$, $k$ subsets $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{k-1}$ of $\mathcal{E}$, another map $g$ from $\mathcal{E}^k$ to $\{0,1\}$, and an element $S \in \mathcal{G}$. Find a solution $(v_0, \ldots, v_{k-1}) \in \mathcal{V}_0 \times \ldots \mathcal{V}_{k-1}$ such that we have at the same time

(i) $f(v_0) + f(v_1) \cdots + f(v_{k-1}) = S$ (subset-sum condition);
(ii) $g(v_0, \ldots, v_{k-1}) = 0$ ($(v_0, \ldots, v_{k-1})$ is a root of $g$).

Dumer's ISD algorithm, for instance, solves the 2-sum problem in the case where

$$\mathcal{G} = \mathbb{F}_2^\ell, \quad \mathcal{E} = \mathbb{F}_2^{k+\ell}, \quad f(v) = H' v^T$$
$$\mathcal{V}_0 = \{(e_0, 0_{(k+\ell)/2}) \in \mathbb{F}_2^{k+\ell} : e_0 \in \mathbb{F}_2^{(k+\ell)/2},\ |e_0| = p/2\}$$
$$\mathcal{V}_1 = \{(0_{(k+\ell)/2}, e_1) \in \mathbb{F}_2^{k+\ell} : e_1 \in \mathbb{F}_2^{(k+\ell)/2},\ |e_1| = p/2\}$$

and $g(v_0, v_1) = 0$ if and only if $e = h(e')$ is of weight $w$ where $e' = v_0 + v_1$. A solution to the 2-sum problem is then clearly a solution to the decoding problem by construction. The point is that the

2-sum problem can be solved in time which is much less than $|\mathcal{V}_0| \cdot |\mathcal{V}_1|$. For instance, this can clearly be achieved in expected time $|\mathcal{V}_0| + |\mathcal{V}_1| + \frac{|\mathcal{V}_0| \cdot |\mathcal{V}_1|}{|\mathcal{G}|}$ and space $|\mathcal{G}|$ by storing the elements $v_0$ of $\mathcal{V}_0$ in a hashtable at the address $f(v_0)$ and then going over all elements $v_1$ of the other set to check whether or not the address $S - f(v_1)$ contains an element. The term $\frac{|\mathcal{V}_0| \cdot |\mathcal{V}_1|}{|\mathcal{G}|}$ accounts for the expected number of solutions of the 2-sum problem when the elements of $\mathcal{V}_0$ and $\mathcal{V}_1$ are chosen uniformly at random in $\mathcal{E}$ (which is the assumption what we are going to make from on). This is precisely what Dumer's algorithm does. Generally, the size of $\mathcal{G}$ is chosen such that $|\mathcal{G}| = \Theta\left(|\mathcal{V}_i|\right)$ and the space and time complexity are also of this order.

Generalised ISD algorithms are thus composed of a loop in which first a set $\mathscr{S}$ is sampled and then an error vector having a certain form, namely with $p$ error positions in $\mathscr{S}$ and $w - p$ error positions outside of $\mathscr{S}$, is sought. Thus, for each ISD algorithm $A$, we will denote by $Search_A$ the algorithm whose exact implementation depends on $A$ but whose specification is always $Search_A : \mathscr{S}, H, s, w, p \rightarrow \{e \mid e \text{ has weight } p \text{ on } \mathscr{S} \text{ and weight } w - p \text{ on } \overline{\mathscr{S}} \text{ and } s^T = He^T\} \cup \{NULL\}$, where $\mathscr{S}$ is a set of indices, $H$ is the parity-check matrix of the code and $s$ is the syndrome of the error we are looking for. The following pseudo-code gives the structure of a generalised ISD algorithm.

**Algorithm 2:** ISD_Skeleton

**Input**: $H$, $s$, $w$, $p$
**Output**: $e$ of weight $w$ such that $s^T = He^T$

```
repeat
    Sample a set of indices 𝒮 ⊂ {1, ..., n}
    e ← Search_A(𝒮, H, s, w, p)
until |e| = w
return e
```

Thus, if we note $P_A$ the probability, dependent on the algorithm $A$, that the sampled set $\mathscr{S}$ is correct and that $A$ finds $e$ [4], and $T_A$ the execution time of the algorithm $Search_A$, the complexity of generalised ISD algorithms is $O\left(\frac{T_A}{P_A}\right)$. To construct generalised quantum ISD algorithms, we use Bernstein's idea of using Grover search to look for a correct set $\mathscr{S}$. However, now each query made by Grover search will take time which is essentially the time complexity of $Search_A$. Consequently, the complexity of generalised quantum ISD algorithms is given by the following formula:

$$O\left(\frac{T_A}{\sqrt{P_A}}\right) = O\left(\sqrt{\frac{T_A^2}{P_A}}\right). \tag{4}$$

An immediate consequence of this formula is that, in order to halve the complexity exponent of a given classical algorithm, we need a quantum algorithm whose search subroutine is "twice" as efficient.

## 4 Solving the generalised 4-sum problem with quantum walks and Grover search

### 4.1 The Shamir-Schroeppel idea

As explained in Section 3, the more sophisticated ISD algorithms solve during the inner step an instance of the generalised $k$-sum problem. The issue is to get a good quantum version of the classical algorithms used to solve this problem. That this task is non trivial can already be guessed from Dumer's algorithm. Recall that it solves the generalised 2-sum problem in time and space complexity $O\left(V\right)$ when $V = |\mathcal{V}_0| = |\mathcal{V}_1| = \Theta(|\mathcal{G}|)$. The problem is that if we wanted a quadratic

[4] In the case of Dumer's algorithm, for instance, even if the restriction of $e$ to $\mathscr{S}$ is of weight $p$, Dumer's algorithm may fail to find it since it does not split evenly on both sides of the bipartition of $\mathscr{S}$.

speedup when compared to the classical Dumer algorithm, then this would require a quantum algorithm solving the same problem in time $O\left(V^{1/2}\right)$, but this seems problematic since naive ways of quantising this algorithm stumble on the problem that the space complexity is a lower bound on the time complexity of the quantum algorithm. This strongly motivates the choice of ways of solving the 2-sum problem by using less memory. This can be done through the idea of Shamir and Schroeppel [25]. Note that the very same idea is also used for the same reason to speed up quantum algorithms for the subset sum problem in [6, Sec. 4]. To explain the idea, suppose that $\mathcal{G}$ factorises as $\mathcal{G} = \mathcal{G}_0 \times \mathcal{G}_1$ where $|\mathcal{G}_0| = \Theta(|\mathcal{G}_1|) = \Theta(|\mathcal{G}|^{1/2})$. Denote for $i \in \{0,1\}$ by $\pi_i$ the projection from $\mathcal{G}$ onto $\mathcal{G}_i$ which to $g = (g_0, g_1)$ associates $g_i$.

The idea is to construct $f(\mathcal{V}_0)$ and $f(\mathcal{V}_1)$ themselves as $f(\mathcal{V}_0) = f(\mathcal{V}_{00}) + f(\mathcal{V}_{01})$ and $f(\mathcal{V}_1) = f(\mathcal{V}_{10}) + f(\mathcal{V}_{11})$ in such a way that the $\mathcal{V}_{ij}$'s are of size $O(V^{1/2})$ and to solve a 4-sum problem by solving various 2-sum problems. In our coding theoretic setting, it will be more convenient to explain everything directly in terms of the 4-sum problem which is given in this case by

*Problem 5.* Assume that $k+\ell$ and $p$ are multiples of 4. Let

$$\begin{aligned}
\mathcal{G} &= \mathbb{F}_2^{\ell}, \quad \mathcal{E} = \mathbb{F}_2^{k+\ell}, \quad f(v) = H'v^T \\
\mathcal{V}_{00} &\triangleq \{(e_{00}, 0_{3(k+\ell)/4}) \in \mathbb{F}_2^{k+\ell} : e_{00} \in \mathbb{F}_2^{(k+\ell)/4},\ |e_{00}| = p/4\} \\
\mathcal{V}_{01} &\triangleq \{(0_{(k+\ell)/4}, e_{01}, 0_{(k+\ell)/2}) \in \mathbb{F}_2^{k+\ell} : e_{01} \in \mathbb{F}_2^{(k+\ell)/4},\ |e_{01}| = p/4\} \\
\mathcal{V}_{10} &\triangleq \{(0_{(k+\ell)/2}, e_{10}, 0_{(k+\ell)/4}) \in \mathbb{F}_2^{k+\ell} : e_{10} \in \mathbb{F}_2^{(k+\ell)/4},\ |e_{10}| = p/4\} \\
\mathcal{V}_{11} &\triangleq \{(0_{3(k+\ell)/4}, e_{11}) \in \mathbb{F}_2^{k+\ell} : e_{11} \in \mathbb{F}_2^{(k+\ell)/4},\ |e_{11}| = p/4\}
\end{aligned}$$

and $S$ be some element in $\mathcal{G}$. Find $(v_{00}, v_{01}, v_{10}, v_{11})$ in $\mathcal{V}_{00} \times \mathcal{V}_{01} \times \mathcal{V}_{10} \times \mathcal{V}_{11}$ such that $f(v_{00}) + f(v_{01}) + f(v_{10}) + f(v_{11}) = S$ and $h(v_{00} + v_{01} + v_{10} + v_{11})$ is of weight $w$.

Let us explain now how the Shamir-Schroeppel idea allows us to solve the 4-sum problem in time $O(V)$ and space $O\left(V^{1/2}\right)$ when the $\mathcal{V}_{ij}$'s are of order $O\left(V^{1/2}\right)$, $|\mathcal{G}|$ is of order $V$ and when $\mathcal{G}$ decomposes as the product of two groups $\mathcal{G}_0$ and $\mathcal{G}_1$ both of size $\Theta\left(V^{1/2}\right)$. The basic idea is to solve for all possible $r \in \mathcal{G}_1$ the following 2-sum problems

$$\pi_1(f(v_{00})) + \pi_1(f(v_{01})) = r \tag{5}$$

$$\pi_1(f(v_{10})) + \pi_1(f(v_{11})) = \pi_1(S) - r \tag{6}$$

Once these problems are solved we are left with $O\left(V^{1/2}V^{1/2}/V^{1/2}\right) = O\left(V^{1/2}\right)$ solutions to the first problem and $O\left(V^{1/2}\right)$ solutions to the second. Taking any pair $(v_{00}, v_{01})$ solution to (5) and $(v_{10}, v_{11})$ solution to (6) yields a 4-tuple which is a partial solution to the 4-sum problem

$$\pi_1(f(v_{00})) + \pi_1(f(v_{01})) + \pi_1(f(v_{10})) + \pi_1(f(v_{11})) = r + \pi_1(S) - r = \pi_1(S).$$

Let $\mathcal{V}_0''$ be the set of all pairs $(v_{00}, v_{01})$ we have found for the first 2-sum problem (5), whereas $\mathcal{V}_1''$ is the set of all solutions to (6). To ensure that $f(v_{00}) + f(v_{01}) + f(v_{10}) + f(v_{11}) = S$ we just have to solve the following 2-sum problem

$$\underbrace{\pi_0(f(v_{00})) + \pi_0(f(v_{01}))}_{f'(v_{00}, v_{01})} + \underbrace{\pi_0(f(v_{10})) + \pi_0(f(v_{11}))}_{f'(v_{10}, v_{11})} = \pi_0(S)$$

and

$$g(v_{00}, v_{01}, v_{10}, v_{11}) = 0$$

where $(v_{00}, v_{01})$ is in $\mathcal{V}_0''$, $(v_{10}, v_{11})$ is in $\mathcal{V}_1''$ and $g$ is the function whose root we want to find for the original 4-sum problem.

This is again of complexity $O\left(V^{1/2}V^{1/2}/V^{1/2}\right) = O\left(V^{1/2}\right)$. Checking a particular value of $r$ takes therefore $O\left(V^{1/2}\right)$ operations. Since we have $\Theta\left(V^{1/2}\right)$ values to check, the total complexity is $O\left(V^{1/2}V^{1/2}\right) = O(V)$, that is the same as before, but we need only $O\left(V^{1/2}\right)$ memory to store all intermediate sets.

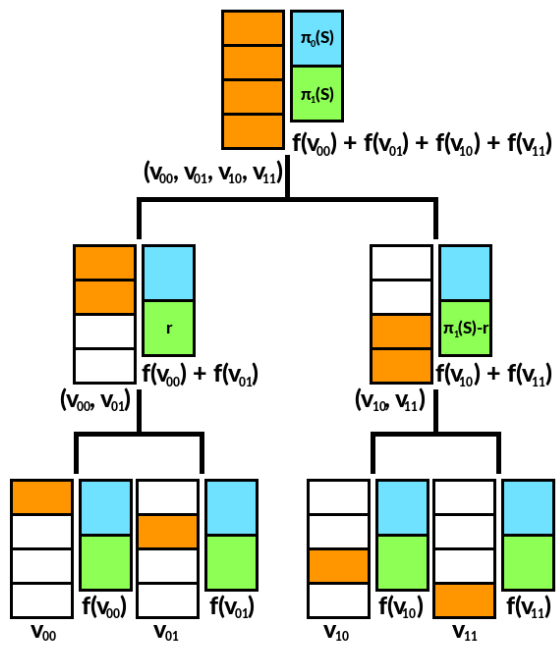


**Fig. 1.** The Shamir-Schroeppel idea in the decoding context (see Problem 5): the support of the elements of $\mathcal{V}_{ij}$ is represented in orange, while the blue and green colours represent $\mathcal{G}_0$ resp. $\mathcal{G}_1$.

### 4.2 A quantum version of the Shamir-Schroeppel algorithm

By following the approach of [6], we will define a quantum algorithm for solving the 4-sum problem by combining Grover search with a quantum walk with a complexity given by

**Proposition 2.** *Consider the generalised* 4*-sum problem with sets* $\mathcal{V}_u$ *of size* $V$*. Assume that* $\mathcal{G}$ *can be decomposed as* $\mathcal{G} = \mathcal{G}_0 \times \mathcal{G}_1$*. There is a quantum algorithm for solving the* 4*-sum problem running in time* $\tilde{O}\left(|\mathcal{G}_1|^{1/2} V^{4/5}\right)$ *as soon as* $|\mathcal{G}_1| = \Omega\left(V^{4/5}\right)$ *and* $|\mathcal{G}| = \Omega\left(V^{8/5}\right)$*.*

This is nothing but the idea of the algorithm [6, Sec. 4] laid out in a more general context. The idea is as in the classical algorithm to look for the right value $r \in \mathcal{G}_1$. This can be done with Grover search in time $O\left(|\mathcal{G}_1|^{1/2}\right)$ instead of $O\left(|\mathcal{G}_1|\right)$ in the classical case. The quantum walk is then used to solve the following problem:

*Problem 6.* Find $(v_{00}, v_{01}, v_{10}, v_{11})$ in $\mathcal{V}_{00} \times \mathcal{V}_{01} \times \mathcal{V}_{10} \times \mathcal{V}_{11}$ such that

$$\begin{aligned} \pi_1(f(v_{00})) + \pi_1(f(v_{01})) &= r \\ \pi_1(f(v_{10})) + \pi_1(f(v_{11})) &= \pi_1(S) - r \\ \pi_0(f(v_{00})) + \pi_0(f(v_{01})) + \pi_0(f(v_{10})) + \pi_0(f(v_{11})) &= \pi_0(S) \\ g(v_{00}, v_{01}, v_{10}, v_{11}) &= 0. \end{aligned}$$

For this, we choose subsets $\mathcal{U}_i$'s of the $\mathcal{V}_i$'s of a same size $U = \Theta\left(V^{4/5}\right)$ and run a quantum walk on the graph whose vertices are all possible 4-tuples of sets of this kind and two 4-tuples $(\mathcal{U}_{00}, \mathcal{U}_{01}, \mathcal{U}_{10}, \mathcal{U}_{11})$ and $(\mathcal{U}'_{00}, \mathcal{U}'_{01}, \mathcal{U}'_{10}, \mathcal{U}'_{11})$ are adjacent if and only if we have for all $i$'s but one $\mathcal{U}'_i = \mathcal{U}_i$ and for the remaining $\mathcal{U}'_i$ and $\mathcal{U}_i$ we have $|\mathcal{U}'_i \cap \mathcal{U}_i| = U - 1$. Notice that this graph is nothing but $J^4(V, U)$. By following [6, Sec. 4] it can be proved that

**Proposition 3.** *Under the assumptions that* $|\mathcal{G}_1| = \Omega\left(V^{4/5}\right)$ *and* $|\mathcal{G}| = \Omega\left(V^{8/5}\right)$*, it is possible to set up a data structure of size* $O(U)$ *to implement this quantum walk such that*
*(i) setting up the data structure takes time* $O(U)$*;*
*(ii) checking whether a new 4-tuple leads to a solution to the problem above (and outputting the solution in this case) takes time* $O(1)$*,*
*(iii) updating the data structure takes time* $O(\log U)$*.*

The proof which we give is adapted from [6, Sec. 4].

*Proof.*

1. **Setting up the data structure takes time $O(U)$.**
The data structure is set up more or less in the same way as in classical Shamir-Schroeppel's algorithm, i.e. by solving two 2-sum problems first and then using the result to solve a third and last 2-sum problem. There are however the following differences:

(i) We no longer keep the results in a hashtable but in some other type of ordered data structure which allows for the insertion, deletion and search operations to be done in $O(\log U)$ time. For instance, [6] chose radix trees. More detail will be given when we look at the UPDATE operation.
(ii) Because we no longer use hashtables, we will need two data structures at each step, one to keep track of $f(v_{00})$ along with the associated $v_{00}$, $f(v_{00}) + f(v_{01})$ along with the associated $(v_{00}, v_{01})$, etc. and another to keep track of $v_{00}$, $(v_{00}, v_{01})$, etc. separately. If we denote the first family of data structures by $\mathcal{D}_f$ and the second family by $\mathcal{D}_\mathcal{V}$, this gives a total of 13 data structures (7 of type $\mathcal{D}_\mathcal{V}$ and 6 of type $\mathcal{D}_f$, because no data structure is needed to store the sum of all four vectors which is simply $S$).

Solving the first two 2-sum problems takes time $|\mathcal{U}_{i0}| + |\mathcal{U}_{i1}| + \frac{|\mathcal{U}_{i0}|.|\mathcal{U}_{i1}|}{|\mathcal{G}_1|}$, $i = 0, 1$, which is $O(U)$ because $|\mathcal{G}_1| = \Omega\left(V^{4/5}\right) = \Omega(U)$. Denote by $\mathcal{U}_0$ resp. $\mathcal{U}_1$ the set of solutions to these two problems. These solutions are used to solve the second 2-sum, problem, which takes time $|\mathcal{U}_0| + |\mathcal{U}_1| + \frac{|\mathcal{U}_0|.|\mathcal{U}_1|}{|\mathcal{G}_0|} = O(U)$ due to $\mathcal{G}_0 = \mathcal{G}/\mathcal{G}_1$ and $|\mathcal{G}| = \Omega\left(V^{8/5}\right)$.

Thus, setting up the data structure takes time $O(U)$.
2. **Updating the data structure takes time $O(\log U)$.**
Recall that the data structures are chosen such that the insertion, deletion and search operations take $O(\log U)$ time, and also that there are two data structures pertaining to each vector or pair of vectors, for a total of 13 data structures.

Recall also that the update step consists in moving from one vertex of the Johnson graph $J^4(V, U)$ to one that is adjacent to it. Suppose, without loss of generality, that we move from the vertex $(\mathcal{U}_{00}, \mathcal{U}_{01}, \mathcal{U}_{10}, \mathcal{U}_{11})$ to $(\mathcal{U}'_{00}, \mathcal{U}_{01}, \mathcal{U}_{10}, \mathcal{U}_{11})$. Thus, a $v_{00} \in \mathcal{U}_{00}$ has been replaced by a $u_{00} \in \mathcal{U}_{00}$.

Then, the low cost of the update step relies upon the following fundamental insight: there are in all $U$ possible ways of writing the sum $\pi_1(f(u_{00})) + \pi_1(f(v_{01}))$ (one for each $v_{01} \in \mathcal{U}_{01}$). But we have one further constraint which is that this sum needs to be equal to a given $r \in \mathcal{G}_1$. Thus, there are on average $\frac{|\mathcal{U}_{ij}|}{|\mathcal{G}_1|} = O(1)$ values of $v_{01} \in \mathcal{U}_{01}$ which give a solution.

Note that the same argument applies for the number of $(v_{10}, v_{11}) \in \mathcal{U}_1$ that fulfil the condition

$$\pi_0(f(u_{00})) + \pi_0(f(v_{01})) + \pi_0(f(v_{10})) + \pi_0(f(v_{11})) \;=\; \pi_0(S)$$

for a given $(u_{00}, v_{01}) \in \mathcal{U}_0$ (where $\pi_0(S) \in \mathcal{G}_0$), for in this case there are on average $\frac{|\mathcal{U}_0|}{|\mathcal{G}_0|} = O(1)$ such elements.

This allows us to proceed as follows: we impose a constant limit on the number of $v_{01} \in \mathcal{U}_{01}$ that correspond to a given $u_{00} \in \mathcal{U}_{00}$ at each update operation. A similar limit is imposed on the number of $(v_{10}, v_{11}) \in \mathcal{U}_1$. The probability of reaching this limit is negligeable, and if it is reached, we re-initialise the data structure, so this does not modify the overall complexity of the algorithm. Note also that there is no problem when the opposite situation happens, i.e. when there are no $v_{01} \in \mathcal{U}_{01}$ corresponding to a given $u_{00} \in \mathcal{U}_{00}$. Indeed, while this may result in the data structure being depleted, this is a temporary situation and the data structure will be refilled over time as more suitable elements occur.

We now enumerate the steps needed to update the data structure. What we need to do is to remove the old element $v_{00}$ and everything that has been constructed using it, and add $u_{00}$ and everything that it allows to construct (within the limits discussed above). First, to remove $v_{00}$ and the other elements it affects, we need to do the following:
(a) Find and delete $v_{00}$ from the data structure $\mathcal{D}_{\mathcal{U}_{00}}$.
(b) Calculate $f(v_{00})$, then find and delete it from the data structure $\mathcal{D}_{f_{00}}$.
(c) Find at most a constant number of $(v_{00}, v_{01})$ in $\mathcal{D}_{\mathcal{U}_0}$ and remove them.
(d) For each of these $(v_{00}, v_{01})$, calculate $f(v_{00}) + f(v_{01})$ and remove it from $\mathcal{D}_{f_0}$.
(e) Find at most a constant number of $(v_{00}, v_{01}, v_{01}, v_{11})$ in $\mathcal{D}_{\mathcal{U}}$ and remove them.
This step uses operations of negligeable cost (calculating $f(v_{00})$, etc.) and the number of operations of cost $\log(U)$ which it uses is bounded by a constant. Thus it takes time $O(\log(U))$.

To add $u_{00}$ and other new elements depending on it, we proceed as follows:
(a) Insert $u_{00}$ in $\mathcal{D}_{\mathcal{U}_{00}}$.
(b) Calculate $f(u_{00})$, then insert it in $\mathcal{D}_{f_{00}}$.
(c) Calculate $x = r - \pi_1(f(u_{00}))$ and find if there are elements $y$ in $\mathcal{D}_{f_{01}}$ such that $\pi_1(y) = x$. For a constant number of associated $v_{01}$, insert $(u_{00}, v_{01})$ in $\mathcal{D}_{\mathcal{U}_0}$ and in $\mathcal{D}_{f_0}$ associated with $r$.
(d) Similarly there are a constant number of $(v_{01}, v_{11})$ that need to be updated, for those calculate $g(u_{00}, v_{01}, v_{10}, v_{11})$. If it is equal to zero, insert $(v_{00}, v_{01}, v_{10}, v_{11})$ in $\mathcal{D}_{\mathcal{U}}$.
It is easy to see that this step also takes time $O(\log(U))$.

3. **Checking whether a new 4-tuple leads to a solution of the problem takes time $O(1)$.**
Checking that the right 4-tuple is in $\mathcal{D}_{\mathcal{U}}$ requires looking for it in $\mathcal{D}_{\mathcal{U}}$ at the first step of the algorithm. This costs $O(\sqrt{U})$ using Grover search. At the following steps of the algorithm, it is enough to check the new elements (whose number is bounded by a constant) that have been added to $\mathcal{D}_{\mathcal{U}}$. So the checking cost is $O(1)$ overall. □

Proposition 2 is essentially a corollary of this proposition.

*Proof (Proof of Proposition 2).* Recall that the cost of the quantum walk is given by $T_s + \frac{1}{\sqrt{\varepsilon}}\left(T_c + \frac{1}{\sqrt{\delta}}T_u\right)$ where $T_s, T_c, T_u, \varepsilon$ and $\delta$ are the setup cost, the check cost, the update cost, the proportion of marked elements and the spectral gap of the quantum walk. From Proposition 3, we know that $T_s = O(U) = O\left(V^{4/5}\right)$, $T_c = O(1)$, and $T_u = O(\log U)$. Recall that the spectral gap of $J(V, U)$ is equal to $\frac{V}{U(V-U)}$ by (2). This quantity is larger than $\frac{1}{U}$ and by using Theorem 2 on the cartesian product of Johnson graphs, we obtain $\delta = \Theta\left(\frac{1}{U}\right)$.

Now for the proportion of marked elements we argue as follows. If Problem 6 has a solution $(v_{00}, v_{01}, v_{10}, v_{11})$, then the probability that each of the sets $\mathcal{U}_i$ contains $v_i$ is precisely $U/V = \Theta\left(V^{-1/5}\right)$. The probability $\varepsilon$ that all the $\mathcal{U}_i$'s contain $v_i$ is then $\Theta\left(V^{-4/5}\right)$. This gives a total cost of

$$O\left(V^{4/5}\right) + O\left(V^{2/5}\right)\left(O(1) + O\left(V^{2/5}\right)O(\log U)\right) = \tilde{O}\left(V^{4/5}\right).$$

When we multiply this by the cost of Grover's algorithm for finding the right $r$ we have the aforementioned complexity. □

### 4.3 Application to the decoding problem

When applying this approach to the decoding problem we obtain

**Theorem 3.** *We can decode $w = \omega n$ errors in a random linear code of length $n$ and rate $R = \frac{k}{n}$ with a quantum complexity of order $\tilde{O}\left(2^{\alpha_{SSQW}(R,\omega)n}\right)$ where*

$$\alpha_{SSQW}(R,\omega) \triangleq \min_{(\pi,\lambda)\in\mathcal{R}} \left( \frac{H_2(\omega) - (1-R-\lambda)H_2\left(\frac{\omega-\pi}{1-R-\lambda}\right) - \frac{2}{5}(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right)}{2} \right)$$

$$\mathcal{R} \triangleq \left\{ (\pi,\lambda)\in[0,\omega]\times[0,1) \colon \lambda = \frac{2}{5}(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right), \pi \leq R+\lambda, \lambda \leq 1-R-\omega+\pi \right\}$$

*Proof.* Recall (see (4)) that the quantum complexity is given by

$$\tilde{O}\left(\frac{T_{\mathrm{SSQW}}}{\sqrt{P_{\mathrm{SSQW}}}}\right) \tag{7}$$

where $T_{\mathrm{SSQW}}$ is the complexity of the combination of Grover's algorithm and quantum walk solving the generalised 4-sum problem specified in Problem 6 and $P_{\mathrm{SSQW}}$ is the probability that the random set of $k+\ell$ positions $\mathscr{S}$ and its random partition in 4 sets of the same size that are chosen is such that all four of them contain exactly $p/4$ errors. Note that $p$ and $\ell$ are chosen such that $k+\ell$ and $p$ are divisible by 4. $P_{\mathrm{SSQW}}$ is given by

$$P_{\mathrm{SSQW}} = \frac{\binom{\frac{k+\ell}{4}}{\frac{p}{4}}^4 \binom{n-k-\ell}{w-p}}{\binom{n}{w}}$$

Therefore

$$(P_{\mathrm{SSQW}})^{-1/2} = \tilde{O}\left(2^{\frac{H_2(\omega)-(1-R-\lambda)H_2\left(\frac{\omega-\pi}{1-R-\lambda}\right)-(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right)}{2}n}\right) \tag{8}$$

where $\lambda \triangleq \frac{\ell}{n}$ and $\pi \triangleq \frac{p}{n}$. $T_{\mathrm{SSQW}}$ is given by Proposition 2:

$$T_{\mathrm{SSQW}} = \tilde{O}\left(|\mathcal{G}_1|^{1/2} V^{4/5}\right)$$

where the sets involved in the generalised 4-sum problem are specified in Problem 6. This gives

$$V = \binom{\frac{k+\ell}{4}}{\frac{p}{4}}$$

We choose $\mathcal{G}_1$ as

$$\mathcal{G}_1 = \mathbb{F}_2^{\lceil \frac{\ell}{2} \rceil} \tag{9}$$

and the assumptions of Proposition 2 are verified as soon as

$$2^\ell = \Omega\left(V^{8/5}\right).$$

which amounts to

$$2^\ell = \Omega\left(\binom{\frac{k+\ell}{4}}{\frac{p}{4}}^{8/5}\right)$$

This explains the condition

$$\lambda = \frac{2}{5}(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right) \tag{10}$$

found in the definition of the region $\mathcal{R}$. With the choices (9) and (10), we obtain

$$\begin{aligned} T_{\mathrm{SSQW}} &= \tilde{O}\left(V^{6/5}\right) \\ &= \tilde{O}\left(2^{\frac{3}{10}(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right)n}\right) \end{aligned} \tag{11}$$

Substituting for $P_{\mathrm{SSQW}}$ and $T_{\mathrm{SSQW}}$ the expressions given by (8) and (11) finishes the proof of the theorem. □

## 5 Improvements obtained by the representation technique and "1 + 1 = 0"

There are two techniques that can be used to speed up the quantum algorithm of the previous section.

*The representation technique.* It was introduced in [15] to speed up algorithms for the subset-sum algorithm and used later on in [18] to improve decoding algorithms. The basic idea of the representation technique in the context of the subset-sum or decoding algorithms consists in (i) changing slightly the underlying (generalised) $k$-sum problem which is solved by introducing sets $\mathcal{V}_i$ for which there are (exponentially) many solutions to the problem $\sum_i f(v_i) = S$ by using redundant representations, (ii) noticing that this allows us to put additional subset-sum conditions on the solution.

In the decoding context, instead of considering sets of errors with non-overlapping support, the idea that allows us to obtain many different representations of a same solution is just to consider sets $\mathcal{V}_i$ corresponding to errors with overlapping supports. In our case, we could have taken instead of the four sets defined in the previous section the following sets

$$\begin{aligned} \mathcal{V}_{00} = \mathcal{V}_{10} &\triangleq \{(e_{00}, 0_{(k+\ell)/2}) \in \mathbb{F}_2^{k+\ell} : e_{00} \in \mathbb{F}_2^{(k+\ell)/2},\ |e_{00}| = p/4\} \\ \mathcal{V}_{01} = \mathcal{V}_{11} &\triangleq \{(0_{(k+\ell)/2}, e_{01}) \in \mathbb{F}_2^{k+\ell} : e_{01} \in \mathbb{F}_2^{(k+\ell)/2},\ |e_{01}| = p/4\} \end{aligned}$$

Clearly a vector $e$ of weight $p$ can be written in many different ways as a sum $v_{00} + v_{01} + v_{10} + v_{11}$ where $v_{ij}$ belongs to $\mathcal{V}_{ij}$. This is (essentially) due to the fact that a vector of weight $p$ can be written in $\binom{p}{p/2} = \tilde{O}\left(2^p\right)$ ways as a sum of two vectors of weight $p/2$.

The point is that if we apply now the same algorithm as in the previous section and look for solutions to Problem 5, there is not a single value of $r$ that leads to the right solution. Here, about $2^p$ values of $r$ will do the same job. The speedup obtained by the representation technique is a consequence of this phenomenon. We can even improve on this representation technique by using the $1 + 1 = 0$ phenomenon as in [4].

*The "$1 + 1 = 0$" phenomenon.* Instead of choosing the $\mathcal{V}_i$'s as explained above we will actually choose the $\mathcal{V}_i$'s as

$$\mathcal{V}_{00} = \mathcal{V}_{10} \triangleq \{(e_{00}, 0_{(k+\ell)/2}) \in \mathbb{F}_2^{k+\ell} : e_{00} \in \mathbb{F}_2^{(k+\ell)/2},\ |e_{00}| = \frac{p}{4} + \frac{\Delta p}{2}\} \tag{12}$$

$$\mathcal{V}_{01} = \mathcal{V}_{11} \triangleq \{(0_{(k+\ell)/2}, e_{01}) \in \mathbb{F}_2^{k+\ell} : e_{01} \in \mathbb{F}_2^{(k+\ell)/2},\ |e_{01}| = \frac{p}{4} + \frac{\Delta p}{2}\} \tag{13}$$

A vector $e$ of weight $p$ in $\mathbb{F}_2^{k+\ell}$ can indeed by represented in many ways as a sum of 2 vectors of weight $\frac{p}{2} + \Delta p$. More precisely, such a vector can be represented in $\binom{p}{p/2}\binom{k+\ell-p}{\Delta p}$ ways. Notice that this number of representations is greater than the number $2^p$ that we had before. This explains why choosing an appropriate positive value $\Delta p$ allows us to improve on the previous choice.

The quantum algorithm for decoding follows the same pattern as in the previous section: (i) we look with Grover's search algorithm for a right set $\mathscr{S}$ of $k+\ell$ positions such that the restriction $e'$ of the error $e$ we look for is of weight $p$ on this subset and then (ii) we search for $e'$ by solving a generalised 4-sum problem with a combination of Grover's algorithm and a quantum walk. We will use for the second point the following proposition which quantifies how much we gain when there are multiple representations/solutions:

**Proposition 4.** *Consider the generalised 4-sum problem with sets $\mathcal{V}_u$ of size $O(V)$. Assume that $\mathcal{G}$ can be decomposed as $\mathcal{G} = \mathcal{G}_0 \times \mathcal{G}_1 \times \mathcal{G}_2$. Furthermore assume that there are $\Omega(|\mathcal{G}_2|)$ solutions to the 4-sum problem and that we can fix arbitrarily the value $\pi_2(f(v_{00}) + f(v_{01}))$ of a solution to the 4-sum problem, where $\pi_2$ is the mapping from $\mathcal{G} = \mathcal{G}_0 \times \mathcal{G}_1 \times \mathcal{G}_2$ to $\mathcal{G}_2$ which maps $(g_0, g_1, g_2)$ to $g_2$. There is a quantum algorithm for solving the 4-sum problem running in time $\tilde{O}\left(|\mathcal{G}_1|^{1/2} V^{4/5}\right)$ as soon as $|\mathcal{G}_1| \cdot |\mathcal{G}_2| = \Omega\left(V^{4/5}\right)$ and $|\mathcal{G}| = \Omega\left(V^{8/5}\right)$.*

*Proof.* Let us first introduce a few notations. We denote by $\pi_{12}$ the "projection" from $\mathcal{G} = \mathcal{G}_0 \times \mathcal{G}_1 \times \mathcal{G}_2$ to $\mathcal{G}_1 \times \mathcal{G}_2$ which associates to $(g_0, g_1, g_2)$ the pair $(g_1, g_2)$ and by $\pi_0$ the projection from $\mathcal{G}$ to $\mathcal{G}_0$ which maps $(g_0, g_1, g_2)$ to $g_0$. As in the previous section, we solve with a quantum walk the following problem: we fix an element $r = (r_1, r_2)$ in $\mathcal{G}_1 \times \mathcal{G}_2$ and find (if it exists) $(v_{00}, v_{01}, v_{10}, v_{11})$ in $\mathcal{V}_{00} \times \mathcal{V}_{01} \times \mathcal{V}_{10} \times \mathcal{V}_{11}$ such that

$$\begin{aligned}
\pi_{12}(f(v_{00})) + \pi_{12}(f(v_{01})) &= r \\
\pi_{12}(f(v_{10})) + \pi_{12}(f(v_{11})) &= \pi_{12}(S) - r \\
\pi_0(f(v_{00})) + \pi_0(f(v_{01})) + \pi_0(f(v_{10})) + \pi_0(f(v_{11})) &= \pi_0(S) \\
g(v_{00}, v_{01}, v_{10}, v_{11}) &= 0.
\end{aligned}$$

The difference with Proposition 2 is that we do not check all possibilities for $r$ but just all possibilities for $r_1 \in \mathcal{G}_1$ and fix $r_2$ arbitrarily. As in Proposition 2, we perform a quantum walk whose complexity is $\tilde{O}\left(V^{4/5}\right)$ to solve the aforementioned problem for a fixed $r$. What remains to be done is to find the right value for $r_1$ which is achieved by a Grover search with complexity $O\left(|\mathcal{G}_1|^{1/2}\right)$. □

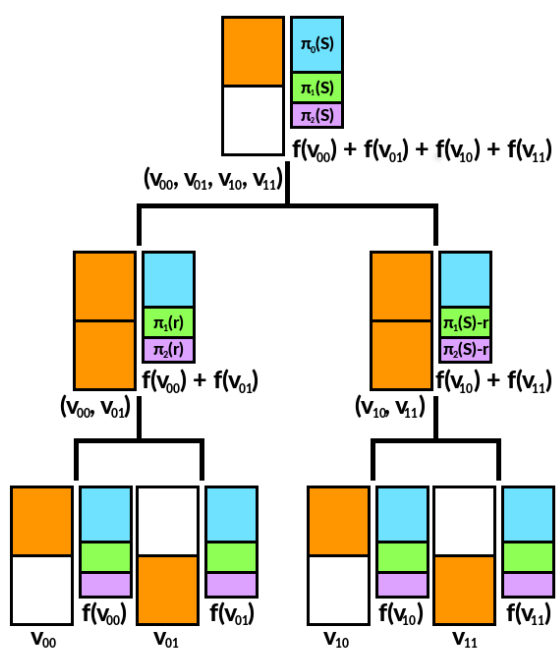


**Fig. 2.** The representation technique: the support of the elements of $\mathcal{V}_{ij}$ is represented in orange, while the blue, green and violet colours represent $\mathcal{G}_0$ resp. $\mathcal{G}_1$, resp. $\mathcal{G}_2$.

By applying Proposition 4 in our decoding context, we obtain

**Theorem 4.** *We can decode $w = \omega n$ errors in a random linear code of length $n$ and rate $R = \frac{k}{n}$ with a quantum complexity of order $\tilde{O}\left(2^{\alpha_{MMTQW}(R,\omega)n}\right)$ where*

$$\begin{aligned}
\alpha_{MMTQW}(R,\omega) &\triangleq \min_{(\pi,\Delta\pi,\lambda)\in\mathcal{R}} \left(\frac{\beta(R,\lambda,\pi,\Delta\pi)+\gamma(R,\lambda,\pi,\omega)}{2}\right) \\
&\textit{with} \\
\beta(R,\lambda,\pi,\Delta\pi) &\triangleq \frac{6}{5}(R+\lambda)H_2\left(\frac{\pi/2+\Delta\pi}{R+\lambda}\right) - \pi - (1-R-\lambda)H_2\left(\frac{\Delta\pi}{1-R-\lambda}\right), \\
\gamma(R,\lambda,\pi,\omega) &\triangleq H_2(\omega) - (1-R-\lambda)H_2(\frac{\omega-\pi}{1-R-\lambda}) - (R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right)
\end{aligned}$$

*where $\mathcal{R}$ is the subset of elements $(\pi,\Delta\pi,\lambda)$ of $[0,\omega]\times[0,1)\times[0,1)$ that satisfy the following constraints*

$$\begin{aligned}
0 \le \Delta\pi &\le R+\lambda-\pi \\
0 \le \ \pi \ &\le \min(\omega, R+\lambda) \\
0 \le \ \lambda \ &\le 1-R-\omega+\pi \\
\pi \ &= 2\left((R+\lambda)H_2^{-1}\left(\frac{5\lambda}{4(R+\lambda)}\right) - \Delta\pi\right)
\end{aligned}$$

*Proof.* The algorithm picks random subsets $\mathscr{S}$ of size $k+\ell$ with the hope that the restriction to $\mathscr{S}$ of the error of weight $w$ that we are looking for is of weight $p$. Then it solves for each of these subsets the generalised 4-sum problem where the sets $\mathcal{V}_{ij}$ are specified in (12) and (13), and $\mathcal{G}$, $\mathcal{E}$, $f$ and $g$ are as in Problem 6. $g$ is in this case slightly more complicated for the sake of analysing the algorithm. We have $g(v_{00},v_{01},v_{10},v_{11}) = 0$ if and only if (i) $v_{00}+v_{01}+v_{10}+v_{11}$ is of weight $p$ (this is the additional constraint we use for the analysis of the algorithm) (ii) $f(v_{00})+f(v_{01})+f(v_{10})+f(v_{11}) = \Sigma(s,H,\mathscr{S})$ and (iii) $h(v_{00}+v_{01}+v_{10}+v_{11})$ is of weight $w$.

From (4) we know that the quantum complexity is given by

$$\tilde{O}\left(\frac{T_{\text{MMTQW}}}{\sqrt{P_{\text{MMTQW}}}}\right) \tag{14}$$

where $T_{\text{MMTQW}}$ is the complexity of the combination of Grover's algorithm and quantum walk solving the generalised 4-sum problem specified above and $P_{\text{MMTQW}}$ is the probability that the restriction $e'$ of the error $e$ to $\mathscr{S}$ is of weight $p$ and that this error can be written as $e' = v_{00}+v_{01}+v_{10}+v_{11}$ where the $v_{ij}$ belong to $\mathcal{V}_{ij}$. It is readily verified that

$$P_{\text{MMTQW}} = \tilde{O}\left(\frac{\binom{k+\ell}{p}\binom{n-k-\ell}{w-p}}{\binom{n}{w}}\right)$$

By using asymptotic expansions of the binomial coefficients we obtain

$$(P_{\text{MMTQW}})^{-1/2} = \tilde{O}\left(2^{\frac{H_2(\omega)-(1-R-\lambda)H_2\left(\frac{\omega-\pi}{1-R-\lambda}\right)-(R+\lambda)H_2\left(\frac{\pi}{R+\lambda}\right)}{2}n}\right) \tag{15}$$

where $\lambda \triangleq \frac{\ell}{n}$ and $\pi \triangleq \frac{p}{n}$. To estimate $T_{\text{SSQW}}$, we can use Proposition 4. The point is that the number of different solutions of the generalised 4-sum problem (when there is one) is of order

$$\tilde{\Omega}\left(\binom{p}{p/2}\binom{k+\ell-p}{\Delta p}\right).$$

At this point, we observe that

$$\log_2\left(\binom{p}{p/2}\binom{k+\ell-p}{\Delta p}\right) = p + (k+\ell-p)H_2\left(\frac{\Delta p}{k+\ell-p}\right) + o(n)$$

when $p$, $\Delta p$, $\ell$, $k$ are all linear in $n$. In other words, we may use Proposition 4 with $\mathcal{G}_2 = \mathbb{F}_2^{\ell_2}$ with

$$\ell_2 \triangleq p + (k+\ell-p) H_2\left(\frac{\Delta p}{k+\ell-p}\right). \tag{16}$$

We use now Proposition 4 with $\mathcal{G}_2$ chosen as explained above. $V$ is given in this case by

$$V = \binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\Delta p}{2}} = \tilde{O}\left(2^{\frac{(R+\lambda)H_2\left(\frac{\pi/2+\Delta\pi}{R+\lambda}\right)n}{2}}\right)$$

where $\Delta\pi \triangleq \frac{\Delta p}{n}$. We choose the size of $\mathcal{G}$ such that

$$|\mathcal{G}| = \tilde{\Theta}\left(V^{8/5}\right) \tag{17}$$

which gives

$$2^{\ell} = \tilde{\Theta}\left(\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\Delta p}{2}}^{8/5}\right).$$

This explains why we impose

$$\lambda = \frac{8}{5}\frac{R+\lambda}{2} H_2\left(\frac{\pi/2+\Delta\pi}{R+\lambda}\right)$$

which is equivalent to the condition

$$\frac{5\lambda}{4(R+\lambda)} = H_2\left(\frac{\pi/2+\Delta\pi}{R+\lambda}\right)$$

which in turn is equivalent to the condition

$$\pi = 2\left((R+\lambda)H_2^{-1}\left(\frac{5\lambda}{4(R+\lambda)}\right) - \Delta\pi\right) \tag{18}$$

found in the definition of the region $\mathcal{R}$. The size of $\mathcal{G}_1$ is chosen such that

$$|\mathcal{G}_1| \cdot |\mathcal{G}_2| = \mathbb{F}_2^{\lceil \frac{\ell}{2} \rceil}. \tag{19}$$

By using (16) and (17), this implies

$$|\mathcal{G}_1| = \tilde{\Theta}\left(\frac{V^{4/5}}{2^{p+(k+\ell-p)H_2\left(\frac{\Delta p}{k+\ell-p}\right)}}\right) \tag{20}$$

With the choices (19) and (18), we obtain

$$\begin{aligned} T_{\mathrm{MMTQW}} &= \tilde{O}\left(|\mathcal{G}_1|^{1/2} \cdot V^{4/5}\right) \\ &= \tilde{O}\left(\frac{V^{6/5}}{2^{\frac{p}{2}+\frac{k+\ell-p}{2}H_2\left(\frac{\Delta p}{k+\ell-p}\right)}}\right) \\ &= \tilde{O}\left(2^{\left[\frac{3}{5}(R+\lambda)H_2\left(\frac{\pi/2+\Delta\pi}{R+\lambda}\right)-\frac{\pi}{2}-\frac{R+\lambda-\pi}{2}H_2\left(\frac{\Delta\pi}{R+\lambda-\pi}\right)\right]n}\right) \end{aligned} \tag{21}$$

Substituting for $P_{\mathrm{MMTQW}}$ and $T_{\mathrm{MMTQW}}$ the expressions given by (15) and (21) finishes the proof of the theorem. □

## 6 Computing the complexity exponents

We used the software SageMath to numerically find the minima giving the complexity exponents in Theorems 3 and 4 using golden section search and a recursive version thereof for two parameters. We compare in Figure 3 the exponents $\alpha_{\mathrm{Bernstein}}(R, \omega_{\mathrm{GV}})$, $\alpha_{SSQW}(R, \omega_{\mathrm{GV}})$ and $\alpha_{MMTQW}(R, \omega_{\mathrm{GV}})$ that we have obtained with our approach. It can be observed that there is some improvement upon $\alpha_{\mathrm{Bernstein}}$ with both algorithms especially in the range of rates between 0.3 and 0.7.

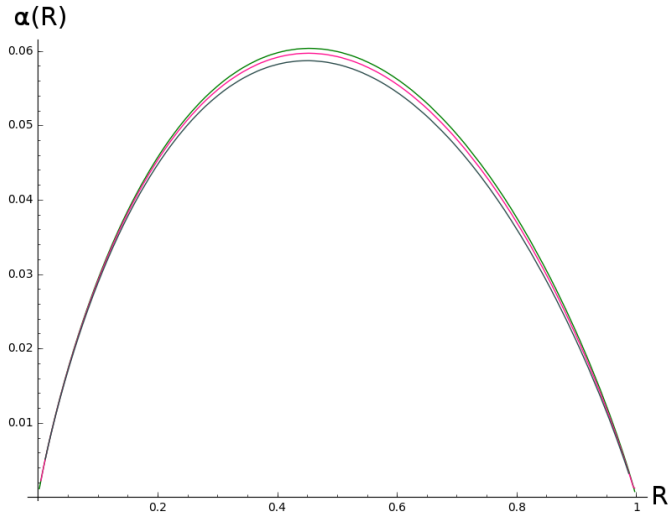


**Fig. 3.** $\alpha_{\text{Bernstein}}$ in green, $\alpha_{SSQW}$ in pink, $\alpha_{MMTQW}$ in grey.

## 7 Concluding remarks

One may wonder why our best algorithm is a version of MMT's algorithm and not BJMM's algorithm or May and Ozerov's algorithm. We did try to quantise BJMM's algorithm, but it turned out to have worse time complexity than MMT's algorithm (for more details, see [16]). This seems to be due to space complexity constraints. Space complexity is indeed a lower bound on the quantum complexity of the algorithm. It has actually been shown [3, Chap. 10, Sec. 3] that BJMM's algorithm uses more space than MMT's algorithm, even when it is optimised to use the least amount of space. Moreover, it is rather insightful that in all cases, the best quantum algorithms that we have obtained here are not direct quantised versions of the original Dumer or MMT's algorithms but quantised versions of modified versions of these algorithms that use less memory than the original algorithms.

The case of the May and Ozerov algorithm is also intriguing. Again the large space complexity of the original version of this algorithm makes it a very challenging task to obtain a "good" quantised version of it.

Finally, it should be noticed that while sophisticated techniques such as MMT, BJMM [18, 4] or May and Ozerov [19] have managed to improve rather significantly upon the most naive ISD algorithm, namely Prange's algorithm [23], the improvement that we obtain with more sophisticated techniques is much more modest when we consider our improvements of the quantised version of the Prange algorithm [5]. Moreover, the improvements we obtain on the exponent $\alpha_{\text{Bernstein}}(R, \omega)$ are smaller when $\omega$ is smaller than $\omega_{\text{GV}}$. Considering the techniques for proving that the exponent of classical ISD algorithms goes to the Prange exponent when the relative error weight goes to 0 [9], we conjecture that it should be possible to prove that we actually have $\lim_{\omega \to 0^+} \frac{\alpha_{\text{MMTQW}}(R,\omega)}{\alpha_{\text{Bernstein}}(R,\omega)} = 1$.

## References

1. Ambainis, A. Quantum walk algorithm for element distinctness. *SIAM J. Comput. 37* (2007), 210–239.
2. Barg, A. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity* (Oct. 1997).
3. Becker, A. *The representation technique, Applications to hard problems in cryptography.* PhD thesis, Université Versailles Saint-Quentin en Yvelines, Oct. 2012.

4. Becker, A., Joux, A., May, A., and Meurer, A. Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012* (2012), Lecture Notes in Comput. Sci., Springer.
5. Bernstein, D. J. Grover vs. McEliece. In *Post-Quantum Cryptography 2010* (2010), N. Sendrier, Ed., vol. 6061 of *Lecture Notes in Comput. Sci.*, Springer, pp. 73–80.
6. Bernstein, D. J., Jeffery, S., Lange, T., and Meurer, A. Quantum algorithms for the subset-sum problem. In *Post-Quantum Cryptography 2011* (Limoges, France, June 2013), vol. 7932 of *Lecture Notes in Comput. Sci.*, pp. 16–33.
7. Bernstein, D. J., Lange, T., and Peters, C. Smaller decoding exponents: ball-collision decoding. In *Advances in Cryptology - CRYPTO 2011* (2011), vol. 6841 of *Lecture Notes in Comput. Sci.*, pp. 743–760.
8. Boyer, M., Brassard, G., Høyer, P., and Tapp, A. Tight bounds on quantum searching. *Fortsch. Phys. 46* (1998), 493.
9. Canto-Torres, R., and Sendrier, N. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016* (Fukuoka, Japan, Feb. 2016), Lecture Notes in Comput. Sci., pp. 144–161.
10. Cvetković, D. M., Doob, M., and Sachs, H. *Spectra of graphs : theory and application.* New York : Academic Press, 1980.
11. Dumer, I. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory* (Moscow, 1991), pp. 50–52.
12. Finiasz, M., and Sendrier, N. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2009* (2009), M. Matsui, Ed., vol. 5912 of *Lecture Notes in Comput. Sci.*, Springer, pp. 88–105.
13. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual ACM Symposium on the Theory of Computation* (New York, NY, 1996), ACM Press, New York, pp. 212–219.
14. Grover, L. K. Quantum computers can search arbitrarily large databases by a single query. *Phys. Rev. Lett. 79* (1997), 4709–4712.
15. Howgrave-Graham, N., and Joux, A. New generic algorithms for hard knapsacks. In *Advances in Cryptology - EUROCRYPT 2010* (2010), H. Gilbert, Ed., vol. 6110 of *Lecture Notes in Comput. Sci.*, Sringer.
16. Kachigar, G. Étude et conception d'algorithmes quantiques pour le décodage de codes linéaires. Master's thesis, Université de Rennes 1, France, Sept. 2016.
17. Magniez, F., Nayak, A., Roland, J., and Santha, M. Search via quantum walk. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing* (2007), STOC '07, pp. 575–584.
18. May, A., Meurer, A., and Thomae, E. Decoding random linear codes in $O(2^{0.054n})$. In *Advances in Cryptology - ASIACRYPT 2011* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Comput. Sci.*, Springer, pp. 107–124.
19. May, A., and Ozerov, I. On computing nearest neighbors with applications to decoding of binary linear codes. In *Advances in Cryptology - EUROCRYPT 2015* (2015), E. Oswald and M. Fischlin, Eds., vol. 9056 of *Lecture Notes in Comput. Sci.*, Springer, pp. 203–228.
20. McEliece, R. J. *A Public-Key System Based on Algebraic Coding Theory.* Jet Propulsion Lab, 1978, pp. 114–116. DSN Progress Report 44.
21. Niederreiter, H. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory 15*, 2 (1986), 159–166.
22. Overbeck, R., and Sendrier, N. Code-based cryptography. In *Post-quantum cryptography* (2009), D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds., Springer, pp. 95–145.
23. Prange, E. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory 8*, 5 (1962), 5–9.
24. Santha, M. Quantum walk based search algorithms. In *5th TAMC* (2008), pp. 31–46. arXiv/0808.0059.
25. Schroeppel, R., and Shamir, A. A $T=O(2^{n/2})$, $S=O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM J. Comput. 10*, 3 (1981), 456–464.
26. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput. 26*, 5 (1997), 1484–1509.
27. Stern, J. A method for finding codewords of small weight. In *Coding Theory and Applications* (1988), G. D. Cohen and J. Wolfmann, Eds., vol. 388 of *Lecture Notes in Comput. Sci.*, Springer, pp. 106–113.
28. Szegedy, M. Quantum speed-up of markov chain based algorithms. In *Proc. of the 45th IEEE Symposium on Foundations of Computer Science* (2004), pp. 32–41.

# A Proofs for Section 2

We want to prove the following theorem.

**Theorem 2 (Cartesian product of Johnson graphs).** *Let $J(n,r) = (\mathcal{V}, \mathcal{E})$, $m \in \mathbb{N}$ and $J^m(n,r) := \times_{i=1}^m J(n,r) = (\mathcal{V}_m, \mathcal{E}_m)$. Then:*

1. *$J^m(n,r)$ has $\binom{n}{r}^m$ vertices and is $md$-regular where $d = r(n-r)$.*
2. *We will write $\delta(J)$ resp. $\delta(J^m)$ for the spectral gaps of $J(n,r)$ resp. $J^m(n,r)$. Then: $\delta(J^m) \geq \frac{1}{m}\delta(J)$.*
3. *The random walk associated with $J^m(n,r)$ is aperiodic, irreducible and reversible for all positive $m$, $n$ and $r < n$.*

We need the following results for the proof.

**Theorem 5 (Cartesian product of $d$-regular graphs).** *Let $n \in \mathbb{N}$ and $G_1, ..., G_n$ be undirected $d$-regular graphs. Then $G^n = \times_{i=1}^n G_i$ has $\prod_{i=1}^n |V_i|$ vertices and is $nd$-regular.*

The proof of this theorem is immediate.

**Theorem 6 (Spectral gap of product graphs).** *Let $G_1$ and $G_2$ be $d_1$- resp. $d_2$-regular graphs with eigenvalues of the associated Markov chain $1 = \lambda_1 \geq \ ... \ \geq \lambda_{k_1}$ resp. $1 = \mu_1 \geq \ ... \ \geq \mu_{k_2}$. Denote by $\delta_i$ the spectral gap of $G_i$, $i = 1, 2$.*
*Then the spectral gap $\delta$ of the product graph $G_1 \times G_2$ fulfils:*

$$\delta \geq \frac{\min(\delta_2 d_2, \delta_1 d_1)}{d_1 + d_2}$$

*Proof.* We first recall the following result (see [10], Chapter 2, Section 5, Theorems 2.23 and 2.24): The Markov chain associated to the graph $G_1 \times G_2$ has $k_1 k_2$ eigenvalues which are $\nu_{i,j} = \frac{d_1\lambda_i + d_2\mu_j}{d_1+d_2}$. In particular, $\delta = \frac{d_1+d_2-\max_{(i,j)\neq(1,1)}|d_1\lambda_i+d_2\mu_j|}{d_1+d_2}$.
As the eigenvalues of $G_1$ and $G_2$ are ordered from largest to smallest, we have the following:

$$\max_{i=2,\ldots,k_1} |\lambda_i| = \max(\lambda_2, -\lambda_{k_1})$$

$$\max_{j=2,\ldots,k_2} |\mu_i| = \max(\mu_2, -\mu_{k_2})$$

Furthermore

$$d_1\delta_1 = d_1 - d_1 \max_{i=2,\ldots,k_1} |\lambda_i| = d_1 - d_1 \max(\lambda_2, -\lambda_{k_1}) \leq d_1 + d_1\lambda_{k_1}$$

$$d_2\delta_2 = d_2 - d_2 \max_{j=2,\ldots,k_2} |\mu_j| = d_2 - d_2 \max(\mu_2, -\mu_{k_2}) \leq d_2 + d_2\mu_{k_2}$$

Which taken together entail

$$d_1 + d_2 + d_1\lambda_{k_1} + d_2\mu_{k_2} \geq d_1\delta_1 + d_2\delta_2$$

Moreover

$$\begin{aligned}\max_{(i,j)\neq(1,1)} |d_1\lambda_i + d_2\mu_j| &= \max(d_1\lambda_1 + d_2\mu_2, d_1\lambda_2 + d_2\mu_1, -d_1\lambda_{k_1} - d_2\mu_{k_2}) \\ &= \max(d_1 + d_2\mu_2, d_1\lambda_2 + d_2, -d_1\lambda_{k_1} - d_2\mu_{k_2})\end{aligned}$$

Therefore

$$(d_1+d_2)\delta = d_1+d_2-\max_{(i,j)\neq(1,1)} |d_1\lambda_i+d_2\mu_j| = \min(d_2 - d_2\mu_2, d_1 - d_1\lambda_2, d_1 + d_2 + d_1\lambda_{k_1} + d_2\mu_{k_2})$$

Finally

$$\begin{aligned}
\delta &= \frac{d_1 + d_2 - \max_{(i,j)\neq(1,1)} |d_1\lambda_i + d_2\mu_j|}{d_1 + d_2} \\
&= \frac{\min\left(d_2 - d_2\mu_2, d_1 - d_1\lambda_2, d_1 + d_2 + d_1\lambda_{k_1} + d_2\mu_{k_2}\right)}{d_1 + d_2} \\
&\geq \frac{\min\left(d_2 - d_2\max(\mu_2, -\mu_{k_2}), d_1 - d_1\max(\lambda_2, -\lambda_{k_1}), d_1\delta_1 + d_2\delta_2\right)}{d_1 + d_2} \\
&\geq \frac{\min\left(\delta_2 d_2, \delta_1 d_1, d_1\delta_1 + d_2\delta_2\right)}{d_1 + d_2} \\
&\geq \frac{\min\left(\delta_2 d_2, \delta_1 d_1\right)}{d_1 + d_2}
\end{aligned}$$

□

*Proof (Theorem 2).* Point (1) is immediate by Theorem 5.
Point (2) is proved using induction.
Indeed, for $m = 2$ we have, by Theorem 6 :

$$\delta(J^2) \geq \frac{\delta(J)d}{2d} = \frac{1}{2}\delta(J)$$

And for $m \geq 2$, supposing that $\delta(J^m) \geq \frac{1}{m}\delta(J)$, we have, using Theorem 6 and point (1) of this theorem :

$$\begin{aligned}
\delta(J^{m+1}) &\geq \frac{\min\left(md\delta(J^m), d\delta(J)\right)}{md + d} \\
&\geq \frac{\min\left(\frac{md}{m}\delta(J), d\delta(J)\right)}{md + d} \\
&\geq \frac{1}{m+1}\delta(J)
\end{aligned}$$

Point (3) just follows from the fact that $J^m(n, r)$ is regular, undirected, connected and has positive spectral gap by using the previous point. □