Quantum Information Processing (2023) 22:143
https://doi.org/10.1007/s11128-023-03884-8

A generalisation of the Phase Kick-Back

Joaquín Ossorio-Castillo[1] · **Ulises Pastor–Díaz**[1] · **José M. Tornero**[1]

Received: 9 October 2022 / Accepted: 17 February 2023 / Published online: 13 March 2023

Abstract
In this paper, we present a generalisation of the Phase Kick-Back technique, which is central to some of the classical algorithms in quantum computing. We will begin by recalling the Phase Kick-Back technique to then introduce the new generalised version for $f : \{0,1\}^n \to \{0,1\}^m$ functions using the eigenvalues of the oracle function $\mathbf{U}_f$. After that, we will present a new generalised version of the Deutsch–Jozsa problem and how it can be solved using the previously defined technique. We will also deal with a generalised version of the Bernstein–Vazirani problem and solve it using the generalised Phase Kick-Back. Finally, we show how we can use this technique to obtain an algorithm for Simon's problem that improves the classical one.

Keywords Quantum algorithms · Phase Kick-Back · Deutsch–Jozsa · Bernstein–Vazirani · Boolean functions

Mathematics Subject Classification 68Q12 · 68Q09 · 81P68

1 Introduction: Phase Kick-Back and notation

The term *Phase Kick-Back* is taken from [1]. This technique is central to some classical quantum algorithms, such as the Deutsch–Jozsa algorithm, the Bernstein–Vazirani algorithm, Simon's algorithm or Grover's algorithm, and first appeared in [2] for solving the Deutsch–Jozsa problem. Here, we will generalise the Deutsch–Jozsa problem

✉ Ulises Pastor–Díaz
upastor@us.es

Joaquín Ossorio-Castillo
jqnssr@gmail.com

José M. Tornero
tornero@us.es

[1] Departamento de Álgebra, Facultad de Matemáticas, Universidad de Sevilla Avda, Reina Mercedes s/n, 41012 Sevilla, Spain

and the Bernstein–Vazirani problem and give a new algorithm for Simon's problem that improves the classical one.

Some of these problems have already been generalised in different directions. In [3], multidimensional versions for the Deutsch–Jozsa problem and Bernstein–Vazirani were first considered and solved. This was further expanded in [4] by considering the problems of evenly distributed and evenly balanced functions.

Regarding other directions, in [5] the Deutsch–Jozsa problem is generalised by considering functions that are balanced in abelian subgroups of $\{0,1\}^n$. In [6, 7], the problem of filtering and distinguishing quantum states is studied, while in [8] a problem where Boolean functions that are balanced in certain subsets of $\{0,1\}^n$ is proposed. Furthermore, Deutsch's problem is generalised in [9] by determining arbitrary Boolean functions $f : \{0,1\}^2 \to \{0,1\}$.

The Bernstein–Vazirani problem was also generalised in [10, 11] by considering and solving the 2-dimensional hidden linear function problem using shallow quantum circuits.

The Deutsch–Jozsa algorithm—and thus the Phase Kick–Back technique—has been implemented using different quantum models, such as by the application of NMR in [12], or via Rydberg blockade interaction in [13].

Let us introduce the notation we will use, which will be that of [1, 14]. These two books, along with [15, 16], can be consulted for more context on the topic of quantum computing.

Remark 1 First of all, we will call the elements $\mathbf{x} \in \{0,1\}^n$ binary strings and note them in bold, underlining their structure as vectors in the space $\mathbb{F}_2^n$.

Let $\mathbf{y}, \mathbf{z} \in \{0,1\}^n$ be two strings, written

$$\mathbf{y} = y_{n-1} \dots y_1 y_0, \qquad \mathbf{z} = z_{n-1} \dots z_1 z_0,$$

and let $\oplus$ denote the exclusive or addition (which is addition modulo 2). We define the exclusive or operation for strings as the exclusive or bitwise, that is,

$$\mathbf{y} \oplus \mathbf{z} = (y_{n-1} \oplus z_{n-1}) \dots (y_1 \oplus z_1)\,(y_0 \oplus z_0)\,,$$

and we will denote the pairing in $\{0,1\}^n$ (not a scalar product, though) by

$$\mathbf{y} \cdot \mathbf{z} = (y_0 \cdot z_0) \oplus \dots \oplus (y_{n-1} \cdot z_{n-1})\,.$$

Note that, as the xor operation is performed bitwise, we have

$$\mathbf{x} \cdot (\mathbf{y} \oplus \mathbf{z}) = (\mathbf{x} \cdot \mathbf{y}) \oplus (\mathbf{x} \cdot \mathbf{z}).$$

We will also write $\mathbf{0}$ to refer to the zero n-string $\mathbf{0} = 00 \cdots 0$.

To represent quantum states, we will use the Bra-Ket or Dirac notation, where given a binary string $\mathbf{x} \in \{0,1\}^n$ of length n we represent the n-dimensional qubit state of the computational basis corresponding to $\mathbf{x}$ by $|\mathbf{x}\rangle_n$. For one-dimensional qubit systems, we will often simply write the ket $|\mathbf{x}\rangle$ without the subindex. If we have more than

one qubit system, we will write the number of qubits of each register separated by commas. For example, in $|\mathbf{x}\rangle_{n,m,r}$ we would have three registers of n, m and r qubits, respectively. We will often note $N = 2^n$.

Let R be an $m \times n$ Boolean matrix—i.e., a matrix whose components are either 0 s or 1 s—and let $\mathbf{r}_i$ be the binary string determined by the i-th file of R, we will define the result of the operation $R \cdot \mathbf{x}$ as the string whose i-th component is $\mathbf{r}_i \cdot \mathbf{x}$ (that is, the usual matrix–vector operation).

We will say that a Boolean function is a function $f : \{0,1\}^n \to \{0,1\}^m$. It is well known—consult [15] for more information—that given a Boolean function one can construct the quantum gate $\mathbf{U}_f$ whose effect is the following:

$$\mathbf{U}_f\Big(|\mathbf{x}\rangle_n \otimes |\mathbf{y}\rangle_m\Big) = |\mathbf{x}\rangle_n \otimes |\mathbf{y} \oplus f(\mathbf{x})\rangle_m.$$

Before proceeding, we must recall the *Hadamard basis*:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

The result in which the *Phase Kick-Back* technique is based is the following:

Lemma 1 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $\mathbf{U}_f$ be the quantum gate that computes it. Then, in the $n+1$ qubit system, vectors of the form $|\mathbf{x}\rangle_n \otimes |-\rangle$ are eigenvectors with eigenvalue $(-1)^{f(\mathbf{x})}$ for every $\mathbf{x} \in \{0,1\}^n$.*

The *Phase Kick-Back* technique is almost always used to mark the amplitudes of the states of the computational basis whose image through f is 1. In that sense, we would have

$$\Big(\mathbf{H}_n|\mathbf{0}\rangle_n\Big) \otimes |-\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle_n\right) \otimes |-\rangle,$$

where $\mathbf{H}_n$ is the Hadamard matrix of dimension n, which can be defined as:

$$\mathbf{H}_n = \mathbf{H}^{\otimes n}, \text{ where } \mathbf{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and whose effect on an element of the computational basis $\mathbf{x} \in \{0,1\}^n$ is the following:

$$\mathbf{H}_n|\mathbf{x}\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle_n.$$

This can be easily proven by induction. In particular, when $\mathbf{x} = \mathbf{0}$, we would have:

$$\mathbf{H}_n|\mathbf{0}\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}\in\{0,1\}^n} |\mathbf{z}\rangle_n.$$

Summarising, we would have a summation over all the states of the computational basis, all of them with the same amplitude in the first n-qubit register. The idea of the Phase Kick-Back is to apply $\mathbf{U}_f$ to this state and mark the aforementioned elements with a negative amplitude.

$$\mathbf{U}_f\left(\frac{1}{\sqrt{2^n}}\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}\rangle_n\otimes|-\rangle\right)=\left(\frac{1}{\sqrt{2^n}}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{x}\rangle_n\right)\otimes|-\rangle.$$

2 Generalised Phase Kick-Back

Let us now present a generalisation of the Phase Kick-Back idea. This approach was already suggested in [3]. This generalisation will consist on the expansion of the technique to general Boolean functions $f:\{0,1\}^n\to\{0,1\}^m$, where the target qubit—the one in the second register of the Deutsch–Jozsa algorithm—becomes a register of m qubits.

During this generalisation, we will take $\mathbf{U}_f$ as presented before and we will notate the states given by $\mathbf{H}_n|\mathbf{y}\rangle_n$ as $|\gamma_\mathbf{y}\rangle_n$, where $|\mathbf{y}\rangle_n$ are the elements of the computational basis.

Let us begin by presenting an analogous version to that of Lemma 1, which will constitute the core idea of this technique.

Lemma 2 *Let $|\gamma_\mathbf{y}\rangle_m=\mathbf{H}_m|\mathbf{y}\rangle_m$ with $\mathbf{y}\in\{0,1\}^m$. Then, for each $\mathbf{x}\in\{0,1\}^n$, the vector $|\mathbf{x}\rangle_n\otimes|\gamma_\mathbf{y}\rangle_m$ is an eigenvector of $\mathbf{U}_f$ with eigenvalue $(-1)^{\mathbf{y}\cdot f(\mathbf{x})}$.*

Proof We know that

$$|\gamma_\mathbf{y}\rangle_m=\frac{1}{\sqrt{2^m}}\sum_{\mathbf{z}\in\{0,1\}^m}(-1)^{\mathbf{y}\cdot\mathbf{z}}|\mathbf{z}\rangle_m.$$

If we now apply $\mathbf{U}_f$ to $|\mathbf{x}\rangle_n\otimes|\gamma_\mathbf{y}\rangle_m$, we get the following:

$$\begin{aligned}\mathbf{U}_f\Big(|\mathbf{x}\rangle_n\otimes|\gamma_\mathbf{y}\rangle_m\Big)&=|\mathbf{x}\rangle_n\otimes\left(\frac{1}{\sqrt{2^m}}\sum_{\mathbf{z}\in\{0,1\}^m}(-1)^{\mathbf{y}\cdot\mathbf{z}}|\mathbf{z}\oplus f(\mathbf{x})\rangle_m\right).\\&=(-1)^{\mathbf{y}\cdot f(\mathbf{x})}|\mathbf{x}\rangle_n\otimes\left(\frac{1}{\sqrt{2^m}}\sum_{\mathbf{z}\in\{0,1\}^m}(-1)^{\mathbf{y}\cdot\mathbf{z}\oplus\mathbf{y}\cdot f(\mathbf{x})}|\mathbf{z}\oplus f(\mathbf{x})\rangle_m\right)\\&=(-1)^{\mathbf{y}\cdot f(\mathbf{x})}|\mathbf{x}\rangle_n\otimes\left(\frac{1}{\sqrt{2^m}}\sum_{\mathbf{z}\in\{0,1\}^m}(-1)^{\mathbf{y}\cdot(\mathbf{z}\oplus f(\mathbf{x}))}|\mathbf{z}\oplus f(\mathbf{x})\rangle_m\right)\\&=(-1)^{\mathbf{y}\cdot f(x)}|\mathbf{x}\rangle\otimes|\gamma_\mathbf{y}\rangle_m,\end{aligned}$$

as for a fixed $f(\mathbf{x})$, $|\mathbf{z}\oplus f(\mathbf{x})\rangle_m$ runs through all of $\{0,1\}^m$ just as $\mathbf{z}$ does. □

As we can see, it is a completely analogous idea to the previous one, with the difference that we can now choose a *marker*, $\mathbf{y} \in \{0,1\}^n$, which will work as a fixed reference and multiply each $f(\mathbf{x})$.

3 A first approach to the GPK

We will consider the Boolean function $f : \{0,1\}^3 \to \{0,1\}^2$ which will eliminate the last bit, that is, $f(xyz) = xy$ where $x, y, z \in \{0,1\}$ and xyz stands for the concatenation of bits.

To use our new tool, we will need a 5-qubit system divided into a 3-qubit register and a 2-qubit register, both of them starting on $|\mathbf{0}\rangle$:

$$|\varphi_0\rangle_5 = |\mathbf{0}\rangle_3 \otimes |\mathbf{0}\rangle_2.$$

We will begin by choosing a marker, i.e., the $\mathbf{y} \in \{0,1\}^2$ that will encode the information we want to look for in f. In this case, we will take $\mathbf{y} = 01$, that is, we will mark those values whose image through f is 01 or 11. To do so, we will begin by preparing the second register to $\mathbf{y}$, which is easily achieved by applying the Pauli $\mathbf{X}$ gate on the last qubit.

$$|\varphi_1\rangle_5 = \left(\mathbf{I}^{\otimes 4} \otimes \mathbf{X}\right) |\varphi_0\rangle_5 = |\mathbf{0}\rangle_3 \otimes |01\rangle_2.$$

Once we have prepared our basic state, we will apply Hadamard gates to all qubits to obtain a superposition state.

$$|\varphi_2\rangle_5 = \mathbf{H}_5 |\varphi_1\rangle_5 = \left(\frac{1}{\sqrt{8}} \sum_{\mathbf{x}\in\{0,1\}^3} |\mathbf{x}\rangle_3\right) \otimes |\gamma_{01}\rangle_2 = \frac{1}{\sqrt{8}} \sum_{\mathbf{x}\in\{0,1\}^3} \Big(|\mathbf{x}\rangle_3 \otimes |\gamma_{01}\rangle_2\Big).$$

Let us remark now that each state of the aforementioned superposition satisfies the conditions of Lemma 2.1, and thus, if we apply the $\mathbf{U}_f$ gate, we will mark the states of the superposition depending on their image.

$$|\varphi_3\rangle_5 = \mathbf{U}_f |\varphi_2\rangle_5 = \frac{1}{\sqrt{8}} \sum_{\mathbf{x}\in\{0,1\}^3} (-1)^{f(\mathbf{x})\cdot 01} \Big(|\mathbf{x}\rangle_3 \otimes |\gamma_{01}\rangle_2\Big).$$

Remark 2 Another way of looking at this Generalised Phase Kick-Back idea is to write the state $|\gamma_{\mathbf{y}}\rangle$ as a tensor product of $|+\rangle$ and $|-\rangle$ states. As an example, in the instance we are dealing with we have:

$$|\gamma_{01}\rangle_2 = |+\rangle \otimes |-\rangle.$$

In general, for a given $\mathbf{y} \in \{0,1\}^m$ we would have a $|+\rangle$ in the i-th position if the i-th bit of $\mathbf{y}$ is 0 and $|-\rangle$ if it is 1. In that sense, we could look at this Generalised Phase Kick-Back as a cascade of Phase Kick-Backs in those positions of $\mathbf{y}$ in which there is a 1.

Let us now focus our attention on the first 3-qubit register and use that $f(\mathbf{x}) \cdot 01 = \mathbf{x} \cdot 010$:

$$|\varphi_4\rangle_3 = \frac{1}{\sqrt{8}} \sum_{\mathbf{x}\in\{0,1\}^3} (-1)^{f(\mathbf{x})\cdot 01} |\mathbf{x}\rangle_3 = \frac{1}{\sqrt{8}} \sum_{\mathbf{x}\in\{0,1\}^3} (-1)^{\mathbf{x}\cdot 010} |\mathbf{x}\rangle_3.$$

And finally, if we apply Hadamard gates to this 3-qubit system, we will get the state $|010\rangle_3$.

$$|\varphi_5\rangle_3 = \mathbf{H}_3 |\varphi_4\rangle_3 = |010\rangle_3.$$

It is not clear now how this idea is helpful, as the final result is directly determined by the initial $\mathbf{y}$ we chose, and if we had fixed $\mathbf{y} = 10$, then the final result would have been 100. However, suppose now that we do not know which of the bits f eliminates, and we want to determine which one it is. We only have three possibilities, and we could easily check with one classical call to f which of the bits is eliminated—simply compute $f(010)$—but it is interesting to do it by using our new tool.

Lemma 3 *Let $f : \{0,1\}^n \to \{0,1\}^{n-1}$ be a Boolean function that eliminates one bit; then, we can use the algorithm above to determine which bit is eliminated.*

Proof To do so, we just apply the generalised version of the algorithm mentioned above $n-1$ times, using each time one of the vectors of the canonical basis of $\{0,1\}^{n-1}$ as an $\mathbb{F}_2$ vector space. If we denote by $\mathbf{e}_i$ the string of bits whose only 1 is in the i-th position (starting by 0)—i.e., the i-th element of the canonical basis—then each of the $n-1$ iterations of the algorithm would go as follows:

$$|\varphi_0\rangle_{n,n-1} = |\mathbf{0}\rangle_n \otimes |\mathbf{0}\rangle_{n-1}.$$

First, we obtain $\mathbf{e}_i$ in the second register by applying the $\mathbf{X}$ gate wherever we need:

$$|\varphi_1\rangle_{n,n-1} = |\mathbf{0}\rangle_n \otimes |\mathbf{e}_i\rangle_{n-1}.$$

Second, we apply Hadamard gates:

$$|\varphi_2\rangle_{n,n-1} = (\mathbf{H}_{2n-1})\, |\varphi_1\rangle_{2n-1}.$$

Then, we use the GPK (Generalised Phase Kick-Back):

$$|\varphi_3\rangle_{n,n-1} = \mathbf{U}_f |\varphi_2\rangle_{2n-1}.$$

And finally, we apply Hadamard gates to the first register and measure:

$$|\varphi_4\rangle_{n,n-1} = (\mathbf{H}_n \otimes \mathbf{I}^{\otimes(n-1)})|\varphi_3\rangle_{n,n-1}.$$

After we have done so with all $n-1$ possible $\mathbf{e}_i$, we will have obtained $n-1$ of the n vectors of the canonical basis of $\mathbb{F}_2^n$, and the one left indicates which of the bits is eliminated. □

This, of course, does not give us an improvement of any sort over the classical case—it is actually the opposite, as we could have just computed the image of $\mathbf{x} = 101010\ldots$ and checked for repeated characters—but it illustrates the inner workings of the technique.

Some other examples such as this could be constructed. Another one is the problem of, given an f that switches one unknown bit, finding out which one. However, we will now focus on a problem in which this idea allows for an improvement over the classical situation.

4 The generalised Deutsch–Jozsa problem

An easy follow-up to the previous section would be to solve a generalised version of the Deutsch–Jozsa problem using this technique. As we have seen, many such generalisations have been considered, but the one we propose generalises the evenly balanced one proposed both in [3] and in [4].

Definition 1 (Generalised Deutsch–Jozsa problem.) We say that a Boolean function is balanced if half of the input values output one string and the other half output another.

Given then a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ that can either be constant or balanced, we will denote by Generalised Deutsch–Jozsa problem the one of finding out in which of the cases are we.

The Deutsch–Jozsa problem is clearly one instance of this general problem where $m = 1$, and thus, we will show how we can solve this problem by using an algorithm inspired by that of Deutsch and Jozsa.

Remark 3 It is clear that if we want to solve this problem using classical deterministic methods, we will need something of the order of $\mathcal{O}(2^{n-1})$ applications of f. We will see how we can improve this with a quantum algorithm to an order of $\mathcal{O}(m)$ calls to f. Note also that this includes the already known case where $m = 1$.

Let us limit ourselves to the instance where constant means that $f(\mathbf{x}) = \mathbf{0}$ for every $\mathbf{x} \in \{0,1\}^n$ and balanced means that half of the values are $\mathbf{0}$ and the other half a fixed string different from $\mathbf{0}$.

Given $\mathbf{e}_i = 0^{(m-1)-i}\ 1\ 0^i$, where $i = 0, \ldots, m-1$, we will repeat the following algorithm for each $\mathbf{e}_i$, but it could actually be done for any binary string $\mathbf{y} \in \{0,1\}^m$.

$\mathbb{STEP}1$

$|\varphi_0\rangle_{n,m} = |\mathbf{0}\rangle_n \otimes |\mathbf{0}\rangle_m.$

We begin with two registers of n and m qubits, both at the state $|\mathbf{0}\rangle$.

$\mathbb{STEP}$ 2

$|\varphi_1\rangle_{n,m} = \left(\mathbf{I}^{\otimes n} \otimes \mathbf{I}^{\otimes(m-1-i)} \otimes \mathbf{X} \otimes \mathbf{I}^{\otimes i}\right) |\varphi_0\rangle_{n,m}$.

We apply the $\mathbf{X}$ gate to achieve the desired $|\mathbf{e}_i\rangle$ state in the second register. If we want any other binary string $\mathbf{y}$ to act as a marker, we should apply the corresponding $\mathbf{X}$ gates in the necessary positions.

$\mathbb{STEP}$ 3

$|\varphi_2\rangle_{n,m} = \mathbf{H}_{n+m}|\varphi_1\rangle_{n,m}$.

We apply Hadamard gates to obtain the desired superposition in the first register and $|\gamma_{\mathbf{e}_i}\rangle$ in the second.

$\mathbb{STEP}$ 4

$|\varphi_3\rangle_{n,m} = \mathbf{U}_f|\varphi_2\rangle_{n,m}$.

We apply $\mathbf{U}_f$ to use the GPK technique.

$\mathbb{STEP}$ 5

$|\varphi_4\rangle_{n,m} = \left(\mathbf{H}^{\otimes n} \otimes \mathbf{I}^{\otimes m}\right) |\varphi_3\rangle_{n,m}$.

At this point, the second register might be discarded and we apply Hadamard gates to the first one.

$\mathbb{STEP}$ 6

We measure the first register and name the result δ_i.

If after repeating these steps for each i we obtain only $\delta_i = \mathbf{0}$ strings, then the function is constant; otherwise it is balanced.

Definition 2 (Generalised Phase Kick-Back algorithm.) The only variable in the algorithm is the choice of the marker $\mathbf{y}$ used for the Phase Kick-Back. We will refer to this algorithm as *GPK algorithm for* $\mathbf{y}$ or GPK($\mathbf{y}$). From now on, the notation regarding this algorithm will be the same as before.

Theorem 1 (Correctness of the algorithm) *The aforementioned algorithm correctly determines whether a function is constant or balanced in the case where the image set of f includes $\mathbf{0}$.*

Proof Given $i = 0, \ldots, m-1$, let us keep track of the states step by step:

As we are applying the $\mathbf{X}$ gate on the i-th qubit of the second register (counting from 0), then

$$|\varphi_1\rangle_{n,m} = |\mathbf{0}\rangle_n \otimes |\mathbf{e}_i\rangle_m,$$

Next,

$$|\varphi_2\rangle_{n,m} = \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle_n \otimes |\gamma_{\mathbf{e}_i}\rangle_m,$$

just by the definition of $|\gamma_{\mathbf{e}_i}\rangle$ and the known effect of Hadamard gates on the $|\mathbf{0}\rangle$ state. Finally, we obtain

$$|\varphi_3\rangle_{n,m} = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} |\mathbf{x}\rangle_n\right) \otimes |\gamma_{\mathbf{e}_i}\rangle_m,$$

by applying Lemma 2.

If we focus now only on the first register, we will have the following state:

$$\frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} |\mathbf{x}\rangle_n.$$

Then, after applying the Hadamard gates, we will have:

$$\begin{aligned}\mathbf{H}_n \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} |\mathbf{x}\rangle_n &= \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} \mathbf{H}_n |\mathbf{x}\rangle_n \\ &= \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{x}\cdot\mathbf{z}} |\mathbf{z}\rangle_n\right) \\ &= \frac{1}{N} \sum_{\mathbf{z}\in\{0,1\}^n} \left[\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i \oplus \mathbf{x}\cdot\mathbf{z}}\right] |\mathbf{z}\rangle_n.\end{aligned}$$

It is easy to check that if the function is constant and equal to $\mathbf{0}$, then regardless of the value of i the amplitude of $|\mathbf{0}\rangle_n$ in the previous superposition is the following:

$$\frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} = \frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^0 = 1.$$

Thus, we will always obtain $\delta_i = \mathbf{0}$ no matter which marker we use.

If f is not constant, then when $f(\mathbf{x}) \neq \mathbf{0}$ there must be an $i \in \{0, \ldots, m-1\}$ for which $f(\mathbf{x}) \cdot \mathbf{e}_i = 1$. If we take such a $\mathbf{e}_i$, then the amplitude for $|\mathbf{0}\rangle_n$ is:

$$\frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{(f(\mathbf{x})\cdot\mathbf{e}_i)\oplus(\mathbf{x}\cdot\mathbf{0})} = \frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i} = 0,$$

because $f(\mathbf{x})$ is balanced, and thus, half the elements of the sum will be 1 and the other half -1. This implies that we would get a result different from $\mathbf{0}$ for that i. □

Note that the choice of the canonical basis is not compulsory and that we could have chosen any other basis of $\mathbb{F}_2^m$ as our markers.

The same idea works for the general case of the Generalised Deutsch–Jozsa problem.

Theorem 2 (General correctness) *The previous algorithm correctly determines whether a function is constant or balanced.*

Proof The only thing left to analyse is the final amplitudes in the general case. To do so, we need to recall that the final state is:

$$\frac{1}{N}\sum_{\mathbf{z}\in\{0,1\}^n}\left[\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{(f(\mathbf{x})\cdot\mathbf{e}_i)\oplus(\mathbf{x}\cdot\mathbf{z})}\right]|\mathbf{z}\rangle_n.$$

If we analyse now the amplitude of $|\mathbf{z}\rangle_n = |\mathbf{0}\rangle_n$, we would be left with:

$$\frac{1}{N}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{(f(\mathbf{x})\cdot\mathbf{e}_i)\oplus(\mathbf{x}\cdot\mathbf{0})} = \frac{1}{N}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})\cdot\mathbf{e}_i}.$$

If $f(\mathbf{x})$ is constant, then $f(\mathbf{x})\cdot\mathbf{e}_i$ is either always 0 or always 1, as $\mathbf{x}$ varies. Whichever the case, the final amplitude will be either 1 or -1, and thus, we will always get $\mathbf{0}$ at the end of the algorithm.

On the other hand, if $f(\mathbf{x})$ is balanced with possible values $\mathbf{f}_1, \mathbf{f}_2 \in \{0,1\}^m$ such that $\mathbf{f}_1 \neq \mathbf{f}_2$, then there is a $i \in \{0,\ldots,m-1\}$ such that $\mathbf{f}_1\cdot\mathbf{e}_i \neq \mathbf{f}_2\cdot\mathbf{e}_i$, and for that i the amplitude of $\mathbf{z}=\mathbf{0}$ would be:

$$\frac{1}{N}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})\cdot\mathbf{e}_i}.$$

As the function is balanced between $\mathbf{f}_1$ and $\mathbf{f}_2$, that amplitude is 0 and thus we would get a result different from $\mathbf{0}$. □

This algorithm not only allows us to distinguish constant and balanced functions, but it also allows us to determine the values of the function. In the balanced situation, it would not be possible to do that efficiently in a deterministic way.

Corollary 1 *It is possible to determine the possible values of f by applying the aforementioned algorithm and making a classical call to the function.*

Proof Let us begin by the case in which the possible images are $\mathbf{0}$ and $\mathbf{f}_1$. In this situation, the values of i for which we obtain a result different from $\delta_i = \mathbf{0}$ mark the bits of $\mathbf{f}_1$ that are different from 0, thus determining exactly the value of $\mathbf{f}_1$, so $\mathbf{f}_1 = \boldsymbol{\lambda} = \lambda_{m-1}\ldots\lambda_1\lambda_0$, where we define λ_i as:

$$\lambda_i = \begin{cases} 0 & \text{if } \delta_i = \mathbf{0} \\ 1 & \text{otherwise.} \end{cases}$$

In the general case, if we note the two possible images by $\mathbf{f}_1$ and $\mathbf{f}_2$, the $\boldsymbol{\lambda} = \lambda_{m-1}\ldots\lambda_1\lambda_0$ string tells us that the Boolean bitwise difference between $\mathbf{f}_1$ and $\mathbf{f}_2$—i.e., $\mathbf{f}_1\oplus\mathbf{f}_2$. Thus, we would know that $\mathbf{f}_1 = \mathbf{f}_2\oplus\boldsymbol{\lambda}$. If we now classically calculate one

of the possible images—for instance $f(\mathbf{0})$—we would be able to retrieve both values. □

Remark 4 We also have to point out that we have solved the problem by applying the quantum gate $\mathbf{U}_f$ m times, which is an exponential improvement over the deterministic classical situation when m is of linear order with respect to n.

Remark 5 There is a pattern that will reappear in the following section, which is that the GPK algorithm is unable to detect translations. That is, given two Boolean functions $f_1, f_2 : \{0,1\}^n \to \{0,1\}^m$ for which there is an $\mathbf{s} \in \{0,1\}^n$ such that $f_1(\mathbf{x}) = f_2(\mathbf{x}) \oplus \mathbf{s}$ for every $\mathbf{x} \in \{0,1\}^n$, if we analyse the first register of $|\varphi_4\rangle_{n+m}$ for function f_2 using $\mathbf{y} \in \{0,1\}^m$ as a marker, we get:

$$\frac{1}{N} \sum_{\mathbf{z} \in \{0,1\}^n} \left[\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f_2(\mathbf{x}) \cdot \mathbf{y} \oplus \mathbf{x} \cdot \mathbf{z}} \right] |\mathbf{z}\rangle_n .$$

And if we now use that $f_2(\mathbf{x}) = f_1(\mathbf{x}) \oplus \mathbf{s}$, we get:

$$\begin{aligned} &\frac{1}{N} \sum_{\mathbf{z} \in \{0,1\}^n} \left[\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{(f_1(\mathbf{x}) \oplus \mathbf{s}) \cdot \mathbf{y} \oplus \mathbf{x} \cdot \mathbf{z}} \right] |\mathbf{z}\rangle_n \\ = (-1)^{\mathbf{s} \cdot \mathbf{y}} &\frac{1}{N} \sum_{\mathbf{z} \in \{0,1\}^n} \left[\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f_1(\mathbf{x}) \cdot \mathbf{y} \oplus \mathbf{x} \cdot \mathbf{z}} \right] |\mathbf{z}\rangle_n . \end{aligned}$$

And, as we can observe, we end up getting a quantum state equivalent to the one we would get by applying the GPK algorithm for the function f_1, which does not affect the probabilities of the final result. This is the reason behind the fact that what we get in the general case of the balanced situation in the Generalised Deutsch–Jozsa algorithm is the sum of the two possible values $\boldsymbol{\lambda}$, and why we must make an extra step to find both values.

Remark 6 In order to solve the generalised Deutsch–Jozsa problem we have computed m applications of the GPK algorithm with the elements of the computational basis as markers. What we want to show now is that this choice of markers is not compulsory and that any basis of $\{0,1\}^m$ would suffice.

Let $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \{0,1\}^m$ be any such basis; we will compute now the GPK algorithm for each of these markers. It becomes clear that if $f(\mathbf{x}) \cdot \mathbf{y}_i$ is constant for all $\mathbf{x} \in \{0,1\}^n$, then the result of the i-th iteration of the algorithm will be $\mathbf{0}$, while if $f(\mathbf{x}) \cdot \mathbf{y}_i = 0$ for half of the values and 1 for the other half, then the result will be any other binary string.

Let $\boldsymbol{\lambda} = \mathbf{f}_1 \oplus \mathbf{f}_2$ be the sum of the two possible values of the function as before—if the function is constant we would have $\boldsymbol{\lambda} = \mathbf{0}$—then what we end up with is a system of equations:

$$\{\mathbf{y}_i \cdot \boldsymbol{\lambda} = \delta_i \mid i = 1, \ldots, m\},$$

where $\boldsymbol{\lambda}$ is the string of unknowns. This system is always made up of m linearly independent equations, as the $\mathbf{y}_i$ are a basis of $\{0,1\}^m$, so the sole solution will be the desired $\boldsymbol{\lambda}$.

5 A Bernstein–Vazirani inspired algorithm

Once again we will put our focus on generalising an already known problem which was first studied in [17]. The problem we propose was already solved in [3], but we will further expand the idea and use it to better understand the technique. Let us begin by recalling the Bernstein–Vazirani problem in the one-dimensional situation.

Definition 3 (Bernstein–Vazirani problem.) Let $f : \{0,1\}^n \to \{0,1\}$ be a function such that there is an $\mathbf{r} \in \{0,1\}^n$ for which $f(\mathbf{x}) = \mathbf{r} \cdot \mathbf{x}$, we want to find the binary string $\mathbf{r}$.

Let us note that the condition stated in the Bernstein–Vazirani problem just asks for f to be linear. This is relevant because in the generalisation of this problem we will consider a linear $f : \{0,1\}^n \to \{0,1\}^m$ and ask to exactly determine it.

Regarding the complexity of this problem, we should note that a linear function $f : \{0,1\}^n \to \{0,1\}$ can be determined in n calls to f, as we only have to calculate the image through f of the elements of one basis of $\{0,1\}^n$. In particular, we can calculate $f(\mathbf{e}_i)$ for each element in the canonical basis and the i-th element of $\mathbf{r}$ would be $r_i = f(\mathbf{e}_i)$. The exact same can be done in the general case.

We will show how we can solve this problem with a quantum algorithm making a single call to $\mathbf{U}_f$. The algorithm we will describe is exactly the same as we used to solve the Deutsch–Jozsa problem.

First, we will have two registers of n and 1 qubits, respectively:

$$|\varphi_0\rangle_{n,1} = |\mathbf{0}\rangle_n \otimes |1\rangle$$

We can obtain the $|1\rangle$ in the second register by applying the $\mathbf{X}$ gate to the last qubit. Secondly, we will apply Hadamard gates to all the qubits in order to obtain the state:

$$|\varphi_1\rangle_{n,1} = \mathbf{H}_n|\varphi_0\rangle_{n,1} = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle_n\right) \otimes |-\rangle,$$

where $N = 2^n$. This state is now ready to use the Phase Kick-Back technique by applying $\mathbf{U}_f$:

$$|\varphi_2\rangle_{n,1} = \mathbf{U}_f|\varphi_1\rangle_{n,1} = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle_n\right) \otimes |-\rangle.$$

Using now that $f(\mathbf{x}) = \mathbf{r} \cdot \mathbf{x}$, we arrive at:

$$|\varphi_2\rangle_{n,1} = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\mathbf{r} \cdot \mathbf{x}} |\mathbf{x}\rangle_n \right) \otimes |-\rangle.$$

Recalling the effect of $\mathbf{H}_n$ on the computational basis, we can easily check that the first register of this state is exactly $\mathbf{H}_n |\mathbf{r}\rangle$, so after applying $\mathbf{H}_n$ to the first register we obtain:

$$|\varphi_3\rangle_{n,1} = (\mathbf{H}_n \otimes \mathbf{I}) |\varphi_2\rangle_{n,1} = |\mathbf{r}\rangle_n \otimes |-\rangle.$$

Then, after measuring the first register we will obtain $\mathbf{r}$.

Let us consider now a slight modification to the Bernstein–Vazirani problem.

Definition 4 (Modified Bernstein–Vazirani problem.) Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean affine function—i.e., a Boolean function such that there are $\mathbf{r} \in \{0,1\}^n$ and $r_0 \in \{0,1\}$ for which $f(\mathbf{x}) = r_0 \oplus \mathbf{r} \cdot \mathbf{x}$ for all $\mathbf{x}$—then we want to exactly determine said function.

This problem can be solved by the previous algorithm with just a final step to determine r_0.

Proposition 1 *The Bernstein–Vazirani algorithm solves the modified Bernstein–Vazirani problem with certainty with a final classical deterministic call to f to determine r_0.*

Proof Following the previous exposition of the Bernstein–Vazirani algorithm, the only difference in this situation is that we would end up with the state:

$$\begin{aligned} |\varphi_2\rangle_{n,1} &= \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{r_0 \oplus \mathbf{r} \cdot \mathbf{x}} |\mathbf{x}\rangle_n \right) \otimes |-\rangle \\ &= (-1)^{r_0} \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\mathbf{r} \cdot \mathbf{x}} |\mathbf{x}\rangle_n \right) \otimes |-\rangle. \end{aligned}$$

This is equivalent to the state we had in the previous situation, and thus, we would end up getting $\mathbf{r}$ after measuring $|\varphi_3\rangle_{n,1}$.

To get r_0, we must only classically calculate $f(\mathbf{0}) = r_0$. □

Again, we arrive at the same pattern, where the GPK cannot distinguish a translation in f, but only the linear structure it has.

This idea can be used to generalise the Bernstein–Vazirani problem to arbitrary dimensions.

Definition 5 (Generalised Bernstein–Vazirani problem.) Let $f : \{0,1\}^n \to \{0,1\}^m$ be an affine function, i.e., one such that there is an $m \times n$ matrix R and an $\mathbf{r}_0 \in \{0,1\}^m$ for which $f(\mathbf{x}) = \mathbf{r}_0 \oplus R \cdot \mathbf{x}$. The Generalised Bernstein–Vazirani problem is that of exactly determining f.

Remark 7 Let us analyse the classical deterministic complexity of this problem. It is easy to prove that we can exactly determine R by calculating $f(\mathbf{e}_i)$ for each element of the computational basis, as the binary string determined by the i-th file of R, $\mathbf{r}_i$, will be exactly $f(\mathbf{e}_i) \oplus \mathbf{r}_0$. We can finally calculate $\mathbf{r}_0$ by computing $f(\mathbf{0})$, so the total calls to f will be $n+1$.

It can be seen that with the GPK we can do this with $m+1$ calls to the function, so in a way we will switch the roles of $\{0,1\}^n$ and $\{0,1\}^m$.

We will now prove that we can solve the Generalised Bernstein–Vazirani problem by computing m iterations of the GPK algorithm by each of the elements of the computational basis of $\{0,1\}^m$ and a final classical computation of $f(\mathbf{0})$.

Theorem 3 (Correctness of the algorithm) *It is possible to exactly determine the matrix R by computing* $\mathrm{GPK}(\mathbf{e}_i)$ *for each of the elements $\mathbf{e}_i$ of the computational basis of* $\{0,1\}^m$.

Proof We will only prove that the result of the algorithm $\mathrm{GPK}(\mathbf{e}_i)$ is the binary string that determines the i-th row of R, which is an $\mathbf{r}_i$ such that $f(\mathbf{x})_i = (\mathbf{r}_0)_i \oplus \mathbf{r}_i \cdot \mathbf{x}$.

Let us calculate the amplitude of $\mathbf{r}_i$ in the final state of the GPK algorithm using $\mathbf{e}_i$ as marker.

$$|\varphi_4\rangle_n = \frac{1}{N} \sum_{\mathbf{z}\in\{0,1\}^n} \left[\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i \oplus \mathbf{x}\cdot\mathbf{z}} \right] |\mathbf{z}\rangle_n .$$

Therefore, the amplitude of $\mathbf{r}_i$ is:

$$\frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{e}_i \oplus \mathbf{x}\cdot\mathbf{r}_i} = \frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{(\mathbf{r}_0 \oplus \mathbf{r}_i\cdot\mathbf{x})\oplus \mathbf{x}\cdot\mathbf{r}_i}.$$

As $f(\mathbf{x}) \cdot \mathbf{e}_i = (\mathbf{r}_0)_i \oplus \mathbf{r}_i \cdot \mathbf{x}$. If we expand now the expression, we get:

$$\frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{(\mathbf{r}_0 \oplus \mathbf{r}_i\cdot\mathbf{x})\oplus \mathbf{x}\cdot\mathbf{r}_i} = (-1)^{\mathbf{r}_0} \frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{\mathbf{x}\cdot(\mathbf{r}_i \oplus \mathbf{r}_i)} = (-1)^{\mathbf{r}_0},$$

and we are assured to get $\mathbf{r}_i$.

Once again, GPK only allows us to determine R, but tells us nothing about the translation $\mathbf{r}_0$, which we have to classically determine by computing $f(\mathbf{0})$. □

Remark 8 The choice of computing the GPK algorithm with the elements of the computational basis is actually arbitrary. If we chose to do so with any other basis, we would end up getting the matrix of the linear application in said basis.

Again, we see that the GPK is more effective when applied to functions with a certain linear structure.

6 A new algorithm to solve Simon's problem

The last problem we will try to solve by means of the GPK algorithm will be Simon's problem. This problem was introduced by Daniel Simon in [18]. Further generalisations of this problem can be found in [1].

Definition 6 (Simon's problem.) Let $f : \{0,1\}^n \to \{0,1\}^n$ be a Boolean function which satisfies that there is a secret $\mathbf{s} \in \{0,1\}^n$ such that if $f(\mathbf{z}) = f(\mathbf{y})$ then $\mathbf{z} = \mathbf{y} \oplus \mathbf{s}$ or $\mathbf{y} = \mathbf{z}$. This kind of function is called a Simon function, and Simon's problem is to determine such $\mathbf{s}$.

Simon's problem is solved by means of Simon's algorithm, which works as follows.

$\mathbb{STEP}$ 1
$|\psi_0\rangle_{n,n} = |\mathbf{0}\rangle_n \otimes |\mathbf{0}\rangle_n$.
We begin with the $\mathbf{0}$ state.

$\mathbb{STEP}$ 2

$$|\psi_1\rangle_{n,n} = (\mathbf{H}_n \otimes \mathbf{I}_m)\,|\psi_0\rangle_{n,n} = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle_n\right) \otimes |\mathbf{0}\rangle_m.$$

We get a superposition of all the states in the computational basis with the same amplitude in the first register.

$\mathbb{STEP}$ 3

$$|\psi_2\rangle_{n,n} = \mathbf{U}_f |\psi_1\rangle_{n,n} = \frac{1}{\sqrt{N}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle_n \otimes |f(\mathbf{x})\rangle_n.$$

We apply the $\mathbf{U}_f$ gate, so now we can measure the second register.

$\mathbb{STEP}$ 4
We measure the second register, getting a specific value $f(\mathbf{x})$ and collapsing the first register to:
$|\psi_3\rangle_n = \frac{1}{\sqrt{2}} \left(|\mathbf{x}\rangle_n + |\mathbf{x} \oplus \mathbf{s}\rangle_n\right)$, for some $\mathbf{x} \in \{0,1\}^n$, getting a superposition of two related states.

$\mathbb{STEP}$ 5

$$|\psi_4\rangle_n = \mathbf{H}_n |\psi_3\rangle_n = \frac{1}{\sqrt{2N}} \sum_{\mathbf{z}\in\{0,1\}^n} \left((-1)^{\mathbf{z}\cdot\mathbf{x}} + (-1)^{\mathbf{z}\cdot(\mathbf{x}\oplus\mathbf{s})}\right) |\mathbf{z}\rangle_n.$$

If we further analyse this final state, we get:

$$|\psi_4\rangle_n = \frac{1}{\sqrt{2N}} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{z}\cdot\mathbf{x}} \left(1 + (-1)^{\mathbf{z}\cdot\mathbf{s}}\right) |\mathbf{z}\rangle_n.$$

And the amplitude of $\mathbf{z}$ is 0 if $\mathbf{z}\cdot\mathbf{s} = 1$ and $(-1)^{\mathbf{z}\cdot\mathbf{x}}\sqrt{2/N}$ if $\mathbf{z}\cdot\mathbf{s} = 0$.
We thus get a uniform probability distribution over all the states $\mathbf{z}$ such that $\mathbf{z}\cdot\mathbf{s} = 0$ where the probability of each such $\mathbf{z}$ is $p(\mathbf{z}) = 2/N$. The idea is now to iterate this algorithm to get enough independent states of that kind and solve the corresponding linear system.

We will now present an alternative to this algorithm using the GPK. The main idea is that we will arrive at a superposition of the same basic states, but with different amplitudes.

Proposition 2 *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a Simon function and $\mathbf{y} \in \{0,1\}^n$. If we apply* GPK($\mathbf{y}$) *to f we obtain a superposition of the same basic states as in Simon's algorithm.*

Proof We will follow the notation in Definition 2. The final state of the algorithm after discarding the second register in Step 4 will be:

$$|\varphi_4\rangle_n = \frac{1}{N} \sum_{\mathbf{z}\in\{0,1\}^n} \left[\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{y}\oplus\mathbf{x}\cdot\mathbf{z}} \right] |\mathbf{z}\rangle_n.$$

If we study the amplitude of each $\mathbf{z}$ in this superposition, we get two distinct cases. If $\mathbf{z}\cdot\mathbf{s} = 1$, then $\mathbf{z}\cdot\mathbf{x} \neq \mathbf{z}\cdot(\mathbf{x}\oplus\mathbf{s})$, which together with the fact that $f(\mathbf{x}) = f(\mathbf{x}\oplus\mathbf{s})$ implies that the amplitude is 0. In the other case, we cannot easily determine the amplitude of $\mathbf{z}$.

Thus, we end up with a sum of states of the computational basis, which fulfil the same property as those in Simon's algorithm but with different probabilities. □

At this point in Simon's algorithm, we would iterate the algorithm until we get enough linearly independent states to solve the system and find $\mathbf{s}$. Our algorithm improves this.

Theorem 4 (Random marker selection algorithm) *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a Simon function with $\mathbf{s} \in \{0,1\}^n$ as its secret string. If we apply the* GPK *algorithm with random marker selection among $\{0,1\}^n$, then the probability of obtaining a given $\mathbf{z} \in \{0,1\}^n$ as a result is:*

$$p(\mathbf{z}) = \begin{cases} 2/N & \text{if } \mathbf{z}\cdot\mathbf{s} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Proof We will simply prove that choosing the marker at random among all the possible elements of $\{0,1\}^n$ we get a uniform distribution as in the Simon's algorithm.

The probability of choosing a given $\mathbf{y} \in \{0,1\}^n$ is $1/N$, so the final probability of getting a certain $\mathbf{z} \in \{0,1\}^n$ at the end of the algorithm would be:

$$p(\mathbf{z}) = \frac{1}{N} \sum_{\mathbf{y}\in\{0,1\}^n} \alpha_{\mathbf{y}}(\mathbf{z})^2,$$

where $\alpha_{\mathbf{y}}(\mathbf{z})$ is the amplitude of $\mathbf{z}$ in the final state of the GPK. As we have seen, this amplitude is:

$$\alpha_{\mathbf{y}}(\mathbf{z}) = \frac{1}{N} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{y}\oplus\mathbf{x}\cdot\mathbf{z}}.$$

Therefore,

$$\begin{aligned} p(\mathbf{z}) &= \frac{1}{N^3} \sum_{\mathbf{y}\in\{0,1\}^n} \left[\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})\cdot\mathbf{y}\oplus\mathbf{x}\cdot\mathbf{z}} \right]^2 \\ &= \frac{1}{N^3} \sum_{\mathbf{y}\in\{0,1\}^n} \sum_{\mathbf{x}\in\{0,1\}^n} \left[(-1)^{f(\mathbf{x})\cdot\mathbf{y}\oplus\mathbf{x}\cdot\mathbf{z}} \left(\sum_{\mathbf{x}'\in\{0,1\}^n} (-1)^{f(\mathbf{x}')\cdot\mathbf{y}\oplus\mathbf{x}'\cdot\mathbf{z}} \right) \right] \\ &= \frac{1}{N^3} \sum_{\mathbf{y}\in\{0,1\}^n} \sum_{\mathbf{x},\mathbf{x}'\in\{0,1\}^n} (-1)^{\left(f(\mathbf{x})\oplus f(\mathbf{x}')\right)\cdot\mathbf{y}\oplus\left(\mathbf{x}\oplus\mathbf{x}'\right)\cdot\mathbf{z}} \\ &= \frac{1}{N^3} \sum_{\mathbf{x},\mathbf{x}'\in\{0,1\}^n} \left[\sum_{\mathbf{y}\in\{0,1\}^n} (-1)^{\left(f(\mathbf{x})\oplus f(\mathbf{x}')\right)\cdot\mathbf{y}\oplus\left(\mathbf{x}\oplus\mathbf{x}'\right)\cdot\mathbf{z}} \right]. \end{aligned}$$

If we now analyse this final expression, in particular the sum on $\mathbf{y}$ for each particular pair $\mathbf{x}$, $\mathbf{x}'$, we get:

$$\sum_{\mathbf{y}\in\{0,1\}^n} (-1)^{\left(f(\mathbf{x})\oplus f(\mathbf{x}')\right)\cdot\mathbf{y}\oplus\left(\mathbf{x}\oplus\mathbf{x}'\right)\cdot\mathbf{z}}.$$

It is easy to see that if $f(\mathbf{x}) \neq f(\mathbf{x}')$, then this sum is 0, as $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{z}$ are fixed and $\left(f(\mathbf{x}) \oplus f(\mathbf{x}')\right) \cdot \mathbf{y}$ would take value 1 for half of the values of $\mathbf{y}$ and 0 for the other half.

Considering this, in the situation of Simon's problem we would get:

$$\begin{aligned} p(\mathbf{z}) &= \frac{1}{N^3} \sum_{\mathbf{x}\in\{0,1\}^n} \sum_{\mathbf{y}\in\{0,1\}^n} \left((-1)^{(\mathbf{x}\oplus\mathbf{x})\cdot\mathbf{z}} + (-1)^{(\mathbf{x}\oplus\mathbf{x}\oplus\mathbf{s})\cdot\mathbf{z}} \right) \\ &= \frac{1}{N^3} \sum_{\mathbf{x}\in\{0,1\}^n} \sum_{\mathbf{y}\in\{0,1\}^n} \left(1 + (-1)^{\mathbf{s}\cdot\mathbf{z}} \right), \end{aligned}$$

which is 0 if $\mathbf{s}\cdot\mathbf{z}$ is 1 and $2/N$ if $\mathbf{z}\cdot\mathbf{s} = 0$. □

In each iteration of Simon's algorithm, we have the same probability of obtaining each string such that $\mathbf{x}\cdot\mathbf{s} = 0$, including the string $\mathbf{0}$, which does not give us any information. Let us prove that we are reducing the probability of obtaining $\mathbf{0}$ without hurting the balance among the rest of the strings.

Corollary 2 *The* GPK *with random marker selection among* $\{0,1\}^n \setminus \{\mathbf{0}\}$ *improves Simon's algorithm.*

Proof If we chose now $\mathbf{0}$ as marker, we would get $\mathbf{0}$ with complete certainty, so if we eliminate the possibility of choosing $\mathbf{0}$ as a marker we will reduce the probability of getting $\mathbf{0}$ as a result to:

$$\frac{2}{N} - \frac{1}{N} = \frac{1}{N},$$

while the probability of every other state $\mathbf{z}$ such that $\mathbf{z} \cdot \mathbf{s} = 0$ would increase to:

$$2\frac{N-1}{N(N-2)}.$$

□

Remark 9 (Example 2.) In order to show an example of this algorithm, we will take $f : \{0,1\}^4 \to \{0,1\}^4$ as follows:

$$\begin{aligned} f(0000) = f(0101) = 1111 \;\; f(0001) = f(0100) = 0001 \\ f(0010) = f(0111) = 0010 \;\; f(0011) = f(0110) = 0011 \\ f(1000) = f(1101) = 0100 \;\; f(1001) = f(1100) = 0101 \\ f(1010) = f(1111) = 0110 \; f(1011) = f(1110) = 0111, \end{aligned}$$

In this case, the secret is $\mathbf{s} = 0101$. Our first step will be to randomly choose a marker different from $\mathbf{0}$, so each possible marker will have $1/15$ probability. Let us suppose that we chose $\mathbf{y} = 0111$. After applying GPK(0111) to f, we would get the state:

$$\begin{aligned} \frac{1}{16}\big(-4|0000\rangle_4 - 4|0010\rangle_4 - 4|0101\rangle_4 \\ -4|0111\rangle_4 - 4|1000\rangle_4 - 4|1010\rangle_4 - 4|1101\rangle_4 + 12|1111\rangle_4\big), \end{aligned}$$

which is composed only by states of the computational basis $|\mathbf{x}\rangle_4$ with $\mathbf{x} \cdot \mathbf{s} = 0$, but not all with the same amplitude. If we measured we would get one of them, which will probably be 1111. This binary string will translate into an equation:

$$s_0 \oplus s_1 \oplus s_2 \oplus s_3 = 0.$$

where $\mathbf{s} = s_0 s_1 s_2 s_3$. If we repeated the algorithm again, we would choose a new marker $\mathbf{y} = 0011$ and apply GPK(0101) to f, which would in turn give us:

$$|1101\rangle_4.$$

After measuring, we would get 1101 with complete certainty, which will translate into the equation:

$$s_0 \oplus s_1 \oplus s_3 = 0.$$

We would repeat the algorithm until we get three independent linear equations. The main improvement over Simon's algorithm is that we have globally diminished the probability of getting $\mathbf{0}$ after each iteration of the algorithm, as we are not choosing $\mathbf{0}$ as a marker.

Acknowledgements This work was supported by the *Ministerio de Ciencia e Innovación* under Project PID2020-114613GB-I00 (MCIN/AEI/10.13039/501100011033) and by the *Junta de Andalucía* and *ERDF* under Project P20-01056.

Funding Funding for open access publishing: Universidad de Sevilla/CBUA

Data Availability Statement Data sharing was not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors report that there are no competing interests to declare.

References

1. Kaye, P., Laflamme, R., and Mosca, M. An Introduction to Quantum Computing. OUP Oxford, (2007)
2. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proc. R. Soc. Lond. A **439**, 553–558 (1992)
3. Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **454**(1969), 339–354 (1998)
4. Chi, D.P., Kim, J., Lee, S.: Initialization-free generalized Deutsch-Jozsa algorithm. J. Phys. A: Math. Gen. **34**(25), 5251 (2001)
5. Holmes, R.R., Texier, F.: A generalization of the Deutsch-Jozsa quantum algorithm. Far East J. Math. Sci. **9**(3), 319–326 (2003)
6. Bergou, J.A., Herzog, U., Hillery, M.: Quantum filtering and discrimination between sets of Boolean functions. Phys. Rev. Lett. **90**(25), 257901 (2003)
7. Bergou, J.A., Herzog, U., Hillery, M.: Optimal unambiguous filtering of a quantum state: an instance in mixed state discrimination. Phys. Rev. A **71**(4), 042314 (2005)
8. Ballhysa, E. A Generalization of the Deutsch-Jozsa Algorithm and the Development of a Quantum Programming Infrastructure. PhD thesis, MS Thesis, Boğaziçi University, (2004)
9. Nagata, K., Nakamura, T.: Generalization of Deutsch's Algorithm. Int. J. Theor. Phys. **59**(8), 2557–2561 (2020)
10. Bravyi, S., Gosset, D., König, R.: Quantum advantage with shallow circuits. Science **362**(6412), 308–311 (2018)
11. Watts, A. B., Kothari, R., Schaeffer, L., Tal, A.: Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In: Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), ACM, New York (2019)
12. Collins, D., Kim, K.W., Holton, W.C., Sierzputowska-Gracz, H., Stejskal, E.: NMR quantum computation with indirectly coupled gates. Phys. Rev. A **62**(2), 022304 (2000)
13. Chen, A.: Implementation of Deutsch-Jozsa algorithm and de-termination of value of function via Rydberg blockade. Opt. Express **19**(3), 2037–2045 (2011)
14. Nielsen, M.A. and Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2010)
15. Lipton, R.J. and Regan, K.W.: Introduction to Quantum Algorithms via Linear Algebra. MIT Press (2021)

16. Ossorio-Castillo, J. and Tornero, J.M.: Quantum computing from a mathematical perspective: a description of the quantum circuit model. arXiv preprint arXiv:1810.08277 (2018)
17. Bernstein, E., Vazirani, U.: Quantum complexity theory. SIAM J. Comput. **26**(5), 1411–1473 (1997)
18. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(eq5), 1474–1483 (1997)