mathematics

IMPACT FACTOR 2.3

CITESCORE 4.6

Article

A Quantum Algorithm for the Classification of Patterns of Boolean Functions

Theodore Andronikos, Constantinos Bitsakos, Konstantinos Nikas, Georgios I. Goumas and Nectarios Koziris

Special Issue

Quantum Computing and Networking

Edited by

Prof. Dr. Jehn-Ruey Jiang and Dr. YungYu Zhang

MDPI

https://doi.org/10.3390/math13111750

mathematics

MDPI

Article

A Quantum Algorithm for the Classification of Patterns of Boolean Functions

Theodore Andronikos [1,*,†], Constantinos Bitsakos [2,†], Konstantinos Nikas [2,†], Georgios I. Goumas [2,†] and Nectarios Koziris [2,†]

[1] Department of Informatics, Ionian University, 7 Tsirigoti Square, 49100 Corfu, Greece
[2] Computing Systems Laboratory, National Technical University of Athens, Heroon Polytechniou 9, 15780 Zografou, Greece; kbitsak@cslab.ece.ntua.gr (C.B.); knikas@cslab.ece.ntua.gr (K.N.); goumas@cslab.ece.ntua.gr (G.I.G.); nkoziris@cslab.ece.ntua.gr (N.K.)
* Correspondence: andronikos@ionio.gr
† These authors contributed equally to this work.

Abstract: This paper introduces a novel quantum algorithm that is able to classify a hierarchy of classes of imbalanced Boolean functions. The fundamental characteristic of imbalanced Boolean functions is that the proportion of elements in their domain that take the value 0 is not equal to the proportion of elements that take the value 1. For every positive integer, n, the hierarchy contains a class of n-ary Boolean functions defined according to their behavioral pattern. The common trait of all the functions belonging to the same class is that they possess the same imbalance ratio. Our algorithm achieves classification in a straightforward manner as the final measurement reveals the unknown function with a probability of 1.0. Let us also note that the proposed algorithm is an optimal oracular algorithm because it can classify the aforementioned functions with just a single query to the oracle. At the same time, we explain in detail the methodology we followed to design this algorithm in the hope that it will prove general and fruitful, given that it can be easily modified and extended to address other classes of imbalanced Boolean functions that exhibit different behavioral patterns.

Keywords: quantum algorithm; Boolean function; pattern; oracle; the Deutsch–Jozsa algorithm; classification

MSC: 68Q12

Academic Editors: Jehn-Ruey Jiang and YungYu Zhang

Received: 12 March 2025
Revised: 14 May 2025
Accepted: 19 May 2025
Published: 25 May 2025

Citation: Andronikos, T.; Bitsakos, C.; Nikas, K.; Goumas, G.I.; Koziris, N. A Quantum Algorithm for the Classification of Patterns of Boolean Functions. *Mathematics* **2025**, *13*, 1750. https://doi.org/10.3390/math13111750

1. Introduction

The endeavor to construct quantum computers that surpass the capabilities of classical computers poses a significant challenge in our era. It is important to acknowledge that this goal has not yet been realized. However, substantial progress is evident, as illustrated by IBM's advancements with the 127-qubit Eagle [1], the 433-qubit Osprey [2], the 1121-qubit Condor [3], and the latest and most powerful R2 Heron [4]. These developments indicate a swift movement towards the practical application of quantum technology. All these suggest that quantum technology has reached a level of maturity that warrants careful consideration in the development and implementation of algorithms targeting difficult problems.

The imperative to enhance the scale of quantum computers represents the most significant obstacle to their potential application in industrial-scale problems. It has become evident that advancing quantum computers beyond the Noisy Intermediate-Scale Quantum (NISQ) level will necessitate scientific breakthroughs and the resolution of various

technological hurdles. In our assessment, the most promising strategy to address the scaling dilemma currently lies in the advancement of distributed quantum computing systems. In the realm of classical computing, the concept of interlinking smaller processors to distribute computational tasks has emerged as a solution to scaling difficulties. This principle is believed to be equally relevant to quantum computing, where the scaling challenge encourages the exploration of connecting smaller quantum computers. A distributed quantum computing system would comprise a network of quantum nodes, each possessing a specific number of qubits for processing and the capability to transmit both classical and quantum information. Nevertheless, the inherent differences between quantum and classical computing introduce unique challenges, not present in classical networks, in the design of networked quantum computers. Fortunately, recently there have been significant technological advancements in hardware [5,6] and design concepts [7,8]. In fact, very recently, researchers demonstrated distributed quantum computing by employing a photonic network interface to effectively connect two distinct quantum processors, thus creating a unified and fully integrated quantum computer [9,10]. It is our firm belief that we are entering the era of distributed quantum computing.

In this work, we introduce a new quantum algorithm that classifies classes of Boolean functions that are characterized by specific patterns that exhibit imbalance. The fundamental characteristic of these imbalanced Boolean functions is that the proportion of elements in their domain that take the value 0 is not equal to the proportion of elements that take the value 1. We refer to this algorithm as the Boolean Function Pattern Quantum Classifier, or BFPQC for short.

Quantum classification of Boolean functions is important because it bridges theoretical quantum advantages, such as query complexity reductions, with practical applications in cryptography, quantum machine learning, and optimization. In cryptography, Boolean functions in block ciphers must satisfy properties like nonlinearity, balancedness, or correlation immunity. Quantum classification can accelerate the evaluation or optimization of such functions, aiding in cryptanalysis or designing secure systems. Quantum algorithms like the Deutsch–Jozsa algorithm demonstrate exponential speedup for certain Boolean function properties (e.g., determining if a function is constant or balanced). While this is a toy problem, it hints at quantum advantages for classification tasks involving Boolean functions with specific structures. Quantum kernel methods, such as those using Quantum Support Vector Machines (QSVMs), map Boolean inputs into high-dimensional quantum feature spaces via quantum circuits. This may reveal patterns in Boolean functions that are intractable classically, especially for functions with complex dependencies. In machine learning, many real-world datasets involve binary or categorical features. Quantum classification of Boolean functions can enhance classical machine learning models by exploiting quantum parallelism to process high-dimensional binary inputs more efficiently. Variational Quantum Classifiers use parameterized quantum circuits to learn Boolean functions. For example, a VQC can be trained to classify inputs based on a target Boolean function (e.g., parity, majority, or threshold functions), potentially requiring fewer parameters or shallower circuits than classical neural networks for certain tasks. While current hardware limitations (e.g., noise) restrict large-scale deployment, near-term systems excel in niche problems with structured data or specific function properties. As fault-tolerant quantum computers emerge, the ability to classify complex Boolean functions efficiently could revolutionize computational tasks.

We have drawn inspiration mainly from the many sophisticated works studying various extensions of the Deutsch–Jozsa algorithm. Already in [11], the authors examined a multidimensional version of the Deutsch–Jozsa problem. This was further expanded in [12] by considering evenly distributed and evenly balanced functions. Subsequently, in [13], the

Deutsch–Jozsa algorithm was extended for balanced functions in finite Abelian subgroups. Another generalization appeared in [14]. Later, the researchers in [15] generalized the Deutsch–Jozsa problem and gave an optimal algorithm. A more recent clever generalization of the Deutsch–Jozsa algorithm can be found in [16]. Useful applications of the Deutsch–Jozsa algorithm were also obtained in [17,18]. Two particularly interesting works towards establishing a distributed version of the Deutsch–Jozsa algorithm were [19,20]. In a related development, the authors in [21] extended Deutsch's algorithm for binary Boolean functions.

Important general results for query complexity in the oracle model were presented in [22], where a sophisticated upper bound for the number of Boolean functions that can be distinguished with k quantum queries was derived, and in [23], where two methods for proving lower bounds on quantum query complexity were studied. A noteworthy result was obtained in [24], where the problem of quantum learning from a noisy quantum example oracle was investigated, and it was shown that the class of parity functions can be learned in logarithmic time from corrupted quantum queries. We should also mention that oracular algorithms geared towards computing Boolean functions or achieving classification are often encountered in the literature on quantum learning and quantum machine learning. In [25], it was shown that the class of polynomial-size Disjunctive Normal Form expressions is efficiently learnable with respect to the uniform distribution by a quantum algorithm using a quantum example oracle. The authors in [26,27] demonstrated an important correlation between classical and quantum learning for both the models of exact learning and probably approximately correct learning from random examples. In [28], the authors introduce an advanced and sophisticated general technique for quantum concept learning. In a related work [29], the number of quantum queries required to identify an unknown multilinear polynomial of degree d in n variables over a finite field was established. The authors in [30] compared a quantum and a classical machine designed for learning Boolean functions in order to address how a quantum system improves the machine learning behavior, and concluded that the quantum machine has a wider acceptable region, induced by quantum superposition. The researchers in [31] presented quantum algorithms for performing nearest-neighbor classification and k-means clustering that promise significant reductions in their query complexity relative to their classical counterparts. Quantum oracles were shown to reduce the time required to train a deep restricted Boltzmann machine and provide a richer and more comprehensive framework for deep learning than classical computing in [32].

We present our algorithm in the form of game, called the Classification Game, that features the familiar characters of Alice and Bob. It is anticipated that the entertaining aspect of games will facilitate a better understanding of the technical concepts involved. Since their introduction in 1999 [33,34], quantum games have gained considerable popularity, as quantum strategies often outperform classical ones [35,36]. A notable illustration of this is the well-known Prisoners' Dilemma [34], which serves as a prime example and is applicable to various other abstract quantum games [37]. Although quantum games are fun, they can be used to solve critical problems like Quantum Key Distribution, Quantum Secret Sharing, and Quantum Private Comparison (see [38,39]). It is worth noting that many classical systems can be turned into quantum versions, including political frameworks, as demonstrated in recent studies [40].

Contribution. In this paragraph we present what we believe to be the main novelties introduced in this work.

- Numerous sophisticated studies have been published in the literature that employ quantum oracles to classify Boolean functions, such as those that expand upon the Deutsch–Jozsa algorithm. The vast majority of them explore balanced Boolean func-

tions. However, as far as we are aware, there has been no previous research dedicated to imbalanced Boolean functions, which are characterized by an unequal number of elements in their domain that yield the values 0 and 1. This article introduces a novel quantum algorithm designed to classify a specific hierarchy of imbalanced Boolean function classes. For each positive integer, n, this hierarchy includes a class of Boolean functions, which are defined according to their behavioral characteristics. A defining feature of all functions within the same class is their shared imbalance ratio.

- Our algorithm achieves classification in a straightforward manner, as the final measurement determines the unknown function with a probability of 1.0. A thorough complexity analysis of our algorithm is given in Section 5.1. This analysis proves that our algorithm is an optimal oracular algorithm, capable of conclusively classifying all Boolean functions belonging to the designating hierarchy with a probability of 1.0 using just a single query to the oracle. At the same time, we show that our algorithm is superior to any deterministic classical algorithm for the same task.
- Of equal importance is the detailed explanation of the methodology followed in the development of this algorithm that we provide in Section 4. This is done with the expectation that it will prove both general and beneficial, as it can be readily adapted and expanded to tackle other classes of imbalanced Boolean functions that display varying behavioral patterns. The intuition of our methodology can be distilled in the phrase "from behavior to pattern vectors and then to unitary classifiers". By appropriately modifying Definition 9 to include different pattern bases, one may construct unitary classifiers for these patterns, leading to new classification algorithms.

Organization

This article is structured in the following way. Section 1 introduces the topic and includes references to the relevant literature. Section 2 offers a brief overview of key concepts, which serves as a basis for grasping the technical details. Section 3 contains a comprehensive exposition to our algorithm, including a detailed small-scale example to build intuition. The general form of the algorithm is formally presented in Section 4. Finally, the paper wraps up with a summary and a discussion of the algorithm's nuances in Section 5.

2. Notation and Terminology

2.1. Boolean Functions and Oracles

Let us first fix the notation and terminology we shall be using in the rest of this paper.

- $\mathbb{B}$ is the binary set $\{0, 1\}$.
- A bit vector, $\mathbf{b}$, of length n is a sequence of n bits: $\mathbf{b} = b_{n-1} \dots b_0$. Two special bit vectors are the zero and the one bit vectors, denoted by $\mathbf{0}$ and $\mathbf{1}$, in which all the bits are zero and one, respectively : $\mathbf{0} = 0 \dots 0$ and $\mathbf{1} = 1 \dots 1$.
- To make clear, when we refer to the bit vector $\mathbf{b} \in \mathbb{B}^n$, we write $\mathbf{b}$ in boldface. Often, it is convenient to view $\mathbf{b}$ as the binary representation of the integer b.
- Each bit vector, $\mathbf{b} \in \mathbb{B}^n$, can also be viewed as the binary representation of one of the 2^n basis kets that form the computational basis of the 2^n-dimensional Hilbert space.

Definition 1 (Boolean function)**.** *A* Boolean *function, f, is a function from $\mathbb{B}^n$ to $\mathbb{B}$, $n \geq 1$.*

Oracles are an important concept in quantum computing and play a crucial role in many quantum algorithms. An oracle is a black box that encodes a specific function or information into a quantum circuit, allowing quantum algorithms to solve problems more efficiently than classical algorithms in certain cases. It is used to evaluate the function

or check a condition without revealing the internal details of how the function works. In quantum algorithms, oracles are often used to mark solutions to a problem or to provide information about a function's behavior. For the purposes of our work, the following definition suffices.

Definition 2 (Oracle and unitary transform)**.** *An oracle is a black box implementing a Boolean function, f. The idea here is that, being a black box function, we know nothing about its inner workings apart from the fact that it works correctly. Thus, it can be used for the construction of a unitary transform, U_f, that captures the behavior of f.*

Henceforth, we shall assume that the corresponding unitary transform, U_f, implements the standard schema

$$U_f : |y\rangle\,|\mathbf{x}\rangle \rightarrow |y \oplus f(\mathbf{x})\rangle\,|\mathbf{x}\rangle\,. \tag{1}$$

In the literature, this type of oracle is sometimes referred to as a Deutsch–Jozsa oracle. We note in passing that there also other variations of oracles, such as the Grover oracle, which is typically used to mark solutions to a problem. In this work, every oracle and unitary transform is assumed to satisfy (1) and is used to deduce a function from its behavior. The standard measure of complexity in oracular algorithms is the query complexity, i.e., the number of queries to the oracle used by the algorithm.

For completeness, we recall the states $|+\rangle$ and $|-\rangle$, which are defined as

$$|+\rangle = H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{2}$$

$$|-\rangle = H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3}$$

To obtain any useful information from schema (1), we set $|y\rangle$ equal to $|-\rangle$, in which case (1) takes the following familiar form:

$$U_f : |-\rangle\,|\mathbf{x}\rangle \rightarrow (-1)^{f(\mathbf{x})}\,|-\rangle\,|\mathbf{x}\rangle\,. \tag{4}$$

Figures 1 and 2 give a visual outline of the unitary transforms, U_f, that implement schemata (1) and (4), respectively. In all the quantum circuits used in this work, including those depicted in Figures 1 and 2, the following conventions are used.

- The way of ordering the qubits adheres to the Qiskit [41] convention, i.e., the little-endian qubit indexing convention, where the least significant qubit is at the top of the figure and the most significant at the bottom.
- IR is the quantum input register that contains n qubits.
- OR is the single-qubit output register that is initialized to arbitrary state $|y\rangle$ in Figure 1 and to state $|-\rangle$ in Figure 2.
- U_f is the unitary transform. Its precise mathematical expression depends on f and is hidden. However, it is taken for granted that is satisfies relation (1) in Figure 1 and relation (4) in Figure 2.

We mention that in the literature it is very common to use the word "promise" when referring to a particular property of the Boolean function f, meaning that we are guaranteed, or, if you prefer, we are certain with a probability of 1.0 that f satisfies the property in question. A prominent such example comes from the Deutsch–Jozsa algorithm, where we are given the promise that f is either constant, or balanced.

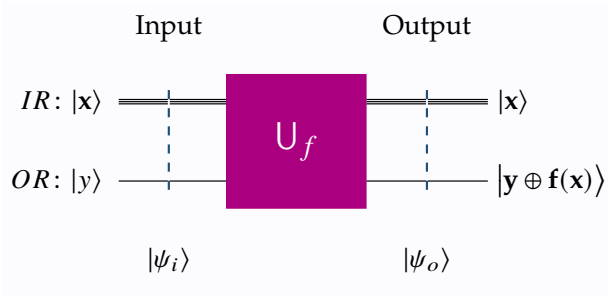

Figure 1. This figure shows the unitary transform, U_f, which is based on the oracle for the function f and implements standard schema (1).

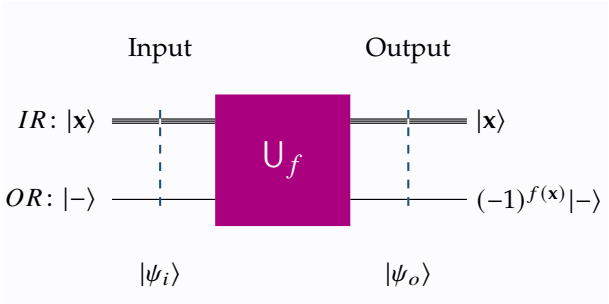

Figure 2. This figure shows the unitary transform, U_f, again based on the oracle for the function f, but now implementing schema (4).

Extending the operation of addition modulo 2 to bit vectors is a natural and fruitful generalization.

Definition 3 (Bitwise addition modulo 2)**.** *Given two bit vectors,* $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$*, with* $\mathbf{x} = x_{n-1} \dots x_0$ *and* $\mathbf{y} = y_{n-1} \dots y_0$*, we define their bitwise sum modulo* 2*, denoted by* $\mathbf{x} \oplus \mathbf{y}$*, as*

$$\mathbf{x} \oplus \mathbf{y} \coloneqq (x_{n-1} \oplus y_{n-1}) \dots (x_0 \oplus y_0) \,. \tag{5}$$

Following the standard approach, we use the same symbol, $\oplus$, to denote the operation of addition modulo 2 two between bits, and the bitwise sum modulo 2 between two bit vectors because the context always makes clear the intended operation.

2.2. A Brief Recap of the Deutsch–Jozsa Algorithm

In this subsection, we recall a few standard operations between bit vectors that will facilitate the forthcoming exposition, and give a brief reminder of the Deutsch–Jozsa algorithm.

Definition 4 (Inner product modulo 2)**.** *The* inner product modulo 2 *is a function from* $\mathbb{B}^n \times \mathbb{B}^n$ *to* $\mathbb{B}$ *that takes as inputs two bit vectors,* $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$*, and returns their inner product denoted by* $\mathbf{x} \bullet \mathbf{y}$*. If* $\mathbf{x} = x_{n-1} \dots x_0$ *and* $\mathbf{y} = y_{n-1} \dots y_0$*, then* $\mathbf{x} \bullet \mathbf{y}$ *is defined as*

$$\mathbf{x} \bullet \mathbf{y} \coloneqq x_{n-1}y_{n-1} \oplus \dots \oplus x_0 y_0 \,, \tag{6}$$

where $\coloneqq$ *stands for "is defined as", and* $\oplus$ *is addition modulo* 2*.*

The fundamental relation that expresses the n-fold Hadamard transform of an arbitrary basis ket $|\mathbf{x}\rangle$ uses the inner product modulo 2. Its proof can be found in most standard textbooks, e.g., [42,43].

$$H^{\otimes n}|\mathbf{x}\rangle = 2^{-\frac{n}{2}} \sum_{\mathbf{z} \in \mathbb{B}^n} (-1)^{\mathbf{z} \bullet \mathbf{x}} |\mathbf{z}\rangle \,. \tag{7}$$

The Deutsch–Jozsa algorithm is a famous quantum algorithm that solves a specific problem faster than any classical algorithm. It determines whether the given Boolean function $f\colon \mathbb{B}^n \to \mathbb{B}$ is constant (produces the same output for all inputs) or balanced (outputs 0 for half the inputs and 1 for the other half) with a single function evaluation, compared with up to $2^{n-1}+1$ evaluations classically.

The algorithm takes as input an oracle for the unknown function, f. The oracle is of the Deutsch–Jozsa variation and the function f is promised to be either constant or balanced. The goal is to establish whether f is constant or balanced with a single oracle query. The abstract quantum circuit in Figure 3 visualizes the implementation of the Deutsch–Jozsa algorithm.

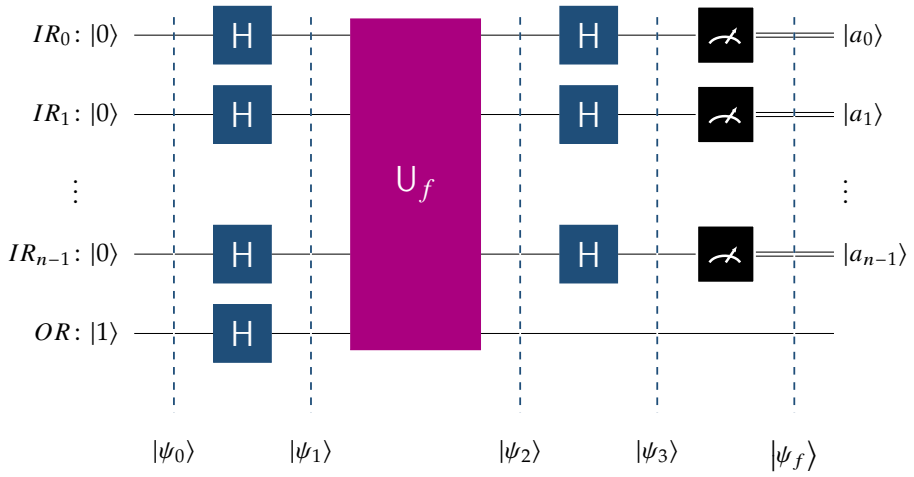

Figure 3. This figure visualizes the abstract quantum circuit for the implementation of the Deutsch–Jozsa algorithm.

In the above Figure 3, the following notation is employed.

- IR is the quantum input register that contains n qubits and starts its operation at state $|0\rangle^{\otimes n}$.
- OR is the single-qubit output register initialized to state $|1\rangle$.
- H is the Hadamard transform.
- U_f is the unitary transform corresponding to the oracle for the unknown function, f. The latter is promised to be constant or balanced.

Referring to Figure 3 and recalling that ordering the qubits adheres to the Qiskit [41] convention, we may express the initial state, $|\psi_0\rangle$, as shown below

$$|\psi_0\rangle = |1\rangle\, |0\rangle^{\otimes n}\,.$$

Applying Hadamard gates to all qubits leads to the next state, $|\psi_1\rangle$, which, by taking into into account (7) for $|\mathbf{x}\rangle = |0\rangle^{\otimes n}$ and (3), can be written as

$$|\psi_1\rangle = H|1\rangle\, H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(2^{-\frac{n}{2}}\sum_{\mathbf{z}\in\mathbb{B}^n}|\mathbf{z}\rangle\right) = 2^{-\frac{n}{2}}\,|-\rangle\sum_{\mathbf{z}\in\mathbb{B}^n}|\mathbf{z}\rangle\,.$$

The oracle's action implementing schema (4) evolves the state to $|\psi_2\rangle$

$$|\psi_2\rangle = 2^{-\frac{n}{2}}\,|-\rangle\sum_{\mathbf{z}\in\mathbb{B}^n}(-1)^{f(\mathbf{z})}|\mathbf{z}\rangle\,.$$

Intuitively, this means that the oracle has encoded the output values of the Boolean function f, into the relative phase $\sum_{\mathbf{z}\in\mathbb{B}^n}(-1)^{f(\mathbf{z})}$ of state $|\psi_2\rangle$. It is at this point exactly that

we encounter the *phase kickback* phenomenon. Phase kickback is a fundamental trait of many quantum algorithms because an operation on the ancilla qubit induces a phase shift on the control qubits. In our case, the control qubits are the n qubits of the quantum input register, IR, and the ancilla qubit is the single-qubit output register, OR. To successfully employ this technique, it is necessary for the ancilla qubit to be in state $\frac{|0\rangle-|1\rangle}{\sqrt{2}} = H|1\rangle = |-\rangle$ (recall Equation (3)). Using phase kickback is crucial because the oracle instead of writing does $f(\mathbf{z})$ explicitly to a qubit, and encodes it as a relative phase $(-1)^{f(\mathbf{z})}$. This approach not only forfeits the need for additional storage qubits, but also allows the desired interference to happen cleanly. An obvious way to generate $|-\rangle$ is to initialize the ancilla qubit to state $|1\rangle$ and, subsequently, apply the Hadamard transform, H, as depicted in the quantum circuit of Figure 3. Modern-day quantum computers allow for the immediate initialization of qubits to states other than $|0\rangle$ and $|1\rangle$, such as $|-\rangle$. We have taken advantage of this capability by directly initializing to $|-\rangle$ the output register, OR, playing the role of the ancilla qubit, in all of the quantum classification circuits shown in the rest of this paper. The classification algorithm we present in this work also relies on the Deutsch–Jozsa oracle type and utilizes the phase kickback technique to encode the behavioral pattern of the unknown function, f, into the relative state of the quantum input register, IR.

Subsequently, by applying Hadamard gates to the n qubits of the quantum input register, IR, we drive the system into the $|\psi_3\rangle$ described below

$$|\psi_3\rangle = 2^{-n}\ |-\rangle \sum_{\mathbf{w}\in\mathbb{B}^n} \sum_{\mathbf{z}\in\mathbb{B}^n} (-1)^{f(\mathbf{z})\oplus \mathbf{w}\bullet\mathbf{z}} |\mathbf{w}\rangle\ . \tag{8}$$

Focusing on the amplitude of the basis state $\mathbf{w} = |0\rangle^{\otimes n}$ within state $|\psi_3\rangle$, we see that it is given by the next equation

$$2^{-n} \sum_{\mathbf{z}\in\mathbb{B}^n} (-1)^{f(\mathbf{z})}\ , \tag{9}$$

which enables us to distinguish between the following two antidiametrical cases.

(**DJ**$_1$) If f is constant, then (9) reduces to ± 1.0.
(**DJ**$_2$) If f is balanced, then (9) reduces to 0.0.

Thus, if we measure $|0\rangle^{\otimes n}$ we are absolutely certain that f is constant, whereas any other outcome means that f is balanced.

From a historical perspective, this algorithm was one of the first to show a clear quantum advantage, paving the way for more complex quantum algorithms like Shor's and Grover's. Its speedup is evident because the quantum algorithm uses 1 query vs. up to $2^{n-1}+1$ classically. Furthermore, it demonstrates quantum parallelism by evaluating f on all inputs simultaneously, and interference by amplifying desired outcomes.

3. The Basic Concepts Behind the BFPQC Algorithm

In this paper, we introduce a new quantum algorithm that differentiates and classifies a class of Boolean function that is characterized by a specific collection of patterns demonstrating imbalance. In view of its intended purpose, we call this algorithm the Boolean Function Pattern Quantum Classifier, or BFPQC for short. The current section gives the definitions regarding the main concepts, and presents a toy scale example illustrating its operation.

By giving rise to different elements of the computational basis with a probability of 1.0. Our algorithm is an oracular algorithm because it relies on a oracle to achieve the classification. Its efficiency is demonstrated by the fact that it is optimal because it requires just one single query to complete its task.

Here we solve what is commonly referred to in the quantum literature as a promise problem, i.e., a problem where the input is promised to belong to a specific set. Promise algorithms are not required to work correctly on any input that does not satisfy the promise. Many quantum algorithms are designed to solve promise problems. For example, in the Deutsch–Jozsa algorithm, the promise is that the function is either constant or balanced. The Deutsch–Jozsa algorithm is designed to distinguish between these two cases efficiently, but it does not need to handle functions that are neither constant nor balanced. The same applies to our case: the BFPQC algorithm can correctly handle any function that belongs to a rigorously defined hierarchy, but it will not output the correct answer if this is not the case.

Our plan of action consists of the following successive steps.

($\mathbf{S}_1$) We focus on imbalanced Boolean functions, i.e., those with the property that the number of elements in their domain that take the value 0 is not equal to the number of elements that take the value 1.

($\mathbf{S}_2$) We employ the concept of pattern vectors to capture the behavior of imbalanced Boolean functions. For each positive integer $n \geq 1$, we define a set of 2^{2n} pattern vectors that all have an equal imbalance ratio, which is always $< \frac{1}{2}$.

($\mathbf{S}_3$) Identifying an appropriate set of pattern vectors enables the construction of the corresponding unitary transform that accomplishes the classification.

Definition 5 (Pattern vector)**.** *Given the Boolean function $f\colon \mathbb{B}^n \to \mathbb{B}$, $n \geq 1$, we define the concept of the unique* pattern vector *that encodes the behavior of f.*

- *The pattern vector $\mathbf{p} = p_{2^n-1} \ldots p_1\ p_0$ of f is the element of $\mathbb{B}^{2^n}$, such that $p_i = f(\mathbf{i})$, where $\mathbf{i}$ is the binary bit vector representing integer i, $0 \leq i \leq 2^n - 1$. In other words, the pattern vector, $\mathbf{p}$, lists the binary values of $f(\mathbf{i})$ as $\mathbf{i}$ ranges over $\mathbb{B}^n$. To enhance comprehension, we visualize the details below.*

Position:	$2^n - 1$	…	1	0
	↓	…	↓	↓
Position in binary:	**1…11**	…	**0…01**	**0…00**
	↓	…	↓	↓
Function value:	$f(\mathbf{1}\ldots\mathbf{11})$	…	$f(\mathbf{0}\ldots\mathbf{01})$	$f(\mathbf{0}\ldots\mathbf{00})$
	↓	…	↓	↓
Pattern bit:	p_{2^n-1}	…	p_1	p_0

- *Given the pattern vector $\mathbf{p} = p_{2^n-1} \ldots p_1\ p_0$ of f, its* negation, *denoted by $\overline{\mathbf{p}}$, is the pattern vector $\overline{p_{2^n-1}} \ldots \overline{p_1}\ \overline{p_0}$, which corresponds to the function $\overline{f}$.*

It is clear by the preceding Definition 5 that there is a one-to-one correspondence between Boolean functions and patterns vectors. We could say that a Boolean function and its pattern vector are the two sides of the same coin. Therefore, just as knowing the behavior of a Boolean function enables the construction of its pattern vector, conversely, the pattern vector contains all the information necessary to reconstruct the Boolean function. This duality is emphasized by Figure 4.

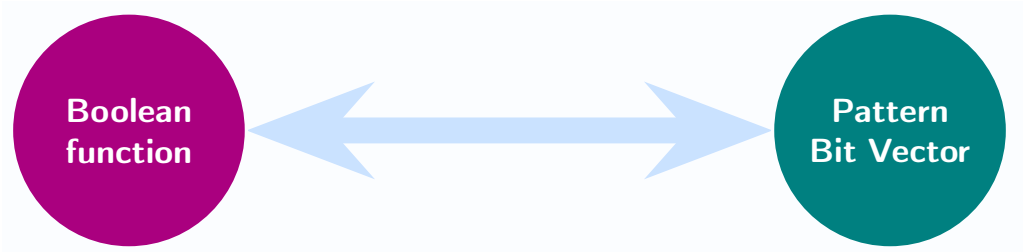

Figure 4. The duality between Boolean functions and their pattern bit vectors.

Definition 6 (Orthogonal pattern vectors)**.** *Consider the distinct pattern vectors* $\mathbf{p}$ *and* $\mathbf{q}$*, corresponding to the Boolean functions* $f, g\colon \mathbb{B}^n \to \mathbb{B}$*.* $\mathbf{p}$ *and* $\mathbf{q}$ *are* orthogonal *if* $\mathbf{p} \oplus \mathbf{q}$ *contains* 2^{n-1} 0 *s and* 2^{n-1} 1 *s.*

Definition 7 (Imbalance ratio)**.** *Given a pattern vector,* $\mathbf{p}$*, of length* 2^n*, let* $\mathbb{0}_{\mathbf{p}}$ *and* $\mathbb{1}_{\mathbf{p}}$ *denote the number of* 0*s and* 1*s appearing in* $\mathbf{p}$*. The* imbalance ratio *of* $\mathbf{p}$ *is defined as*

$$\rho := \min\left\{\frac{\mathbb{0}_{\mathbf{p}}}{2^n}, \frac{\mathbb{1}_{\mathbf{p}}}{2^n}\right\}. \tag{10}$$

If $\mathbf{p}$ *is the pattern vector of* f*, we shall also say that* ρ *is the imbalance ratio of* f*. In the same spirit, if* P *and* F *are a collection of pattern vectors and a collection of Boolean functions with the common imbalance ratio* ρ*, respectively, we will speak of* ρ *being the imbalance ratio of* P *and* F*.*

As we pointed out previously, we visualize the execution of the BFPQC algorithm as the evolution of the Classification Game played between our prolific stars Alice and Bob, according to the following rules.

- ($\mathbf{G}_1$) Bob is free to choose any Boolean function, provided that it belongs to the promised class of functions.
- ($\mathbf{G}_2$) Bob wins the game if Alice fails to recognize the chosen function with one try. Otherwise, Alice is the winner.
- ($\mathbf{G}_3$) In terms of implementing the game as a quantum circuit, Bob chooses the hidden oracle, while Alice furnishes the classifier.

Before we proceed to introduce more technical machinery, we give a toy scale example to build intuition.

Example 1 (A toy scale example)**.** *Let us consider the following two families of Boolean functions defined on* $\mathbb{B}^2$*.*

$$\left\{\begin{array}{l} f_0(x_1, x_0) := \overline{x_1} \wedge \overline{x_0} \\ f_1(x_1, x_0) := \overline{x_1} \wedge x_0 \\ f_2(x_1, x_0) := x_1 \wedge \overline{x_0} \\ f_3(x_1, x_0) := x_1 \wedge x_0 \end{array}\right\} \tag{11}$$

$$\left\{\begin{array}{l} g_0(x_1, x_0) := x_1 \vee x_0 \\ g_1(x_1, x_0) := x_1 \vee \overline{x_0} \\ g_2(x_1, x_0) := \overline{x_1} \vee x_0 \\ g_3(x_1, x_0) := \overline{x_1} \vee \overline{x_0} \end{array}\right\} \tag{12}$$

Their truth values and pattern vectors are given in Tables 1 and 2 below.

Table 1. The truth values and the pattern vectors of f_0, f_1, f_2, and f_3.

	00	**01**	**10**	**11**	**Pattern Vector**
f_0	1	0	0	0	0001
f_1	0	1	0	0	0010
f_2	0	0	1	0	0100
f_3	0	0	0	1	1000

Table 2. The truth values and the pattern vectors of g_0, g_1, g_2, and g_3.

	00	**01**	**10**	**11**	**Pattern Vector**
g_0	0	1	1	1	1110
g_1	1	0	1	1	1101
g_2	1	1	0	1	1011
g_3	1	1	1	0	0111

The four functions f_0, f_1, f_2, and f_3 exhibit a common pattern: for precisely one element, $\mathbf{x} \in \mathbb{B}^2$, their value is 1*, while for the remaining three elements their value is* 0*. Symmetrically, the four functions g_0, g_1, g_2, and g_3 exhibit an analogous motif, i.e., for precisely one element, $\mathbf{x} \in \mathbb{B}^2$, their value is* 0*, while for the remaining three elements their value is* 1*. Obviously, this is because $g_i = \overline{f_i}$, $0 \leq i \leq 3$. The imbalance ratio, ρ, for both families is the same, namely $\rho = \frac{1}{4}$. The four pattern vectors shown in Table 1 are pairwise orthogonal and constitute the set $P_2 = \{0001, 0010, 0100, 1000\}$. An important observation at this point is that, although f_i and g_i are logically different, within our quantum context f_i and g_i are indistinguishable because they lead to the same state.*

For future reference, we gather the Boolean functions f_i into one set, which we call F_2. Given any function in F_2, it is easy to construct the corresponding oracle using standard quantum gates. Accordingly, it is possible to distinguish among the four Boolean functions f_i. Hence, given the promise that the unknown function, f, is one of the above four Boolean functions, and having the corresponding oracle, the aim of the Classification Game is to construct a quantum circuit that allows Alice to win with absolute certainty, i.e., with a probability of 1.0*. The initial segment of such a circuit is shown in Figure 5. U_f is the oracle of the hidden function, chosen by Bob. After the application of the unitary transform, U_f, the state of the quantum input register, IR, will be $|\psi_2\rangle$. As is the norm in such cases, we ignore from now on the output register, OR, since its state remains $|-\rangle$. It is quite straightforward to verify the precise dependency of $|\psi_2\rangle$ on each of the functions in F_2, which is shown in Table 3. The important observation here is that each of the four f_i leads to a different $|\psi_2\rangle$, which means that we can distinguish and classify them. However, as expected, state $|\psi_2\rangle$ is the same for each pair of functions f_i and g_i, which means that they are indistinguishable.*

Alice now employs a unitary transform that can differentiate among the four Boolean functions f_0, f_1, f_2, and f_3, such as Q_2. The matrix representation of Q_2 is given by Equation (13). It is easy to verify that the action of Q_2 on the four possible states, $|\psi_2\rangle$, leads to the states shown in Table 4, which are precisely the basis kets of the computational basis B_4. It is straightforward to build Q_2 using standard quantum gates readily available in contemporary quantum computers. Below we show in Figure 6 such a construction that requires only Hadamard, Z, and controlled-Z gates:

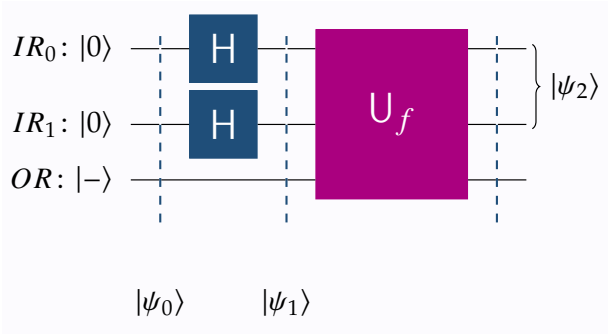

Figure 5. This figure visualizes the initial segment of a quantum circuit that can be used for the classification of the functions in F_2.

Table 3. The four Boolean functions f_0, f_1, f_2, and f_3 drive the quantum circuit of Figure 5 to the four different states shown below. In contrast, f_i and g_i, $0 \leq i \leq 3$, are indistinguishable because they lead to the same state.

The State $\|\psi_2\rangle$	
Function	$\|\psi_2\rangle$
f_0	$-\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle + \frac{1}{2}\|10\rangle + \frac{1}{2}\|11\rangle$
f_1	$-\frac{1}{2}\|00\rangle - \frac{1}{2}\|01\rangle + \frac{1}{2}\|10\rangle + \frac{1}{2}\|11\rangle$
f_2	$-\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle - \frac{1}{2}\|10\rangle + \frac{1}{2}\|11\rangle$
f_3	$-\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle + \frac{1}{2}\|10\rangle - \frac{1}{2}\|11\rangle$
g_0	$-\frac{1}{2}\|00\rangle - \frac{1}{2}\|01\rangle - \frac{1}{2}\|10\rangle - \frac{1}{2}\|11\rangle$
g_1	$-\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle - \frac{1}{2}\|10\rangle - \frac{1}{2}\|11\rangle$
g_2	$-\frac{1}{2}\|00\rangle - \frac{1}{2}\|01\rangle + \frac{1}{2}\|10\rangle - \frac{1}{2}\|11\rangle$
g_3	$-\frac{1}{2}\|00\rangle - \frac{1}{2}\|01\rangle - \frac{1}{2}\|10\rangle + \frac{1}{2}\|11\rangle$

Regarding the schematic of Figure 5, we note the following.

- *IR_0 is the least significant qubit and IR_1 is the most significant qubit of the quantum input register, IR, that contains 2 qubits.*
- *OR is the single-qubit output register that is initialized to state $|-\rangle$.*
- *H is the Hadamard transform.*
- *U_f is the unitary transform that is based on the oracle for the unknown function, f, and satisfies relation (4).*

$$Q_2 = (H \otimes H)\ CZ\ (Z \otimes Z)\ (H \otimes H) \tag{13}$$

$$Q_2 = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \tag{14}$$

Therefore, the quantum algorithm that classifies each Boolean function contained in F_2 can be visualized by the quantum circuit depicted in Figure 7. Alice surely wins because the action of the classifier Q_2 results in the final state of the system being one of the four basis kets of the computational basis $B_4 = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Specifically, if the oracle encodes f_i, the final state will be $|\mathbf{i}\rangle$, where $\mathbf{i}$ is the binary representation of the index i, $0 \leq i \leq 3$. Therefore, upon the final measurement, Alice will surmise the correct hidden function with a probability of 1.0.

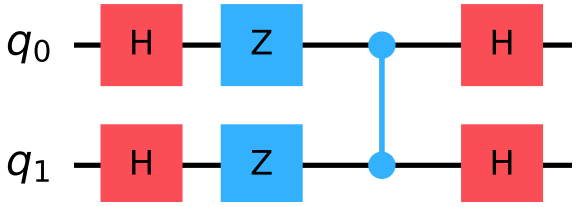

Figure 6. This figure shows the quantum circuit that implements the unitary transform, Q_2, for the classification of the Boolean functions in F_2.

Table 4. This table contains the outcome of the action of Q_2 on the four possible states, $|\psi_2\rangle$, outlined in Table 3.

	$\mathbf{f_0}$	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$
$\mathbf{Q_2}$ action on $\|\boldsymbol{\psi_2}\rangle$	$Q_2 \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	$Q_2 \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	$Q_2 \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	$Q_2 \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$
Outcome	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \|00\rangle$	$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \|01\rangle$	$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \|10\rangle$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \|11\rangle$

An actual implementation of the abstract quantum circuit of Figure 7 in Qiskit [41] using the oracle for the function f_2 is depicted in Figure 8. Let us clarify that in all the figures in this paper the qubit numbering follows the "little-endian" convention, where is the rightmost qubit is the least significant qubit (LSQ), and the leftmost qubit is the most significant qubit (MSQ).

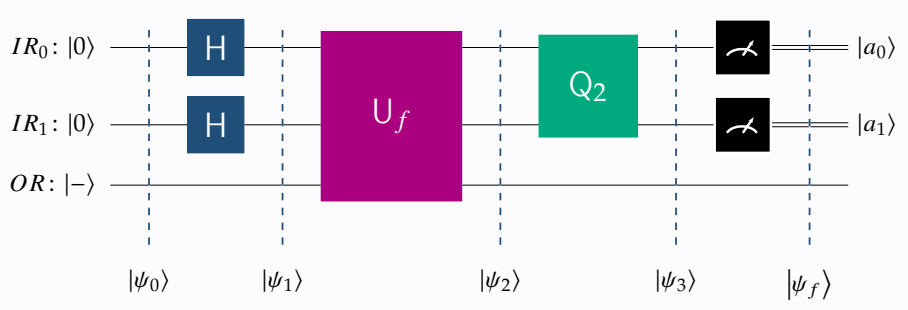

Figure 7. This figure visualizes the abstract quantum circuit that implements the BFPQC algorithm for the classification of the functions contained in F_2.

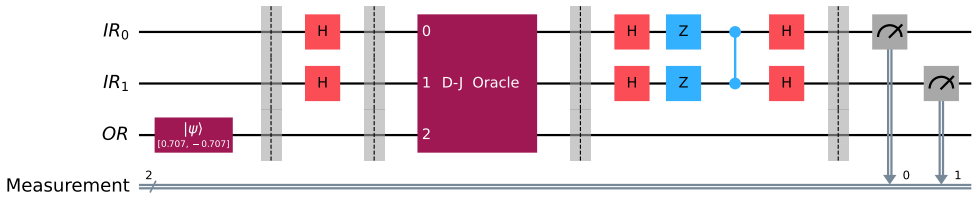

Figure 8. This figure shows the quantum circuit that implements the BFPQC algorithm for the classification of the Boolean functions in F_2 using the oracle for f_2.

Figure 9 shows the state of the quantum circuit of Figure 8 after the oracle but before the action of Q_2. The main property of Q_2 is its ability to distinguish and classify the superpositions of the basis kets of the computational basis $B_4 = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ in which exactly three kets appear

with same sign and the fourth ket appears with the opposite sign. Thus, the state of the quantum circuit after the action of Q_2 will be $|10\rangle$, as depicted in Figure 10.

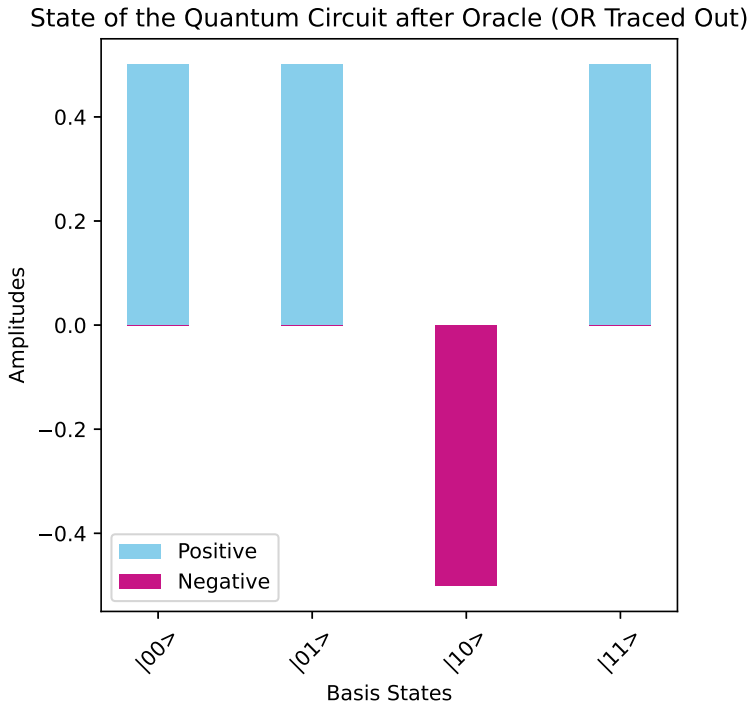

Figure 9. This is the state of the quantum circuit of Figure 8 before the action of Q_2.

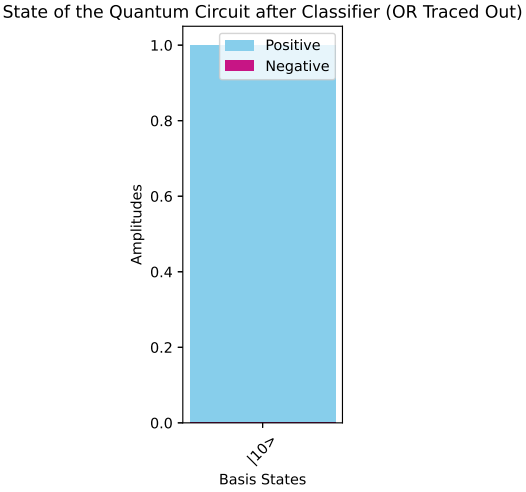

Figure 10. This is the state of the quantum circuit of Figure 8 after the action of Q_2.

We observe that the four different Boolean functions f_i give rise to four different orthonormal states, $|\psi_2\rangle$. Thus, the task of differentiating among the four Boolean functions f_i can be reduced to the task of using a unitary transform that maps the four orthonormal states $|\psi_2\rangle$ to the computational basis $B_4 = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

4. The General Form of the BFPQC Algorithm

In this section we present the general form of the BFPQC algorithm. For this purpose, we extend the definitions given in the previous section.

Definition 8 (Pattern basis). *A* pattern basis *of rank* $2n$, $n \geq 1$ *is a collection of* 2^{2n} *pairwise orthogonal pattern vectors of length* 2^{2n}, *denoted by* P_{2n}.

The initial pattern basis, P_2, is the set consisting of the following four pairwise orthogonal pattern vectors:

$$P_2 := \{0001, 0010, 0100, 1000\} . \tag{15}$$

Starting from P_2, we may define an infinite hierarchy of pattern bases. The details are explained below.

Definition 9 (Pattern hierarchy)**.** *We recursively define a* hierarchy *of pattern bases P_{2n}, $n \geq 1$ as follows.*

($\boldsymbol{PH}_0$) *If $n = 1$, the corresponding pattern basis is the set P_2, as defined by (15).*

($\boldsymbol{PH}_1$) *Let P_{2n} contain the pattern vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m$; then, the pattern basis $P_{2(n+1)}$ consists of the pattern vectors with the following syntax structure:*

$$\begin{aligned} P_{2(n+1)} := \{\ & \mathbf{p}_0\mathbf{p}_0\mathbf{p}_0\overline{\mathbf{p}}_0, \mathbf{p}_1\mathbf{p}_1\mathbf{p}_1\overline{\mathbf{p}}_1, \ldots, \mathbf{p}_m\mathbf{p}_m\mathbf{p}_m\overline{\mathbf{p}}_m \\ & \mathbf{p}_0\mathbf{p}_0\overline{\mathbf{p}}_0\mathbf{p}_0, \mathbf{p}_1\mathbf{p}_1\overline{\mathbf{p}}_1\mathbf{p}_1, \ldots, \mathbf{p}_m\mathbf{p}_m\overline{\mathbf{p}}_m\mathbf{p}_m, \\ & \mathbf{p}_0\overline{\mathbf{p}}_0\mathbf{p}_0\mathbf{p}_0, \mathbf{p}_1\overline{\mathbf{p}}_1\mathbf{p}_1\mathbf{p}_1, \ldots, \mathbf{p}_m\overline{\mathbf{p}}_m\mathbf{p}_m\mathbf{p}_m, \\ & \overline{\mathbf{p}}_0\mathbf{p}_0\mathbf{p}_0\mathbf{p}_0, \overline{\mathbf{p}}_1\mathbf{p}_1\mathbf{p}_1\mathbf{p}_1, \ldots, \overline{\mathbf{p}}_m\mathbf{p}_m\mathbf{p}_m\mathbf{p}_m, \ \} \end{aligned} \tag{16}$$

An easy conclusion of the above definition is that every P_{2n}, $n \geq 1$, contains 2^{2n} pairwise orthogonal pattern vectors. Henceforth, we shall assume that the 2^{2n} pattern vectors contained in P_{2n} are enumerated as $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{2^{2n}-1}$ according to the order prescribed by Formula (16).

Example 2 (Pattern basis P_4)**.** *To facilitate the understanding of Definition 9, we list P_4 in Table 5 to show how it is derived from $P_2 = \{0001, 0010, 0100, 1000\}$.*

Table 5. This table contains the pattern vectors of P_4.

$\mathbf{P_2}$ **Pattern Vectors**	$\mathbf{p}_0 = 0001$	$\mathbf{p}_1 = 0010$	$\mathbf{p}_2 = 0100$	$\mathbf{p}_3 = 1000$
$\mathbf{p}_i\mathbf{p}_i\mathbf{p}_i\overline{\mathbf{p}}_i$	$\mathbf{r}_0$: 0001000100011110	$\mathbf{r}_1$: 0010001000101101	$\mathbf{r}_2$: 0100010001001011	$\mathbf{r}_3$: 1000100010000111
$\mathbf{p}_i\mathbf{p}_i\overline{\mathbf{p}}_i\mathbf{p}_i$	$\mathbf{r}_4$: 0001000111100001	$\mathbf{r}_5$: 0010001011010010	$\mathbf{r}_6$: 0100010010110100	$\mathbf{r}_7$: 1000100001111000
$\mathbf{p}_i\overline{\mathbf{p}}_i\mathbf{p}_i\mathbf{p}_i$	$\mathbf{r}_8$: 0001111000010001	$\mathbf{r}_9$: 0010110100100010	$\mathbf{r}_{10}$: 0100101101000100	$\mathbf{r}_{11}$: 1000011110001000
$\overline{\mathbf{p}}_i\mathbf{p}_i\mathbf{p}_i\mathbf{p}_i$	$\mathbf{r}_{12}$: 1110000100010001	$\mathbf{r}_{13}$: 1101001000100010	$\mathbf{r}_{14}$: 1011010001000100	$\mathbf{r}_{15}$: 0111100010001000

Definition 10 (Functions from patterns)**.** *To each pattern basis $P_{2n} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{2^{2n}-1}\}$ of rank $2n$, we associate the class $F_{2n} = \{f_0, f_1, \ldots, f_{2^{2n}-1}\}$ of Boolean functions from $\mathbb{B}^{2n}$ to $\mathbb{B}$, such that $\mathbf{p}_i$ is the pattern bit vector corresponding to f_i, $0 \leq i \leq 2^{2n} - 1$.*

Hence, a hierarchy, P_{2n}, of pattern bases induces a corresponding hierarchy, F_{2n}, of classes of Boolean functions. In what follows we shall also assume that the 2^{2n} Boolean functions contained in F_{2n} are enumerated as $f_0, f_1, \ldots, f_{2^{2n}-1}$ following the same enumeration with the pattern vectors of P_{2n}. By construction, the pattern vectors and, consequently, the pattern bases satisfy the following important relations.

($\mathbf{R}_1$) As we have mentioned in Example 1, the imbalance ratio, ρ, of $P_2 = \{1000, 0100, 0010, 0001\}$ is $\rho = \frac{1}{4}$.

($\mathbf{R}_2$) The recursive Definition 9 of the pattern hierarchy implies that the imbalance ratio satisfies the recurrence relation given below

$$\rho_{2n} = \frac{1}{4} + \frac{1}{2}\,\rho_{2n-2} \qquad (n \geq 2) , \tag{17}$$

where ρ_{2n-2} and ρ_{2n} are the imbalance ratios of P_{2n-2} and P_{2n}, respectively.

($\mathbf{R}_3$) After some manipulation, the above recurrence relation can be transformed into the next closed form

$$\rho_{2n} = \frac{1}{2} - \frac{1}{2^{n+1}} \qquad (n \geq 1)\ . \tag{18}$$

($\mathbf{R}_4$) The above closed-form formula enables us to surmise that

$$\rho_{2n} < \frac{1}{2} \qquad (n \geq 1)\ , \tag{19}$$

which proves that every pattern basis and all classes of Boolean functions in their respective hierarchies have the imbalance ratio $< \frac{1}{2}$, or, in simpler terms, all the Boolean functions we classify are indeed imbalanced as we have previously asserted.

Our purpose is to realize the Boolean Function Pattern Quantum Classifier algorithm through a family of quantum circuits denoted by QCPC_{2n}, $n \geq 1$, such that QCPC_{2n} classifies the class of Boolean functions F_{2n}, which consist of functions that follow the motif prescribed by the elements of the pattern basis B_{2n}. In these quantum circuits, the critical component for the classification is the Q_{2n} unitary classifier, defined below.

Definition 11 (A hierarchy of unitary classifiers)**.** *We recursively define a* hierarchy *of unitary classifiers, denoted by Q_{2n}, $n \geq 1$, as follows.*

($\boldsymbol{QH}_0$) *If $n = 1$, the corresponding classifier is Q_2, as expressed by (13) with the matrix representation given by Figure 6.*

($\boldsymbol{QH}_1$) *Given Q_{2n}, the classifier $Q_{2(n+1)}$ is defined as*

$$Q_{2(n+1)} \coloneqq Q_2 \otimes Q_{2n} = Q_2^{\otimes(n+1)} \qquad (n \geq 1)\ . \tag{20}$$

Example 3 (Unitary classifier Q_4)**.** *It is instructive to show in detail how the matrix representation of the unitary classifier Q_4 is derived. By Definition 11, we know that*

$$Q_4 \overset{(20)}{=} Q_2 \otimes Q_2 \overset{(6)}{=} \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \otimes \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

$$= \left[\begin{array}{cc|cc} -\frac{1}{2}Q_2 & \frac{1}{2}Q_2 & \frac{1}{2}Q_2 & \frac{1}{2}Q_2 \\ \frac{1}{2}Q_2 & -\frac{1}{2}Q_2 & \frac{1}{2}Q_2 & \frac{1}{2}Q_2 \\ \hline \frac{1}{2}Q_2 & \frac{1}{2}Q_2 & -\frac{1}{2}Q_2 & \frac{1}{2}Q_2 \\ \frac{1}{2}Q_2 & \frac{1}{2}Q_2 & \frac{1}{2}Q_2 & -\frac{1}{2}Q_2 \end{array}\right]$$

$$= \left[\begin{array}{cccc|cccc|cccc|cccc}
\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
-\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
-\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\
-\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
\hline
-\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
\hline
-\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
\hline
-\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\
\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4}
\end{array}\right]. \quad (21)$$

We use the unitary classifier Q_{2n}, $n \geq 1$, as the main component in the construction of the family of quantum circuits QCPC_{2n}, for the classification of the class of Boolean functions F_{2n}.

Definition 12 (A family of quantum classifiers)**.** *To each unitary classifier $Q_{2n}, n \geq 1$, we associate the quantum circuits $QCPC_{2n}$, for the classification of the class of Boolean functions F_{2n}.*

- *The first member of this family, the $QCPC_2$ quantum circuit, takes the form depicted in Figure 7 and can classify the Boolean functions in F_2.*
- *The general $QCPC_{2n}$ quantum circuit takes the abstract form visualized in Figure 11. It is endowed with the oracle U_f encoding the behavior of the Boolean function f, which is promised to belong to F_{2n}.*

Therefore, the abstract quantum circuit that implements the BFPQC algorithm for the classification of the class of Boolean functions $F_{2n}, n \geq 1$, is outlined in Figure 11. To avoid any ambiguity, we explain the notation used in this figure.

- IR is the quantum input register that contains $2n$ qubits and starts its operation at state $|\mathbf{0}\rangle$.
- OR is the single-qubit output register initialized to $|-\rangle$.
- H is the Hadamard transform.
- U_f is the unitary transform corresponding to the oracle for the unknown function, f. The latter is promised to be an element of F_{2n}.
- Q_2 is the fundamental building block of Q_{2n}, as evidenced by Equation (20).

Classification works as follows: the Boolean functions contained in F_{2n} are enumerated as $f_0, f_1, \ldots, f_{2^{2n}-1}$. Assuming the oracle encodes the function f_i with index i, the outcome of the final measurement of the quantum circuit QCPC_{2n} after the action of the classifier $Q_2^{\otimes 2n}$ will be $|\mathbf{i}\rangle$, where $\mathbf{i}$ is the binary representation of the index i, i.e., one of the basis kets of the computational basis. Our algorithm is optimal because it requires just a single query to classify the hidden function.

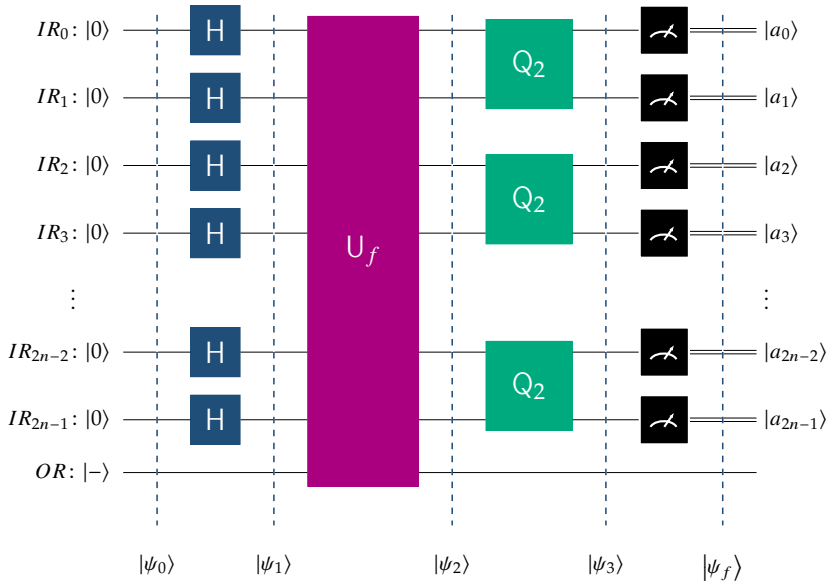

Figure 11. This figure visualizes the general quantum circuit QCPC_{2n} that implements the BFPQC algorithm for the classification of the functions contained in F_{2n}.

The method we used to devise the BFPQC algorithm is visualized in Figure 12. We are confident that this methodology is general and fruitful, in the sense that it can be used as a starting point to define additional quantum classification algorithms by establishing different hierarchies of pattern bases and classifiers.

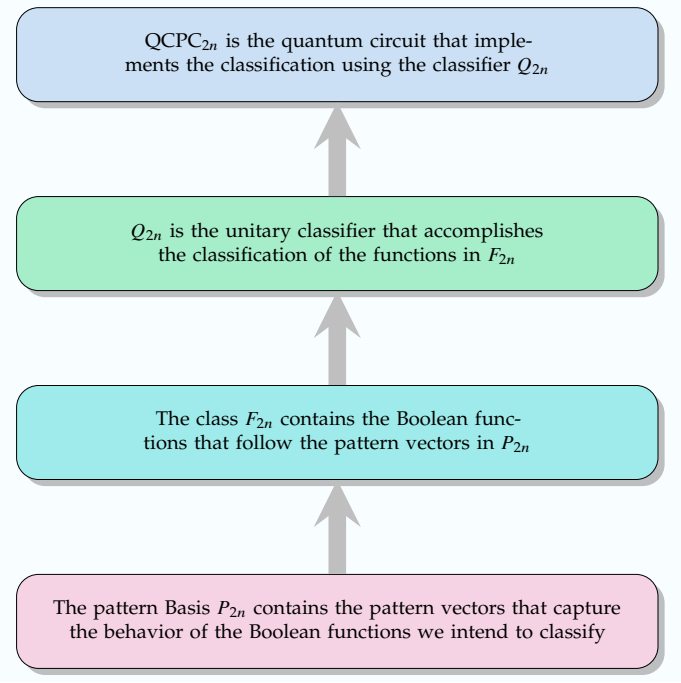

Figure 12. This diagram visualizes the main stages of the methodology we employed to create the BFPQC algorithm.

We close this section by giving a more interesting example targeting functions of F_4.

Example 4 (Classifying functions of F_4). *Let us assume that Bob has to choose a Boolean function from F_4, the promised class of functions in this case. Say that Bob chooses f_3, the behavior of which is given by the pattern vector* $\mathbf{r}_3$: 1000 1000 1000 0111, *listed in Example 2. Alice makes her move by employing the classifier $Q_4 = Q_2^{\otimes 2}$. In this case, the concrete implementation in Qiskit [41] of the*

general quantum circuit of Figure 11 takes the form shown in Figure 13, where Bob uses the oracle for the function f_3 and Alice the classifier Q_4.

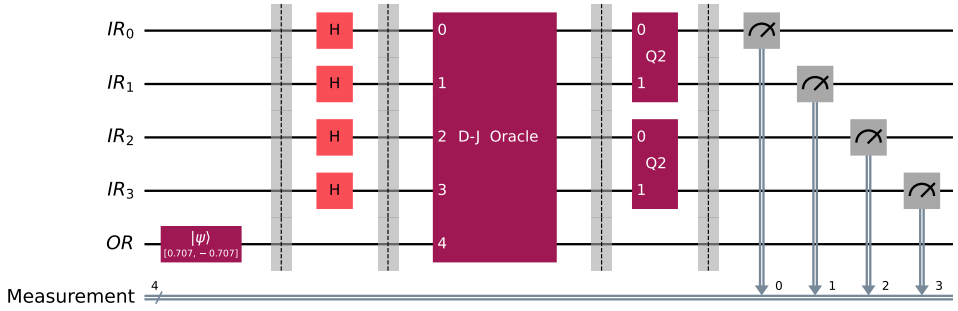

Figure 13. This figure shows the implementation of the BFPQC algorithm for the classification of the Boolean functions in F_4, assuming Bob has chosen the oracle for the function f_3 and Alice has employed the classifier Q_4.

After the oracle, and before the action of the classifier, the state of the system is shown in Figure 14. After the action of the classifier, the state of the system is just $|\mathbf{0011}\rangle$*. Therefore, measuring the quantum circuit depicted in Figure 13 will output the bit vector* 0011 *with probability* 1 *(as corroborated by the measurements contained in Figure 15), which is the binary representation of the index of f_3. Alice surely wins the game, as anticipated.*

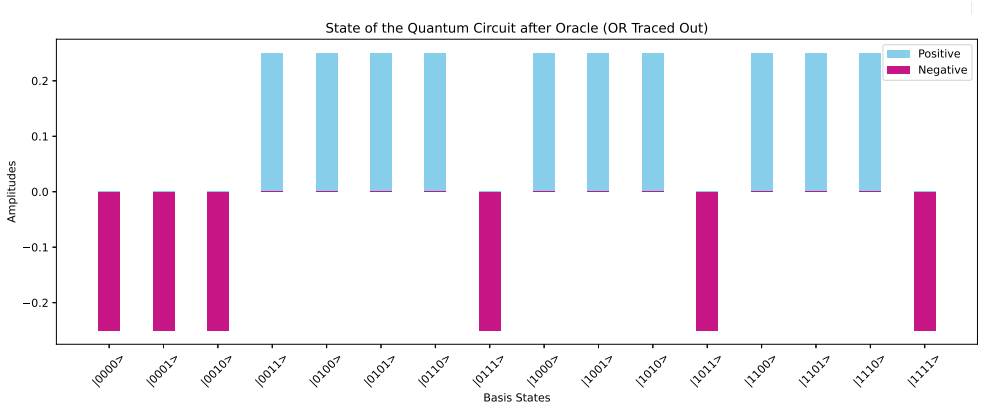

Figure 14. This is the state of the quantum circuit of Figure 13 after the oracle but before the action of $Q_2^{\otimes 2}$.

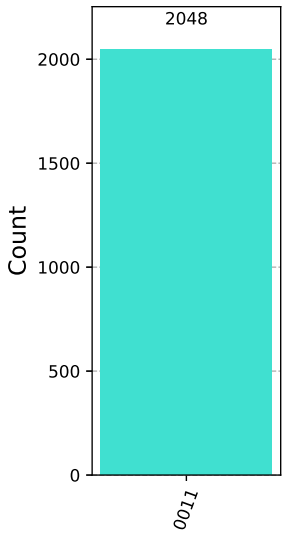

Figure 15. The measurement outcome of the quantum circuit of Figure 13.

Execution on IBM quantum computers.

After validating the correctness of the quantum circuit depicted in Figure 13 via simulation, we used Qiskit to execute it on a real IBM quantum computer. At the moment of running the experiment, the backends that were available to us (via our free-tier account) were `ibm_brisbane` (127 qubits) and `ibm_sherbrooke` (127 qubits). Both IBM Brisbane and IBM Sherbrooke quantum computers are 127-qubit backends based on IBM's Eagle r3 processor, making them architecturally similar. They are designed for utility-scale quantum computing, meaning that they are capable of running complex algorithms and physics simulations that push the boundaries of classical computation, and suit our circuit's modest requirements (number of qubits and number of gates). Each time we picked the least busy backend with enough qubits using the typical command:

```
backend = service.least_busy(simulator=False, operational=True,
          min_num_qubits=5)
```

The operation of the BFPQC algorithm is formalized in Theorem 1.

Theorem 1 (Classification via the BFPQC algorithm)**.** *Let* f_i, $0 \leq i \leq 2^{2n} - 1$ *be a Boolean function contained in the class* $F_{2n} = \{f_0, f_1, \ldots, f_{2^{2n}-1}\}$. *Upon measuring the quantum circuit* $QCPC_{2n}$ *containing the oracle for* f_i *and the unitary classifier* Q_{2n}, *we obtain with a probability of* 1.0 $|\mathbf{i}\rangle$, *where* $\mathbf{i}$ *is the binary representation of the index* i, *and* $|\mathbf{i}\rangle$ *is one of the basis kets of the computational basis.*

5. Discussion and Conclusions

This section contains the complexity analysis of the BFPQC algorithm followed by a brief discussion summarizing the novelties of this work. We begin by considering the complexity aspects of the BFPQC algorithm.

5.1. Discussion on Complexity

From a classical viewpoint, we may devise an oracular deterministic algorithm that can solve the classification problem for functions defined via pattern bases as in Definition 10. Such an algorithm can achieve this task by querying the oracle on different inputs. As we have explained in Section 2, the standard measure of complexity in oracular algorithms is the query complexity, i.e., the number of queries to the oracle used by the algorithm. In that respect, the proposed quantum algorithm BFPQC is superior to any deterministic classical algorithm. This is because the BFPQC algorithm can conclusively identify any function in F_{2n}, $n \geq 1$, with just a single query to the oracle, whereas any deterministic classical algorithm requires polynomially more queries. To be exact, if c_{2n} is the number of calls to the oracle that any deterministic classical algorithm makes in the worst case, in order to classify any function in F_{2n}, we can show with induction on n that

$$c_{2n} = 3 + 2(n-1) \qquad (n \geq 1)\,. \tag{22}$$

Implementation-wise, the quantum circuits $QCPC_{2n}$ used for the execution of the BFPQC algorithm require very modest quantum resources. To quantify these resources, we use standard terminology that can be found in most textbooks (see for details [44–47]). Recall that the depth of a quantum circuit is the maximal sequence of one- and two-qubit gates between the initial state preparation and the final measurement, the width of a quantum circuit is the total number of qubits (including ancillae qubits), and the size is the total number of gates used. To make an analogy with classical computation, one could say that the depth and width of a quantum circuit correspond to its time and space complexity. The quantum circuit $QCPC_{2n}$ depicted in Figure 11, which implements the classification, has constant depth, excluding the oracle, namely 5, taking into account that the subcircuits

realizing the classifier Q_2 have depth 4 (recall Figure 6). Its width is $2n+1$, where the constant 1 is due to the output qubit OR.

5.2. Other Possible Hierarchies of Boolean Functions

As we have shown, the proposed BFPQC algorithm can classify imbalanced Boolean functions that belong to the hierarchy of classes of Boolean functions $F_2, F_4, \ldots, F_{2n}, \ldots$. This hierarchy of Boolean functions is constructed from the corresponding hierarchy of imbalanced pattern bases $P_2, P_4, \ldots, P_{2n}, \ldots$, as ordained by Definition 10. Taking into account the fact that our purpose was to tackle imbalanced Boolean functions, we see that the initial pattern basis in this hierarchy must necessarily be P_2 (recall Definition 9) because this is the least such basis that exhibits imbalance in terms of the length of the pattern vector. It is trivial to verify that there can be no pattern basis with a pattern vector of length 2 that captures imbalance. For instance, the pattern basis $B_2 := \{00, 01\}$ consists of pattern vectors that correspond to a constant and a perfectly balance function. Regarding the rest of the hierarchy, there could be variations by appropriately modifying Definition 11. The simplest possibility would be to define a different hierarchy of unitary classifiers as follows

$$H \otimes Q_2,\ H^{\otimes 2} \otimes Q_2, \ldots\ . \tag{23}$$

The above approach leads to a totally different hierarchy of classes of Boolean functions $G_2, G_3, \ldots, G_n, G_{n+1}, \ldots$. However, excluding G_2, all other classes, G_n, contain both perfectly balanced and imbalanced functions. In our view, it is preferable to choose the construction outlined in Definition 11 because it guarantees that all Boolean functions in the same class, F_{2n}, are imbalanced and with exactly the same imbalance ratio, ρ_{2n}, as given by Equation (18).

5.3. Conclusions

The starting point of this work was the realization that, although the relevant literature is full of sophisticated studies that extend and generalize the Deutsch–Jozsa algorithm and investigate balanced Boolean functions, it still lacks works focusing on imbalanced Boolean functions. By imbalanced we mean Boolean functions that are characterized by an unequal number of elements in their domain that take the values 0 and 1. This article presents a new quantum algorithm aimed at categorizing a particular hierarchy of imbalanced Boolean function classes.

For each positive integer $n \geq 1$, this hierarchy encompasses a class, F_{2n}, of $2n$-ary Boolean functions, which are delineated based on their behavioral traits. A distinguishing characteristic of all functions within the same class is their common imbalance ratio. Our algorithm enables classification in a clear and intuitive way, as the final measurement identifies the unknown function with a probability of 1.0. It is crucial to emphasize that the proposed algorithm is an optimal oracular algorithm, capable of categorizing the specified functions with a single query to the oracle. We note that, within this quantum context, f_i and its negation, $\overline{f_i}$, are indistinguishable because they lead to the same state.

In closing, we emphasize that, in addition to a concrete algorithm, we offer a comprehensive description of the methodology utilized in the creation of this algorithm. This is done with the hope that it will prove both general and advantageous, as it can be easily modified and expanded to address other categories of imbalanced Boolean functions that exhibit diverse behavioral patterns. This can be easily achieved by appropriately modifying Definition 9 to include different pattern bases.

Author Contributions: Conceptualization, T.A., C.B. and K.N.; methodology, T.A., G.I.G. and N.K.; validation, C.B., K.N. and G.I.G.; formal analysis, T.A., G.I.G. and N.K.; investigation, C.B. and K.N.; writing—original draft preparation, T.A., C.B. and G.I.G.; writing—review and editing, T.A. and N.K.; visualization, C.B. and K.N.; supervision, T.A., G.I.G. and N.K.; project administration, T.A. and N.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chow, J.; Dial, O.; Gambetta, J. IBM Quantum Breaks the 100-Qubit Processor Barrier. 2021. Available online: https://www.ibm.com/quantum/blog/127-qubit-quantum-processor-eagle/ (accessed on 7 January 2025).
2. IBM. IBM Unveils 400 Qubit-Plus Quantum Processor. 2022. Available online: https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two/ (accessed on 7 January 2025).
3. Gambetta, J. The Hardware and Software for the Era of Quantum Utility Is Here. 2023. Available online: https://www.ibm.com/quantum/blog/quantum-roadmap-2033/ (accessed on 7 January 2025).
4. IBM. IBM Launches Its Most Advanced Quantum Computers, Fueling New Scientific Value and Progress Towards Quantum Advantage. 2024. Available online: https://newsroom.ibm.com/2024-11-13-ibm-launches-its-most-advanced-quantum-computers,-fueling-new-scientific-value-and-progress-towards-quantum-advantage/ (accessed on 7 January 2025).
5. Photonic. Photonic Demonstrates Distributed Entanglement Between Modules, Marking Significant Milestone Toward Scalable Quantum Computing and Networking. 2024. Available online: https://photonic.com/news/photonic-demonstrates-distributed-entanglement-between-modules/ (accessed on 7 January 2025).
6. Nu Quantum. Announcing the Qubit-Photon Interface (QPI): Towards Unlocking Modular and Scalable Distributed Quantum Computing. 2024. Available online: https://www.nu-quantum.com/news/qubit-photon-interface-qpi-towards-unlocking-modular-and-scalable-distributed-quantum-computing/ (accessed on 7 January 2025).
7. Cacciapuoti, A.S.; Illiano, J.; Viscardi, M.; Caleffi, M. Multipartite Entanglement Distribution in the Quantum Internet: Knowing When to Stop! *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 6041–6058. [CrossRef]
8. Illiano, J.; Caleffi, M.; Viscardi, M.; Cacciapuoti, A.S. Quantum MAC: Genuine Entanglement Access Control via Many-Body Dicke States. *IEEE Trans. Commun.* **2024**, *72*, 2090–2105. [CrossRef]
9. Main, D.; Drmota, P.; Nadlinger, D.P.; Ainley, E.M.; Agrawal, A.; Nichol, B.C.; Srinivas, R.; Araneda, G.; Lucas, D.M. Distributed quantum computing across an optical network link. *Nature* **2025**, *638*, 383–388. [CrossRef]
10. Oxford News. First Distributed Quantum Algorithm Brings Quantum Supercomputers Closer. 2025. Available online: https://www.ox.ac.uk/news/2025-02-06-first-distributed-quantum-algorithm-brings-quantum-supercomputers-closer/ (accessed on 7 February 2025).
11. Cleve, R.; Ekert, A.; Macchiavello, C.; Mosca, M. Quantum algorithms revisited. *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.* **1998**, *454*, 339–354. [CrossRef]
12. Chi, D.P.; Kim, J.; Lee, S. Initialization-free generalized Deutsch-Jozsa algorithm. *J. Phys. A Math. Gen.* **2001**, *34*, 5251. [CrossRef]
13. Holmes, R.R.; Texier, F. A Generalization of the Deutsch-Jozsa Quantum Algorithm. *Far East J. Math Sci.* **2003**, *9*, 319–326.
14. Ballhysa, E.; Say, A.C.C. Generating Equiprobable Superpositions of Arbitrary Sets for a New Generalization of the Deutsch-Jozsa Algorithm. In *Computer and Information Sciences—ISCIS 2004*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 966–975. [CrossRef]
15. Qiu, D.; Zheng, S. Generalized Deutsch-Jozsa problem and the optimal quantum algorithm. *Phys. Rev. A* **2018**, *97*, 062331. [CrossRef]
16. Ossorio-Castillo, J.; Pastor–Díaz, U.; Tornero, J. A generalisation of the Phase Kick-Back. *Quantum Inf. Process.* **2023**, *22*, 143. [CrossRef]
17. Qiu, D.; Zheng, S. Revisiting Deutsch-Jozsa algorithm. *Inf. Comput.* **2020**, *275*, 104605. [CrossRef]
18. Zhengwei, X.; Daowen, Q.; Guangya, C.; Gruska, J.; Mateus, P. Testing Boolean Functions Properties. *Fundam. Informaticae* **2021**, *182*, 321–344. [CrossRef]
19. Tănăsescu, A.; Mina, M.Z.; Popescu, P.G. Non-local quantum functions and the distributed Deutsch-Jozsa algorithm. *Phys. Lett. A* **2019**, *383*, 2168–2171. [CrossRef]
20. Li, H.; Qiu, D.; Luo, L. Distributed Generalized Deutsch-Jozsa Algorithm. In *Computing and Combinatorics*; Springer Nature: Singapore, 2025; pp. 214–225. [CrossRef]

21. Nagata, K.; Nakamura, T. Generalization of Deutsch's Algorithm. *Int. J. Theor. Phys.* **2020**, *59*, 2557–2561. [CrossRef]
22. Farhi, E.; Goldstone, J.; Gutmann, S.; Sipser, M. Bound on the number of functions that can be distinguished with k quantum queries. *Phys. Rev. A* **1999**, *60*, 4331–4333. [CrossRef]
23. Høyer, P.; Spalek, R. Lower Bounds on Quantum Query Complexity. *Bull. EATCS* **2005**, *87*, 78–103.
24. Cross, A.W.; Smith, G.; Smolin, J.A. Quantum learning robust against noise. *Phys. Rev. A* **2015**, *92*, 012327. [CrossRef]
25. Bshouty, N.H.; Jackson, J.C. Learning DNF over the Uniform Distribution Using a Quantum Example Oracle. *SIAM J. Comput.* **1998**, *28*, 1136–1153. [CrossRef]
26. Servedio, R.; Gortler, S. Quantum versus classical learnability. In Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, IL, USA, 18–21 June 2001; pp. 138–148. [CrossRef]
27. Servedio, R.A.; Gortler, S.J. Equivalences and Separations Between Quantum and Classical Learnability. *SIAM J. Comput.* **2004**, *33*, 1067–1092. [CrossRef]
28. Hunziker, M.; Meyer, D.A.; Park, J.; Pommersheim, J.; Rothstein, M. The geometry of quantum learning. *Quantum Inf. Process.* **2009**, *9*, 321–341. [CrossRef]
29. Montanaro, A. The quantum query complexity of learning multilinear polynomials. *Inf. Process. Lett.* **2012**, *112*, 438–442. [CrossRef]
30. Yoo, S.; Bang, J.; Lee, C.; Lee, J. A quantum speedup in machine learning: Finding anN-bit Boolean function for a classification. *New J. Phys.* **2014**, *16*, 103014. [CrossRef]
31. Wiebe, N.; Kapoor, A.; Svore, K.M. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Inf. Comput.* **2015**, *15*, 316–356. [CrossRef]
32. Wiebe, N.; Kapoor, A.; Svore, K.M. Quantum deep learning. *Quantum Inf. Comput.* **2016**, *16*, 541–587. [CrossRef]
33. Meyer, D.A. Quantum strategies. *Phys. Rev. Lett.* **1999**, *82*, 1052. [CrossRef]
34. Eisert, J.; Wilkens, M.; Lewenstein, M. Quantum games and quantum strategies. *Phys. Rev. Lett.* **1999**, *83*, 3077. [CrossRef]
35. Andronikos, T.; Sirokofskich, A. The Connection between the PQ Penny Flip Game and the Dihedral Groups. *Mathematics* **2021**, *9*, 1115. [CrossRef]
36. Andronikos, T. Conditions that enable a player to surely win in sequential quantum games. *Quantum Inf. Process.* **2022**, *21*, 268. [CrossRef]
37. Koh, D.E.; Kumar, K.; Goh, S.T. Quantum Volunteer's Dilemma. *arXiv* **2024**, arXiv:2409.05708. [CrossRef]
38. Bennett, C.H.; Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing. In Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India, 9–12 December 1984; IEEE Computer Society Press: Washington, DC, USA, 1984; pp. 175–179.
39. Andronikos, T.; Sirokofskich, A. A Multiparty Quantum Private Equality Comparison Scheme Relying on $|GHZ3\rangle$ States. *Future Internet* **2024**, *16*, 309. [CrossRef]
40. Andronikos, T.; Stefanidakis, M. A Two-Party Quantum Parliament. *Algorithms* **2022**, *15*, 62. [CrossRef]
41. Qiskit. Qiskit Is the World's Most Popular Software Stack for Quantum Computing. 2025. Available online: https://www.ibm.com/quantum/qiskit/ (accessed on 7 January 2025).
42. Mermin, N. *Quantum Computer Science: An Introduction*; Cambridge University Press: Cambridge, UK, 2007. [CrossRef]
43. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*; Cambridge University Press: Cambridge, UK, 2010.
44. Kaye, P.; Laflamme, R.; Mosca, M. *An Introduction to Quantum Computing*; OUP Oxford: Oxford, UK, 2007.
45. Williams, C.P. *Explorations in Quantum Computing*; Springer: London, UK, 2011. [CrossRef]
46. Kasirajan, V. *Fundamentals of Quantum Computing*; Springer International Publishing: Berlin/Heidelberg, Germany, 2021. [CrossRef]
47. Stancil, D.D.; Byrd, G.T. *Principles of Superconducting Quantum Computers*; Wiley: Hoboken, NJ, USA, 2022. [CrossRef]