BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 67 (71), Numărul 2, 2021
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ
DOI:10.2478/bipie-2021-0009

IMPLEMENTATION OF THE BERNSTEIN-VAZIRANI QUANTUM ALGORITHM USING THE QISKIT FRAMEWORK

BY

ALEXANDRU-GABRIEL TUDORACHE[*], VASILE-ION MANTA and SIMONA CARAIMAN

“Gheorghe Asachi” Technical University of Iași, Romania,
Faculty of Automatic Control and Computer Engineering

Received: March 29, 2021
Accepted for publication: May 17, 2021

Abstract. This paper describes the basics of quantum computing and then focuses on the implementation of the Bernstein-Vazirani algorithm, which can be seen as an extension of the Deutsch-Josza problem (that solves the question on whether a function is balanced or not). The idea behind the B-V algorithm is that someone can find a secret number (sequence of bits) using only one measurement, unlike the classical counter-part, that requires n measurements, where n is the number of bits of the secret number. The implementation of this algorithm, using the Python programming language, along with the Qiskit framework (an open-source library for quantum operations from IBM), illustrates how to create and simulate a circuit for such an algorithm. The circuit is dynamically generated for the required number (which in practice is received from a different source) and is used to measure the probability of each qubit. The algorithm can also be extended for different types of data and can be used for signal or image processing, as well as applications in cryptography.

Keywords: Quantum computing; Bernstein-Vazirani; quantum algorithm; Qiskit.

[*]Corresponding author; *e-mail*: alexandru.tudorache93@gmail.com

1. Introduction

The quantum computing universe is one of the most exciting subjects in the “information technology” field, especially with the latest physical innovations, such as the creation of D-Wave, IBM Q System One (a 20-qubit quantum computer) and Google’s quantum computer, which recently claimed “quantum supremacy”, by performing a calculation in 200 seconds instead of 10,000 years, as it world have taken on a classical computer (they use a 54-qubit Sycamore processor). Today it is possible not only to simulate the circuits on classical computers, but to also run them and measure the results on real platforms.

The most basic element in quantum processing is the *qubit* (quantum bit), which, unlike the classical bit, that can either be 0 or 1, can simultaneously be in multiple states (superposition) with different probabilities – these basic states, named *ket*, are $|0\rangle$ and $|1\rangle$. A superposition state, called $|\psi\rangle$, can be defined as (McMahon, 2008):

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers, that verify $|\alpha|^2 + |\beta|^2 = 1$; $|\alpha|^2$ is the probability that a qubit is in the $|0\rangle$ state and $|\beta|^2$ is the probability that a qubit is in the $|1\rangle$ state.

In order for a quantum algorithm to be implemented, a circuit that uses different quantum gates for certain operations is required. All quantum gates are described by unitary transforms, so they are all reversible; the number of inputs is equal to the number of outputs; so, the input values can be obtained by knowing the output values for each gate. Some of the basic gates, on one or more qubits, are briefly presented below. The following gates act on a single qubit:

- the **NOT** gate acts similarly to its classical counter-part; its action is linear, so the state $a|0\rangle + b|1\rangle$ changes to $a|1\rangle + b|0\rangle$.
- the **Z** gate inverts the phase, by changing a qubit’s state from $a|0\rangle + b|1\rangle$ to $a|0\rangle - b|1\rangle$.
- the **Hadamard** gate can be applied to any of the basic states and the result is a mix of the two with equal probability, that is:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2}$$

and

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\ 0\end{bmatrix} - \begin{bmatrix}0\\ 1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{3}$$

The following gates act on multiple qubits:

- the **CNOT** gate (controlled-NOT), the quantum version of the XOR gate, as can be seen below:

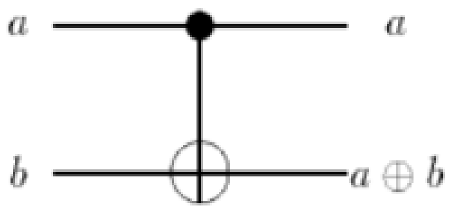

Fig. 1 – CNOT gate.

$|a\rangle$ is called the control qubit, and $|b\rangle$ is the target qubit. If the control qubit is in the $|0\rangle$ state, then the target qubit is unchanged; if the control qubit is in state $|1\rangle$, the target qubit is inversed. The gate's action ($|ab\rangle \rightarrow |a(a \oplus b)\rangle$) can also be expressed as:

$$|00\rangle \rightarrow |00\rangle,\ |01\rangle \rightarrow |01\rangle,\ |10\rangle \rightarrow |11\rangle,\ |11\rangle \rightarrow |10\rangle.$$

- the **SWAP** gate inverts the values of the qubits: $|ab\rangle \rightarrow |ba\rangle$.
- the **CCNOT** gate, presented in the figure below:

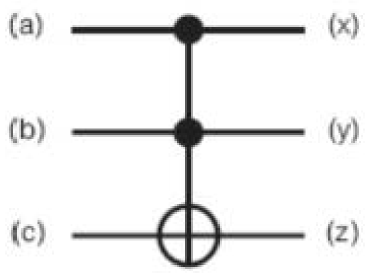

Fig. 2 – CCNOT gate (Toffoli).

The a and b inputs don't change their states ($x = a$ and $y = b$); input from c will change its state by calculating the XOR value between c and the logic product (AND) of a and b, according to the formula:

$$z = c \oplus (\mathrm{a} \wedge \mathrm{b}). \tag{4}$$

The current paper presents the implementation of the Bernstein-Vazirani algorithm using the Python programming language (and the Qiskit framework), by dynamically creating a circuit using the gates above, and the way this can be applied to the image processing field.

2. Quantum Algorithms

In order to utilize the full potential of the quantum universe, different properties such as quantum *entanglement* and *phase kick-back* have been used in various quantum algorithms.

One important quantum property is the phenomenon called *phase kick-back*, where in specific conditions, the CNOT gate can affect not only the target qubit, but also the control one. For example, by applying the CNOT gate to the qubits in the following states we obtain:

$$\text{CNOT: } \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \rightarrow \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right). \tag{5}$$

The idea behind this effect is that $\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$ (the target qubit in the formula above) is an *eigenvector* (or *eigenstate*) of the NOT gate with the *eigenvalue* -1 and an *eigenvector* of the identity gate (I) with the *eigenvalue* 1 (Kaye *et al*., 2007).

Starting from this observation, the concept of the quantum oracle was developed, such as, for a function that takes 2 qubits as input, U_f, that implements a random function $f: \{0,1\} \rightarrow \{0,1\}$, with $U_f: |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, we can write:

$$U_f: |x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \rightarrow (-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \tag{6}$$

An algorithm that needs only one measurement in order to find whether a function $f: \{0,1\} \rightarrow \{0,1\}$ is constant or balanced is the Deutsch algorithm. A function f is constant if $f(0) = f(1)$, which implies $f(0) \oplus f(1) = 0$ and balanced if $f(0) \neq f(1)$ ($f(0) \oplus f(1) = 1$); the classical method requires two measurements. The Deutsch-Josza algorithm is a generalization of the Deutsch algorithm – the function f operates on n bits and has a single bit as output $f: \{0,1\}^n \rightarrow \{0,1\}$. The problem remains the same, to find out if f is constant or balanced (f is constant if $f(x)$ has the same value for any x, and balanced if

$f(x) = 0$ for half of the entries and $f(x) = 1$ for the other half). The classical solution requires $2^{n-1} + 1$ queries (half + 1), but the quantum solution needs only one query.

The Bernstein-Vazirani algorithm is an extension of the Deutsch-Josza algorithm. Let there be a Boolean function f, which takes as input a string of bits and returns 0 or 1, $f: \{0,1\}^n \rightarrow \{0,1\}$. The function calculates $f(x) = s \cdot \mathrm{x}\ (\mathrm{mod}2)$; the problem is to find the certain secret string s (more details can be found on https://qiskit.org/textbook/ch-algorithms/bernstein-vazirani.html).

3. Implementation of the B-V Algorithm

The language of choice for implementing and testing the B-V algorithm is Python, and the chosen framework is Qiskit – "an open-source quantum computing software development framework for leveraging today's quantum processors in research, education and business" (as defined on the official page, https://qiskit.org/). Qiskit offers support for both the execution and simulation of the code written for quantum applications and algorithms.

It is made up of 4 components:

- Qiskit Terra contains a set of tools for writing quantum programs at the circuit level; it uses different optimizations for the available physical quantum processor and manages the execution of the experiments;
- Qiskit Aer is the high-performance simulator component – it contains optimized C++ simulator backends for the circuits compiled using Qiskit Terra, along with tools needed to analyze various noise models;
- Qiskit Aqua contains a set of quantum algorithms that can be used in different applications;
- Qiskit Ignis is a framework focused on the study of noise in quantum systems.

One way to apply or implement the B-V algorithm for a common scenario would be to consider the case of a simple image and then to select a certain property of each pixel (or group of pixels) as input (secret number) for the algorithm. The quantum circuit used to implement the B-V algorithm, when applied to a simple grey-scale image, uses 8 qubits for each pixel representation, as the gray value requires 8 bits (values are ranging from 0 to 255); there is also an ancilla qubit (only one is required for the entire image, but for demonstration purposes I drew the same one after each group of 8). The current implementation can generate the circuit dynamically (regardless of the size of the binary representation). The circuit can be created in the following steps:

Step 1. Each of the 8 qubits is initialized with the $|0\rangle$ state at first and then set up in superposition using the Hadamard gate.

Step 2. The auxiliary qubit must be set to the eigenvector state $\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$ – this is done by setting it first in the $|1\rangle$ state and then using the Hadamard gate.

Step 3. For each bit of value 1, we add a CNOT gate, where the control qubit is the corresponding qubit from the image and the target qubit is the auxiliary qubit.

Step 4. There is one more set of Hadamard gates for all the image qubits; basically, applying the Hadamard gate twice will not modify the qubit's state. If the qubit's initial state is $|0\rangle$ and the ancilla qubit is not used (the corresponding bit from the gray value is 0) the Hadamard gates will have no impact. If there is a CNOT gate using the image's qubit and the ancilla qubit, the intermediary result (after the first Hadamard gate and the CNOT gate) will be the same as if the qubit's initial state was $|1\rangle$: $\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Therefore, after the second Hadamard gate, the qubit's state will be $|1\rangle$.

In the following section, the algorithm is applied to a simple image:

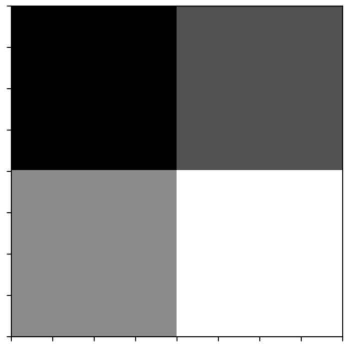

Fig. 3 – Sample 2x2 image.

The grey levels for this image are:

$$\begin{bmatrix} 38 & 96 \\ 136 & 217 \end{bmatrix}.$$

For each pixel, the generated circuits are presented bellow; the grey intensity is saved in the vector $q_7q_6q_5q_4q_3q_2q_1q_0$; each of them has a probability of 100% (simulated result):

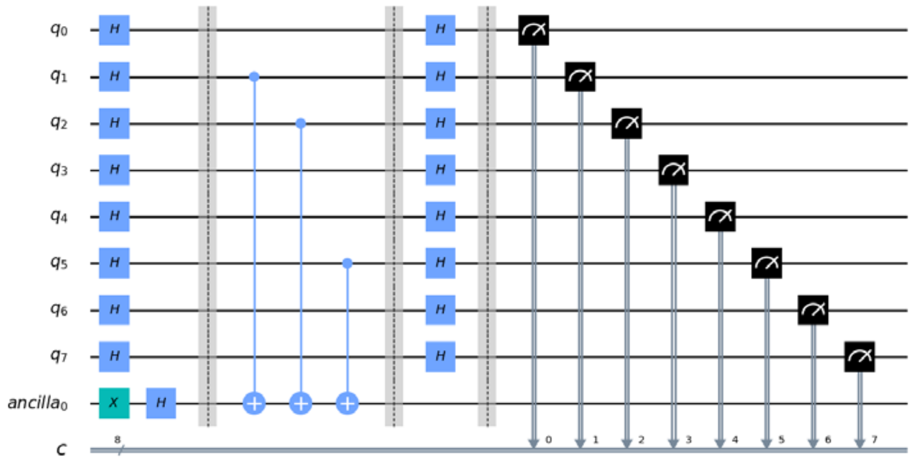

Fig. 4 – Circuit for grey level 38 (00100110).

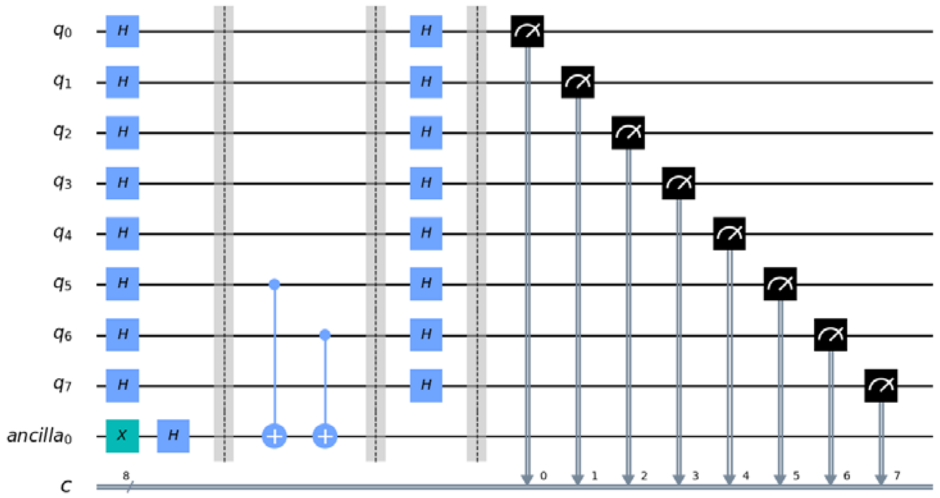

Fig. 5 – Circuit for grey level 96 (01100000).

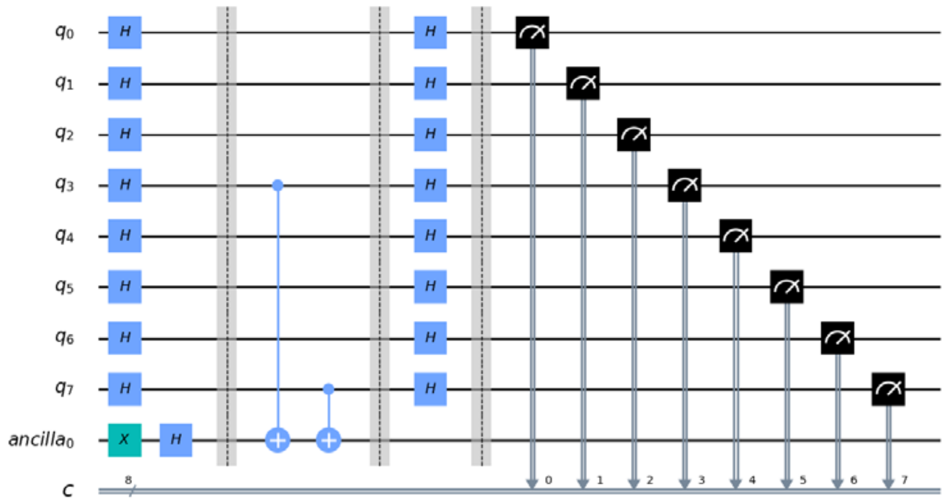

Fig. 6 – Circuit for grey level 136 (10001000).

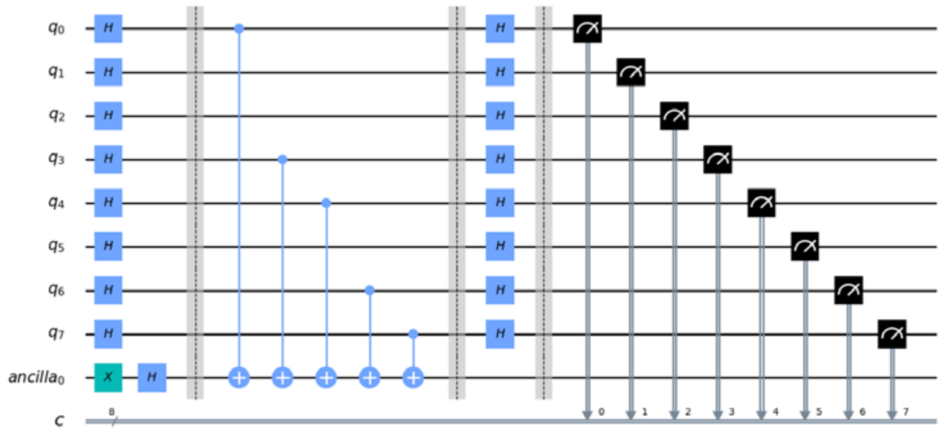

Fig. 7 – Circuit for grey level 217 (11011001).

3. Conclusions

This paper presents a way of generating a circuit that implements the Bernstein-Vazirani algorithm and then shows how this can be applied to a grey-scale image. Thanks to the IBM platform, this algorithm could be tested and its measurement results analyzed after the experiment execution. The idea can be extended to different types of data images (RGB for example), but also various data types in general, such as signals from external devices; a way of encrypting the data can also be used before the algorithm is applied.

A key point, used in the experiments from this paper, is the ability to use the phase kick-back effect, exemplified here with the help of the Bernstein-Vazirani algorithm. This can also be done in combination with some of the most popular quantum image representation techniques, for further data processing, depending on the scenario. Among these methods we mention the following:

- FRQI, flexible representation for quantum images (Le *et al.*, 2011), which allows the representation of an image by taking into account its color and position, according to the following formula:
$$|\psi\rangle = cos\ \theta_i|0\rangle + sin\ \theta_i|1\rangle,$$
with θ being the notation for the angle vector, used to specify the color;
- NEQR, novel enhanced quantum representation (Zhang *et al.*, 2013), that offers information about the color (grey level) and position for each qubit;
- NCQI, novel quantum representation of color digital images (Sang *et al.*, 2017), building on NEQR and being used for color images, represented using 3 color channels (RGB); the image state, for each pixel, can be written as:
$$|\psi(y,x)\rangle = \left|R_{nr\_q-1} \dots R_0 G_{nr\_q-1} \dots G_0 B_{nr\_q-1} \dots B_0\right\rangle,$$
using *nr_q* to indicate the number of qubits for each color channel;
- QRCI, new quantum representation model of color digital images (L. Wang *et al.*, 2019), which uses the bit-plane to identify the image state, for each of the 3 color components.

A survey that analyzes multiple quantum image representations, which could be extended through the phase kick-back effect (and is therefore useful as a general technique, depending on the nature of the quantum circuit), has also been done a couple of years ago (Yan *et al.*, 2016) and is a source of great knowledge for researches that want to familiarize themselves with the field of quantum image processing.

REFERENCES

Kaye P., Laflamme R., Mosca M., *An Introduction to Quantum Computing*, Oxford University Press, New York, 2007.

Le P.Q., Dong F., Hirota K., *A Flexible Representation of Quantum Images for Polynomial Preparation, Image Compression, and Processing Operations*, Quantum Inf Process 10, 63-84 (2011).

McMahon D., *Quantum Computing Explained*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2008.

Sang J., Wang S., Li, Q., *A Novel Quantum Representation of Color Digital Images*, Quantum Inf Process 16, 42 (2017).

Wang L., Ran Q., Ma J.Y., Yu S., Tan L., *QRCI: A New Quantum Representation Model of Color Digital Images*, Optics Communications, 438, 147-158 (2019).

Yan F., Iliyasu A.M., Venegas-Andraca S.E, *A SUrvey of Quantum Image Representations*, Quantum Inf Process 15, 1-35 (2016).

Zhang Y., Lu K., Gao Y., Wang, M., *NEQR: A Novel Enhanced Quantum Representation of Digital Images*, Quantum Inf Process 12, 2833-2860 (2013).

https://qiskit.org/, *Qiskit – Welcome to Quantum*, 2019.

https://qiskit.org/textbook/ch-algorithms/bernstein-vazirani.html, *Bernstein-Vazirani Algorithm*, 2019.

IMPLEMENTAREA ALGORITMULUI CUANTIC BERNSTEIN-VAZIRANI FOLOSIND BIBLIOTECA QISKIT

(Rezumat)

Această lucrare prezintă bazele procesării cuantice a informației, concentrându-se apoi pe implementarea algoritmului cuantic Bernstein-Vazirani, care poate fi văzut ca o extensie a algoritmului Deutsch-Josza, ce determină dacă o funcție este sau nu balansată. Ideea care stă la baza algoritmului B-V este faptul că un număr secret (secvență de biți) poate fi determinat cu o singură măsurătoare (un singur pas), spre deosebire de alternativa clasică, care necesită n operații, unde n este numărul de biți pe care se poate reprezenta numărul secret. Implementarea acestui algoritm, folosind limbajul de programare Python, împreună cu framework-ul Qiskit (o bibliotecă de tip open-source pentru operații cuantice scrisă de cercetătorii de la IBM), ilustrează cum se poate crea și simula un circuit pentru un astfel de algoritm. Circuitul este generat în mod dinamic pentru numărul cerut (care în practică este primit de la o sursă externă), fiind folosit pentru a măsura probabilitatea fiecărui qubit. Algoritmul poate fi extins pentru diverse tipuri de date și poate fi folosit pentru procesări de semnale sau imagini, precum și în aplicații din domeniul criptografiei.