Quantum-Inspired Evolutionary Algorithms on IBM Quantum Experience

Yoshio Rubio *Member, IAENG*, Cynthia Olvera, and Oscar Montiel *Member, IAENG*

***Abstract*—Quantum computing has been proposed as a possible accelerator for a myriad of complex computational problems. From this, quantum-inspired methodologies have emerged as methods that take the principles and restrictions of quantum theory to solve classical problems on classical computers. Quantum-inspired methodologies have proven advantageous in solving optimization problems and in learning over traditional nature-inspired methods. Since these algorithms operate on classical computers, the next unanswered question is important and valid: Can quantum-inspired evolutionary algorithms take advantage of quantum computers? The present work attempts to shed some light on this question, implementing quantum-inspired evolutionary algorithms for numerical optimization on quantum hardware. We present statistical metrics of their performance on the IBM Q quantum computer and compared them to their execution on a GPU-based quantum simulator, and the IBM quantum simulator.**

***Index Terms*—quantum computing, optimization, IBM Q, quantum-inspired, genetic algorithm.**

I. Introduction

SINCE its conception, quantum computing has been seen as the panacea for computer science [1]. Although this idea has prevailed since Deutsch's proposal [2], for almost two decades, quantum acceleration has remained a promise with an extended compliance date [3]. At present, many efforts to achieve useful quantum acceleration, as well as the development of practical quantum algorithms in different fields, have been conducted. Some scientific and technological fields exploiting the advantages of quantum computing are cybersecurity [4], quantum medicine [5], drug development [6], traffic optimization [7], [8], artificial intelligence [9], [10], and an ever-growing list of useful applications.

Quantum-inspired metaheuristics (QIM) are optimization methods that merge evolutionary computation and quantum computing. QIMs use quantum phenomena such as superposition, entanglement, and quantum measurements as inspiration to solve optimization problems [11]. There are many proposals of QIMs for numerical and combinatorial optimization [12], [13], [14]. A branch of QIMs is quantum-inspired evolutionary algorithms (QIEA) that use quantum phenomena and evolution mechanisms to solve optimization problems. The results of QIEA show that they might outperform traditional evolutionary algorithms (EA) on overall generational convergence and accuracy [15], [16], [17].

The number of proposals of QIEAs that exploit current quantum device architectures are limited [18]. Although full-scale quantum computers are still unattainable, small-scale devices with few qubits and limited or no error correction are available. Hence, the implementation of quantum algorithms aiming for quantum speedup has to adapt to these devices in order to prove the advantages of quantum computing.

The currently available quantum computers have been named noisy intermediate-scale quantum (NISQ) devices; a term coined by John Preskill. The *noisy* adjective in NISQ indicates the imperfect control over the qubits, placing limitations on what devices can achieve [19]. NISQ devices can be used to test the advantages that quantum computing might bring, which has given a boost to the research of developing and implementing quantum algorithms [1], [3].

Some of the most recent applications of quantum algorithms tested in real quantum devices are quantum approximate optimization algorithm for combinatorial problems [20], [21], and quantum machine learning [22], [23]. Nonetheless, they do not deal with quantum evolutionary metaheuristics. The work that best addresses evolution was presented in [24], where a quantum individual was modeled in conjunction with its biological behavior, such as interaction, evolution, mutation, self-replication, and death, within a natural selection scenario.

In this work, we present the circuit model development of two quantum evolutionary algorithms (QEA) for the IBM quantum computers using the IBM Quantum Experience cloud platform. We developed a methodology that properly translates QIEAs to QEA in quantum circuit form for the quantum computers to achieve this; specifically, we developed fully quantum versions of the most commons variants of QIEAs. Our proposal is compatible with any quantum computer-based on quantum circuits and reduces the depth of quantum circuits with an average depth of three gates per quantum individual, which is a significant contribution to the state-of-the-art regarding quantum metaheuristics for NISQ devices.

To evaluate the proposals, we used one-dimensional numerical functions and ran several statistical tests. We also study the limitations of further implementation of QIEAs on quantum computers.

The paper is organized as follows: in Section II, we explain the theoretical background of QIMs and the description of QIEAs in detail. In Section III, we describe our proposal to translate QIEAs to QEAs. The experiments that were carried

Manuscript received May 04, 2021; revised September 16, 2021. This work was supported by the Instituto Politécnico under Grant SIP20200053 and 20210320, and the Commission of Operation and Promotion of Academic Activities of IPN (COFAA).

Y. Rubio is an External Advisor of Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI) del Instituto Politécnico Nacional (IPN), 1310 Instituto Politécnico Nacional Ave., Nueva Tijuana 22430, Tijuana, Baja California, Mexico (email: rrubio@citedi.mx).

C. Olvera is a PhD Candidate of Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI) del Instituto Politécnico Nacional (IPN), 1310 Instituto Politécnico Nacional Ave., Nueva Tijuana 22430, Tijuana, Baja California, Mexico (colvera@citedi.mx).

O. Montiel is a Profersor of Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI) del Instituto Politécnico Nacional (IPN), 1310 Instituto Politécnico Nacional Ave., Nueva Tijuana 22430, Tijuana, Baja California, Mexico. Corresponding author phone +52(664)6231344; fax: +52(664)6231344 (email: oross@ipn.mx).

out using the QEAs implemented in a quantum computer are detailed in Section IV. The comparative and statistical results are shown in Section V. Finally, we discuss our results in Section VI.

II. Quantum-inspired evolutionary metaheuristics

The interaction between quantum computing and evolutionary computation can be organized into three main branches: evolutionary-designed quantum algorithms (EDQA), quantum evolutionary algorithms, and quantum-inspired metaheuristics, as shown in Figure 1.

The main goal of EDQAs is to create new quantum algorithms using genetic programming. EDQAs run on classic computers to find the best design of algorithms that will run on quantum devices.

Quantum evolutionary algorithms are optimization methods designed to operate on quantum devices. As in any optimization method, their goal is to find the optimal solution for a given problem. At the present, there is no general consensus on the best way of developing NISQ devices [1], which impacts the design of QEAs that are able to run on actual quantum computers.

Quantum-inspired metaheuristics are nature-inspired methods that use quantum phenomena to solve optimization problems[11]. In principle, QIMs are not designed to run on NISQ devices; at most, they run over limited quantum simulators. Many of the existing methods are quantum versions of traditional methods that translate some of their components using quantum mechanics analogies. Based on their classical counterparts, most of QIMs can be classified in one of the following:

- Quantum swarm algorithms.
- Quantum-inspired social algorithms.
- Quantum versions of physics-based methods.
- Quantum-inspired evolutionary algorithms.

Different natural behaviors inspire quantum swarm algorithms (QSAs). For example, the quantum variant of particle swarm optimization (QPSO) uses quantum wells as a substitution of the traditional PSO operators[25], while the quantum firefly algorithm (QFA) is inspired by the movement of charged particles [26], [27], [28], [29].

Quantum-inspired social algorithms (QISA) utilize human interactions, beliefs in society, and opinions to direct the evolutionary process of quantum individuals [30], [31].

Quantum versions of physics-based methods can mix Newtonian physics with quantum physics or only use quantum physics as inspiration. For example, quantum gravitational search algorithms (QGSA) use the combination of quantum phenomena and gravitational forces to search for the optimal value [32], [33], while quantum annealing (QA) exploits quantum tunneling to find problems where the search space is discrete with many local minima [34].

Quantum-inspired evolutionary algorithms use the Darwinian interpretation of evolution: biological populations change through time, and the fittest survive, passing their genes to their descendants [35]. QIEAs evolve a population of solutions composed of individuals encoded in quantum chromosomes using quantum bits as building blocks. The evolution of the population is done by applying quantum operations on the quantum chromosomes [36], [37].

Implementing any quantum-inspired algorithm in a quantum simulator is a difficult task, and it is more challenging to implement them on NISQ devices. Not all quantum-inspired algorithms are designed in the same way, and many of them only use some quantum principles as metaphors for classical operations, which limits the feasibility of their implementation in quantum devices. For example, quantum metaheuristics such as QPSO [38], QGSA [39], and some variations of QFA [28] only use quantum mechanics as metaphors to implement update mechanisms for their populations. These quantum operations simplify quantum phenomena, represented as addition and multiplication operations, with added randomness that simulates the probability distribution in quantum measurements.

Of all the quantum-inspired metaheuristics, QIEAs have the highest qualifications to be implemented in NISQ devices based on semiconductor qubits. In their purest form, QIEAs encode their populations using qubits and evolve them using a set of rules and quantum gates, which are easy to represent using quantum circuits.

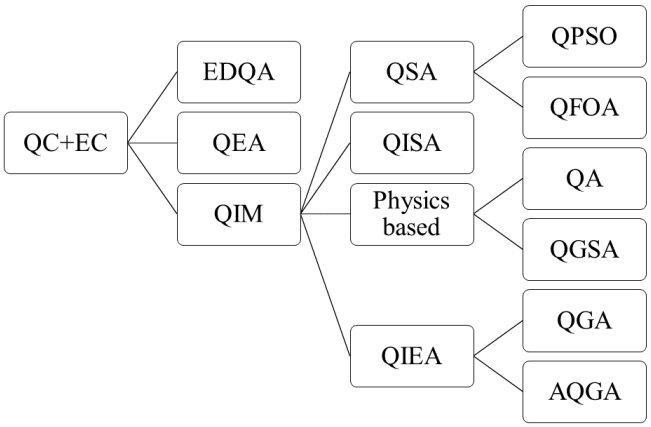


Fig. 1: There are three main branches derived from the interaction of quantum computing and evolutionary computation.

In this paper, we present the implementation into a NISQ device of two QIEAs: the quantum genetic algorithm and the adaptive quantum genetic algorithm. Most of the modern works that use the term quantum-inspired evolutionary algorithm or quantum genetic algorithm (QGA) use the work of Han and Kim [40], [36] as a reference. In our case, we used the term QGA to reference the work of Han and Kim, with any other derivation or improvement of their work being a variant.

In the following subsections, we describe in detail the quantum genetic algorithm and the adaptive quantum genetic algorithm.

A. Quantum genetic algorithm

Genetic algorithms (GA) use a set of individuals (population) and evolve them to solve optimization problems. Every individual of the population has a set of traits encoded using chromosomes. These traits represent a possible solution to the problem. A fitness function derived from an objective function is used to qualify how close the evaluated individual

is from the optimal solution. Using a set of selection, reproduction, and mutation operators, the population evolves to select the fittest individuals that satisfy the optimization problem.

Quantum genetic algorithms are the quantum version of the classical genetic algorithms. The base for most QGAs is the population encoding using quantum bits or qubits [41]. A qubit is the smallest unit of information in quantum computing [42], composed of two states $|0\rangle$ y $|1\rangle$, which are orthonormal vectors that represent the computational basis:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1}$$

A single qubit can be in the state $|0\rangle$, $|1\rangle$, or in a superposition of both states, see equation 2. The coefficients α and β are called probability amplitudes. Because each qubit has a probability amplitude for the two given states of the computational basis, it is said that the qubit is in a *superposition* of states $|0\rangle$ and $|1\rangle$.

$$|\psi_j\rangle = \alpha_j|0\rangle + \beta_j|1\rangle \quad \text{with} \quad \alpha_j, \beta_j \in \mathbb{C}^2, \quad |\alpha_j|^2 + |\beta_j|^2 = 1 \tag{2}$$

Each individual is encoded using a quantum chromosome $|\Psi_i\rangle$, which is a set of n qubits [36], [40]. To calculate the fitness of the individual at any given time t, first, the quantum chromosome has to be measured. When the quantum chromosome is measured, every qubit collapses to a bit, forming a classical chromosome, as shown in (3); the symbol $\mathscr{O}$ represents the measurement operation.

$$|\Psi_i\rangle^t = \begin{bmatrix} \alpha_{i,1}^t & \alpha_{i,2}^t & \cdots & \alpha_{i,n}^t \\ \beta_{i,1}^t & \beta_{i,2}^t & \cdots & \beta_{i,n}^t \end{bmatrix} \xrightarrow{\mathscr{O}} [1|0| \cdots |1]_i^t \tag{3}$$

Measurements in quantum computation are done by using measurement operators M which are matrices designed to get information about the probability of measuring any given state m. The operators (also called observables) are Hermitian operators that have spectral decomposition $M = \sum \lambda_m P_m$; where P_m is a projector onto the eigenspace of M with eigenvalue λ_m[42]. Using Born's rule, we can calculate the probability of measuring λ_m and with that calculate the post-measurement state as:

$$|\psi_j'\rangle = \frac{P_m|\psi_j\rangle}{\sqrt{\langle\psi_j|P_m|\psi_j\rangle}} \tag{4}$$

After we get the classical chromosome, the fitness can be calculated as in any GA. The probability amplitudes of qubits in a quantum chromosome can be modified through quantum gates. A quantum gate is a unitary operator $\mathbb{U}$, which satisfies $\mathbb{U}^\dagger\mathbb{U} = \mathbb{U}\mathbb{U}^\dagger$, where $\mathbb{U}^\dagger$ is the hermitian adjoint of $\mathbb{U}$.

Most of the proposals for QGAs use the $R_y(\theta)$ gate to update the individual. The $R_y(\theta)$ modifies the probability of measuring a $|0\rangle$ or a $|1\rangle$ state, and is represented by:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{5}$$

In (6) the rotation gate $R_y(\theta)$ acts over the qubit $[\alpha_j, \beta_j]^T$, generating the new quantum state $[\alpha_j', \beta_j']^T$.

$$\begin{bmatrix} \alpha_j' \\ \beta_j' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix}. \tag{6}$$

Another useful quantum gate is the Hadamard gate, or H gate, which is given by (7). This gate acts on a single qubit changing the $|0\rangle$ state to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and the $|1\rangle$ state to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The idea of using the H gate is to use it whenever a classical random population or individual needs to be created since this gate can put a qubit in an equal superposition state on the computational basis.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7}$$

The steps to implement the QGA are given in Algorithm 1. As previously mentioned, in a QGA, the quantum chromosome $|\Psi_i\rangle$ represents a quantum individual formed by n qubits. When $|\Psi_i\rangle$ is measured, it generates a binary string of n bits. The first step in the QGA is to generate the initial quantum population and the initial set of solutions, see lines 3–5. To achieve this, an array of N quantum registers $Q^t = \{|\Psi_1\rangle^t, |\Psi_2\rangle^t, \ldots, |\Psi_N\rangle^t\}$ with n qubits is set to the $|0\rangle$ state, i.e. $Q^t \leftarrow \{|0\rangle^{\otimes n}, ..., |0\rangle^{\otimes n}\}$, where $t = 0$. Every quantum chromosome in the quantum population is then put into superposition using the Hadamard transform $H^{\otimes n}$, line 3. The initial solutions B_i^t are generated using a quantum measurement operator, line 4.

The initial solutions are then evaluated using a fitness function and ordered based on their fitness. The binary strings and the quantum individuals are also ranked based on the fitness of their corresponding solutions, and the best individual is stored, see lines 6–8.

The next step is to evolve the population to find the optimal solution, see lines 9–18. The evolution of the quantum population is done by changing the probability amplitude of all the n-qubits in the basis states $\{|0\rangle, |1\rangle\}$ using the R_y rotation gate. This rotation operation is illustrated using the 2-D qubit representation shown in Figure 2, where the x-axis represents the probability amplitude of $|0\rangle$ state, the y-axis the probability amplitude of $|1\rangle$ state, and the positive or negative signs of the axis indicate the phase value which does not affect the outcome. The idea behind quantum rotation is to change the probability amplitude of every qubit in each quantum chromosome in the direction of the best individual. In other words, the evolution of the quantum population increments the probability that the outcome of the measurements of every quantum chromosome is close to the best individual but with a low probability of measuring any other value on the search space. This inherently can resolve issues that local optimal values might generate.

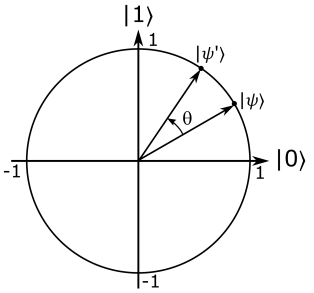


Fig. 2: The quantum qubit $|\psi\rangle$ rotates θ degrees in the direction of state $|1\rangle$.

The direction ω of the rotation angle is calculated by

Algorithm 1: Quantum genetic algorithm

Data: Quantum population $Q^t \leftarrow \{|0\rangle^{\otimes n}, ..., |0\rangle^{\otimes n}\}$
Result: Best solution b

1 **begin**
2 $t \leftarrow 0$;
3 Put in superposition every quantum individual, $Q^t \leftarrow H^{\otimes n} \otimes |\Psi_i\rangle^t$;
4 Measure each quantum individual, $B^t \leftarrow \frac{P_m|\Psi_i\rangle^t}{\sqrt{\langle\Psi_i|P_m|\Psi_i\rangle^t}}$;
5 Evaluate the population and store the solutions, $S^t \leftarrow f(B_i^t)$;
6 Order the solutions based on the fitness, $S^t \leftarrow S_{\downarrow}^t$, where $\downarrow$ indicates descending order;
7 Order the binary strings based on the solution index, $B^t \leftarrow B_{\downarrow}^t$;
8 Store the best individual, $b \leftarrow B_1^t$;
9 **repeat**
10 Calculate the angle direction for every qubit, $\omega \leftarrow \texttt{QRS}(|\Psi_i\rangle^{t+1}, S_i^t, B_i^t, b)$;
11 Update every quantum individual, $|\Psi_i\rangle^{t+1} \leftarrow Ry_i^j(\omega * \theta)) \otimes |\Psi_i\rangle^t$;
12 Measure each quantum individual, $B^{t+1} \leftarrow \frac{P_m|\Psi_i\rangle^{t+1}}{\sqrt{\langle\Psi_i|P_m|\Psi_i\rangle^{t+1}}}$;
13 Evaluate the population, $S^{t+1} \leftarrow f(B_i^{t+1})$;
14 Order the solutions based on the fitness, $S^{t+1} \leftarrow S_{\downarrow}^{t+1}$;
15 Order the binary strings based on the solution index, $B^{t+1} \leftarrow B_{\downarrow}^{t+1}$;
16 Store the best individual, $b \leftarrow B_1^{t+1}$;
17 $t \leftarrow t + 1$;
18 **until** *stop criteria*;
19 **end**

using a quantum rotation scheme (QRS), see line 10. The QRS is a lookup table in which the best individual b is compared against any other individual x^t in the population. The direction value $\omega \in \{0, +1, -1\}$ is calculated by comparing both individuals quantum state at x^t.

There are several proposals of rotation schemes that have been published, as described in [43]. One of the most popular QRS was proposed by Han [36], which is described in Table I. In this Table, the first two columns show each bit in the binary strings of x^t and b. The third column compares the individual's aptitude; here, we compare whether the current individual x^t is better ($f(x_i^t) < f(b)$) or worse ($f(x_i^t) > f(b)$) than the present best individual b. In the last columns are the four possible combinations of the phases in each qubit of x^t. Therefore, we selected the rotation direction ω based on which case our current qubit x_j^t is (columns 1 to 4).

As an example, we compare an arbitrary individual x^t and the best individual b of a given population. Here, the value of the j-th bit of x^t is 0, the phases α and β of the j-th qubit of x^t are both positive, the j-th bit of b is 1, and the current individual x is worse than the best individual ($f(x_i^t) > f(b)$). Analyzing Table I, we can see that this case falls under row four and column four, which indicates that $\omega = 1$.

After we have calculated the direction ω, the quantum individual is updated using equation 5 with a fix value of θ, see line 11. Once the population is updated, it is measured and stored in binary strings, see lines 12–13. Next, the population is evaluated using the fitness function $f(\cdot)$. Finally, the solutions are sorted based on their fitness, and the best individual is selected, see lines 14–16. This process is repeated until the stop criterion is met, e.g., a specific number of generations.

B. Adaptive quantum genetic algorithm

Wang, Liu, Zhi, and Fu proposed the adaptive quantum genetic algorithm (AQGA) as an improvement of the QGA [37]. The AQGA eliminates the need for a quantum rotating scheme; it uses an adaptive angle instead of a fixed angle and adds two new quantum operations to increase population diversity.

The AQGA is described in Algorithm 2. The initial quantum population is generated using the same steps of the QGA, lines 3-8 in Algorithm 1. In the evolution step, the rotation scheme is substituted by a mathematical formulation, see line 16. The direction of the rotation is calculated using a determinant of the qubit amplitudes as follows:

$$\mathbf{D} = \begin{vmatrix} \alpha_{b,j} & \alpha_{i,j} \\ \beta_{b,j} & \beta_{i,j} \end{vmatrix} \tag{8}$$

where, $[\alpha_{b,j}, \beta_{b,j}]_t^T$ are the probability amplitudes of a qubit of the best individual at the generation t, and $[\alpha_{i,j}, \beta_{i,j}]_t^T$ are the probability amplitudes associated to an element of the quantum population. So, each qubit in the quantum chromosomes of every individual is compared to the qubits of the best individual. When $\mathbf{D} \neq 0$, the direction of the rotation angle is $-sgn(\mathbf{D})$; otherwise, the rotating angle can be $\{1, -1\}$.

For the rotation angle magnitude θ, the AQGA uses a dynamic angle instead of a fixed angle, see line 17. The rotation angle magnitude begins at a maximum value at $t = 0$ and diminishes its value through every generation. The idea is to use big rotation angles at the beginning to get a diverse population and small rotation angles in the last generations to increase the chances of getting optimal solutions. The update of the population is done using the same process as in QGA, see line 18.

Once updated, the AQG adds a quantum mutation operator, see line 19. The objective of the quantum mutation operator is to enable randomly selected individuals to deviate from the current evolutionary path preventing local optimal stagnation. The quantum mutation operator swaps the probability amplitudes $\{\alpha_j, \beta_j\}$ of randomly selected qubits in the quantum population. To swap the values of the probability amplitudes, the Pauli-X gate $X = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ is used.

The second operator added is the quantum disaster operation, see lines 21 to 23. The disaster conditions engage when the best solution does not change over a certain amount of generations. The objective of this operator is to prevent convergence to locally optimal solutions by applying a significant disturbance to some individuals in the population and generating some random individuals. The disturbance is generated by putting some qubits of the individuals into superposition, removing some individuals from the population and generating new individuals, or by a combination of both.

TABLE I: Quantum rotation scheme proposed by Han [36].

x_{ij}^t	b_j	$f(x_i^t) > f(b)$	$\alpha_{ij}^t \beta_{ij}^t > 0$	$\alpha_{ij}^t \beta_{ij}^t < 0$	$\alpha_{ij}^t = 0$	$\beta_{ij}^t = 0$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	-1	+1	$\pm$ 1	0
1	0	0	-1	+1	$\pm$ 1	0
1	0	1	+1	-1	0	$\pm$ 1
1	1	0	+1	-1	0	$\pm$ 1
1	1	1	+1	-1	0	$\pm$ 1

Algorithm 2: The Adaptive quantum genetic algorithm

Data: Quantum population $Q^t \leftarrow \{|0\rangle^{\otimes n}, ..., |0\rangle^{\otimes n}\}$
Result: Best solution b

begin
  $t \leftarrow 0$;
  Put in superposition every quantum individual, $Q^t \leftarrow H^{\otimes n} \otimes |\Psi_i\rangle^t$;
  Measure each quantum individual, $B^t \leftarrow \frac{P_m|\Psi_i\rangle^t}{\sqrt{\langle\Psi_i|P_m|\Psi_i\rangle^t}}$;
  Evaluate the population and store the solutions, $S^t \leftarrow f(B_i^t)$;
  Order the solutions based on the fitness, $S^t \leftarrow S_{\downarrow}^t$, where $\downarrow$ indicates descending order;
  Order the binary strings based on the solution index, $B^t \leftarrow B_{\downarrow}^t$;
  Store the best individual, $b \leftarrow B_1^t$;
  repeat
    Calculate the angle direction for every qubit, $\mathbf{D} \leftarrow \begin{vmatrix} \alpha_{b,j} & \alpha_{i,j} \\ \beta_{b,j} & \beta_{i,j} \end{vmatrix}$,
    if $\mathbf{D} == 0$ **then**
      $\omega \leftarrow \{-1, 1\}$;
    else
      $\omega \leftarrow -sgn(\mathbf{D})$;
    end
    Calculate the rotation angle, $\theta_t \leftarrow \theta_{max} - \frac{\theta_{max}-\theta_{min}}{t_{max}} * t$;
    Update every quantum individual, $|\Psi_i\rangle^{t+1} \leftarrow Ry_i^j(\omega * \theta_t)) \otimes |\Psi_i\rangle^t$;
    Apply mutation operation at random, $|\psi_{i,j}\rangle^{t+1} \leftarrow X_{i,j} \otimes |\psi_{i,j}\rangle^t$;
    Check disaster condition, and apply it
    $|\Psi_i\rangle^{t+1} \leftarrow |0\rangle^{\otimes n}$,
    $|\Psi_i\rangle^{t+1} \leftarrow H^{\otimes n} \otimes |\Psi_i\rangle^t$
    where $i = \{N - r, N - r + 1, ..., N\}$;
    Measure each quantum individual, $B^{t+1} \leftarrow \frac{P_m|\Psi_i\rangle^{t+1}}{\sqrt{\langle\Psi_i|P_m|\Psi_i\rangle^{t+1}}}$;
    Evaluate the population, $S^{t+1} \leftarrow f(B_i^{t+1})$;
    Order the solutions based on the fitness, $S^{t+1} \leftarrow S_{\downarrow}^{t+1}$;
    Order the binary strings based on the solution index, $B^{t+1} \leftarrow B_{\downarrow}^{t+1}$;
    Store the best individual, $b \leftarrow B_1^{t+1}$;
    $t \leftarrow t + 1$;
  until *stop criteria*;
end

In our implementation, we replace the r worst individuals with new individuals. After the disaster operator, the new population is measured, new binary strings are generated, evaluated, and sorted. Then, the process is repeated for a certain amount of generations.

III. Implementing a QIEA in a quantum circuit

To implement a QIEA on a NISQ device, the architecture of the quantum device must be considered The IBM quantum device encodes quantum algorithms with quantum circuit representation [44]. Quantum circuits use quantum qubits, quantum gates, and measurements to implement quantum algorithms.

The first step to create a quantum circuit for the QIEA is to encode the information using qubits. Most QIEAs represent qubits using a two-axis circle, where the x-axis indicates the probability amplitude of $|0\rangle$ state, and the y-axis the probability amplitude of $|1\rangle$ state. In quantum circuits, qubits can be represented by their geometrical form:

$$|\psi_j\rangle = \exp(i\phi)(\cos(\theta/2)|0\rangle + \exp(i\gamma)\sin(\theta/2)|1\rangle), \quad (9)$$

where θ, γ and ϕ are real numbers, $0 \leq \theta \leq \pi$ and $0 \leq \gamma \leq 2\pi$, and i is the imaginary portion of the complex number. The sphere that represents the equation above is called the Bloch sphere, which is illustrated in Figure 3a. Since the value of ϕ does not hinder any information, only the polar and azimuth angles are considered.

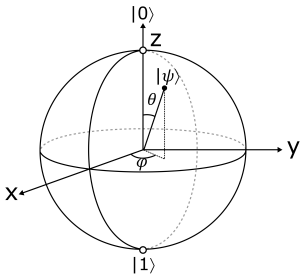
Quantum individuals can be encoded in the circuit representation using the concepts mentioned above as quantum registers with n qubits. Individuals and their evolution are kept in the quantum computer and handled as a quantum circuit. At $t = 0$, for every individual, each qubit is set to $|0\rangle$, then a Hadamard gate is applied to each qubit. Then, the individuals are measured, generating the initial population. Because measuring quantum registers destroys quantum information, it is necessary to generate a new quantum circuit at every generation t.

The next step is to update the quantum population in the quantum circuits. As previously stated, in the QIEA, a quantum individual is updated using a quantum rotation scheme to change the probability amplitudes of its qubits.

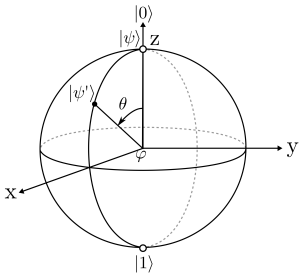
Taking into account the Bloch sphere, see Figure 3a, for the QIEAs, the polar angle gives the most helpful information since it provides the probability of measuring the quantum states $|0\rangle$ and $|1\rangle$. Any rotation in the azimuth angle does not change the probability of measuring the basis states. The aforementioned is represented in Figure 3b, where the value of the qubit is updated only using the polar angle. Since the polar angle is the only angle to be rotated, the update is done

TABLE II: Qubit rotation scheme proposed by Nicolao, Schirru, and Monteiro [45].

x_{ji}^g	b_i	$f(x_j^g) < f(b)$	$\langle\sigma_x\rangle > 0$	$\langle\sigma_x\rangle < 0$	$\langle\sigma_x\rangle = 0$	$\langle\sigma_z\rangle = 0$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	+1	-1	0	±1
0	1	1	-1	+1	±1	0
1	0	0	-1	+1	±1	0
1	0	1	+1	-1	0	±1
1	1	0	0	0	0	0
1	1	1	0	0	0	0



(a) Bloch sphere



(b) Qubit update using the rotation angle θ over the polar axis.

Fig. 3: Rotation of the qubit in the Bloch sphere.

by using the $R_y(\theta)$ gate over the qubits, which is analogous to the update operation of QIEA in classical devices.

The next step is to calculate the rotation direction to perform the update. For the rotation scheme, we decided to adopt the proposal of Nicolao, Schirru, and Monteiro [45]. The qubit rotation scheme is described in Table II.

Additionally to selecting a rotation scheme, given the nature of the Bloch sphere representation, we need to calculate the position of the qubit to determine in what direction to move to any desired state. To do this, we calculate the expectation values of the Pauli spin operators [42] for the x-axis and z-axis, see equation (10). With the expectation values, we calculate the position of the qubit in the sphere, and with that decision, if the direction needs to be positive or negative to be closer to the desired state.

$$\begin{aligned} \langle\sigma_x\rangle &= \langle\psi|\sigma_x|\psi\rangle, \\ \langle\sigma_z\rangle &= \langle\psi|\sigma_z|\psi\rangle \end{aligned} \tag{10}$$

Once the magnitude θ and direction ω of the rotation angle are calculated for each qubit in $|\Psi_i\rangle$, the qubits are rotated using the $R_y(\theta)$ gate.

The mutation operation is implemented using the Pauli-X gate, applied randomly in some qubits of the quantum circuit. After that, each quantum circuit representing a quantum individual is measured and evaluated using the fitness function. The population is ranked and the process is repeated until the stop criteria is met.

The IBM Quantum Experience provides cloud access to several quantum devices, and a quantum simulator also accessed through the cloud. The proposed methodology creates a quantum circuit of n qubits, applying equal superposition, updates the quantum circuit, uploads the circuit to the cloud service, measures the quantum circuit in the quantum device, and retrieves the classical data.

A notable limitation of implementing QIEAs with tens of generations in current quantum devices is the depth of the circuit. If at every generation we applied $R_y(\theta)$ gate in each qubit in the quantum circuit, the minimum depth of the circuit for each quantum individual would be $g+1$, where g is the number of generations. Since the mutation operator only acts on rare occasions, the average depth will not change. To reduce the depth of the quantum circuits, we store the cumulative rotation angle in a matrix and apply the $R_y(\theta)$ gate with this cumulative angle.

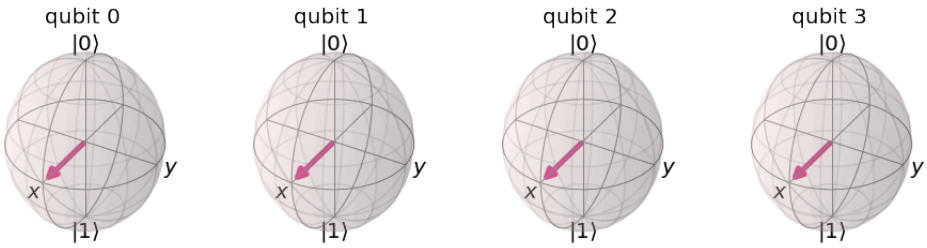
An example of the evolution of a 4-qubit quantum individual is shown in Figure 4. In $t = 0$, all of the qubits are in an equal superposition state, which means that any $\{0, 1\}^4$ state can be measured, see Figure 4a. At every generation, each individual in the population modifies the probability amplitudes of their qubits, evolving to the optimal value. Figure 4b shows how the 4-qubit quantum individual has changed the probability amplitudes of its qubits, intending to get near the optimal value.

The quantum circuit after several generations can be seen in Figure 5. At this point, the amplitude of each qubit has changed, moving closer to the optimal value, and a mutation has occurred in q_3. Even though several generations have passed, the depth of the circuit is three, which significantly reduces the time to execute the algorithm in the quantum computer. Reducing the depth of the circuit to an average of three gates enables evolving the quantum population for tens or hundreds of generations, which significantly benefits the results of QIEAs.

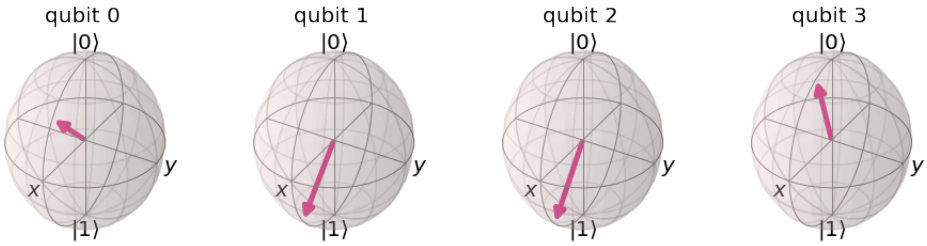
All the modifications mentioned above allow the implementation of QIEAs in a circuit based quantum devices like the one provided by IBM.

IV. Experiments

At present, hardware limitations can reduce the desired performance of QIEAs on quantum devices considerably. The main factors are cloud connectivity, the low number of qubits, and the noisy nature of the NISQ configuration. Cloud connectivity affects the retrieval time of results; this means that small quantum populations with a limited number of generations are desired. Until July 7th of 2021, IBM had



(a) The quantum individual at $t = 0$.



(b) Evolution of the quantum individual trough four generations ($t = 4$), each qubit evolves to optimal value.

Fig. 4: Evolution of a 4-qubit quantum individual.

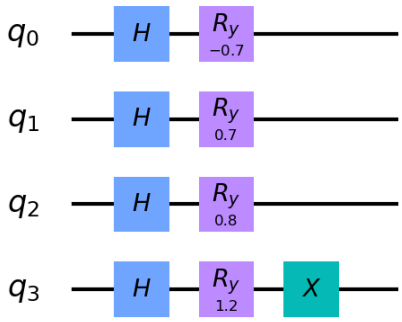


Fig. 5: Example of update of quantum circuit of a 4-qubit quantum individual. The presence of an X gate in q_3 indicates a quantum mutation ocurred at some point.

one machine with 15 qubits available to the public (the *ibm_16_melbourne* backend) and several 5-qubit devices, reducing the available precision for optimization problems. Additionally, the inherently noisy nature of these devices gives a higher error rate and a lower performance than most simulation processes, which affects the fidelity of the evolution of the quantum population. Currently, any possible computational gain is lost by the cloud connection.

Considering these limitations, to test our proposal's validity, we implemented the QGA with mutation and the AQGA to solve 1-dimensional optimization problems on IBM quantum devices. Although the traditional QGA does not include the mutation operation, in our experience, the mutation adds a better performance overall for QGA. The probability of mutation for both algorithms is 0.05. In the case of the AQGA, the calculus of the rotation direction that uses the determinant $\mathbf{D}$ was replaced by the rotation scheme proposed by Nicolao [45], shown in Table II. This decision was taken because any computational speedup given by the determinant $\mathbf{D}$ is lost by the cloud connection and using the selected rotation scheme showed better results.

The magnitude of the rotation angle θ was set to 0.1π for the QGA, whereas for the AQGA the magnitude of θ_t was dynamically adjusted according to the generation using $\theta_t = \theta_{max} - (\theta_{max} - \theta_{min}) * t/t_{max}$ with $\theta_{max} = 0.15\pi$ and $\theta_{min} = 0.05\pi$.

We selected 12 test functions; all of them are minimization problems. The test functions for the experiments can be seen in Table III.

We used a limited population of 10 individuals for the experiments, each one with ten qubits, and evolved through ten generations. Every quantum circuit was measured 1024 times without error mitigation, and each optimization process was executed 30 times.

We selected two control sets using a GPU-based quantum simulator developed by our research lab [46] and the IBMQ QASM cloud quantum simulator. Each optimization process ran 50 times in each quantum simulator (IBM QASM and the GPU-based simulator) to get a better statistical representation of the algorithm behavior.

V. Results

For each experiment, we calculated the mean, standard deviation, and accuracy rate. Execution accuracy is calculated as follows: every solution with a difference of 1×10^{-3} of the optimal value was classified as accurate. The results of the QGA with the mutation operator are described in Table IV; the best results are highlighted in bold. On most of the problems, the algorithm showed better performance (higher accuracy and lower standard deviation) in both quantum simulators than when executed on the quantum device. For the *ibm_16_melbourne* implementation (IBM Q16), the average accuracy rate on all the functions was 0.59, obtaining a higher accuracy in just 1 of the 12 problems. On the other hand, the IBM QASM implementation achieved an average accuracy rate of 0.67 and the best performance in 4 of the 12 problems, while the implementation in the Quantum

TABLE III: Test functions for the experiments.

Name	Formula	Minimum	Range
TF1	$f(x) = \sin(x) + \sin(\frac{10}{3}x)$	-1.8995	$[2.7, 7.5]$
TF2	$f(x) = -\sum_{k=1}^{6} k \sin[(k+1)x + k]$	-16.5310	$[-10, 10]$
TF3	$f(x) = -(16x^2 - 24x + 5)\exp(-x)$	-3.8504	$[1.9, 3.9]$
TF4	$f(x) = -(1.4 - 3x)\sin(18x)$	-1.4890	$[0, 1.2]$
TF5	$f(x) = \sin(x) + \sin(\frac{10}{3}x) + \log(x) - 0.84x + 3$	-1.6013	$[2.7, 7.5]$
TF6	$f(x) = 2\cos(x) + \cos(2x)$	-1.5	$[-\pi/2, 2\pi]$
TF7	$f(x) = \sin^3(x) + \cos^3(x)$	-1.0	$[0, 2\pi]$
TF8	$f(x) = -\exp(-x)\sin(2\pi x)$	-0.7886	$[0, 4]$
TF9	$f(x) = \begin{cases} (x-2)^2, & \text{if } x \leq 3 \\ 2\log(x-2) + 1, & \text{otherwise} \end{cases}$	0	$[0, 6]$
TF10	$f(x) = -(x - \sin(x))\exp(-x^2)$	-0.0634	$[-10, 10]$
TF11	$f(x) = x\sin(x) + x\cos(2x)$	-9.5083	$[0, 10]$
TF12	$f(x) = \exp(-3x) - \sin^3(x)$	-1	$[0, 20]$

TABLE IV: Accuracy, mean and standard deviation of QGA with mutation in different platforms. Best results in bold.

	Quantum Simulator			IBM QASM			IBM Q16		
Function	Acc	Mean	SD	Acc	Mean	SD	Acc	Mean	SD
TF1	**0.74**	-1.8955	0.0066	**0.74**	-1.8959	0.0083	0.43	-1.8958	0.0043
TF2	**0.3**	-16.2906	0.6906	0.24	-15.8887	1.1943	0.20	-15.6634	1.5717
TF3	0.8	-3.8501	0.0005	**0.88**	-3.8503	0.0003	0.60	-3.8497	0.0006
TF4	0.7	-1.4846	0.0085	**0.72**	-1.4843	0.0008	0.23	-1.3824	0.1424
TF5	**0.92**	-1.5994	0.0093	0.76	-1.5923	0.0217	0.76	-1.5869	0.0608
TF6	**0.96**	-1.4998	0.0004	0.94	-1.4992	0.0038	0.86	-1.4981	0.0056
TF7	**0.98**	-0.9998	0.0002	**0.98**	-0.9998	0.0002	0.96	-0.9994	0.0030
TF8	**0.64**	-0.7853	0.0076	0.44	-0.7825	0.0088	0.46	-0.7837	0.0053
TF9	**0.92**	0.0002	0.0007	0.82	0.0014	0.0044	0.70	0.0161	0.0271
TF10	**0.92**	-0.0632	0.0003	0.80	-0.0628	0.0012	0.90	-0.0630	0.0005
TF11	**0.48**	-9.4329	0.2219	0.26	-9.4608	0.1392	0.10	-9.2756	0.2593
TF12	0.68	-0.9978	0.0043	0.56	-0.9922	0.0271	**0.90**	-0.9997	0.0006
Average	0.75	—	0.08	0.67	—	0.12	0.59	—	0.17

Simulator obtained the best accuracy in 9 of the 12 problems with an average rate of 0.75.

Moreover, in terms of standard deviation, the IBM Q16 had the higher mean standard deviation (SD) with 0.17; the simulators show better performance, with 0.12 mean SD in the IBM QASM, and 0.08 for the Quantum Simulator. These results indicate that, albeit the execution on the quantum computer yielded the lowest accuracy and higher SD, the algorithm did indeed converge to the optimal region in the majority of the runs but it was unable to reach the goal minimum value of $1 \times 10^{-}3$ in 40% of the runs (with the most difference in TF2 and TF11).

In the second experiment, the functions that had an accuracy lower than 0.5 in every platform were tested again, incrementing the number of generations to 20. As shown in Table V. The QGA improved its convergence in both simulators; however, its performance on the quantum device was rather inconsistent. In two of the five functions tested (TF2 and TF4), the QGA running on IBM Q16 obtained better results since the accuracy increased while the mean and standard deviation decreased. Nevertheless, in the other three problems, the QGA accuracy performance diminished, which indicates that the real issue concerning the non-optimal results of the IBM Q quantum computer does not lie in the algorithm's structure. Even if the number of generations increased and the population size grew, the results may remain out of the imposed convergence region.

The results of the AQGA implementation are displayed in Table VI. The results of the average accuracy were very similar to those attained by the QGA for the three platforms. For both implementations, the Quantum Simulator and the IBM QASM, the average accuracy rate slightly increased to 0.79 and 0.70, respectively. For the IBM Q16 implementation, the mean accuracy decreased to 0.57, with lower performance in half of the test functions. The mean SD measured was 0.17, 0.13, and 0.08 for the IBM Q16, IBM QASM, and GPU Quantum Simulator, respectively. These results indicate that the improvements of the AQGA did not enhance the algorithm's performance in a real quantum device. Moreover, the SD over the three platforms did not increment, which indicates that the overall precision remained the same.

Tables IV and VI indicate that TF2 and TF11 are big outliers in the performance metrics. Both test functions have the lowest accuracy and highest SD for both QIEAs in all the test platforms.

If we remove the outliers (TF2 and TF11), the IBM Q16 average accuracy is 0.68 for the QGA, and 0.66 for the AQGA, with an average SD the outliers of 0.025 and 0.005 for the QGA and AQGA, respectively. For the IBM QASM, the QGA reported an average accuracy of 0.76 with a average SD of 0.008, while in the AQGA the average accuracy value is 0.77 with an average SD of 0.005. The average accuracy of the Quantum Simulator is 0.826 for the QGA and 0.862 for the AQGA, with a SD of 0.004 and 0.015, respectively.

Figure 6 shows the average SD for every platform without TF2 and TF11. The lowest precision was for the AQGA using the Quantum Simulator and the QGA using the IBM Q16; the QASM simulator obtained the highest precision for the QGA, followed by the Quantum Simulator for the QGA.

The mean values of Tables IV and VI indicate that for many of the functions, the results on the three platforms are very close to the desired precision. QIEAs such as QGA and AQGA, are susceptible to having issues with fine tuning with lower epoch count due to randomness in the measurement.

TABLE V: Accuracy, mean and standard deviation of QGA for the lowest performing functions using 20 generations.

	Quantum Simulator			IBM QASM			IBM Q16		
Function	Acc	Mean	SD	Acc	Mean	SD	Acc	Mean	SD
TF1	-	-	-	-	-	-	0.16	-1.8180	0.1270
TF2	0.44	-16.4640	0.2612	0.42	-16.5109	0.0399	0.43	-16.4950	0.0857
TF4	-	-	-	-	-	-	0.8	-1.4856	0.0073
TF8	-	-	-	-	-	-	0.40	-0.7834	0.0049
TF11	0.80	-9.4311	0.1900	0.42	-9.5048	0.0084	0.06	-9.2354	0.2608

TABLE VI: Accuracy, mean and standard deviation of AQGA in different platforms. Best results in bold.

	Quantum Simulator			IBM QASM			IBM Q16		
Function	Acc	Mean	SD	Acc	Mean	SD	Acc	Mean	SD
TF1	0.74	-1.8962	0.0062	0.90	-1.8990	0.0014	**0.93**	-1.8992	0.0009
TF2	0.13	-16.3041	0.6641	**0.3**	-15.7705	1.3122	0.13	-15.6616	1.7464
TF3	0.84	-3.8502	0.0004	**0.93**	-3.8502	0.0005	0.43	-3.8498	0.0005
TF4	**0.84**	-1.4644	0.0812	0.70	-1.4846	0.0007	0.43	-1.4702	0.0106
TF5	0.72	-1.5937	0.0318	0.73	-1.5939	0.0255	**0.83**	-1.6005	0.0013
TF6	**0.94**	-1.4992	0.0042	0.90	-1.4996	0.0006	0.56	-1.4891	0.0134
TF7	0.98	-0.9999	0.0002	0.93	-0.9994	0.0013	**1.0**	-0.9999	0.0001
TF8	**0.74**	-0.7866	0.0040	0.63	-0.7856	0.0054	0.30	-0.7829	0.0056
TF9	**0.96**	0.0002	0.0004	0.66	0.0033	0.0071	0.76	0.0029	0.0113
TF10	**1.0**	-0.0633	0.0002	0.76	-0.0628	0.0012	0.90	-0.0628	0.0001
TF11	**0.54**	-9.4360	0.1805	0.26	-9.5083	0.1849	0.06	-9.5083	0.2592
TF12	**0.82**	-0.9949	0.0263	0.63	-0.9973	0.0269	0.46	-0.9960	0.0004
average	0.79	—	0.08	0.70	—	0.13	0.57	—	0.17

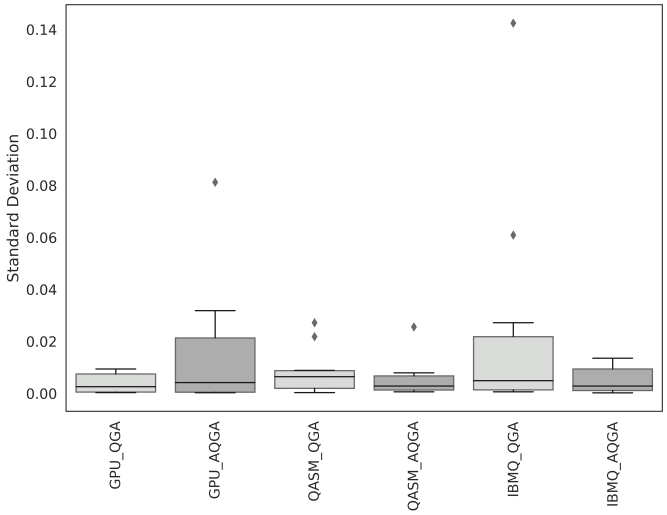


Fig. 6: Precision of the QGA and AGA implementations measured by the standard deviation. GPU refers to the GPU-based Quantum Simulator.

In this regard, local search algorithms might increment the accuracy if the best solution is not trapped in local optima. To test this, we implemented gradient descent in the QGA that runs in the IBM QASM; the results can be found in Table VII. The idea of implementing the QGA on the IBM QASM instead of IBM Q16 to reduce the influence of the classic algorithm on the quantum algorithm, i.e., that most of the optimization is done by the classical segment of the algorithm instead of the quantum elements of it.

As Table VII indicates, the average accuracy incremented from 0.73 to 0.87, and the standard deviation was reduced considerably from 0.0845 to 0.0037. The reduction of the standard deviation means that most of the solutions were in the same neighborhood which indicates that the best individuals are not stuck in local optima and can be fine-tuned with a local search algorithm.

We performed Welch's t-test to determine if the distribution of the results between the different platforms differs significantly. Therefore, we compared the IBM Q16 performance when running the QGA and AQGA against the IBM QASM and the GPU-based quantum simulator. Table VIII shows the hypothesis test result of the QGA running on the three different platforms. Similarly, Table IX show the hypothesis test result of the AQGA. The hypothesis H0 results on both algorithms indicate that there is no significant statistical difference between data sets. These results suggest that the difference in accuracy might be reduced with future hardware improvement, lowering the impact of noise in quantum circuits.

VI. Discussion and future work

This work presents a quantum circuit development to implement a QIEA on a circuit-based quantum computer. Due to their nature, not all QIMs are candidates for implementing easily (or at all) on available quantum devices. Because of that, we selected the QGA and the AQGA for

TABLE VII: Accuracy, mean and standard deviation of QGA with mutation using gradient descent.

Function	Acc	Mean	SD
TF1	0.76	-1.8985	0.0020
TF2	1.0	-16.5321	2.71×10^{-10}
TF3	1.0	-3.8504	7.43×10^{-5}
TF4	1.0	-1.4997	0.0005
TF5	1.0	-1.5994	0.0093
TF6	0.96	-1.4998	0.0004
TF7	0.98	-0.9998	0.0002
TF8	0.73	-0.7874	0.0022
TF9	0.92	0.0002	0.0007
TF10	0.92	-0.0632	0.0003
TF11	0.5	-9.5008	0.023
TF12	0.68	-0.9984	0.0021

TABLE VIII: Statistical hypothesis test of the QGA when tested on three different platforms.

Parameters	IBM Q16	IBM QASM	Quantum Simulator
Mean	-3.1678	-3.2414	-3.209
Variance	19.8886	21.3602	20.5329
Stand. Dev.	4.4597	4.6216	4.5313
d.o.f.	—	22	22
Critical value	—	3.792	3.792
t-value	—	0.0397	0.0224
Hypothesis H0	—	True	True
In all the cases, the significance level value is $\alpha = 0.001$; d.o.f means degrees of freedom.			

TABLE IX: Statistical hypothesis test of the AQGA when tested in the three different platforms.

Parameters	IBM Q16	IBM QASM	Quantum Simulator
Mean	-3.1931	-3.204	-3.2407
Variance	20.1019	20.3307	21.4001
Stand. Dev.	4.4844	4.509	4.626
d.o.f.	—	22	22
Critical value	—	3.792	3.792
t-value	—	0.0059	0.0256
Hypothesis H0	—	True	True
In all the cases, the significance level value is $\alpha = 0.001$; d.o.f means degrees of freedom.			

implementation. The translation from QIEA to QEA was done by taking into account the expectation value of Pauli spin operators instead of the probability amplitudes of the qubits.

The experiments were executed in three quantum platforms, which included a GPU-based Quantum Simulator, the IBM cloud QASM simulator, and IBM Q16 (*ibm_16_melbourne* backend). The accuracy metric shows that, for most of the experiments, the results of both simulators were very similar to each other. This fact demonstrates that the translation from comparing probability amplitudes to comparing expectation values of the Pauli spin operators for determining the orientation of the evolution was correct.

Nonetheless, according to the results, the behavior of the algorithms on the IBM Q16 varies considerably from that of the IBMQ QASM simulator. For example, compared to the simulator, the quantum device obtained a global mean 13% lower for the QGA and 24% lower for the AQGA. Furthermore, in terms of precision, the IBMQ QASM shows a lower average SD than the results in the Quantum Simulator and the IBM Q16. The main reason for this discrepancy might be related to the noisy nature of the quantum device and the lack of an error mitigation phase.

We theorize that the errors in measurements lead to a loss in subtle evolution for later stages, which might prevent the convergence of the population to the optimal value. Experiments with a higher amount of individuals seem to validate this hypothesis; see Table V. Furthermore, since the accuracy and SD improved for the simulators but not for the quantum device, the lower performance might point towards a cumulative error in the rotation and measurement in the quantum circuit.

Welch's test indicates that there is no statistical difference between the results of the three platforms. This is an indicator that noise correction might improve the results in quantum devices for QIEA optimization, and further improvements in hardware can put up to par classical and quantum implementations.

Quantum measurement plays an essential role in QIEAs since it is the most powerful tool for getting both diversity and convergence to the optimal value at the same time. Therefore, it is important to assess how much the measurement error influences the performance of the algorithms, and how much in reality is related to the structural design of the QEAs. In earlier generations, measurement error might not affect the algorithm's convergence since the diversity aids in creating several evolutionary routes. The problem arise in later generations when a delicate tuning of the values is needed to move the population in the right convergence direction. The noise generated by quantum measurement might drastically change the value of the outcome binary strings, generating a zig-zag pattern near the optimal value.

To study if the results of QIEAs improved using a local search technique, we ran tests using gradient descent in the IBM QASM. The results indicate that most of the best quantum individuals are near the optimal value and do not get stuck in local optima. This reinforces the idea that novel techniques for fine-tuning in later generations with noise

corrections plays an important role in running QIEAs on quantum devices.

From the analysis of the results, the lowest performance was obtained in functions with several local minima (TF2 has near 20 in the established range, TF11 and TF12 have 3). In addition, the value of the gradients of these test functions near the local and global minima are big. A slight variation in the individual can easily distance it from the optimal value, which might induce a stale evolution. As mentioned before, the simulation platforms had better performance when incrementing the population size and generations, but this increment was not drastic. The lack of performance increase might reflect an inner limitation of QA and AQGA, which might be solved by QIEAs that implement fine tuning in latter epochs with a quantum circuit design.

We will finish this discussion by answering the opening question "can quantum-inspired evolutionary algorithms take advantage of quantum computers? Throughout this paper, we have demonstrated that it is technically viable to translate a quantum-inspired metaheuristic —designed for a non-quantum-computer simulator— that provides optimal solutions to a quantum computer based on the circuit model. However, at present, the rate of repeatability (number of hits) of optimal solutions is low. We have also discussed that noise is a problem attributed to the fact that quantum technology is still in the early stages of development. Therefore, we conclude that it is worth continuing with the development of quantum evolutionary algorithms that exploit the limits of current NISQ devices to take advantage of future quantum technology. We expect that the noise problems responsible for not obtaining a high repeatability rate will eventually be reduced. Meanwhile, it is important to analyze QIEA performance in small populations with a few amount of generations to fully take advantage of current quantum devices despite their resource limitations.

REFERENCES

[1] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, ser. Quantum Science and Technology. Springer, Cham, 2018.

[2] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[3] M. Brooks, "Beyond quantum supremacy: the hunt for useful quantum computers," *Nature*, vol. 574, pp. 19–21, 2019, https://doi.org/10.1038/d41586-019-02936-3. [Online]. Available: https://www.nature.com/articles/d41586-019-02936-3

[4] R. Hughes and J. Nordholt, "Refining quantum cryptography," *Science*, vol. 333, no. 6049, pp. 1584–1586, 2011.

[5] Y. Rubio, O. Montiel, and R. Sepúlveda, "Quantum inspired algorithm for microcalcification detection in mammograms," *Information Sciences*, vol. 480, pp. 305 – 323, 2019.

[6] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6:1–6:20, 2018.

[7] H. Hussain, M. bin Javaid, F. S. Khan, A. Khalique, and A. Dalal, "Optimal control of traffic signals using quantum annealing," 2019.

[8] S. Feld, C. Roch, T. Gabor, C. Seidel, F. Neukart, I. Galter, W. Mauerer, and C. Linnhoff-Popien, "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer," *Frontiers in ICT*, vol. 6, p. 13, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2019.00013

[9] A. Al-Rabadi, "Intelligent control of singularly-perturbed reduced order eigenvalue-preserved quantum computing systems via artificial neural identification and linear matrix inequality transformation," *International Journal of Computer Science*, vol. 37, 2010.

[10] N. H. Nguyen, E. C. Behrman, and J. E. Steck, "Quantum learning with noise and decoherence: a robust quantum neural network," *Quantum Machine Intelligence*, vol. 2, no. 1, pp. 1–15, 2020.

[11] O. H. Montiel Ross, "A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers," *IEEE Access*, vol. 8, pp. 814–838, December 2019.

[12] J. Wright and I. Jordanov, "Quantum inspired evolutionary algorithms with improved rotation gates for real-coded synthetic and real world optimization problems," *Integrated Computer-Aided Engineering*, vol. 24, no. 3, pp. 203–223, 2017, https://doi.org/10.3233/ICA-170545.

[13] A. K. Beheshti, S. R. Hejazi, and M. Alinaghian, "The vehicle routing problem with multiple prioritized time windows: A case study," *Computers and Industrial Engineering*, vol. 90, pp. 402–413, 2015, https://doi.org/10.1016/j.cie.2015.10.005.

[14] Y. Li, M. Tian, G. Liu, C. Peng, and L. Jiao, "Quantum optimization and quantum learning: A survey," *IEEE Access*, vol. 8, pp. 23 568–23 593, 2020, https://doi.org/10.1109/ACCESS.2020.2970105.

[15] X. Wu and S. Wu, "An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 28, pp. 1441–1457, 2017, https://doi.org/10.1007/s10845-015-1060-6.

[16] L. R. da Silveira, R. Tanscheit, and M. Vellasco, "Quantum inspired evolutionary algorithm for ordering problems," *Expert Systems with Applications*, vol. 67, pp. 71 – 83, 2017, https://doi.org/10.1016/j.eswa.2016.08.067.

[17] H. Talbi and A. Draa, "A new real-coded quantum-inspired evolutionary algorithm for continuous optimization," *Applied Soft Computing*, vol. 61, pp. 765–791, 2017, https://doi.org/10.1016/j.asoc.2017.07.046.

[18] M. Udrescu, L. Prodan, and M. Vlăduţiu, "Implementing quantum genetic algorithms: A solution based on grover's algorithm," in *Proceedings of the 3rd Conference on Computing Frontiers*. New York, NY, USA: Association for Computing Machinery, 2006, p. 71–82, https://doi.org/10.1145/1128022.1128034.

[19] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, https://doi.org/10.22331/q-2018-08-06-79.

[20] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," https://arxiv.org/abs/1907.02359, 2019.

[21] P. Dupuy de la Grand'rive and J.-F. Hullo, "Knapsack problem variants of qaoa for battery revenue optimisation," https://arxiv.org/abs/1908.02210, 2019.

[22] W.-J. Liu, P.-P. Gao, W.-B. Yu, Z.-G. Qu, and C.-N. Yang, "Quantum relief algorithm," *Quantum Information Processing*, vol. 17, no. 10, Sep 2018, http://dx.doi.org/10.1007/s11128-018-2048-x.

[23] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," https://arxiv.org/abs/1912.08278, 2019.

[24] U. Alvarez-Rodriguez, M. Sanz, L. Lamata, and E. Solano, "Artificial life in quantum technologies," *Scientific Reports*, vol. 6, p. 20956, 2016, https://doi.org/10.1038/srep20956.

[25] J. Niu, Y. Zhang, and Z. Li, "Quantum-behaved particle swarm optimization algorithm based on dynamic dual-population joint-search mechanism," *International Journal of Computer Science*, vol. 46, 2019.

[26] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, 2004, pp. 325–331 Vol.1.

[27] B. Haddar, M. Khemakhem, S. Hanafi, and C. Wilbaut, "A hybrid quantum particle swarm optimization for the multidimensional knapsack problem," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 1 – 13, 2016, https://doi.org/10.1016/j.engappai.2016.05.006.

[28] F. B. Ozsoydan and A. Baykasoğlu, "Quantum firefly swarms for multimodal dynamic optimization problems," *Expert Systems with Applications*, vol. 115, pp. 189–199, 2019, https://doi.org/10.1016/j.eswa.2018.08.007.

[29] X. Lai, J.-K. Hao, Z.-H. Fu, and D. Yue, "Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem," *Expert Systems with Applications*, vol. 149, p. 113310, 2020, https://doi.org/10.1016/j.eswa.2020.113310.

[30] R. Pavithr and Gursaran, "Quantum inspired social evolution (qse) algorithm for 0-1 knapsack problem," *Swarm and Evolutionary Computation*, vol. 29, pp. 33–46, 2016, https://doi.org/10.1016/j.swevo.2016.02.006.

[31] Y.-N. Guo, P. Zhang, J. Cheng, C. Wang, and D. Gong, "Interval multi-objective quantum-inspired cultural algorithms," *Neural Computing and Applications*, vol. 30, no. 3, pp. 709–722, 2018, https://doi.org/10.1007/s00521-016-2572-5.

[32] A. Ibrahim, A. Mohamed, and H. Shareef, "A novel quantum-inspired binary gravitational search algorithm in obtaining optimal power quality monitor placement," *Journal of Applied Sciences*, vol. 12, pp. 822–830, 09 2012, https://doi.org/10.3923/jas.2012.822.830.

[33] H. Nezamabadi-pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 40, pp. 62 – 75, 2015, "https://doi.org/10.1016/j.engappai.2015.01.002.

[34] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355

[35] D. Câmara, "1 - evolution and evolutionary algorithms," in *Bio-inspired Networking*, D. Câmara, Ed. Elsevier, 2015, pp. 1 – 30, "http://www.sciencedirect.com/science/article/pii/B9781785480218500016.

[36] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, Dec 2002.

[37] H. Wang, J. Liu, J. Zhi, and C. Fu, "The improvement of quantum genetic algorithm and its application on function optimization," *Mathematical Problems in Engineering*, vol. 2013, 2013, https://doi.org/10.1155/2013/730749.

[38] F. P. Mahdi, P. Vasant, M. Abdullah-Al-Wadud, J. Watada, and V. Kallimani, "A quantum-inspired particle swarm optimization approach for environmental/economic power dispatch problem using cubic criterion function," *International Transactions on Electrical Energy Systems*, vol. 28, no. 3, p. e2497, 2018, https://doi.org/10.1002/etep.2497.

[39] M. M. Moghadam, H. H. Nezamabadi-Pour, and M. M. Farsangi, "A quantum behaved gravitational search algorithm," *Intelligent Information Management*, vol. 4, no. 6, pp. 390–395, 2012, https://doi.org/10.4236/iim.2012.46043.

[40] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 2, July 2000, pp. 1354–1360 vol.2, https://doi.org/10.1109/CEC.2000.870809.

[41] H. Miyajima, M. Fujisaki, and N. Shigei, "Quantum search algorithms in analog and digital models," *International Journal of Computer Science*, vol. 39, 2012.

[42] P. Wittek, "Quantum mechanics," in *Quantum Machine Learning*, P. Wittek, Ed. Boston: Academic Press, 2014, pp. 25–39. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128009536000037

[43] H. Xiong, Z. Wu, H. Fan, G. Li, and G. Jiang, "Quantum rotation gate in quantum-inspired evolutionary algorithm: A review, analysis and comparison study," *Swarm and Evolutionary Computation*, vol. 42, pp. 43 – 57, 2018.

[44] A. Córcoles, E. Magesan, S. Srinivasan, A. Cross, M. Steffen, J. Gambetta, and J. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Communications*, vol. 6, no. 679, 2015, https://doi.org/10.1038/ncomms7979.

[45] A. dos Santos Nicolau, R. Schirru, and A. M. M. de Lima], "Nuclear reactor reload using quantum inspired algorithm," *Progress in Nuclear Energy*, vol. 55, pp. 40 – 48, 2012, https://doi.org/10.1016/j.pnucene.2011.11.001.

[46] O. Montiel, Y. Rubio, C. Olvera, and A. Rivera, "Quantum-inspired acromyrmex evolutionary algorithm," *Scientific Reports*, vol. 9, no. 12181, 2019, https://doi.org/10.1038/s41598-019-48409-5.