# Quantum Encoded Quantum Evolutionary Algorithm for the Design of Quantum Circuits

Georgiy Krylov Department of Computer Science Nazarbayev University Astana,Kazakhstan Email: gkrylov@nu.edu.kz Martin Lukac Department of Computer Science Nazarbayev University Astana,Kazakhstan Email: martin.lukac@nu.edu.kz

Abstract-In this paper we present Quanrum Encoded Quantum Evolutionary Algorithm (QEQEA) and compare its performance against a a classical GPU accelerated Genetic Algorithm (GPUGA). The proposed QEQEA differs from existing quantum evolutionary algorithms in several points: representation of candidates circuits is using qubits and qutrits and the proposed evolutionary operators can in theory be implemented on quantum computer provided a classical control exists. The synthesized circuits are obtained by a set of measurements performed on the encoding units of quantum representation. Both algorithms are accelerated in GPGPU. The main target of this paper, is not to propose a completely novel quantum genetic algorithm but to rather experimentally estimate the advantages of certain components of genetic algorithm being encoded and implemented in a quantum compatible manner. The algorithms are compared and evaluated on several reversible and quantum circuits. The results demonstrate that on one hand the quantum encoding and quantum implementation compatible implementation provides certain disadvantages with respect to the classical evolutionary computation. On the other hand, encoding certain components in a quantum compatible manner could in theory allow to accelerate the search. This acceleration would in turn counter weight the implementation limitations.

#### I. INTRODUCTION

The direct design of quantum circuits, that is designing quantum or reversible circuits directly using a set of quantum universal gates, suffers from two main problems. First, the optimal method for constructing primitive logic reversible quantum gates for larger number of qubits is not known. The complexity of constructing such gates rises from the fact that component quantum gates applied to at maximum two qubits can be used. Second, designing large reversible and quantum circuits from macros, does not guarantee an exact minimal and optimal design. As a result, there is no real quantum circuit design algorithm for both quantum and reversible circuits using directly quantum primitives.

The synthesis of reversible quantum gates such as gates from the  $C^nU$  family with U being NOT, or SWAP unitary operations, has been solved in general for some sets of Turing universal quantum gates and small number of qubits. For instance, the minimal realization of  $C^2NOT$  gate is known in the  $CNOT/CV/CV^{\dagger}$ , Clifford-T or CH/CZ set of quantum gates. However, in the Ising model the Toffoli gate is not known with certainty as the original specification was found by a stochastic algorithm [1] while in [2] an improved realization was found. Additionally, this situation only gets worse with larger logic gates, where synthesis is done by LUT [3] or replacement of large gates by a group of smaller gates already known [4]. Thus, a synthesis method that designs larger quantum circuits directly using quantum gates would benefit from better minimal cost but also would require faster computers.

The evolutionary approach is one of the possible way to find cheaper realizations of the  $C^nU$  quantum gates. The reason is that while for smaller Turing-universal gate sets and relatively small circuits it is possible to enumerate all possible gate combinations and therefore obtain the less costly minimal gate realization; for larger sets of gates and quantum circuits enumeration would take too long. Consequently, a pseudoevolutionary search can accelerate the search for more optimal realization of small and medium sized quantum gates using evolutionary operators.

However, designing algorithm directly for quantum computer is not a trivial task. To do that the specific principles and constraints of quantum computing [5] has to be taken into account. Naturally, one can use classical algorithms implemented and accelerated in quantum computer such as SAT [6] but such algorithms suffer from the overhead of building classical mechanisms using quantum computational elements. Thus, implementing an algorithm that is as close as possible to quantum information and this manipulation using basic quantum tools is most desirable.

In this paper we propose a quantum encoded quantum evolutionary algorithm (QEQEA) parallelized on GPGPU and we compare it to an equivalently GPGPU accelerated classical evolutionary algorithm. QEQEA uses qubits and qutrits to represent parameters evolved by the quantum evolutionary operators as compared to classical genetic algorithm that uses classical strings. The evolutionary operators of QEQEA are strongly simplified and adapted to be quantum-realizable; the used evolutionary operators are built from unitary evolution and measurement process. The QEQEA, evolves simple quantum gates that are used to build the quantum circuits. From one single population of gates, several quantum circuits are sampled by measurement. As such the QEQEA is intended for ensemble quantum computer approach such as NMR [5] or One-Way quantum computing [7]. Each quantum gate is updated proportionally to fitness values stored in the nonquantum part of the algorithm. The QEQEA and the classical GPUGA are both evaluated on the Ising model of quantum computer [5] due to high complexity and high number of parameters to optimize. The results presented here are aimed to evaluate the difference between these two algorithms rather than provide new state-of-the art circuits realizations in currently used models of quantum gates such as Clifford-T.

In summary the following quantum-like modifications are implemented in QEQEA:

- ensemble-quantum computer inspired set of evolutionary operators,
- population of candidates solutions encoded using qubits and qutrits,
- adaptive mutation as the main driving force of the evolution,
- templates for building interaction gates,
- use of position in the memory to encode circuit information
- measurement based quantum gate and quantum circuit creation.

This paper is organized as follows. Section II introduces required knowledge about genetic algorithms and Section III introduces the required information about quantum computing and quantum circuits. Section IV introduces the proposed model. Section V describe the experimentation and obtained results and Section VI discusses the quantum implementation discrepancies and performance considerations of the proposed algorithm. Finally Section VII concludes the paper.

# II. BACKGROUND

The genetic algorithms are a class of pseudo-random search algorithms inspired by natural evolution. They are used to solve difficult search and optimization problems that are otherwise unsolvable by exhaustive or analytic approaches in reasonable amount of time or with bounded memory. The application of evolutionary algorithm for quantum synthesis is a topic that has been previously studied since the beginning of the century [8], [9], [10]. A lot of work has been done for solving the problem in classical paradigm using different approaches and hardware, however the execution time is a limiting factor even for the most optimal evolutionary and general algorithms [11] directly designing quantum circuits. Thus, Quantum and Quantum Inspired Algorithms were introduced in order to reduce the computation time using principles of quantum mechanics. One of the first evolutionary algorithm inspired by quantum computing was developed in [12]. The most original idea was the extension of quantum inference crossover [12]. In [13] the first definition and requirements for evolutionary quantum algorithms have been introduced.

The most important and challenging requirements are listed below for the clarity of understanding:

- A reasonable method of splitting the problem to subproblems
- "The number of universes required should be identified" [13], that is the number of quantum registers should be well described
- The computations should occur in parallel
- "There must be some form of interaction between all of the universes. The interference must either yield a solution, or new information for the universes to utilize in locating a solution" [13]

Several further studies described the Quantum Genetic Algorithms for general purposes [14] [15] such as for the knapsack problem. The problem of quantum circuits synthesis was studied using Quantum Evolutionary Algorithm (QEA) in [16]. The study [16] used integer representation of population, and demonstrated synthesis with multiple controlled NOT gates. In [17] the design of quantum circuits used qutrits for individual encoding. This allowed for more advantageous usage of mutation and ternary operators. In order to run these algorithms, most of the studies design special quantum encoding and mapping of evolutionary operators that could potentially allow to execute their algorithm on quantum computers. The simulation task is not trivial and requires optimization and performance acceleration by itself. Building quantum simulators is also developing because of the need of benchmark and corrections for upcoming quantum hardware [18].

The previous studies in the quantum and in quantum inspired evolutionary computation fields outline several possible improvements. For problems dealing with higher dimensional space such as multi-qubit complex vector space (Hilbert space) operators efficiency, the selection methodology and fitness evaluation should be evaluated for both performance and accuracy w.r.t to its classical counter parts.

# III. QUANTUM CIRCUITS AND QUANTUM GATES

Information in quantum circuit is represented by a qubit  $|\phi\rangle$  represented by a wave state  $|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$ . Multiple qubits are expanded into a quantum register using Kronecker product such as for two qubits  $|a\rangle$  and  $|b\rangle$  the joint state is

$$|\psi\rangle = |a\rangle \otimes |b\rangle = \alpha_a \alpha_b |00\rangle + \alpha_a \beta_b |01\rangle + \beta_a \alpha_b |10\rangle + \beta_a \beta_b |11\rangle$$
(1)

The logic operations applied upon qubits are specified by unitary matrices. For instance, to negate the qubit  $|a\rangle$  the X operator can be applied (note the coefficients reordered):

$$X|\psi\rangle = \beta_a \alpha_b |00\rangle + \beta_a \beta_b |01\rangle + \alpha_a \alpha_b |10\rangle + \alpha_a \beta_b |11\rangle \quad (2)$$

The Ising model of quantum computing [5] uses three single qubit quantum gates  $R_X(\theta)$ ,  $R_Y(\theta)$  and  $R_Z(\theta)$  and one two-qubit interaction quantum gate  $I_{ZZ}(\theta)$ . The gates are parameterized by an angle of rotation  $\theta$  from the range  $[0, 2\pi]$ . The rotations applied to a single qubit can be visualized using the unitary Bloch sphere on Figure1



Fig. 1. Bloch sphere

A sequence of single and two-qubit operators (gates) applied to a quantum register is called a quantum circuit. Example of a quantum circuit is shown in Figure 2.



Fig. 2. Example of arbitrary quantum circuit in the Ising model

The single qubit rotations can be described by following equations[1]:

• X direction

$$R_x(\theta) = e^{\left(\frac{-i\theta X}{2}\right)} = \cos\left(\frac{\theta}{2}\right) I_2 - i\sin\left(\frac{\theta}{2}\right) X = \begin{bmatrix}\cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{bmatrix}$$
(3)

• Y direction:

$$R_{y}(\theta) = e^{\left(\frac{-i\theta Y}{2}\right)} = \cos\left(\frac{\theta}{2}\right)I_{2} - i\sin\left(\frac{\theta}{2}\right)Y = \begin{bmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{bmatrix}$$
(4)

• Z direction:

$$R_{z}(\theta) = e^{\left(\frac{-i\theta Z}{2}\right)} = \cos\left(\frac{\theta}{2}\right)I_{2} - i\sin\left(\frac{\theta}{2}\right)Z = \begin{bmatrix} e^{-i(\theta/2)} & 0\\ 0 & e^{i(\theta/2)} \end{bmatrix}$$
(5)

The template for the two-qubit interaction in Ising model [1]:

$$J_{ij}(\theta) = e^{\frac{-i\theta}{2}} \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & e^{\theta} & 0 & 0\\ 0 & 0 & e^{\theta} & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}$$
(6)

# IV. QUANTUM EVOLUTIONARY ALGORITHM DESCRIPTION

The proposed approach is distinguished from previous work in the following points:

- synthesis on a level of single qubit rotations and twoqubit interaction gates,
- unique encoding of quantum population using qubits and qutrits,
- GPU accelerated version for simulation of quantum computer behavior; we offer unique mapping of quantum operators that enables optimization for GPU acceleration,

- predefined templates of interaction matrices for simplification of the search,
- evolutionary operators are a combination of adaptive mutation and SU(3) rotations (in case of qutrits).

The proposed algorithm is briefly depicted in Figure 3. The QEQEA does not evolve circuits directly; instead a set of quantum gates (segments) are evolved as a population. The circuits are obtained by probabilistic selection of gates from the population. Each gate is encoded by several quantum parameters and uses measurement procedure for circuit construction.



Fig. 3. High level flow of the quantum evolutionary algorithm

#### A. Quantum Gates Representation

In this work the Ising model of quantum circuits is used. While this model is only theoretically useful, it is the most complex. Generating results in this circuit model is an indicator of the performance of the applied algorithm. There are two types of primitive gates in Ising model: rotations and interactions that were described in Section III, and they were used in the QEQEA. In the QEQEA, each quantum gate is represented by a set of qubits and /or qutrits.

1) Rotation gates: The single qubit gates  $(R_X(\theta), R_Y(\theta))$ and  $R_Z(\theta)$ ) are encoded using one qubit and one qutrit. The angle of rotation  $\theta$  is represented by the qubit parameter specifying its complex amplitudes:  $e^{-i\pi\theta}$ . The axis of rotation is obtained by measuring the state of the qutrit. We repeat the measurement process multiple times to approximate the state of the qutrit, without eliminating uncertainty. The qutrit states:  $\{|0\rangle, |1\rangle, |2\rangle\}$  correspond to rotations around  $\{x, y, z\}$ axis, respectively.

2) Interaction gates: The second type of quantum gate we use is the two-qubit interaction. The interaction gate is equivalent to two parameterized Z rotation gates applied simultaneously to two qubits [1]. By introducing interaction matrices templates, we reduced the number of parameters required to construct the interaction gate to one. This parameter is the angle of rotation is obtained by copying qubit value similar to the case of single qubit gates construction. For numberOfWires

wires, there can be at most  $\binom{numberOfWires}{2}$  possible configurations (we call them templates).

3) Interaction Gate Templates: The interaction gate can be expressed in form of term-wise exponent of scalar multiple of gate parameter value and special diagonal matrix. The elements of this diagonal matrix are +1 and -1 depending on the wires on which the interaction is applied. Notice that the number of possible interaction templates grows slowly considering the size of the problem. Thus, all possible templates can be stored in memory. When an interaction gate is to be inserted in a quantum circuit, parallel exponent of elements of diagonal matrices instead of matrix multiplications between swap gates and interactions are performed. These templates allow to exclude swap gates for interaction between nonneighboring qubits and therefore gives potential to synthesize more optimal circuits.

#### **B.** Population Initialization

In the proposed QEQEA, the evolution is performed by local modification of a set of qubits and qutrits. These elementary quantum information units encode a set of quantum gates that are sampled into several possible quantum circuits. The set of evolved qubits (representing the population) can be conditionally split to two regions : single qubit rotations and interactions region. The population of the qubits and qutrits is defined by following parameters

- *sizeOfIndividual* parameter corresponding to the length of the circuit in gates (segments)
- *sizeOfPopulation* parameter corresponding to number of individuals(circuits) in the evolution. The segments count in population grows to:

$$sizeOfPopulation * sizeOfIndividual$$
 (7)

• *numberOfWires* - parameter corresponding to the number of qubits of the target quantum gate. From this parameter, the *interactionTemplatesNumber* is derived. Thus, *numberOfWires* parameter increases the amount of memory required to store the qubits population to:

# (interaction templates Chose + number Of Wires) \* size Of Population \* size Of Individual (8)

The number of qutrits is fixed at

# number Of Wires\*size Of Population\*size Of Individual,

because qutrits are used only for segments that represent single qubit rotations.

The number of individuals in the population raises the amount of initial information to explore. It also significantly increases the computational complexity. However, the tasks required to synthesize one individual may be executed in parallel and we aim to get most benefit of highly effective parallel capabilities of quantum computers at that stage of the algorithm. Figure 4 describes an example population that would have two individuals, targeting to synthesize the circuit consisting of four gates applied to three input qubits (wires). The first eight qubits encoding the circuit segments correspond to rotation applied on the first input wire (labeled "wire 1" in Figure 4). There are exactly eight qubits in this particular case because the population consists of two individuals of size four. Similarly, the next eight qubits correspond to rotation on the second wire (labeled "wire 2" in Figure 4). Same rules apply to the third set of eight qubits. The remaining twenty four qubits do not have qutrits allocated for them because they belong to interaction region and use pre-calculated templates instead of measured axis (labeled "Interactions" in Figure 4). The figure does not contain qutrits in it, the qutrits are described on the Figure 5.

# C. Segments construction

The population of qubits and qutrits encodes a set of segments. Each single qubit rotation gate is expanded to the width of the full circuit defined by numberOfWires parameter. For interaction gates, templates are expanded to the circuit width before the evolutionary process starts and then are simply retrieved from memory. The obtained segments (rotation and interaction) are used to build the target candidate circuit. The restriction of using only one quantum gate per segment allowed for more efficient acceleration on the GPGPU [2]. Additionally, as described in Section IV-B, the population uses structure to encode following properties

- The wire on which the gate should operate
- The position of a segment inside of the individual

The segment construction procedure consists of five steps and is illustrated in Figure 5.

- Step 1: Measure the qutrits to get an array of axes of rotation
- Step 2: Plug in the qubits values for corresponding matrix template determined by qutrit measure and parameter value defined by qubit
- Step 3: Rearrange the memory in parallel Kronecker Product friendly order
- Step 4: Apply Kronecker product simultaneously to rotation matrices
- Step 5: Construct interaction matrices in a way it was described above and put next to segments obtained from rotation matrices

Note that the allocated memory represents directly units of quantum information. Thus, references to memory allocation represent directly qubit and qutrit allocation in particular order.

#### D. Circuit construction

The proposed quantum circuit design method is a form of evolutionary algorithm heavily altered in order to allow some of its components to be directly mapped into a quantum computer. Additionally, the proposed algorithm is also intended to be efficiently implementable on a highly parallel device such as GPGPU.



Fig. 4. Example of memory allocation for three-qubit circuit of length four with two individuals in the population



Fig. 5. Process of Segment Construction

For the circuit of length of sizeOfIndividual we launch sizeOfIndividual parallel threads each indexed by  $index_{thread}$ . Each thread generates two random numbers:

- which Individual from range 0..sizeOf Population
- *whichRotationOrInteraction* from the range

0..number Of Wires+interaction Templates Number

These values are later used to calculate the index of segment to be plugged in the circuit:

$$segmentIndex = whichRotationOrInteraction * sizeOfIndividual * sizeOfPopulation + whichIndividual * sizeOfIndividual + index_{thread} (10)$$

The result of calculation is stored as reference to a segment in population for  $index_{thread}$  position in the circuit. An example of process is shown in the Figure 6.



Fig. 6. Building a circuit of length three affecting two qubits having two individuals in the population

We repeat this process *sizeOfPopulation* times to generate *sizeOfPopulation* circuits each iteration.

# E. Fitness Evaluation

The fitness value reflects the proximity of the synthesized circuit matrix S, to the target circuit matrix T. The possible values of selected function are ranging from 0 to 1, where 1 represents identical matrices and 0 being the opposite. The approach is based on property of unitary matrices that  $U^{\dagger}*U = I$ . The Equation 11 displays the actual fitness function:

$$fitnessValue(S) = 1 - \sqrt{\frac{size - |tr(S^{\dagger}T)|}{size}}$$
(11)

In this expression, the || operator corresponds to modulo operation. The tr operation represents calculation of the sum of diagonal elements. The *size* is normalization constant and is taken to be equal to  $2^{numberOfWires}$ .

1) Segment Fitness: Each segment used during circuit construction stage (Section IV-D) is assigned with a fitness value. The fitness value assigned to each segment is the same as the fitness value of the circuit it was used to construct.

Additionally, an elitist approach was implemented: if the new fitness value of a segment is better than the previous best value, the states of the qubits and qutrits are preserved, otherwise they get discarded. Finally, each segments fitness is tied to a particular position in a given circuit. That is, the same segment will be represented by various fitness values depending on the position where it was located within the synthesized circuit.

#### F. Evolutionary Search

The main differences between the classical GPUGA and the proposed QEQEA is the mutation operation and the lack of crossover. We use adaptive mutation inspired from the evolutionary strategies approach [19]. The mutation is proportional to the error, i.e. better individuals undergo less significant changes[20]. This approach is argued to be more effective than the mutation with constant probability and mutation range[21]. Additionally, the *probabilityOfMutation* parameter is introduced in the algorithm to make the mutation operation probabilistic.

Each individual undergoes change per iteration of our algorithm with probabilityOfMutation. Every time the mutation is to be performed, there are two equiprobable operations that may happen: qubits or qutrits mutation.

- We use the mutationRange parameter that determines the maximum possible change to qubit parameter. In our algorithm, it is taken to be fraction of  $\pi$ . The formula to calculate the mutation value is  $\pm(1-segmentFitness)*$ mutationRange. The qubit parameters are assumed to stay within  $[0, 2 * \pi]$  range, so after the mutation the resulting parameter is readjusted modulo  $2 * \pi$
- The qutrits mutation is performed by applying the arbitrary SU(3) rotations on a qutrit [22]. Such matrix can be generated using eight parameters: three rotation angles  $\theta_1, \theta_2, \theta_3$  from range  $0 < \theta < \pi/2$  and five phases  $\phi_1, \phi_2, \phi_3, \phi_4, \phi_5$  from range  $0 < \phi < 2 * \pi$ . Equation 12 shows the template used to calculate the mutation on the qutrits, with  $c_k = \cos\theta_k$  and  $s_k = \sin\theta_k$ .

During one step of mutation, one of these nine parameters is generated randomly from a domain of its possible values multiplied by 1-segmentFitness. The constructed operator is then applied to the target qutrit.

The crossover operation was intentionally removed from the model, as our genotype - the array of qubits and qutrits is used to generate a population of circuits. Thus, only one set of qubits and qutrits is evolved and the crossover is replaced by the location dependent segment fitness value.

#### V. RESULTS

#### A. Evaluation of QEQEA

To verify the QEQEA algorithm we tested it on several small quantum gates:  $C^2NOT$ , Peres and CNOT. Table I shows the results of the search for the CNOT gate.

The Table I presents the outputs from the algorithm obtained in the process of synthesizing a CNOT gate. Each row in the table from top to the bottom represent encoded circuit segments in the order they appear in the synthesized circuit. Each row of the table contains all information required to decode information about circuit segment. The first column contains the parameter value  $\theta$  representing the rotation. The second column determines whether the parameter  $\theta$  should be plugged to rotation or interaction template. The third column of the table contains the states of the qutrit, which after measurement indicate the direction of the rotation gate. The value of this column should be ignored if the segment is a two-qubit interaction. The fourth column indicates the axis of rotation obtained as a result of measurement.

TABLE I RESULT OF CNOT GATE SYNTHESIS (sizeOfIndividual=3, sizeOfPopulation = 1)

Parameter $\theta$	Index in memory	Qutrit states	Axis
$\pi/2$	0	-0.43 - 0.16i; 0.85 + 0.08i; 0.03 - 0.24i	у
$3\pi/2$	7	Interaction template between 1 and 2	
$3\pi/2$	2	0.39 - 0.66i; -0.43 + 0.43i; 0.16 - 0.14i	х

thus, Table I represents a CNOT circuit constructed using the following sequence of gates:  $R_{1y}(\theta = 1.570796)J_{12}(\theta = 4.712389)R_{1x}(\theta = 4.712389)$ .

Table II shows the resulting matrix of the obtained  $C^2NOT$  gate with the length of 16 segments. The schematic representation of the target Toffoli gate is shown in Figure 7a). Some terms of the matrix have differences from original Toffoli gate therefore the circuit obtained is not exact, however on average the error per term is  $\approx 0.02$ . The reason of not having exact gate appeared due to the convergence to local maximum.

TABLE II Result of Toffoli gate synthesis ( sizeOfIndividual=16)

0.894	0.000	0.004	0.000	0.101	0.000	0.000	0.000
0.000	0.916	0.000	0.001	0.000	0.080	0.000	0.004
0.004	0.000	0.967	0.000	0.000	0.000	0.029	0.000
0.000	0.004	0.000	0.121	0.000	0.000	0.000	0.875
0.101	0.000	0.000	0.000	0.894	0.000	0.004	0.000
0.000	0.080	0.000	0.004	0.000	0.916	0.000	0.001
0.000	0.000	0.029	0.000	0.004	0.000	0.967	0.000
0.000	0.000	0.000	0.875	0.000	0.004	0.000	0.121

# B. Comparing QEQEA and GA

The comparison of the performance was done between the QEQEA and the non-quantum GPUGA algorithm from [2]. The reason to compare QEQEA with algorithm from [2] is that the GPUGA provides similar algorithmic and acceleration basis for comparison. In fact, the QEQEA was developed as a quantum extension of the original non quantum algorithm. The main differences are:

- Representation: same mapping from memory to individual was implemented. The representation of quantum gate (segment) was performed using a set of real and complex coefficients
- The Evolutionary operators: two point crossover was used and the mutation was a random small alterations of the gate parameters.
- Selection was using the Stochastic Universal Sampling (SUS)
- Evolution occurred on the level of level of circuits, not on the individual gates (segments).

$$\mathbf{U} = \begin{bmatrix} e^{i\phi_1}c_1c_2 & e^{i\phi_3}s_1 & e^{i\phi_4}c_1s_2 \\ e^{-i\phi_4 - i\phi_5}s_2s_3 - e^{i\phi_1 + i\phi_2 - i\phi_3}s_1c_2c_3 & e^{i\phi_2}c_1c_3 & -e^{-i\phi_1 - i\phi_5}c_2s_3 - e^{i\phi_2 - i\phi_3 + i\phi_4}s_1s_2c_3 \\ -e^{-i\phi_2 - i\phi_4}s_2c_3 - e^{i\phi_1 - i\phi_3 + i\phi_5}s_1c_2s_3 & e^{i\phi_5}c_1s_3 & e^{-i\phi_1 - i\phi_2}c_2c_3 - e^{-i\phi_3 + i\phi_4 + i\phi_5}s_1s_2s_3 \end{bmatrix}$$
(12)



Fig. 7. a) Representation of Toffoli and b) Peres logic gates

• In the GPUGA no qutrits were used; we introduced the qutrits in QEQEA in order to avoid allocating extra memory for each type of the rotation gates (x,y,z) direction. This evolution of qutrits could possible reduce computation time required for each population step.

The common parts of both algorithms are in the GPU acceleration and parallelism. The computational overhead that was required for the implementation of the QEQEA is the amount of measurement used during the creation of candidate segments from the encoding qubits and qutrits. Despite these various implementation differences the two algorithms are evaluated for speed of convergence and ability to find the desired solution as many of their components were programmed in a similar manner.

The Table III shows the differences of speed in obtaining the various gates for which we tested both algorithms. Notice that in all cases the classical algorithm was faster than the QEQEA algorithm (iteration of QEQEA takes significantly more time). Thus even if the iteration number is smaller in QEQEA, the GPUGA is faster in real time and was able to converge to better results. The reason is the fact that the QEQEA is evolving gates rather than whole circuits while the classical GA evolves whole circuits. Additionally, the QEQEA generates solutions from a single set of encoding qubits and qutrits. As such there is no crossover because there is only one individual of qubits and qutrits. Consequently, because the main evolution mechanisms are selection and mutation, the proposed QEQEA is more related to evolutionary strategies rather than to genetic algorithm.

TABLE III Comparison of Results and performance between the QEQEA and a classical GPGPU

Function	(	QEQEA	GPGPU		
	Accuracy	No. Generations	Accuracy	No. Generations	
CNOT	1.0000	400	1.0000	200	
Toffoli	0.7047	13000	0.9663	34500	
CCCNOT	0.6464	limit	0.7539	650970	
Peres	0.5693	limit	0.9443	2M	

The first and the third columns of Table III display the accuracy of best results achieved by each algorithm. The iter-

ations number could also serve as a measure for performance comparison, however for the QEQEA this data is only partially available. The reason for that is the search of CCCNOT and Peres gates reached the maximum iterations limit of ten million iterations. However, this fact also means the result could be possibly improved if the higher limit for iterations was set. Careful reader may notice the difference between best available results shown in this table and in [2]. This is due to re-evaluation of accuracy of previously achieved results with respect to the new fitness function described above.

# VI. DISCUSSION ON THE PERFORMANCE AND REALISM OF THE IMPLEMENTATION

While the search for the gates was partially successful (and thus, confirming that the proposed approach converges), the main drawback of the QEQEA was the slower convergence due to the simulated quantum evolution of individual gates. However, the changes implemented are intended to simulate the implementation of the QEQEA using certain quantum components and thus, the main concern was the general convergence and feasibility.

In more details the following design choices of implementation of the QEQEA affected its overall performance.

- 1) First, the structure of the task requires our population being represented as floating point numbers. While this is a limitation for GPGPU it is an advantage for quantum computer where each qubit and qutrit exists in a state-wave state. Using quantum computer an amplitude estimation technique would have to be used in order to estimate the parameter  $\theta$ .
- Second, the use of adaptive mutation approach. The lack of crossover benefits from GPU acceleration, but the algorithm shows performance reduction compared with regular GPU accelerated genetic algorithms [2].
- 3) Third, the genetic algorithms utilizing elitism approach are more in danger of convergence to local maxima compared with the other approaches, but the specificity of the task enforces us to use this approach to preserve control over the population.
- 4) Additionally, observe that in general the QEQEA is less performant compared to the GPGPU (Table III). The main reason is due to the fact QEQEA evolves quantum gates and thus, each gate have fitness representing it general usefulness rather than its usefulness in a particular quantum circuit.

Several design choices making QEQEA computationally tractable prevent it from being directly ported to a quantum computer. The two most important restrictions imposed that would require several changes to the algorithm in order to make it quantum implementable are:

- 1) The measurement approach: for each circuit the qubits are measured and new quantum gates are generated while the unmeasured states of the qubits and qutrits are evolved. In a more realistic setting of the proposed method is to use weak measurement [23] that would allow to preserve the quantum states at least partially. While the usage of protective measurement directly on the encoding qubits and qutrits makes it impossible for the algorithm to be considered implementable on quantum computer (one would need an infinite amount of copies and evolve them in parallel) it allowed us to at least simulate the result of such process.
- 2) The evolution of quantum gates instead of the whole circuits. This was decided in order to avoid entanglement between elements of the quantum circuits in space and in time. While entanglement could be highly beneficial to quantum evolution it also makes the simulation of the evolution much more complex and computationally expensive.

#### VII. CONCLUSION

We introduced a QEQEA as a means for the synthesis of quantum circuits and we compared its performance with a parallelized version of classical GA. The QEQEA features certain components being a possible target for implementation in a quantum computer but in order to keep the implementation computationally tractable several design choices that made it impossible to port directly to a quantum computer.

Additionally even if all components of the algorithm were made quantum-implementation compatible, many components would remain classic. In particular this means, that even if QEQEA evolutionary components are mapped to a quantum computer, fitness function values, circuit information, algorithm flow control and other parameters require to be kept in a classical memory.

The comparison with the classical GPUGA showed that the quantum evolutionary model shows worse performance than the classical evolution. The inferior performance is due to many constraints included in the QEQEA that resulted in strong simplification of the evolutionary process. Consequently the main result is that the evolutionary process for computation as originally proposed in [24] seems to be most efficient when implemented in classical computer. In quantum computer, an efficient implementation requires the entanglement that would made the search much more efficient. However simulating such system on classical computer requires high computational resources and is not easily compared to a classical GA.

As future work, we plan to integrate weak-measurement for circuit generation and an additional mechanism in classical computer that keeps track of gates fitness with respect to a all circuits it was used to build. An even further improvement is to use more complex encoding such as qudits with higher number of bases states and evolve whole quantum circuits.

#### References

 S. Lee, S. J. Lee, T. Kim, J. S. Lee, J. Biamonte, and M. Perkowski, "The cost of quantum gate primitives," *Journal of Multiple-Valued Logic* and Soft Computing, vol. 5-6, no. 12, 2006.

- [2] M. Lukac and G. Krylov, "Study of gpu acceleration in genetic algorithms for quantum circuit synthesis," in 2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL), May 2017, pp. 213–218.
- [3] M. Soeken, M. Roetteler, N. Wiebe, and G. De Micheli, "Hierarchical reversible logic synthesis using luts," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. New York, NY, USA: ACM, 2017, pp. 78:1–78:6. [Online]. Available: http://doi.acm.org/10.1145/3061639.3062261
- [4] M. Szyprowski and P. Kerntopf, "An approach to quantum cost optimization in reversible circuits," in 2011 11th IEEE International Conference on Nanotechnology, Aug 2011, pp. 1521–1526.
- [5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [6] S. Santra, G. Quiroz, G. V. Steeg, and D. Lidar, "Max 2-sat with up to 108 qubits," 2013.
- [7] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Phys. Rev. A*, vol. 68, p. 022312, Aug 2003. [Online]. Available: https://link.aps.org/doi/10.1103/ PhysRevA.68.022312
- [8] M. Lukac and M. Perkowski, "Evolving quantum circuits using genetic algorithm," in *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 177–185.
- [9] K. Dill and M. Perkowski, "Evolutionary minimization of generalized reed-muller forms," in *Proc. of the International Conference on Computational Intelligence and Multimedia*, February 1997.
- [10] F. Hadjam and C. Moraga, "Evolutionary design of reversible digital circuits using imep the case of the even parity problem." in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–6.
- [11] M. Lukac, M. Kameyama, M. Miller, and M. Perkowski, "High speed genetic algorithms in quantum logic synthesis: Low level parallelization vs. representation," *Journal of multiple-valued logic and soft computing*, vol. 20, no. 1-2, pp. 89–120, 2012.
- [12] M. Moore and A. Narayanan, "Quantum-inspired computing," 12 1995.
- [13] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in Proceedings of IEEE International Conference on Evolutionary Computation, May 1996, pp. 61–66.
- [14] A. M. Mohammed, N. A. Elhefnawy, M. M. El-Sherbiny, and M. M. Hadhoud, "Quantum crossover based quantum genetic algorithm for solving non-linear programming," in 2012 8th International Conference on Informatics and Systems (INFOS), May 2012, pp. BIO–145–BIO– 153.
- [15] A. Malossini, E. Blanzieri, and T. Calarco, "Quantum genetic optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 231–241, April 2008.
- [16] S. Ding, Z. Jin, and Q. Yang, "Evolving quantum circuits at the gate level with a hybrid quantum-inspired evolutionary algorithm," *Soft Computing*, vol. 12, no. 11, pp. 1059–1072, Sep 2008. [Online]. Available: https://doi.org/10.1007/s00500-007-0273-9
- [17] M. Lukac, M. Perkowski, and M. Kameyama, "Evolutionary quantum logic synthesis of boolean reversible logic circuits embedded in ternary quantum space using heuristics," 2011.
- [18] T. Hner and D. S. Steiger, "0.5 petabyte simulation of a 45-qubit quantum circuit," 2017.
- [19] A. Auger, "Convergence results for the (1,)-sa-es using the theory of -irreducible markov chains," *Theoretical Computer Science*, vol. 334, no. 1, pp. 35 – 69, 2005. [Online]. Available: http: //www.sciencedirect.com/science/article/pii/S030439750400814X
- [20] L. Hong, J. H. Drake, and E. Özcan, "A step size based self-adaptive mutation operator for evolutionary programming," in *Proceedings* of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ser. GECCO Comp '14. New York, NY, USA: ACM, 2014, pp. 1381–1388. [Online]. Available: http://doi.acm.org/10.1145/2598394.2609873
- [21] S. Marsili Libelli and P. Alba, "Adaptive mutation in genetic algorithms," *Soft Computing*, vol. 4, no. 2, pp. 76–80, Jul 2000. [Online]. Available: https://doi.org/10.1007/s005000000042
- [22] N. V. Vitanov, "Synthesis of arbitrary su(3) transformations of atomic qutrits," *Phys. Rev. A*, vol. 85, p. 032331, Mar 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.85.032331
- [23] O. Oreshkov and T. A. Brun, "Weak measurements are universal," *Phys. Rev. Lett.*, vol. 95, p. 110409, Sep 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.95.110409

[24] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial

Intelligence. Cambridge, MA, USA: MIT Press, 1992.