Evolutionary Quantum Logic Synthesis of Boolean Reversible Logic Circuits Embedded in Ternary Quantum Space using Heuristics

Martin Lukac,* Marek Perkowski,† and Michitaka Kameyama‡

It has been experimentally proven that realizing universal quantum gates using higher-radices logic is practically and technologically possible. We developed a Parallel Genetic Algorithm that synthesizes Boolean reversible circuits realized with a variety of quantum gates on qudits with various radices. In order to allow synthesizing circuits of medium sizes in the higher radix quantum space we performed the experiments using a GPU accelerated Genetic Algorithm. Using the accelerated GA we compare heuristic improvements to the mutation process based on cost minimization, on the adaptive cost of the primitives and improvements due to Baldwinian vs. Lamarckian GA. We also describe various fitness function formulations that allowed for various realizations of well known universal Boolean reversible or quantum-probabilistic circuits.

I. INTRODUCTION

The nature of quantum particles and quantum environment allows quantum elementary particles to have in general more than two states [3, 6]. Depending on the quantity that is being measured, the particles can have two, three, high natural number or even an infinity of quantum states. Depending on the technology used, these particles can be atoms, molecules, photons or potentially even electrons could make suitable candidates. However so far, most of the Quantum Logic Synthesis methods implicitly assume that the underlying mechanism of quantum computing is quantum-binary [4, 10, 13, 25, 31, 40]. This assumption is correct for classical technology where the radix 2 of a logic is the easiest to implement; for instance in CMOS logic, all values of a given logic are constrained to a finite range of allowed voltage. Thus building a multi-valued CMOS logic requires to place the desired number of logic values on the same allowed voltage range. This ultimately makes such implementations impractical as well as technologically infeasible.

In Quantum mechanics, a similar limitation can be found. For instance, it is well known that a quantum system can have degenerate eigenstates; such states have the same value of a given measured quantity. However, in nature, many of the degrees of freedom of elementary particles will have more than two distinct and non degenerate states. The measured quantity can be energy, spin, position of a particle and so on. An example of an elementary particle with various degrees of freedom is a photon. It is currently used for various experiments and is also one of the most promising areas of quantum experimental computing [2, 7–9, 14, 15, 18, 34, 36, 37]. A photon has a large number of degrees of freedom; these are polarization, transverse spatial-mode, arrival-time, photon number, and frequency [15]. Other elementary particles offer large number of degrees of freedom with many states as well. For instance, trapped ions have electronic and vibrational energy levels with higher dimensions.

* lukacm@ecei.tohoku.ac.jp; Department of Information Sciences, Tohoku University, Sendai, Japan

† mperkows@ee.pdx.edu; Department of Electrical Engineering, Portland State University, Portland, OR, USA

‡ kameyama@ecei.tohoku.ac.jp ; Department of Information Sciences, Tohoku University, Sendai, Japan

Thus, assuming that at least k out of n possible states in a degree of freedom in an elementary particle can be attained at a reasonable cost of energy, it can be highly advantageous to use these higher-radix states for the synthesis of quantum logic circuits. Using this approach and using photons, a initial implementation of the Toffoli-sign (TS) gate using only three two-qubit quantum gates was experimentally demonstrated [15]. This work was later extended into realizing generalized Toffoli gates using various methods and approaches such as in [11]. The TS gate approach is similar to the one in [33] (page 183), but this time only with three CNOT gates and four single qubit rotation gates. Moreover, this approach scales better for larger number of control bits [15] (linear versus quadratic in the number of two-qubit quantum gates).

Evolutionary Algorithms allow to explore a very large problem space without knowledge of global structure of the problem space. In other words, an evolutionary algorithm can be used when the knowledge about the problem landscape is almost unknown and can be specified as a constraint satisfaction problem [5]. The required knowledge is in the local fitness function, and as such it means that one needs to know how a solution to the optimization problem is computed.

The Evolutionary Quantum Logic Synthesis (EQLS) has been explored from various points of view in the last decade. On one hand the Genetic Programming has been widely used to synthesize quantum circuits and logic functions [17, 28, 29, 39, 41, 42]. On the other hand GA based methods have been applied to various quantum circuits as well [19, 22, 23, 25, 44]. In general the focus of these approaches is either on a particular function (Reversible, quantum-permutative) or a general approach is used to see how well the evolutionary approach deals with this difficult problem. Several problems in EQLS have already been studied and analyzed, some of them are complexity of the quantum search space, high dimensionality of the quantum space, large number of quantum gates, etc. From these previous studies, it can be concluded that Evolutionary methods are well suited for research aimed to discover novel principles and novel quantum gate realizations of moderate size [19]. Following this reasoning, in this paper the focus is on the discovery and a deeper understanding of dynamics of the EQLS under various experimental conditions in the qudit framework.

In this paper we extend our initial work [24] of EQLS of Boolean reversible and quantum-probabilistic circuits on qu-

dits of higher dimensions ($d = 3$) using structural restrictions and heuristics. We provide a more detailed analysis of evolutionary mechanisms in the heterogeneous quantum logic synthesis and we compare results of the EQLS process with various improvements. The particular improvements to the Genetic Algorithm that we study are the following:

- Comparison of two types of fitness function
- Comparison between a Baldwinian and Lamarckian GA
- Comparison between normal and adaptively weighted primitive sets

The experiments are performed over a selected set of both reversible and quantum gates and discussion of the results is provided.
This paper is structured as follows. Section II describes the quantum circuits and the underlying quantum computing properties. Section III describes the GA used in this work. Section IV presents and discusses the experimental results and Section V concludes this paper.

II. QUANTUM GATES AND QUANTUM CIRCUITS

A. Boolean Quantum Gates

The quantum gates used in this paper are some of the well known Boolean Quantum gates. The used quantum gates can be separated into permutative and non-permutative quantum gates.

1. Permutative Quantum Gates

The permutative gates are also known as logically reversible quantum gates. Permutative gates in this work are the NOT (X) single qubit gate, CNOT and SWAP the two qubit permutative gates and Toffoli (also known as CCNOT), Fredkin (also known as CSWAP), Majority, Miller, Full Adder as the multi-qubit permutative quantum gates. The respective representation of the single and two qubit permutative quantum gates can be found in the Appendix A. Toffoli gate can be simply defined by eq. 1

$$\begin{aligned} CCNOT &= |0\rangle\langle 0| \otimes I^{\otimes^2} + |1\rangle\langle 1| \otimes CNOT \\ &= |00\rangle \otimes I + |01\rangle \otimes I + |10\rangle \otimes I + |11\rangle \otimes X \end{aligned} \tag{1}$$

Fredkin gate is given by eq. 2

$$CSWAP = |0\rangle\langle 0| \otimes I^{\otimes^2} + |1\rangle\langle 1| \otimes SWAP \tag{2}$$

Majority is defined by the logic equation

$$F = ab + bc + ac \tag{3}$$

and there are several implementations that do satisfy this logical requirement. Finally the Miller gate is given by eq. 4

$$Miller = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Finally, the full-adder is defined by two outputs as three inputs and its logic definition is shown in eq. 5.

$$\begin{aligned} S &= a \oplus b \oplus c \\ C &= majority(a, b, c) \end{aligned} \tag{5}$$

2. Non-Permutative Quantum Gates

The non permutative quantum gates used are the Z, Hadamard and their are shown in eq. 6 and eq. 7 respectively.

$$Z = (|0\rangle\langle 0| - |1\rangle\langle 1|) * I \tag{6}$$

$$H = \frac{1}{\sqrt{2}}[|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|] \tag{7}$$

For the multi-qubit non-permutative gates the Controlled-Z (CZ) or the Controlled-H (CH) gates are used as well. The CH gate is given by

$$CH = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes H \tag{8}$$

and the CZ gate is given by

$$CZ = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z \tag{9}$$

B. Ternary Quantum Gates

Circuits considered in this paper and in the original work of Lanyon [15] contain the gate denoted by $[0-2]$. This is a multi-valued permutation quantum gate. Its matrix is shown in equation 10(a). This quantum gates permutes the values of a single qutrit so that when the qutrit state is $|0\rangle$ it will result in $|2\rangle$ and if it is $|2\rangle$ it will become $|0\rangle$. Equation 10(b) shows a similar permutation gate $[1-2]$. This gate has similar functionality as the $[0-2]$ gate but swaps the states $|1\rangle$ and $|2\rangle$. Finally eq. 10(c) shows the $[0-1]$ permutative gate. These three gates are the only truly multi-valued quantum gates used in this work.

$$\begin{aligned} [0-2] &= |0\rangle\langle 2| + |1\rangle\langle 1| + |2\rangle\langle 0| \text{ (a)} \\ [1-2] &= |0\rangle\langle 0| + |1\rangle\langle 2| + |2\rangle\langle 1| \text{ (b)} \\ [0-1] &= |0\rangle\langle 1| + |1\rangle\langle 0| + |2\rangle\langle 2| \text{ (c)} \end{aligned} \tag{10}$$

C. Cost of Quantum Gates

On of the criterion that the designed circuits are evaluated on is their cost. In this paper we are assuming a very simple quantum cost model. This model has been used for the standard quantum Boolean logic synthesis in various previous works [23, 27, 31] and is only applied to single qubit and two qubit quantum gates. all other gates are built from these primitives. Thus, in this paper all two qubit/qutrit quantum gates have a cost of 1 and single qubit gates are ignored.

D. Boolean Quantum Circuits Embedded in Qutrit Quantum Space

Embedded Boolean quantum gates are different from multi-valued gates by the fact that as introduced in [15], these gates behave like regular Boolean gates for Boolean values of the input qutrit but for all other values of the qutrits they behave as identity operators.
For instance the Controlled-Z gate embedded in ternary quantum space is shown in eq. 11.

$$CZ^3 = (|0\rangle\langle 0| + |2\rangle\langle 2|) \otimes I_3 + |1\rangle\langle 1| \otimes Z^3 \qquad (11)$$

Observe that the CZ^3 gate changes the phase of the target qutrit only when the input state is $|11\rangle$; this is the same function as a classical Boolean Controlled-Z gate. On all other values of both the target and the control qutrit it must operate as an Identity operator [15].

An example of a non-permutative quantum gate embedded in a ternary quantum space is shown in eq. 12.

$$H^3 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (12)$$

In this article, the gates embedded in ternary quantum space are denoted by a an exponent of the embedding space; for instance a SWAP gate embedded in ternary space is denoted by SWAP^3 (eq. 13).

$$SWAP^3 = (|0\rangle\langle 0| + |2\rangle\langle 2|) \otimes I_3 + |1\rangle\langle 1| \otimes X \qquad (13)$$

1. *Toffoli Gate*

As shown in papers [15, 38] that are the prime motivation for this work, a Toffoli-Sign (TS) gate is such a gate that it changes the sign for only one particular input minterm (Figure 1(c)). The TS gate become a logical Toffoli gate when two additional Hadamard gates are placed before and after the TS gate on the target qubit (Figure 1(d)). In a similar manner a Toffoli gate different from the logical Toffoli function by a relative phase from a normal Toffoli gate can be constructed using the approach shown in Figure 1(a). In the Figure 1(a) the rotations around the *y* axis of the Bloch sphere use $\theta = \frac{\pi}{4} + n * \frac{\pi}{2}$ [33]. Finally, another and the most common realization of Toffoli gate - using the *Controlled-V/V*†is shown in Figure 1(b).

TABLE I. The K-map of the Toffoli-sign gate: (a) using the $[0-2]$ gate, (b) using the $[1-2]$.

c / ab	0	1
00	000	001
01	010	011
11	110	111
10	100	-101

(a)

c / ab	0	1
00	000	001
01	010	011
11	110	-111
10	100	101

(b)

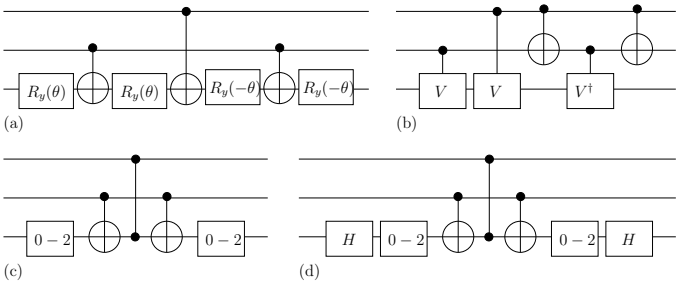

FIG. 1. (a) Toffoli gate with a relative phase difference [33], (b) Toffoli gate realized using the CV/CV† gates, (c) The Toffoli-Sign gate using multi-valued gates, (d) Toffoli gate constructed from Toffoli-Sign gate.

Note that the Toffoli gate from Figure 1(d) swaps the binary states $[101, 100]$. So it works differently from normal Toffoli gate that swap states $[110, 111]$. The Toffoli that swaps $[110, 111]$ states is realized using the same principle but by replacing the $[0-2]$ gate by $[1-2]$ ternary gates (eq. 10). In such case the realized gates swaps effectively the binary states $[110, 111]$ (Table I). The two-qubit gate in the middle of the circuit Figure 1(c) and (d) is a Controlled-Z (CZ) gate.
Finally observe that the Toffoli gate in Figure 1(d) works because the embedded Boolean quantum gates do not mix the ($|0\rangle$,$|1\rangle$) and the $|2\rangle$ quantum subspaces. In fact, it first separates the Boolean space of the target qutrit into two separate quantum spaces ($|0\rangle$, $|1\rangle$) and ($|2\rangle$) and then apply a single phase to one of the states in the ($|0\rangle$, $|1\rangle$) quantum space.
The Toffoli gates introduced here all work on generally different principles from one another: we say that such circuits are realized using *heterogeneous quantum gates* . With respect to classical Quantum Logic Synthesis we define the *heterogeneous quantum logic synthesis* as:

Definition II.1 (Heterogeneous Quantum Logic Synthesis)**.** *A process of designing Quantum circuits using quantum gates of a various radices, acting on both the phase space and on the observable space.*

Observe that unlike in classical Logic Synthesis, the heterogeneous quantum logic synthesis is a quite natural approach; no different technology and not distinctively different computational control protocol are required.

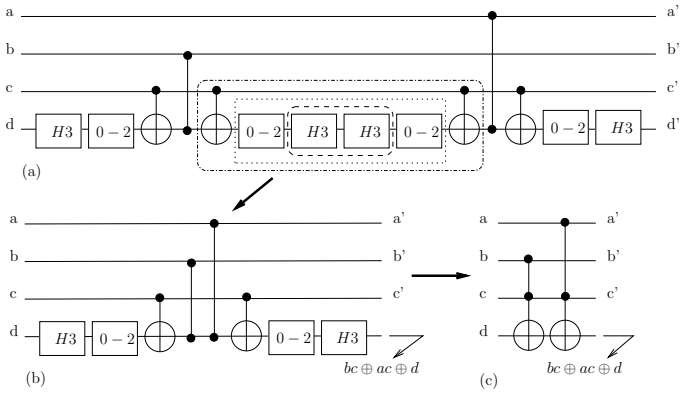

FIG. 2. (a) The realization of a Boolean function using Controlled-Z, controlled-NOT, Hadamard and $[0-2]$ multi-valued quantum gates, (b) the minimization of the number of quantum gates, (c) the Boolean function realized with two Toffoli gates.

As an illustration, the Figures 2 and 3 show the implementation of the $F_1 = bc \oplus ac \oplus d$ and of the $F_2 = (b \oplus a) \cdot c$ functions respectively. Observe that both circuits use the heterogeneous logic synthesis.

The circuit from Figure 2 shows how adjacent self-inverse gates are eliminated - the same process as in standard quantum Boolean logic. Similarly Figure 3 illustrates the gate concatenation based on the same principles as in qubit logic.

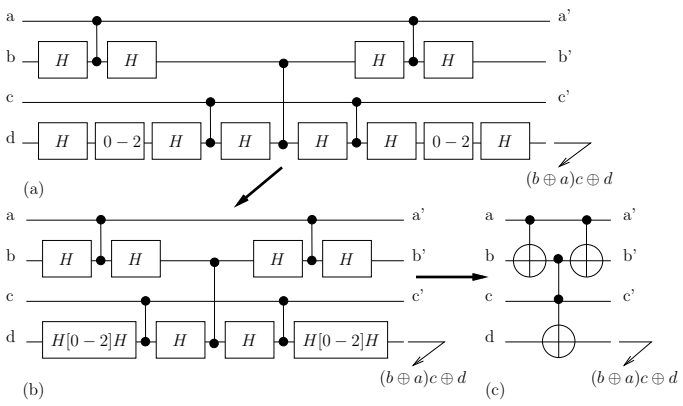

FIG. 3. (a)The realization of a Boolean function using only Controlled-Phase, Hadamard quantum-binary gates and the $[0-2]$ multi-valued quantum gate, (b) the minimized circuit, (c) the same Boolean function realized using only Toffoli and Controlled-NOT gates.

The CZ^3 gate can also be built as a mixture of qubit (control) and qutrit (target) and in such case its unitary matrix will be reduced from $3^3 * 3^3$ to $6 * 6$ rows-by-columns.

The boolean quantum logic synthesis uses two spaces: the phase space and the observable space. The difference between these two spaces can be seen in comparing the bases states. For instance, on one hand the common quantum bases states are $|0\rangle$ and $|1\rangle$. On the other hand more complex basis states are given by $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Observe that applying the NOT operator will change state $|0\rangle$ to $|1\rangle$ and vice versa but the state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ will change into $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ by using the Z quantum gate. This means that in the former set of basis states, the operator changes the value while in the later set of basis states the same operator changes only the phase.

III. GENETIC ALGORITHM FOR QUANTUM LOGIC SYNTHESIS

The GA that was used for the experiments is an extension of our previous work [22, 23]. It designs Boolean quantum circuits embedded in multi-valued quantum space by only evaluating the Boolean input-output either by measuring for the two observables bases $|0\rangle$ and $|1\rangle$ or by comparing the coefficients of the circuit matrix with the target circuit matrix. For clarity we discuss only the most relevant features of the evolutionary search. For a complete description of the used GA the reader should consult [19, 20, 22, 23].

A. The evolutionary process of computation

The process of using GA for EQLS is described in the pseudo code below:

```
0: Initialize the GA and the primitive set
1: Generate Initial population of circuit
3: Evaluate the circuits in the population
2: Until(solution is found
     or maximum generation is reached)
3:    select and replicate the circuits
4:    mutate the circuits
5:    evaluate the circuits
6:    replace old population by the new one
7:    Goto 2
```

B. Primitive Set

The GA builds circuits from a set of gates specified by the user called the *primitive set*. The size of this primitive set determines in a large extent whether yes or no the target circuit is found [19]. This is natural as without any restrictions, each additional gate that can be placed in a quantum circuit of a given size increases the possible combinations of the gates exponentially. The primitive set used in this work contains the following quantum gates:

1. Controlled-NOT (CNOT^3), Controlled-Phase (CZ^3), $[0-2]$, $[1-2]$, Controlled-Hadamard (CH^3), Controlled-$[0-2]$, Controlled-$[0-1]$, Hadamard^3 and Wire.

The GA uses all primitive gates as they are; the control qubit will be always on the wire above the target qubit if the primitive gate is given as such. The only operation that the GA does on the gates is the expansion so that a CNOT gate can be placed on qubits that are not direct neighbors.

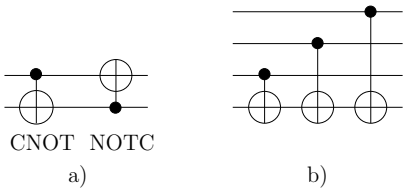

FIG. 4. Example of CNOT gates placement and uniqueness: a) The CNOT and the NOTC quantum gates, b)Example of CNOT gates requiring a unique encoding in the genotype string of a four qubit circuit

The reason to restrict the primitive sets not to include the X single qubit gate is the fact that the $[0-1]$ quantum gate is the X gate embedded in the ternary quantum space as well as because of the fact that $H * Z * H = X$. Finally, as it was shown in our previous experimental work in EQLS the smaller primitive gate set is the higher is the probability that the desired circuit can be found.

C. Circuits encoding

The quantum circuits are presented as strings, each gate is uniquely defined by a triplet of characters. The reason for representing each gate by a triplet of character is purely algorithmic; when designing circuits on many qubits, the number of possible gates grows up quickly and thus a single character encoding does not allow to represent all required gates. As will be seen during the description of the Lamarckian Genetic Algorithm (Section III E) it is required to store a large amount of gates that each require a unique character representation and a single character encoding (8 bits) allows to maximally represent 256 different quantum gates. For example a CNOT gate defined on two adjacent qubits and the same gate defined on two non adjacent qubits will have a different representation in the genotype string. Thus even with a finite set of input primitives, the number of unique representations grows with the width of the circuit. Figure 4(b) shows the different CNOT gates requiring different encoding in a circuit with four qubits.

The eq. 14 shows an example of a circuit represented as a string. For clarity, spaces have been added between each triplet of characters.

$$C = p\ \underline{!!!}\ \underline{\#!!}\ pp\ \underline{\$!!}\ pp\ \underline{"!!}\ \underline{!!!}\ \underline{!!!}\ pp \tag{14}$$

A quantum circuit can be separated into a set of serially connected blocks, each acting on the same number of qutrits; each of the block acts on the exactly same number of qutrits as the whole circuit acts on. In eq. 14 such blocks are delimited by a 'p' character from each side. Each such block is a sub-circuit acting on the same number of qutrits as the whole circuit acts on. The evolutionary algorithm operates on the circuit by either recombining the strings (exchanging the serial blocks) or by mutating elements within these blocks. According to the user-provided parameters, the mutation operator can be structure preserving (when one gate is replaced by another gate with the same number of inputs and outputs), not structure-preserving (a gate is changed into an arbitrary other gate) or a whole block can be erased and replaced by a new randomly generated block. In this experiment the mutation operator is not structure preserving but the probability of mutation is quite low. We use the Stochastic Universal Sampling replication mechanism [1] and the two-point crossover.

D. Structural Restrictions

In non-heterogeneous evolutionary quantum logic synthesis several parameters affect directly the success of the synthesis [19, 21, 22]. In particular a large number of input primitive gates reduces the chance to find the circuit [19]. This is particularly true when designing circuits with many quantum gates; circuits that require long sequences of gates - building blocks - are difficult to be maintained through the evolutionary process [5] and such sequences are generally lost when the crossover evolutionary operator is applied. This problem was experimentally shown as having a major impact on the EQLS performance [19, 20, 22].

In the heterogeneous quantum logic synthesis there is additionally a third space: the radix space. To evolve quantum circuits in a heterogeneous quantum space additional quantum gates are required. This means that the initial primitive set is enlarged by not only Boolean quantum gates embedded in a higher radix quantum space but by also possible multi-valued quantum gates.

However, as was introduced in Section II D 1 the heterogeneous quantum space allowed the design of a novel very low cost Toffoli gate and thus synthesizing circuits in this quantum space could allow the GA to find novel solutions to some of the well known quantum circuits.

In order to deal with the larger input primitive set required for the heterogeneous quantum synthesis a set of heuristics are introduced in this paper.

The first heuristic improvement introduced in this paper are the so-called *structural restrictions*. A structural restriction is a condition imposed on a quantum gate that allows this gate to be located or not to be located on a particular qudit of the quantum circuit. In the experimentations described later in this paper, some of the gates are allowed to be placed only on a specific wire in the quantum circuit. These restrictions are introduced on the observations made from the analysis of the circuits realized in the heterogeneous quantum logic synthesis [15, 24].

The Toffoli gate realized in this manner requires the multi-valued gates to be located only on the bottom (target) qubit. Using such knowledge about the Toffoli gate designed by Lanyon et al. [15], we can verify how fast the Toffoli gate will be designed by the GA without and with the structural restrictions. In this case the restrictions are on all single qubit gates but not on the Wire (Identity) single qubit gate. Thus for the primitive set, the single-qubit gates Hadamard and $[0-2]$ can be forced to be placed only on the bottom qudit (output qudit) of the quantum circuit.

To realize for instance a Fredkin gate one can use the Peres-Toffoli-Feynman (PTF) principle (Figure 5). This means that using two CNOT gates around a Toffoli gate will create a

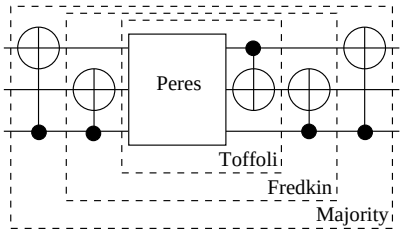

FIG. 5. The PTF principle

Fredkin gate. In the heterogeneous logic synthesis a possible solution is the multi-valued swap gate called SWAP[3]. This gate is shown in eq. 15.

$$S_3 = \begin{array}{c} \\ 00 \\ 01 \\ 02 \\ 10 \\ 11 \\ 12 \\ 20 \\ 21 \\ 22 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 00 & 01 & 02 & 10 & 11 & 12 & 20 & 21 & 22 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \underline{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{1} & 0 \\ 0 & 0 & \underline{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{1} \end{pmatrix} \end{array} \quad (15)$$

Using the gate from eq. 15 as a part of a controlled gate the Fredkin realization could potentially be cheaper than in binary quantum. Figure 6 shows one possible realization of the Fredkin gate; Figure 6(a) shows the Fredkin gate designed using the PTF principle (with Controlled-Phase gates, Hadamard and the ternary $[0-2]$) and Figure 6(b) shows the Fredkin gate after gate minimization.

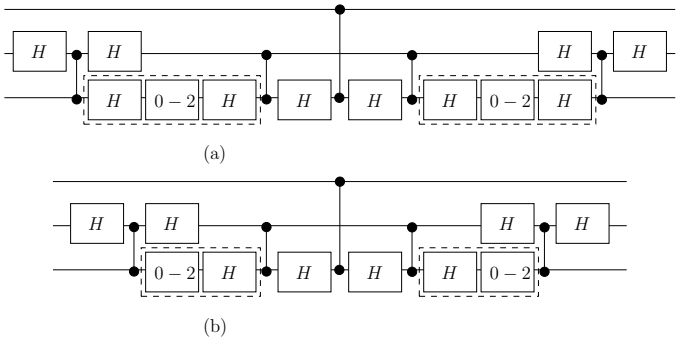

FIG. 6. (a) Fredkin gate realized using the PTF principle, (b) CNOTC gate realized using Toffoli-sign and two NOTC gates.

It is also an open question if the multi-valued swap gate S_3 (eq. 15) can be constructed and what is its cost. So far using the approach studied in this article, it seems that because the CNOT gate (Figure 7) can be realized using only the Controlled-Z and Hadamard gate, the PTF principle seems to be the most natural. The search for novel realization of Feynman, Toffoli or Fredkin gates is one of the challenges of this paper.

The multi-valued swap gate S_3 can also be realized using the approach proposed by Muthukrishan and Stroud [12, 32]. Thus a Fredkin gate can be also realized as shown in Figure 8.

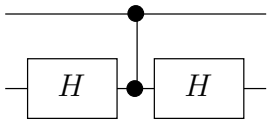

FIG. 7. CNOT gate realized using a CZ and two Hadamard gates.

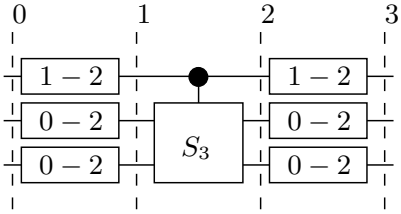

FIG. 8. Fredkin realized using a controlled multi-valued S_3 gate.

This can be shown by analyzing the input-output mapping of the Fredkin gate realization. This Fredkin gate inputs binary inputs and generates binary outputs. The Controlled-S_3 gate is activated only when the top qubit is in state 2; this is obtained by applying a rotation $[1-2]$ to the control qubit. The bottom two target qubits are first projected into the $|1\rangle$, $|2\rangle$ quantum space and then are swapped. To obtain back the binary values the bottom qubits are transformed using again the single $[0-2]$ rotation gates.

Observe that this realization uses only multi-valued quantum gates; the Controlled-S_3 gate is only active when the control qubit is in state $|2\rangle$. Because the input data is restricted to binary values, the usage of the Controlled-S_3 gate is restricted to four input values only (Underlined coefficients in the matrix from eq. 15). This means that the gate Controlled-S_3 gate is designed to provide correct input-output mapping only when the control qubit is in the $|2\rangle$ state and the target qubits are in the quantum space spanned by $|1\rangle$ and$|2\rangle$. When the control qubit is in states $|0\rangle$ or $|1\rangle$ the Controlled-S_3 gate is not active and thus acts as an Identity gate but only on the target qubits being in the $|1\rangle$ and$|2\rangle$ quantum states.

Thus the design of such gate is much simplified. The only required input-output mappings in the S_3 are now much easier to design as they represent only $\frac{2}{3}$ of the diagonal elements of the whole gate; this is essentially the same principle as simply embedding boolean quantum space to a ternary quantum space. As the input qubits will never be in the $|1\rangle$ for the control qubit and $|0\rangle$ for the target qubits the designed quantum gate is not required to be identity for these values from the logical point of view. Thus the gate S_3 is required to be only functionally corresponding to a swap gate only for the qubit values that will be available at its inputs or outputs.In this case these states are $|21\rangle$, $|12\rangle$, $|11\rangle$ and $|22\rangle$.

Naturally, this representation of Fredkin gate is equivalent to a Controlled-SWAP gate realized in higher-radix logic as long as it swaps two distinct values exclusively. Thus the circuit from Figure 8 could be simply realized by a

Controlled-Swap gate in a higher-radix logic assuming it is defined for the $\{100, 101, 110, 111\}$ as a SWAP gate and for $\{000, 001, 010, 011\}$ as the Identity gate. Such gate is a realization of quantum multiplexer in logic of higher dimension (equation 16). The don't care sign "-" represents the fact that the control qubit will never be in the state $|1\rangle$ and thus from the logic point of view it is not important what sub-matrix is defined for control qubit in state $|1\rangle$.

$$CS_{12} = \begin{pmatrix} \begin{pmatrix} I \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} - \end{pmatrix} & 0 \\ 0 & 0 & \begin{pmatrix} SW_{12} \end{pmatrix} \end{pmatrix} \tag{16}$$

Observe that all the gates discussed above and below are unitary because of the following synthesis restrictions imposed:

- All gates are designed using primitive unitary single and two qubit/qutrit quantum gates.
- All primitive quantum gates embedded are also unitary as required by the initial condition imposed in [15]
- Gates not strictly obeying the unitary requirement are such gates that are unitary on all the available input-outputs mapping. This naturally means that the unused part of the quantum space (as the sub-matrix denoted by "-" in Figure 8) must also be unitary.

In a similar manner it can be expected that the majority gate can be realized using the heterogeneous EQLS. The majority gate has the advantage that it requires only one output; the reversible generalization of majority gate was proposed by [30] and is called the Miller gate. The Miller gate can be directly obtained using the PTF principle by surrounding a quantum realized Fredkin gate by two more CNOT gates. However, in this paper we are interested in a novel implementation of the majority/Miller gate using the multi-valued quantum gates. The majority gate outputs a binary value "1" when the number of ones in the input is larger than the number of zeros. The output of the Majority gate is also given using eq. 3.
The goal of designing the majority gate is to verify whether the structural restriction impact will be observable on a single-qubit output reversible function. In the case of the Fredkin gate, we are not aware about any direct solution using the quantum heterogeneous logic synthesis; the only known method of designing the Fredkin gate is the PTF principle that uses however only quantum Boolean gates. Thus the experiments will be conducted also over sets of restrictions in order to verify as many potential solutions as possible.

E. Baldwinian vs. Lamarckian Evolutionary Learning

Before explaining the differences between Baldwinian and Lamarckian implementation of the EQLS we present some required basic concepts.

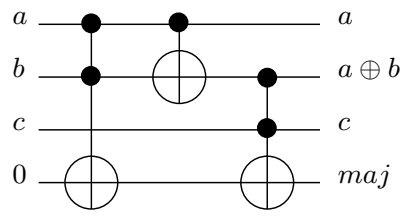

FIG. 9. The majority gate realized using only MCT gates: $ab \oplus ac \oplus bc = ab \oplus (a \oplus b)c$.

1. Cost of Quantum Circuits

One of the well known problems in Quantum Logic synthesis is how to find the quantum circuits with the minimal cost [16, 23, 27]. The cost of an arbitrary quantum circuit is in general calculated from the number of quantum gates it contains. The procedure starts with assigning costs to the primitive gates and then for a given function the cost is the sum of costs of all gates. Let $g = \{g_0, \ldots, g_m\}$ be the set of all available quantum gates and $c = \{c_0, \ldots, c_f\}$ be the set of all possible costs of the quantum gates in g. Also let a function $r(g_p) = c_j$ be a mapping that takes a gate from g as argument and returns the appropriate quantum cost from c. Then for any QC with k gates one can write $r(g^n) = c_j$ where $g_n \in g$ is the n^{th} quantum gate from QC that corresponds to a unique gate in g with the corresponding quantum cost c_j. Using this principle the cost of a QC is given by eq. 17.

$$C_{QC} = \sum_{j=1}^{k} r(g^j) = \sum_{j=1}^{k} c^j \tag{17}$$

with c^j being the cost of the j^{th} quantum gate in the circuit. On the lowest granularity level of quantum logic synthesis, the cost is given by the number of single-qubit and two-qubit electro-magnetic (control) pulses. On this level the cost of well-known in terms of numbers of electromagnetic pulses have been computed in [16]. Designing quantum circuits using control pulses is however quite difficult because even for small quantum gates such as Peres, Toffoli, Fredkin or Miller the number of control pulses reaches 12, 13, 19 and 24 respectively [16]. Thus the design of a slightly larger quantum circuits will contain hundreds of the control pulses. From our previous experience with evolutionary quantum logic synthesis (EQLS), circuits with so many gates are difficult to synthesize because during the evolutionary process the building blocks are too large [19, 20, 22]. Thus, evolutionary synthesis should be not done on the level of control pulses.
In the most popular approach, the quantum cost is computed on the level of quantum logic gates; each logic gate such as Peres or Toffoli quantum gate is assigned a single cost and a gate is used as a macro in the synthesis process. Once a circuit is designed using these high level templates it is transformed to quantum counter parts - in general using the CNOT, CV and CV† quantum gates and the final cost is computed after some local optimization on the level of control pulses or quantum primitives [20, 23, 27, 31]. However this approach

is not guaranteed to produce minimal results on the level of quantum gates because it is not known if such transformations will always give the minimal quantum cost. For instance, using Multi-Controlled Toffoli (MCT) gates there are several high quality synthesizers [26, 43]; however the synthesis using MCT gates does not guarantee an optimal (minimal) quantum cost.
Thus, with respect to the cost of quantum circuit the following dichotomy appears: the problem of Quantum Logic Synthesis is to find the most appropriate method that will avoid the high complexity space (control pulses) and provide the best level of minimization.

2. Correctness of Quantum circuit

Definition III.1 (Correctness of Quantum Circuit)**.** *A quantum circuit is 100% correct if all the logical elements of the circuit correspond to the desired target definition.*

For instance 80% means that the given gate is correct in 80% of the input-output mapping. The mappings used to evaluate the correctness of the quantum circuit can be either measured observables or the elements of the unitary matrix [19]

From logic point of view one can determine the correctness of a quantum circuit from three different perspectives:

- evaluating quantum circuit only on certain output qubits: this can lead to improved quantum cost but for the price that certain qubits remains in superposed quantum state [19]
- evaluation of circuit correctness based only on a set of logic values, is equivalent to evaluation of the logical output after the measurement
- evaluation of the circuit correctness based on matrix coefficient is the most exact method of evaluation but is not practically realizable as it requires to know the exact values of the matrix representing the unitary transform.

In this work the correctness is evaluated on the logical values basis; the phase information of logic values is not taken into the account.

3. Input gate set reduction

In EQLS the problem of synthesizing circuits with the smallest possible cost is difficult because when designing circuits on the level of control pulses the primitive set is too large. This is equivalent to the statement that there are to many degrees of freedom in the selection of the primitive during the process of mutation. However when designing circuits with a small number of gates, the GA is outperformed by non-evolutionary approaches that use reversible logic synthesis design methods such as MMD algorithm [26] (bearing name after the authors initials) or Boolean Decision Diagram based [43]. However these approaches use well known and optimized techniques and thus are not suitable for the synthesis of arbitrary quantum circuits and do not allow discovering novel realizations of well known quantum or reversible functions.
The approach proposed here is aimed to improve the ability of the GA to converge faster when designing quantum circuits with larger numbers of primitive gates. The process is illustrated in Figure 10. The idea behind this adaptive process is

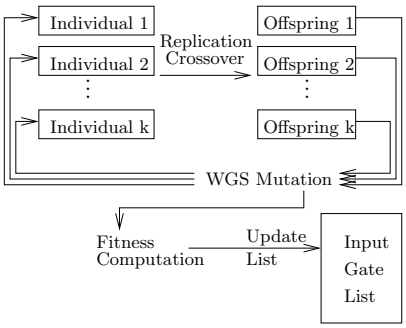

FIG. 10. The process of ranking primitive gates in order to allow faster GA convergence

the following: take the gates from the primitive set and assign to each of them a *usage* and a *fitness* values. *Usage* represents how often the given gate is used while the *fitness* represents the average value of combined fitness of all circuits this gate has been used in. Thus each time a gate is used both the *usage* and the *fitness* value are updated and the gate is re-ranked within the pool of the primitive gates. This dynamically growing and ranked primitive gate set is referred to as the *extended primitive gate set* (EIGS). Thus the gates that are created as replacement during the mutation process are now being selected using the *Weighted Gate Set (WGS) mutation* operator.
To implement the WGS mutation operator a variable *gfitness* is calculated and holds the sum of all gate-fitness values. The WGS mutation operator randomly selects a quantum gate from the circuit to be changed and then selects a new gate that matches both the structural restrictions as well as $r * gfitness$ value where $0 \leq r \leq 1$ is a randomly generated number. This approach allows to at least partially bias the selection of the replacement gates with three selection possibilities: trying new gates, trying good gates or trying any available gates from the primitive set.

4. Baldwinian and Lamarckian Circuit evaluation

The Lamarckian EQLS is based on the well known minimization of the quantum gates. The general rule is given by:

Definition III.2. *Two adjacent quantum gates can be combined into a single quantum gate if they are acting on the same qubits.*

For instance the well known realization of the two-qubit SWAP gate is based on three adjacent CNOT gates (Figure 11). This SWAP gate can be used as an optimized unit on the level of control pulses. Using this approach many quantum circuits can be minimized. Moreover and more important

for EQLS is the fact that the merging of gates on one hand reduces the total number of quantum gates in the quantum circuit and on the other hand increases the number of quantum gates in the primitive gate set. This process of gate merging is also shown in Figures 2 and 3.

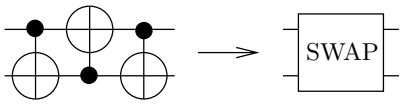

FIG. 11. Merging gates on the same quantum wires creates new larger gates that have been or can be optimized once for all on the control pulse level [16].

This means that when starting form an initial set of quantum gates soon a very large set of quantum gates is available for the mutation operator. Thus using the EIGS and the WGS mutation operator it is possible to partially control which gates are reused and which can be omitted. The combination of EIGS and of the WGS mutation operator is the base of the Lamarckian and Baldwinian algorithms.

The difference between the two approaches (Baldwinian and Lamarckian GA) is in the fact that once a circuit is constructed, it is minimized using the above rules and in the case of the Lamarckian EQLS the new genotype replaces the old one. In the case of the Baldwinian EQLS only the cost and the fitness calculation is done using the optimized circuit representation but the genotype remains unchanged. Thus, in the Lamarckian GA, the actual genotype of the individual is rewritten by the minimized string and new gates are actively created and inserted into the primitive gate set (Row 5 in the Table II).

One of the possible problems when using the proposed Lamarckian EQLS is that if it is not bounded by some hard imposed limits the EIGS would explode in the computer memory. This problem would make the GA practically impossible to run as it would run out of physical memory within few initial generations. The other problem is that, even if we impose a hard limit to the extended set of gates, it is hard to replace the least fit gates: the gates that have been least used and have the lowest gfitness value in this run. This means, that one could imagine an approach where with a limited set of extended quantum gates, the least fit of them are replaced by the newly generated. The problem with this approach is that it would require that all the removed gates are also removed from the population. The alternative approach to this was to impose a hard limit on the number of quantum gates and reduce the number of iterations during the GA run and average the extended set of gates over multiple runs that ended in the same local or global maximum. This allowed us to evaluate the overall convergence of the extended gate set. In the presented experiments the EIGS size was limited to 50 quantum gates.

Table II sums up the main differences between the Baldwinian, the Lamarckian and a classical EQLS. The classical GA does not use any optimization for calculating the cost of the quantum circuits. The Standard key word refers to evolutionary operators that operates on well-known principles in the GA. The Ranked and dynamic primitive gate set means that the EQLS uses the EIGS: the EIGS is growing with novel gates being created and inserted into it, and is dynamically updated using the fitness and usage of the gates in real time. Initial primitive gate set means that the primitive gate set selected by the user does not change during the whole evolutionary process. Observe that for both Baldwinian and Lamarckian GAs the computed fitness values are calculated from the optimized genotype.

TABLE II. Comparisons of differences between Baldwinian and Lamarckian EQLS

	Lamarckian	Baldwinian	Classical
Mutation	WGS	Standard	Standard
Crossover	Standard	Standard	Standard
Input gate set	Ranked, dynamic	Initial	Initial
Genotype	Replaced	Preserved	Preserved
Phenotype	Optimized	Optimized	Standard

F. GPU acceleration

As already described, in this work, all qubit operators are embedded in ternary operators in such manner that for one value of the qudit they act like the Identity. Similarly the circuits evolved are in principle ternary but only subsets of desired input and output values are used for the actual computation and evaluation processes.

Thus the overhead of computation can be quite large. This computational overhead is the main motivation to use a dedicated GPU computational processor for the evaluation of quantum circuits in the EQLS. The GA uses the GPU based matrix computation in order to accelerate the overall computation by taking advantage of the parallelism in the CUDA [35] libraries. The GPU acceleration is used to compute the matrix of the circuit. In general the GPU comes with multiple cores and each core is organized into a two-dimensional (can be virtually organized into a three-dimensional) grid of process threads. In order to implement an optimal matrix multiplication several conditions must be satisfied. Most important is that the grid of the computational processes should be allocated optimally. It means that the matrix sizes should be multiples of the number of the parallel computational processes.

The second constraint of similar importance is that in order to execute GPU computations the data must be transported from the main memory to the GPU memory using the data bus. As this bus is used also for other devices on the computer, it is possible to saturate the bandwidth and thus actually slow down the computation.

The GA used in this work is configured to compute ternary quantum matrices. Such matrices cannot be computed in the most optimal approach in a GPU device that is in general designed to have 2^k computational threads. However even in such case this computation is more efficient than on the CPU as it allows to unload the CPU to perform other tasks in parallel. Also to save the data bus bandwidth the computation of

the circuit matrix representation is done in two steps:

1. Compute the matrix of the circuit by sending all required parallel blocks (matrices) one by one to the GPU
2. Return the computed resulting matrix.

IV. EXPERIMENTS AND RESULTS DISCUSSION

The reversible quantum gates that we attempted to synthesize are the Toffoli gate(Toffoli-Sign), the Fredkin gate, the majority gate, the Miller gate, the $SWAP_3$ gate and the full adder. All gates can be synthesized using the GA when using only the Boolean quantum gates and thus present good benchmarks to evaluate the performance of our Genetic Algorithm. The reason for selecting this set of gates is that they represent a very basic standard of well known and used universal reversible gates.
The experiments are evaluated for up to 10000 generations; this number was previously experimentally determined as to be sufficient to observe if a given gate can be found in a statistically significant manner [19]. All resulting data are represented as average over 50 runs. The default fitness function used is the f_1 fitness function from eq. 18 unless specified explicitly.
Two fitness functions have been used for the experimentation. The f_0 function is a simple fitness function given by eq. 18:

$$f_0 = \frac{1}{1 + error} \tag{18}$$

where $error$ being given by the square difference between the desired and the target circuit output. A more elaborate fitness function that takes into account the cost of the quantum circuit is shown in eq. 19:

$$f_1 = \alpha \frac{1}{1 + error} + \beta \frac{1}{cost} \tag{19}$$

with α and β are constants obeying unity given by $\alpha + \beta = 1$ and $cost$ being the cost of the circuit in the range $[1, \inf]$. The values of the coefficients α and β are experimentally determined and are not the subject of study here. However as can be expected if the influence of the cost is too high the synthesized circuits will be of a smaller cost but will rarely represent the correct function.
The first set of experiments is aimed to show the difference between the usage of the restrictions. For this certain gates from the primitive set have been restricted to only certain wires. These gates are the $[0-2]$, $[1-2]$ and the H^3 and in the restrictions allowed them to be placed only on a single qubit at a time. The results of both runs are reported and compared in Table III. Our algorithm was able to find solutions to most of the benchmark functions in the allocated time and space. Each row in Table III shows the benchmark function name in the first column, and the number of generations if a solution was found for GA run with the structural restrictions (column 2) and without restrictions (column 3). The column entitled "Corr.", corresponds to average correctness of the target gate evaluated using the error described in Section III E 2. For instance the Toffoli gate was generated in 500 generation in average when using the structural restrictions and with the average correctness of 95% (in 5% of the cases the correct gate was not found and the algorithm stopped because reached the maximum allowed generations).

TABLE III. Overview of the results of the synthesis problem

Gate Name	Number of Generations w. Restrictions	wo. Resctrictions	Corr.
Toffoli	500	2500	100%
Majority	500	2000	97%
Fredkin	1500	-	95%
$SWAP_3$	500	1000	98%
Miller	10000	-	90%

Observe that the structural restrictions allow to design the given circuit considerably faster in the case where the solution was found. In the cases where the algorithm did not find any solution, the number in the third column indicates the percentage of correctness of the given gate.
The application of structural restrictions proved to be useful as it allows to directly reduce the time required to find solutions to our benchmarks. Naturally as can be expected, one needs prior knowledge about at least the principles of the desired quantum gates in order to be able to produce a set of restrictions that speed up the evolution. If a wrong set of restrictions is applied then the evolution can both take longer or will not find the target gate at all.
For instance. using the principle proposed in [15] one could restrict the usage of single qudit quantum rotation gates (depending on the number of control qudits [15]) to only the wires where the actual output function is created. This however works only for the MCT gates. All MCT gates have only a single functional output while all other inputs are just passing through and are unchanged on the output. However, the same restrictions will not be successful on a permutative non MCT gate as for instance is the Miller gate; for this gate the restrictions should be less strict as the circuit realizes majority function on each of the output qubits. In such case the gates should not be using structural restrictions.

TABLE IV. Comparison of results when using the Lamarckian GA with the primitive gate set ranking side by side with the Baldwinian GA and the standard not improved GA.

Gate Name	Classical (Gen./Corr.)	Lamarckian and WGS (Gen/Corr.)	Baldwinian (Gen/Corr.)
Toffoli	1500/95%	900/100%	1600/95%
Majority	1000/95%	750/100%	1200/95%
Fredkin	-	1700/100%	2000/96%
Full Adder	120/97%	250/100%	180/98%

The next set of benchmarks illustrates the convergence changes as a consequence of using the Lamarckian and the Baldwinian learning. Table IV shows the comparison of the

three main types of GA tested: the standard (column 2) the Lamarckian (column 3) and the Baldwinian (column 4). Each of the three columns (2, 3 and 4) shows the average number of generations until the target gate was obtained and the average correctness of the obtained result. Observe that using the Lamarckian learning significantly boosted the evolutionary process.

It is particularly interesting to notice that in fact the Baldwinian GA does not really improve the performance of the GA. The reason for this is that because the actual size of the circuits is not modified. Thus during mutation and recombination, same restrictions applies to the Baldwinian GA as to the standard one. The fitness and the cost of the minimized circuit has no effect on the genetic operators and thus as the main constraint is the size of the circuit with the right combination of quantum gates, the Baldwinian GA is not more successful. Table V shows the comparison between the Lamarckian and the Baldwinian evolution. The Lamarckian GA is evaluated with and without using adaptive weighted primitive gate set. Observe that as expected the performance of such Lamarckian GA without the use of the EIGS is much worse than any other GA configuration used.

TABLE V. Comparison of results when using the Lamarckian GA with and without the primitive gate set ranking side by side with the Baldwinian GA

Gate Name	Lamarckian (Gen./Corr.)	Lamarckian and WGS (Gen./Corr.)	Baldwinian (Gen./Corr.)
Toffoli	2100/80%	900/100%	1600/95%
Majority	2000/70%	750/100%	1200/98%
Fredkin	2500/85%	1700/100%	2000/95%
Full Adder	420/97%	250/100%	180/98%

Moreover, the performance of the Lamarckian GA without the use of the EIGS is actually much worse than the performance of the standard non-improved GA. This can be seen when comparing the second column of Table V with the first column of Table IV.

The above benchmarks all used cost related fitness functions from eq. 18. To observe how the fitness function affects the convergence, the following experiments are focused on the correctness of the synthesized quantum circuit (error) and its cost. The Table VI shows how the two main classes of fitness functions affect the convergence of the algorithm. The first column of the Table VI shows the name of the synthesized function. The second column in Table VI illustrates the performance of the GA using the fitness function given by the eq. 18. The third column in Table VI shows the same data but using the fitness function from eq. 19.

Columns two and three in Table VI show the average cost and the average correctness of the target circuit.

For instance the Toffoli gate row shows that using the fitness function f_0 (second column) the average cost of the Toffoli gate was 8 and the average correctness of the synthesized Toffoli gate is 95%. This means that on the 50 runs of the GA in 2 runs the obtained gate was not the exact match to the desired Toffoli gate.

TABLE VI. Comparison of results when using the normal fitness function vs. using fitness function with cost as a component of the fitness value.

Gate Name	Fitness f_0 (Cost/Corr.)	Fitness f_1 (Cost/Corr.)
Toffoli	8/95%	7/87%
Majority	10/95%	9/90%
Fredkin	15/40%	12/60%
Full Adder	10/97%	15/98%

TABLE VII. Comparisons of the rate of success of the GA under the various settings with the average cost of solutions found. The numbers represent averaged percentage of the successful target gate as well as the cost averaged over 20 GA runs.

Gate Name	Standard Succ/Cost	Lamarckian and WGS Succ/Cost	Baldwinian Succ/Cost
Toffoli	98%/20	100%/18	100%/22
Majority	90%/40	100%/25	100%/29
Fredkin	-	50%/25	20%/30
Full Adder	90%/35	100%/28	93%/35

The last set of experiments is intended to demonstrate the improvements given by the Lamarckian GA; the Table VII shows the percentage of the GA success for the tested gate benchmarks. Observe that the Lamarckian GA using the WGS mutation allows to synthesize the circuits with a higher rate of success as well as with a lower cost. The three data columns in Table VII represent the average success of the synthesis (Succ) of the target gate and the average cost (Cost).

As already partially described, when using Lamarckian learning, the problem is that the primitive gate set is actively modified. Because of this it is required to provide additional information about the newly created quantum gates otherwise it will be very difficult for the GA to find the correct circuit. This is because as shown in [19, 23] a large primitive set means that the appropriate set of gates to be used in the target circuit is harder to select as the probabilities of selection are distributed over a larger primitive set. Thus, in our case we add the gate-fitness that allows to implement a ranking of the quantum gates. This means, that as long as a list of available quantum gates can be maintained the GS mutation operator will allow to select a set of gates that have a higher probability to be present in the target final gate.

It is also important to notice that the ranking of the gates depends on the initial population. Thus On one hand, a set of generated gates during one run will end up having a different ordering than when another run is performed. This is natural as which gates are preserved in the memory depends on the initial state of the population. On the other hand, for the same desired quantum gate to be synthesized, statistically the same gates will end up being included in the extended set of quantum gates. This is because, as the evolutionary process advances, the local maximum of fitness also partially determines what gates are used in the circuit.

Figure 12 shows gates from a single run for the Fredkin quan-

tum gate. Observe that the difference between the gates on the top of the list and on the bottom of the list is relatively small. The peak at gate ID 12 represents the identity gate.

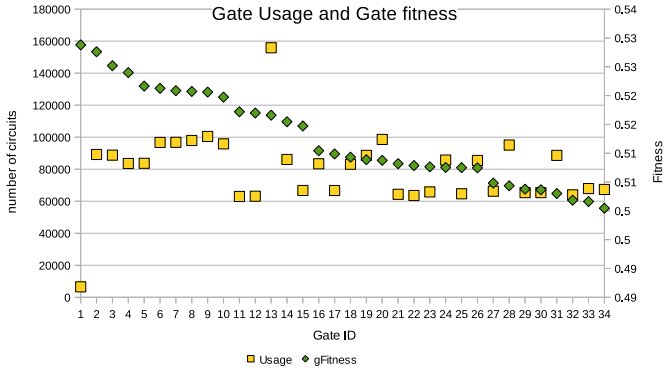

FIG. 12. The final result of the classification of the extended gate set. Usage represents the number of times a given gate has been used in a circuit, and *gfitness* is the average of the fitnesses of all circuits it has been part of.

Unfortunately it is not practically possible to compare gate lists from various runs because the gates that are placed into the EIGS can have similar string encoding in different evolutionary runs but do not necessarily have the same representing unitary matrix. Thus a comparative representation would be meaningless. However, the graph from Figure 12 can be used to set up preferential parameters for further investigations.
The most important result from this search is the novel implementation of the practical Fredkin gate. The circuit is shown in Figure 13. Despite the fact that this gate is realized using on qubit operators, it was discovered by the algorithm while searching for Fredkin gate with elementary gates used for the EQLS in the ternary quantum space. In other words, this gate does not use any properties of the qutrit quantum space to generate the qubit outputs.

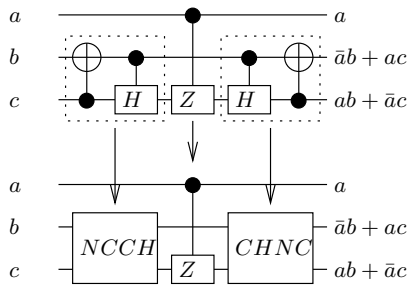

FIG. 13. The discovered Fredkin quantum gate and its minimization. NCCH stands for NOT-Controlled Controlled-Hadamard and CHNC is Controlled-H NOT-Controlled.

The Fredkin gate from Figure 13 is in fact the cheapest realization so far known in the quantum logic synthesis. It is particular because it heavily relies on the phase kick-back from both the controlled-Z and from the controlled-Hadamard gate but does not use the principles of the heterogeneous quantum logic synthesis. Also, when minimized, using the concatenation rule from def. III.2, this gate has the same cost as the Toffoli gate! This is due to the fact that the circuit given by the three middle gates is a Toffoli gate, but unlike in the methodology using the CV/CV†/CNOT gates, it does not require additional CNOT gate to return the control inputs to the initial values (Figure 1b). Thus, it can be minimized to a smaller cost.
Similarly the Majority gate was designed by the GA using the controlled-H and the controlled-Z gate. It is shown in figure 14.

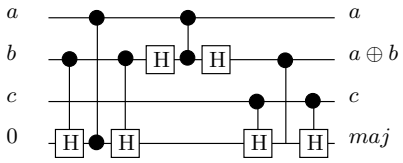

FIG. 14. The majority quantum gate designed using the Controlled-H and the Controlled-Z gates.

Gate name	Best Boolean Cost	Best Calculated	Best Obtained
Toffoli	5	3	3
Fredkin	7	5	5
Miller	9	7	8
Majority	12	8	10
Full-adder	12	12	12

Finally, to show the differences between the cost of quantum circuits designed using heterogeneous quantum logic synthesis the Table IV shows the best different coasts of the various quantum circuits. The second column of Table IV shows the best known quantum coast when quantum boolean gates only are used. The third quantum column shows the equivalent of the best quantum boolean realization when re-synthesized using the heterogeneous quantum synthesis. The fourth column shows the best cost obtained using the experimentation described in this paper.

V. CONCLUSION

First, in this paper we experimentally approached the heterogeneous QLS using an Evolutionary Algorithm with a set of structural restrictions and Lamarckian GA. Both of the proposed improvements to standard GA were beneficial on the benchmark quantum circuits that we were able to synthesize under the prescribed experimental conditions. As such, the Lamarckian GA and the structural restrictions are a quite successful improvement to the EQLS. In the described experiments only circuits of relatively small size have been explored. However even in these benchmarks the improvement is observable and thus similar improvement of performance can be expected for large circuits composed from basic gates designed here.
Second, we designed new set of quantum gates automatically: Toffoli (realization previously known), Fredkin (real-

ization previously unknown), Adder (realization previously unknown) and Majority (realization previously unknown). The obtained results validate the proposed approach not only as a theoretical study but also shows that GA is very well situated for exploration of unknown problems in the QLS area. The future work on this approach is to extended it to a truly hybrid heterogeneous QLS, where qubits, qudits and qutrits are truly mixed in a quantum circuit and its software representation. This has the advantage that the dimension of the circuit is significantly reduced; linearly in the size of the matrices. For instance a circuit with three qubits is represented by a $3^2 \otimes 3^2 \otimes 3^2 = 27^2$ complex coefficients. Assuming that two qudits in such circuit use only $|0\rangle$ and $|1\rangle$ states, a circuit with two qubits and one qutrit has its matrix of size only $2^2 \otimes 2^2 \otimes 3^2 = 12^2$. Moreover, the mixing of qudits of different radices allows a faster selection of operators that can be applied and thus subsequently allows to create more and more sophisticated *structural restrictions*, the concept proposed in this article. The possible disadvantage is that the multiplication of operators for qudits of various radices creates matrices that might be not well suited to be computed quickly using the CUDA approach. The possible improvement of the GPU approach presented is to move all operators (matrices) to the GPU memory, and perform all matrix computations as well as error evaluations directly on the GPU. This means that during an evolutionary run there will be almost no information transmitted between the memory/CPU and GPU but short strings or scalar values.

Appendix A: Permutative Quantum Gates

The NOT (also called X) gate is shown in eq. A1

$$X = |0\rangle\langle 1| + \langle 1||0\rangle \tag{A1}$$

The CNOT gate is shown in eq. A2

$$X = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X \tag{A2}$$

[1] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *In Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pages 14–21, 1987.

[2] Daniel E. Browne and Terry Rudolph. Resource-efficient linear optical quantum computation. *Phys. Rev. Lett.*, 95(1):010501, Jun 2005.

[3] P. A. M. Dirac. *The principles of quantum mechanics*. Clarendon, Oxford, 1984.

[4] G.W Dueck and D. Maslov. Garbage in reversible designs of multiple-output functions. In *Proceedings of RM 2003*, pages 162–170, 2003.

[5] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[6] R. Feynman. *Feynman Lectures on Computation*. Addison Wesley, 1996.

[7] Jaromír Fiurášek. Linear-optics quantum toffoli and fredkin gates. *Physical Review A*, 73:062313, 2006.

[8] Jaromír Fiurášek. Linear optical fredkin gate based on partial-swap gate. *Phys. Rev. A*, 78(3):032317, Sep 2008.

[9] Yan-Xiao Gong, Guang-Can Guo, and Timothy C. Ralph. Methods for a linear optical quantum fredkin gate. *Phys. Rev. A*, 78(1):012305, Jul 2008.

[10] W.N.N. Hung, X. Song, G. Yang, J. Yang, and M Perkowski. Quantum logic synthesis by symbolic reachability analysis. In *Proceedings of DAC*, 2004.

[11] Radu Ionicioiu, Timothy P. Spiller, and William J. Munro. Generalized toffoli gates using qudit catalysis. *Phys. Rev. A*, 80(1):012312, Jul 2009.

[12] Asif Islam Khan, Nadia Nusrat, Samira M. Khan, Masud Hasan, and Mozammel H. A. Khan. Quantum realization of some ternary circuits using muthukrishnan-stroud gates. In *ISMVL*, page 20. IEEE Computer Society, 2007.

[13] A. Khlopotine, M. Perkowski, and P. Kerntopf. Reversible logic synthesis by gate composition. In *Proceedings of IWLS*, pages 261–266, 2002.

[14] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Rev. Mod. Phys.*, 79(1):135–174, Jan 2007.

[15] B. P. Lanyon, M. Barbieri, M. P. Almeida, T. Jennewein, T. C. Ralph, K. J. Resch, G. J. Pryde, J. L. O'Brien, A. Gilchrist, and A. G. White. Quantum computing using shortcuts through higher dimensions, April 2008.

[16] S. Lee, S.J. Kim, J. Biamonte, and M. Perkowski. The cost of quantum gate primitives. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6):561–574, 2006.

[17] A. Leier. *Evolution of Quantum Algorithms Using Genetic Programming*. PhD thesis, University of Dortmund, 2004.

[18] X. Li, Y. Wu, D. Steel, D. Gammon, T.H. Stievater, D.S. Katzer, D. Park, C. Piermarocchi, and L.J. Sham. An all-optical quantum gate in a semiconductor quantum dot. *Science*, 301(5634):809 – 811, 2003.

[19] M. Lukac. *Quantum Logic Synthesis and Inductive Machine Learning*. PhD thesis, Portland State University, 2009.

[20] M. Lukac and M. Perkowski. Combining evolutionary and exhaustive search to find the least expensive quantum circuits. In *Proceedings of ULSI symposium*, 2005.

[21] M. Lukac and M. Perkowski. Using exhaustive search for the discovery of a new family of optimum universal permutative binary quantum gates. In *Proceedings of International Workshop on Logic & Synthesis, Poster Session*, 2005.

[22] M. Lukac and M. Perkowski. Evolutionary approach to quantum symbolic logic synthesis. In *IEEE Congress on Computational Intelligence (WCCI)*, pages 3374–3380, 2008.

[23] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, H. Jee, B.-G. Kim, and Y.-D. Kim. Evolutionary approach to quantum reversible circuit synthesis. *Artif. Intell. Review.*, 20(3-4):361–417, 2003.

[24] M. Lukac, M. Perkowski, and M. Kameyama. Evolutionary quantum logic synthesis of boolean reversible logic circuits embedded in ternary quantum space using structural restrictions. In *Proceedings of the WCCI 2010*, 2010.

[25] M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski.

Automated synthesis of generalized reversible cascades using genetic algorithms. In *Proceedings of Fifth Intern. Workshop on Boolean Problems*, pages 33–45, 2002.

[26] D. Maslov, G. W. Dueck, and D. M. Miller. Synthesis of Fredkin-Toffoli reversible networks. *IEEE Transactions on VLSI*, 13(6):765–769, 2005.

[27] D. Maslov and G.W. Dueck. Improved quantum cost for n-bit Toffoli gates. *IEE Electronic Letters*, 39(25):1790–1791, 2003.

[28] P. Massey, J.A. Clark, and S. Stepney. Evolving quantum circuits and programs through genetic programming. In *Proceedings of the Genetic and Evolutionary Computation conference (GECCO)*, pages 569–580, 2004.

[29] P. Massey, J.A. Clark, and S. Stepney. Evolving of a human-competitive quantum fourier transform algorithm using genetic programming. In *Proceedings of the Genetic and Evolutionary Computation conference (GECCO)*, pages 1657–1664, 2005.

[30] D. M. Miller. Spectral and two-place decomposition techniques in reversible logic. In *Proc. Midwest Symposium on Circuits and Systems, on CD-ROM*, August 2002.

[31] D. M. Miller, D. Maslov, and G. W. Dueck. Synthesis of quantum multiple-valued circuits. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6):431–450, 2006.

[32] A. Muthukrishnan and C. R. Stroud. Multivalued logic gates for quantum computation. *Phys. Rev. A*, 62(5):052309, Oct 2000.

[33] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[34] Michael A. Nielsen. Optical quantum computation using cluster states. *Phys. Rev. Lett.*, 93(4):040503, Jul 2004.

[35] NVIDIA. *NVIDIA CUDA*. NVIDIA, http://www.nvidia.com/object/cuda_learn.html.

[36] J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning. Demonstration of an all-optical quantum controlled-not gate. *Nature*, 426:264–267, 2004.

[37] J. Pachos and H. Walther. Quantum computation with trapped ions in an optical cavity. *Physical Review Letters*, 89(18), 2002.

[38] T. C. Ralph, K. J. Resch, and A. Gilchrist. Efficient toffoli gates using qudits. *Phys. Rev. A*, 75(2):022313, Feb 2007.

[39] B.I.P. Rubinstein. Evolving quantum circuits using genetic programming. In *Congress on Evolutionary Computation (CEC2001)*, pages 114–121, 2001.

[40] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum logic circuits. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, pages 272–275, New York, NY, USA, 2005. ACM Press.

[41] L. Spector. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Kluwer Academic Publishers, 2004.

[42] R. Stadelhofer, W. Banzhaf, and D. Suter. Evolving blackbox quantum algorithms using genetic programming. *Artif. Intell. Eng. Des. Anal. Manuf.*, 22:285–297, 2008.

[43] R. Wille and R. Drechsler. Bdd-based synthesis of reversible logic for large functions. In *Proceedings of the 46th Annual Design Conference*, 2009.

[44] T. Yabuki. Genetic algorithms for quantum circuit design – evolving a simpler teleportation circuit –. In *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, 2000.