Applying Evolutionary Techniques to Quantum Computing Problems

Steven R. Hutsell  
Dept. of Elect. & Computer Engr.  
Portland State University  
Portland, OR 97207–0751  
Email: steven.r.hutsell@intel.com

Garrison W. Greenwood  
Dept. of Elect. & Computer Engr.  
Portland State University  
Portland, OR 97207–0751  
Email: greenwd@ece.pdx.edu

Abstract—A new approach based on Evolution Strategies (ES) is proposed to evolve quantum unitary operators which represent the computational algorithm a quantum computer would perform to solve an arbitrary problem. This approach has superior attributes over previous endeavors by using real valued vectors and not limited to a predefined set of quantum gates. We show this method can be used to solve a variety of key problems from evolving simple quantum gates, quantum oracles, to generalized problem instances. This method is highly extensible and useful for propagating future research of quantum unitary operator development.

I. INTRODUCTION

Quantum Computing (QC) has proven to be a very powerful means of solving certain computationally difficult problems more efficiently than a classical computer. A quantum computer is described by a unitary operator. One of the most challenging tasks is how to construct those quantum operators. In fact, it is unknown how to efficiently generate a quantum operator for an arbitrary problem.

In the past, researchers have used genetic algorithms (GA) and genetic programming (GP) to evolve quantum circuits. But these previous approaches were completely different from the work described in this paper. Lukac and Perkowski [1] used a GA to evolve arbitrary quantum circuits but the target unitary matrix was known beforehand. Similarly Massey, et al. [2] used GP to evolve a circuit to implement a quantum Fourier transform. Again, the target unitary operator was known beforehand.

We have developed a new QC design method which uses an evolution strategy (ES) coupled with a clever method of generating random unitary matrices [3] to evolve the unitary operator which describes a quantum computer. Our method is completely different from the above mentioned research in a very crucial aspect: we don’t evolve quantum circuits equivalent to a target unitary operator, we evolve a unitary operator to produce a target quantum state. Our approach is far more general and opens the possibility of evolving new quantum algorithms.

This new method can evolve solutions to a variety of QC problems. For example, our method can evolve quantum gates—i.e., simple unitary operators which work on a small number of qubits and can be joined together to form more complex operations. We successfully evolved the Hadamard quantum gate which is heavily used for exploiting quantum parallelism. (To be precise, we evolved a slightly different gate with properties identical to the Hadamard gate, but this evolved version is actually more efficient to physically implement.) We can also evolve so-called quantum oracles which are normally developed by hand and are crucial for some of the most powerful quantum algorithms being developed.

The paper is organized as follows: The next section gives a background on quantum computation and describes the random unitary matrix method. In Section III we explain how an ES can evolve quantum unitary operators and Section IV shows several examples of our method including evolving quantum gates, evolving quantum oracles and evolving instance solutions for generalized problems. We conclude with a short summary how we can extend this method for other applications as well as current and future research.

II. BACKGROUND

A. Quantum Computation

Quantum computation has spread into many fields, initially starting with computing [4] and information [5], and quickly spreading to a wide range of fields such as cryptography [6], artificial intelligence [7], game theory [8], economics [9,10], and control systems [11,12].

The computational properties of quantum mechanics were originally investigated by Benioff [13], while the concept that quantum mechanics could be more computationally powerful than a classical Turing machine is attributed to Feynman [14,15]. Since then researchers have developed methods of describing quantum computers and developed quantum algorithms which can solve problems more efficiently than on a classical Turing machine.

Deutsch defined the quantum Turing machine [16] and the concept of quantum circuits (or gates) [17]. Along with Jozsa, they answered the long-standing question that a quantum computer could solve a sample problem more efficiently compared to a classical Turing machine [18]. More examples were found by Berthiaume and Brassard [19], Simon [20], and Bernstein and Vazirani [21]. This led to the discovery by Shor [22] of a polynomial-time quantum
algorithm for two key problems (discrete log and factoring) for which no polynomial-time classical algorithm is known. Another significant discovery is Grover’s algorithm [23] for database searching.

Classical computer systems represent a single bit of information deterministically: the value is either a logic 0 or a logic 1. Quantum computer systems represent a single bit of information as a qubit, which is a unit vector in a complex Hilbert space $C^2$. The ideas are commonly expressed using bra/ket notation introduced by Dirac [4]. The bra symbol is denoted by $\langle x |$ and the corresponding ket is denoted by $| x \rangle$. The ket describes a quantum state and the corresponding bra is its complex conjugate transpose.

Any practical quantum computer manipulates a register of $n$ qubits. If each qubit has an orthonormal basis $\{|0\rangle, |1\rangle\}$, then an $n$ qubit system has a basis expressed by the tensor product: $(C^2)^\otimes n = C^2 \otimes C^2 \otimes \cdots \otimes C^2$. This gives $2^n$ total basis vectors. In general, $|\alpha\rangle$ denotes the tensor product $\bigotimes_{i=0}^{n-1} |\alpha_i\rangle = |\alpha_n\rangle \otimes |\alpha_{n-1}\rangle \otimes \cdots \otimes |\alpha_1\rangle \otimes |\alpha_0\rangle$ which means a quantum register has the value $a = 2^n a_0 + 2^{n-1} a_1 + \cdots + 2^0 a_n$.

A qubit need not exist in only one basis state. Indeed, a qubit can exist as a linear superposition of basis states, $|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle$, where $c_0$ and $c_1$ are complex numbers satisfying $|c_0|^2 + |c_1|^2 = 1$. More generally, the $n$ qubit register can be prepared in a superposition of all possible classical states:

$$|x\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle$$

where the normalization condition $\sum_i |c_i|^2 = 1$ must hold. The complex number $c_i$ is called the amplitude associated with the state $|i\rangle$.

The state of a qubit register is determined by a measurement. In quantum systems, this measurement process projects the system state onto one of the basis states. Referring to Eq. 1, the measurement returns a value of $|i\rangle$ with probability $|c_i|^2$. Any subsequent measurement returns the state $|i\rangle$ with probability 1, which means the measurement process irreversibly alters the state of the system. Measurement also gives another perspective on entanglement: two qubits are irreversibly altered the state of the system. Measurement process projects the system onto one of the basis states.

The most conventional representation of a base state $|i\rangle$ is as a column matrix with the $i^{th}$ entry 1 and all other entries 0. A state $|\psi\rangle$ is therefore represented as a column matrix of the complex amplitudes. That is,

$$|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix}$$

Quantum systems evolve from state to state according to Schrödinger’s equation [24]. Suppose we start in state $|\psi\rangle = \sum c_i |i\rangle$. A linear operator $U$ produces a new state $|\phi\rangle = U |\psi\rangle$. Both states are linear combinations of the same base states, so $|\phi\rangle = \sum c_i' |i\rangle$. This means evolution occurs by modification of the state amplitudes. Note that the normalization condition required of states is satisfied iff $U$ is unitary—i.e., $U^\dagger U = I$.

It is important to emphasize the role superposition plays in quantum computing. Consider a state $|\psi\rangle = \sum c_i |i\rangle$. You can exploit the superposition using the property of quantum interference. Interference allows the exponential number of computations performed in parallel to either cancel or enhance each other. Feynman [24] beautifully describes how light waves can constructively or destructively interfere to produce this effect. The goal of any quantum algorithm is to have a similar phenomena occur—i.e., interference increases the amplitudes of computational results we desire and decreases the amplitudes of the remaining results. It is a unitary operator that would alter these amplitudes.

### B. Unitary Matrix Generation

The method chosen to generate random unitary matrices was developed by Zyczkowski and Kus [3] which conform to the statistical properties of circular unitary ensembles (CUE) originally investigated to describe the spectral properties of quantum objects [25].

We generate the $N \times N$ unitary operator, $U$, as a composition of elementary unitary operators which perform transformations in two-dimensional subspaces. This elementary matrix is denoted by $E_{i,j}^{(\phi, \psi, \chi)}$. The non-zero elements are given by the following rule:

\[
E_{i,j}^{(\phi, \psi, \chi)} = \begin{cases} 
1 & k = 1, \ldots, N \ k \neq i, j \\
\cos(\phi) e^{i\psi} & \\
\sin(\phi) e^{i\chi} & \\
-\sin(\phi) e^{-i\chi} & \\
\cos(\phi) e^{-i\psi} & 
\end{cases}
\]

The angles are taken over the intervals:

\[
0 \leq \phi \leq \frac{\pi}{2} \quad 0 \leq \psi < 2\pi \quad 0 \leq \chi < 2\pi
\]

Using the above elementary operators, we construct the $N-1$ composite rotations:

\[
E_1 = E^{(1,2)}(\phi_{12}, \psi_{12}, \chi_{12}) \\
E_2 = E^{(2,3)}(\phi_{23}, \psi_{23}, 0)E^{(1,3)}(\phi_{13}, \psi_{13}, \chi_{13}) \\
\vdots \\
E_{N-1} = E^{(N-1,N)}(\phi_{N-1,N}, \psi_{N-1,N}, 0) \times \\
E^{(N-2,N)}(\phi_{N-2,N}, \psi_{N-2,N}, 0) \times \\
\cdots \times E^{(1,N)}(\phi_{1N}, \psi_{1N}, \chi_{1N})
\]

The unitary operator is finally defined as:

\[
U = E_1 E_2 E_3 \cdots E_{N-1}
\]
III. EVOLVING A QUANTUM COMPUTER

Recent work has shown evolutionary algorithms can be effectively used to help design quantum computers. For example, genetic algorithms have been used for gate-level synthesis of a quantum computer [1]. However, in this paper, we take a different approach by using an ES to directly evolve unitary operators, which represent solutions to problem instances.

An ES represents individuals as a vector of real values. In particular, the genotype has the structure:

\[ < x_1, x_2, \ldots, x_n, \sigma_1, \sigma_2, \ldots, \sigma_n >, \]

where the first set, \{x\}, are the object parameters and the second set, \{\sigma\}, are the strategy parameters. The object parameters are the basic parameters needed to define the individual. In our case, the \(\phi, \psi, \chi\) angles from the previous section are the object parameters manipulated by the ES. For an \(N \times N\) unitary operator we will need \(N^2\) total object parameters plus \(N^2\) strategy parameters to control the mutation.

The initial state of the quantum system is represented by an \(N\)-element column vector, which has an initial state

\[ |\psi_0\rangle = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \]  

(7)

This particular initialization indicates the initial state of the quantum system is a linear superposition of all possible states with equal probability. Once the unitary operator (\(U\)) is constructed, the operation

\[ U |\psi_0\rangle = |\psi\rangle \]  

(8)

produces \(|\psi\rangle\), which is the result of the quantum computation the evolved \(U\) described. Each individual in the ES represents a unique operator \(U\).

The fitness of an individual can be calculated in a variety of ways depending on whether pre-measurement, post-measurement or probability distribution state vectors are used. Pre-measurement uses the complex state vector \(|\psi\rangle\) produced by equation (8) whereas post-measurement refers to the final real-valued state the quantum system collapses into after an observation is taken. The probability distribution vector is the modulus-squared of the state vector \(|\psi|^2\), which can be compared to a target probability distribution vector.

Given an individual with unitary operator \(U\), an initial state vector \(|\psi_0\rangle\) and a target final state vector \(|\psi^*\rangle\), fitness is calculated with the function

\[ \text{fitness}(U) = \sum_{0}^{N-1} \frac{1}{(|c|^2 - |c^*|^2)^2 + \epsilon} \]  

(9)

where \(c_i\) is the \(i^{th}\) complex component in the resultant state vector as described in equation (2) and \(c_i^*\) is the \(i^{th}\) complex element in the target state vector. The fitness equation above is an example of using probability distribution method since it involves the modulus-squared of complex elements. When designing quantum gates or other sub-circuits, the complex behavior is important so using a pre-measurement type fitness function would be more appropriate.

IV. EXPERIMENTAL RESULTS

This section presents the results from a number of experiments we conducted. All ES runs were executed on an Intel Dual-Xeon 3.6GHz HT platform with 8GB main memory.

A. General Problem Instance

Let \(\mathcal{P}\) be an optimization problem and suppose there are a large enough number of possible solutions so that an exhaustive search is impractical. An algorithm \(\mathcal{A}\) that optimally solves \(\mathcal{P}\) would take an initial solution \(x_0\) and compute the optimal solution \(x^*\). That is, \(x^* = \mathcal{A}(x_0)\). Of course the main difficulty is knowing how to create \(\mathcal{A}\). In terms of QC this is analogous to finding an optimal unitary operator \(U^*\) that can transform an initial state vector into a final state vector such that an observation will yield the optimal solution with probability 1. Since each component of the state vector is a problem solution, the state vector component corresponding to the optimal solution should have a value of “1” and all other components should have a value of “0”.

In this section we describe how to evolve such a unitary operator. The term “general problem instance” is used because we aren’t solving a particular \(\mathcal{P}\). Instead, the optimal solution is assumed to be the first component of the state vector, which means the target vector is a \(|0\rangle\). The initial quantum state is as shown in equation (7). The structure of this problem is shown in figure 1.

\[
\begin{pmatrix}
   u_{00} & u_{01} & u_{02} & \cdots \\
   u_{10} & u_{11} & u_{12} & \cdots \\
   u_{20} & u_{22} & u_{22} & \cdots \\
   \vdots & \vdots & \vdots & \ddots
\end{pmatrix}
\frac{1}{\sqrt{N}}
\begin{pmatrix}
   1 \\
   1 \\
   \vdots \\
   \vdots
\end{pmatrix}
= 
\begin{pmatrix}
   1 \\
   0 \\
   \vdots \\
   \vdots
\end{pmatrix}
\]

Fig. 1. Evolving \(U\) to solve for a known output vector

1, 2, 3, and 4 qubit versions of this problem were investigated. In practice it is unreasonable to assume an optimal \(U^*\) can be found that will produce a final state vector with all zero amplitudes except one. (Nature has very few absolutes) Consequently, the goal was to find a unitary operator that would produce a final state vector that, when measured, would return the optimal solution with some high probability. A (100 + 100) ES was run until the desired state \(|0\ldots0\rangle\) was reached with at least 70% probability—i.e., \(|c_0|^2 \geq 0.7\). The fitness function used is the same as shown in equation (9). Table I shows how many generations were required and the final probability for the desired state.

The 1, 2, and 3 qubit problems were handled quite easily. In fact, in the single-qubit case, the initial pool was so large that it was easy to find an individual well above 70% in the first generation. However, at 4 qubits the increased complexity imposes some run-time restrictions; a higher
number of qubits now require days of run time on a typical workstation. Figure 2 shows the progression of evolving the best fit individual for 2, 3, and 4 qubit problem instances.

<table>
<thead>
<tr>
<th>Qubits</th>
<th>Generations</th>
<th>Probability</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>73.17%</td>
<td>1s</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>71.84%</td>
<td>12s</td>
</tr>
<tr>
<td>4</td>
<td>460</td>
<td>71.70%</td>
<td>1453s</td>
</tr>
</tbody>
</table>

Table I: Generations versus number of qubits

![Target State Probability vs Generations](image)

B. Evolving the Hadamard Gate

The goal here is to see if an ES can evolve the well known and extremely useful Hadamard quantum gate whose unitary matrix is shown below:

\[ H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \] (10)

This gate is extremely useful because it maps the basis states \(|0\rangle\) and \(|1\rangle\) into a superposition of the two states with equal weight. That is,

\[ |0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2} \]

and

\[ |1\rangle \rightarrow (|0\rangle - |1\rangle)/\sqrt{2} \]

To set this up for an ES, we need to use two initial states (input states) that map to two final states (output states). The fitness is then averaged over both inputs. Pre-measurement based targets are used since the primary interest is in obtaining complex behavior. The ES produced the following input and target vectors for the single qubit gate problem:

INPUT [0] = [(1.0+0.0i), (0.0+0.0i)]

TARGET [0] = [(0.707+0.0i), (0.707+0.0i)]

INPUT [1] = [(0.0+0.0i), (1.0+0.0i)]

TARGET [1] = [(0.707+0.0i), (-0.707+0.0i)]

After 100 generations, the most fit individual has the unitary operator:

\[ U = \begin{pmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{pmatrix} \approx \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \] (11)

This result differs from the unitary matrix of the Hadamard gate described in equation (10). However, the probability distribution of this gate given each input state vector behaves exactly the same as the Hadamard gate does. That is, \(|U|0\rangle|^2 = |H|0\rangle|^2 = [50\%, 50\%]\) and \(|U|1\rangle|^2 = |H|1\rangle|^2 = [50\%, 50\%]\). However, a limitation of our random unitary matrix method is evident. For \(N = 2\), the general matrix from equation (6) is anti-symmetric and thus incapable of producing the Hadamard gate. Nevertheless, the gate we evolved is still valid and turns out to be a known gate called the “pseudo-Hadamard” which actually is preferred over the Hadamard when implementing quantum algorithms on Nuclear Magnetic Resonance quantum computers [26].

This result shows an ES can evolve alternate gates that might prove useful.

C. 2-qubit Oracle for Deutsch’s Problem

Deutsch presented a problem [18] for a quantum computer to determine whether a given function, \(f(x)\), was balanced (meaning \(f(x) = 1\) for half of the inputs and 0 for the other half) or constant (meaning \(f(x) = 0\) or always 1). This example is important since for \(n\) bits, a conventional computer requires \(2^n + 2\) queries, whereas on a quantum computer Deutsch’s algorithm exploits quantum parallelism and interference to produce the answer in one query.

The role of a quantum oracle is to make a decision based on its input state in one step. If the inputs of the quantum oracle are in superposition state, then the oracle can make all possible decisions in one step. Figure 3 shows how the quantum circuit is implemented. The input state, \(|\psi0\rangle\), is set to \(|01\rangle\). This is then passed through Hadamard gates to put them into a superposition state which is then processed by \(U\), the oracle we want to design. The output of the oracle will be a superposition of decisions which we can resolve by passing them again through Hadamard gates. This has become a common structure of quantum algorithms. The objective here is to construct an oracle which will determine if \(f(x) = \text{NOT}(x)\) is balanced or constant.

\[ |0\rangle \rightarrow |H\rangle \rightarrow U \rightarrow |H\rangle \rightarrow \] (Fig. 3)

\[ |1\rangle \rightarrow |H\rangle \rightarrow U \rightarrow |H\rangle \]

Obviously \(f(x)\) is a balanced function and the desired pre-measurement output state is known to be \(|\psi\rangle = |0, 0, 0, -1\rangle\).
The ES is modified such that it evolves the quantum oracle, $U$, until it satisfies the equation below. Note that the two Hadamards in parallel can be represented by the tensor product $H \otimes H$, also written as $H^{\otimes 2}$.

$$T = H^{\otimes 2} \times U \times H^{\otimes 2} \times (|0\rangle \otimes |1\rangle)$$ (12)

Using the $(100 + 100)$ ES, a perfect solution was found after only 26 generations ($\sim 5$ seconds runtime). To speed up the convergence, we modified the mutation scheme to increment the rotation angles ($\phi, \theta, \chi$) in steps of size $\pi/12$. The resultant oracle was:

$$U = \begin{pmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0
\end{pmatrix}$$ (13)

This is actually different from the oracle discovered by Deutsch and could possibly have desirable properties over the original.

V. Conclusions & Future Work

We have described how an evolution strategy can be used to evolve unitary operators which represent the computations of a quantum computer. The novel aspect of this work is the unitary operator is evolved rather than specified. We were able to show efficient quantum gates can be rediscovered using our method. Moreover, the described method can evolve unitary operators capable of solving arbitrary optimization problems. Even the oracle for Deutsch’s problem can be evolved using our method. This capability introduces a new method of designing quantum algorithms.

The investigations described here were limited to less than five qubits because larger quantum problems would have had excessive run times when executed, as we did, on a single workstation. Nevertheless, the basic principles described in this paper are sound and more powerful computing platforms will allow investigations on larger problems with a larger number of qubits. Evolutionary algorithms are well-suited for running on parallel processing systems. Our future research efforts will include porting our methods to a parallel processing computer system.

References