Decomposition of Unitary Matrices for Finding Quantum Circuits

Anmer Daskin[1] and Sabre Kais[2]

[1]*Department of Computer Science, Purdue University, West Lafayette, IN, 47907 USA*
[2]*Department of Chemistry and Birck Nanotechnology Center, Purdue University, West Lafayette, IN 47907 USA*

Constructing appropriate unitary matrix operators for new quantum algorithms and finding the minimum cost gate sequences for the implementation of these unitary operators is of fundamental importance in the field of quantum information and quantum computation. Here, we use the group leaders optimization algorithm, which is an effective and simple global optimization algorithm, to decompose a given unitary matrix into a proper-minimum cost quantum gate sequence. Using this procedure, we present new circuit designs for the simulation of the Toffoli gate, the amplification step of the Grover search algorithm, the quantum Fourier transform, the sender part of the quantum teleportation and the Hamiltonian for the Hydrogen molecule. In addition, we give two algorithmic methods for the construction of unitary matrices with respect to the different types of the quantum control gates. Our results indicate that the procedure is effective, general, and easy to implement.

I. INTRODUCTION

Quantum computation promises to solve fundamental, yet otherwise intractable problems in many different fields. It is commonly believed that advancement in quantum computing science will bring new polynomial time algorithms, and some NP-complete problems shall be solvable in polynomial time, too. To advance the quantum computing field, finding circuit designs which can execute algorithms on quantum computers (in the circuit design model of quantum computing) is important. Therefore, it is of fundamental importance to develop new methods with which to overcome the difficulty in forming a unitary matrix describing the algorithm (or the part of the computation), and the difficulty to decompose this matrix into the known quantum gates.

The problem in the decomposition of a given unitary matrix into a sequence of quantum logic gates can be presented as an optimization problem. Williams and Gray [1] suggested the use of genetic programming technique to find new circuit designs for known algorithms, and also presented results for quantum teleportation. Yabuki, Iba [2] and Peng et al. [3] focused on circuit designs for the quantum teleportation by using different genetic algorithm techniques. Spector [4] explains the use of genetic programming to explore new quantum algorithms. Stadelhofer [5] used genetic algorithms to evolve black box quantum algorithms. There are also some other works [6–8] which evolve quantum algorithms or circuits by using genetic programming or genetic algorithms. Review of these procedures can be found in [9].

In this paper, we use the group leaders optimization algorithm (GLOA)-an evolutionary algorithm-to decompose the unitary matrices representing some quantum algorithms or the unitary propagator of a given many-body Hamiltonian. In addition to giving the circuit designs for the simulation of the hydrogen Hamiltonian, we show new circuit designs for the operators of the Grover search algorithm; the sender part of the quantum teleportation; the Toffoli gate; and the quantum Fourier transform. We also present two algorithmic methods to efficiently find the unitary matrix representations of the single- and multiple-control quantum gates. These two methods allow us to use a wide variety of quantum gates in the optimization without affecting the computing performance.

The structure of the paper is as follows: after giving a brief introduction about quantum computation, we describe in Sec.III the algorithmic methods for the construction of the matrix representations of the quantum control gates; Sec.IV and Sec.V are devoted to the optimization problem and the objective function; in Sec.IV, we describe the group leaders optimization algorithm and give the flow chart of the algorithm for the optimization problem; and finally, a brief discussion of the test cases and the result-circuit designs are given in Sec.VII.

II. QUANTUM COMPUTATION

In the circuit model of quantum computing the gates are the unitary operators which carry a system from one state to another state. For a close quantum system described by:

$$i\hbar\frac{d}{dt}\psi(t) = H\psi(t), \tag{1}$$

the solution for fixed times t_0 and t_1 is:

$$\psi(t_1) = e^{-\frac{i}{\hbar}H(t_1-t_0)}\psi(t_0). \tag{2}$$

In Eq.(2), $e^{iH(t_1-t_0)}$ is the unitary operator which carries the system from the state at t_0 to the state at t_1 for time independent system [10]. For time dependent system the unitary operator which implements the desired gate is $Te^{-\frac{i}{\hbar}\int H(t)\,dt}$, where T is the time-ordering operator [11, 12]. These unitary operators implement the quantum gates and a general single quantum gate can be represented as a unitary matrix operator in the computational basis:

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}. \tag{3}$$

Some of known quantum gates for one qubit are:

$$\begin{aligned} I &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ Y &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \end{aligned} \tag{4}$$

where, I is the identity matrix and X, Y, and Z are ,in order, the Pauli matrices σ_x, σ_y, and σ_z. Here, X gate is also known as NOT gate that transforms the quantum state from $|0\rangle$ to the state $|1\rangle$ and vice versa. Some of other unitary operators are given in Table I.

If a set of quantum gates adequately approximate any n-qubit (n≥1) unitary operator to arbitrary accuracy, that set of gates are said to be universal. Any 2-qubit entangling gate with a 1-qubit gate forms a universal set of gates [10]. In this paper, all quantum gate sets used in the optimization obey the definition of the universality.

Using a universal set of quantum gates, any unitary matrix acting on Hilbert Space can in principal be approximated to an arbitrary accuracy. [13]. For an N by N unitary operator on n qubits, where $N = 2^n$, $O(N^2(\ln N)^3)$ number of gates are required to represent all matrix elements of the operator, and so the required number of gates for a unitary operator grows exponentially with respect to the number of qubits [14].

III. UNITARY MATRIX REPRESENTATION AND CONSTRUCTION METHODS FOR QUANTUM GATES

As mentioned in the previous section, quantum gates can be represented as unitary matrices in the computational basis. To find the unitary matrix representation of a gate, it is necessary to show its operation on $\{|0\rangle$ and $|1\rangle\}$ in the computational basis. For instance, the action of X gate is represented as $|0\rangle \rightarrow |1\rangle$, $|1\rangle \rightarrow |0\rangle$. And $|0\rangle\langle 1| + |1\rangle\langle 0|$ gives the corresponding unitary matrix[15], given in Eq.(4). For control gates it is more difficult to find the unitary matrices by operation since the unitary matrix representation of a gate changes with respect to the order of the control and target qubits within the circuit and the number of qubits the quantum gate acts on. As stated in [15], the equivalent unitary matrix to CNOT (controll-NOT) can be found as follows:

$$\begin{aligned} CNOT &= |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| \\ &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \end{aligned} \tag{5}$$

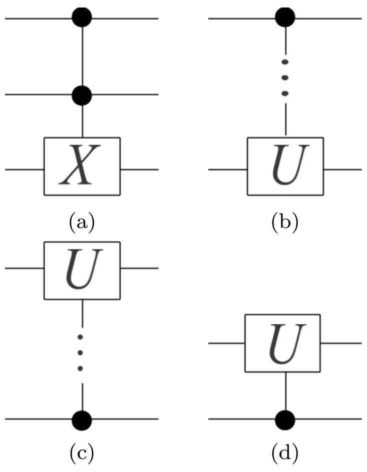

FIG. 1: Various types of quantum control gates: In (a), X gate operating on the target qubit is controlled by two qubits (the Toffoli gate). In (b), the control and target qubits are not neighbors. In (c), the control and target qubits are not neighbors and are in reverse order. In (d) the control and target qubits are neighbors, but in reverse order.

This process becomes tedious when there exist control gates in which some qubits are between the control and the target qubits, and the placement of the control and target qubits are in reverse order as shown in Fig.1. Because this is tedious and and concerns about the efficiency for the optimization process, we use two algorithmic methods to construct unitary matrices for the desired control gates. The first algorithm is for the construction of the unitary matrices for the single-control gates. And the second algorithm gives the unitary matrices for the multiple-control gates.

The main idea of these algorithms is: to use the order of qubits in the state register represented as $|q_0q_1...q_n\rangle$ (n is the number of qubits on which the control gate acts); find the related indices for these qubits in the identity matrix; and using the distance between the control and target qubits in the state register, change the related elements of this matrix with the elements of the controlled elementary gate. In the beginning of both algorithms, an identity matrix is generated having the right dimension for the quantum gate that will be represented. And then, the initial positions of the control and target qubits in this matrix are located by using the order of the control and target qubits in the state register, $|q_0q_1...q_n\rangle$. Finally, the jumping amount from this initial positions in the unitary matrix are determined by using the distance between the control and target qubits, and the elements in the related positions are changed according to the elements of the single elementary gate that acts on the target qubit, and is controlled by the control qubit. The correctness of these algorithms comes from the direct relation between the state register and the matrix representation, and can be seen by constructing unitary matrices for the different types of quantum gates.

A. Unitary matrix construction method for the single-control quantum gates

This algorithm builds a unitary matrix which represents the different types of the control gates which posses only one control and one target qubits. It locates the related elements of the identity matrix, and changes them with the elements of the elementary gate acting upon the target qubit.

Algorithm-1 Unitary matrix construction method for the control gates from an identity matrix

```
Input($target, control$)
{Either the control or the target qubit is zero.}

1: $d = |control - target|$;
2: $U = I$;
   {I is $2^{d+1}$ by $2^{d+1}$ identity matrix.}
3: $t = 2^{target}$;
4: $c = t + 2^{control}$;
5: $m = 2^{d-1} - 1$;
6: for $k = 0$ to $m$ do
7:   $i = c + 2k$;
8:   $j = t + 2k$;
```

```
9:     $U_{ii} = u_{00}; U_{ij} = u_{01}; U_{ji} = u_{10}; U_{jj} = u_{11}$;
       {where $u_{00}, u_{01}, u_{10}$, and $u_{11}$ are the
       matrix elements of the single gate.}
10:  end for
return $U$;
```

In Algorithm-1, d-which is the absolute difference between the control and target qubits-helps to find the number of qubits covered by the control gate and is the key element in the determination of the number of times the for-loop shall run. Since the algorithm works on an identity matrix, in the second line of the algorithm the matrix U is initially defined as a 2^{d+1} by 2^{d+1} identity matrix which shall be the representation of the control gate at the end of the algorithm. The variables c and t are the indices used to find the i and j which determine the indices of elements to be changed inside the loop. After finding the correct indices, the algorithm changes the four elements of U in each iteration with the elements (u_{00}, u_{01}, u_{10}, and u_{11}) of the single gate that operates on the target qubit. Since the loop runs $m+1$ times, the total $4 \times (m+1)$ elements are replaced by the elements of the single gate. At the end of the algorithm, the matrix U represents the desired unitary matrix for the control gate, and is given as an output of the algorithm.

The running time of the Algorithm-1 is $O(N)$ in the worst case since the loop, which is the dominant in the running time, runs $m+1$ times, where the variable m is found as:

$$N = 2^{d+1}; \text{ and } m = 2^{d-1} - 1; \rightarrow m = \frac{N}{4} - 1. \tag{6}$$

Note that if the definition of the identity matrix at the beginning of the algorithm is involved in the running time, then the total running time becomes $O(N^2)$.

B. Unitary matrix construction method for the multiple-control gates

Algorithm-2 Unitary matrix construction method for the multiple-control gates

```
Input($target, control$)
{Either the control or the target qubit is zero.}

1: $d = |control - target|$;
2: $U = I$;
   {$I$ is $2^{d+1}$ by $2^{d+1}$ identity matrix.}
3: $t = 2^{target}$;
4: $c = t + 2^{control}$;
5: $m = 2^{d-1} - 1$;
6: $i = c + 2m$;
7: $j = t + 2m$;
8: $U_{ii} = u_{00}; U_{ij} = u_{01}; U_{ji} = u_{10}; U_{jj} = u_{11}$;
   {where $u_{00}, u_{01}, u_{10}$, and $u_{11}$ are the
   matrix elements of the single gate.}
return $U$;
```

Algorithm-2 creates a unitary matrix which represents the multiple-control quantum gates. Unlike in the regular-control gates, all of the qubits between the control and target qubits are also the control qubits in the multi-control gates. The idea of Algorithm-2 is similar to Algorithm-1, but the absence of a for-loop makes this simpler than Algorithm-1. The algorithm starts with an identity matrix having the dimension of 2^{d+1} by 2^{d+1}, then it changes the required four elements of the identity matrix. Note that for the regular-multi-control gates-the Toffoli gates-the algorithm simply changes the only four elements in the lower-right-hand corner of the identity matrix.

If the definition of U is not included in the running time, the algorithm runs in $O(1)$. Otherwise, it takes $O(N^2)$ time, the same as the running time of Algorithm-1.

IV. IMPLEMENTATION DESIGN OF THE PROBLEM

There are a few things which need to be considered before starting the optimization process. First, the quantum gates; the target and control qubits; and the angle for the rotation gates; need to be described in a way which can be read by the optimization algorithm to find the decomposition of a given unitary matrix in terms of a quantum gate sequence. We use the same idea as in [16], and define each quantum gate with a number in the range of 1 to the maximum number of gates. In our implementation The default set of gates used in the optimization is given in Table II. Since we use general construction methods for the quantum gates, the set of the quantum gates can be very diverse and include the wide variety of quantum gates.

The sequence of the gates (candidate circuit design solution) are represented with a matrix having 4 columns: the first column identifies the type of the gate; the second column is for the target qubit; the third column is for the control qubit; and the fourth and final column represents the angle. The fourth column (angle) for the non-rotation gates, the third column for single (non-control) gates, and the rows defining the control gates with the equal target and control qubits are not taken into consideration (they are all zero in the solution matrix). And for the multiple-control gates, all qubits between the target and control qubits are also taken as control qubits for the gate. The total number of rows in the matrix represents the maximum number of gates that can be used for the solution. This representation is similar to the representation of Miller and Thomson's Cartesian genetic programming for classical circuits [17].

In addition to defining quantum gates, it is also important to be able to simulate all kinds of quantum gate sets. In the previous section, we had explained the algorithms used to construct the unitary matrices for different types of single- and multiple-control quantum gates. Once the matrix representation for a quantum gate is found, the matrix describing the act of the quantum gate for the whole system is found by:

$$U_i = I \otimes I \otimes ... \underbrace{G_i}_{\substack{i_{th} \\ \text{factor}}} ... \otimes I \otimes I, \quad 0 \leq i \leq n. \quad (7)$$

In Eq.(7), G_i is the gate acting on i_{th} qubit (i may be more than one qubit). As an example for the circuit in Fig.2, the whole unitary matrix that represents the act of control-U gate in the circuit can be found to be:

$$U_0 = G \otimes I, \quad (8)$$

where G is an 8 by 8 unitary matrix which represents control-U gate in the circuit and constructed by using Algorithm-1. In Fig.2, the quantum state at ϕ_0 is:

$$|\phi_1\rangle = |\psi_1\psi_2\psi_3\psi_4\rangle. \quad (9)$$

After applying the quantum gate (control-U) to the circuit, the quantum state at ϕ_1 is found as follows:

$$|\phi_2\rangle = U_0|\phi_1\rangle = U_0|\psi_1\psi_2\psi_3\psi_4\rangle. \quad (10)$$

If there is more than one gate on the circuit, the whole computation is defined by combining these gates either by their orders in time or from left to right. For n gates operating on a circuit, the computation can be defined as:

$$|\phi_{last}\rangle = U_n U_{n-1} ... U_2 U_1 U_0 |\phi_{initial}\rangle, \quad 0 \leq i \leq n, \quad (11)$$

where $|\phi_{initial}\rangle$ is the initial, and $|\phi_{last}\rangle$ is the final quantum states. And U_i represents the unitary operator for the i_{th} quantum gate.

The definitions of the optimization problems:

Two slightly different optimization problems can be defined for the circuit model of quantum computation:

- The first problem is: for a given initial quantum state, finding a unitary matrix (or a sequence of the quantum gates) which brings the initial state to the desired state. The error is the difference between desired state and the state found by the optimization algorithm. This problem is the same as finding the right sequence of U matrices in the expression shown in Eq.(11) in a way that will give the desired output $|\phi_{last}\rangle$ when the sequence is applied to the initial state $|\phi_{initial}\rangle$.

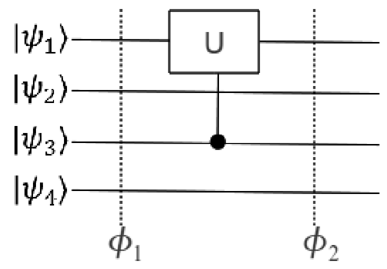

FIG. 2: A simple gate operating on the first qubit and controlled by the third qubit. An 8 by 8 matrix G representing the control-U gate in the circuit is found by using Algorithm-1 . The whole unitary operator, U_0, is a 16 by 16 matrix found by taking the Kronecker product of G and I (the identity matrix for the fourth qubit).

- The other optimization problem can be described as finding a sequence of the quantum gates (or quantum circuit design) which gives the desired unitary matrix. That means, the matrix U_{given} describing the computation will be decomposed into the given quantum gates. Hence, the evolution of the whole quantum system will be defined in terms of quantum gates. The error will be the difference between the unitary matrix given by the user and the unitary matrix found by the optimization.

The second optimization problem is the main interest of this paper. Hence, all experiments have been made to find quantum circuit designs for the known quantum algorithms and the unitary propagator of Hamiltonians.

V. DEFINING OBJECTIVE FUNCTION

The definition of the objective function is an important factor in the optimization process. In our case, there are two factors which need to be optimized: the correctness and the cost of the circuit. Let U_f be the matrix found by the optimization and U_g is the given unitary matrix to be decomposed. The correctness is defined in [18] as:

$$Correctness(C) = \left| \frac{Tr\left(U_g U_f^{\dagger}\right)}{N} \right|, \qquad (12)$$

where $N = 2^n$ (n is the number of qubits), the symbol $\dagger$ represents the complex conjugate transpose of a matrix, and Tr(..) is the trace of a matrix. This definition ignores the global phase differences, which make the optimization easier for many cases. However, for the simulation of the chemical Hamiltonians the global phase must also be taken into consideration so as to get accurate circuits. In order to add the global phase differences to the definition of the correctness, we separately consider the real parts and the imaginary parts of the diagonal elements of the matrix formed by multiplying U_g and $U_f^{\dagger}$. It is obvious that when the real parts are one and the imaginary parts are zero, the correctness is one and $U_f = U_g$. The correctness used in this paper respects the global phase differences, and is defined as:

$$C = \sum_{j=1}^{N} \left| \frac{Re\left(\mathbb{U}_{jj}\right) - \left(Im\left(\mathbb{U}_{jj}\right)\right)^2}{N} \right|, \qquad (13)$$

where $\mathbb{U} = U_g U_f^{\dagger}$, and j is the matrix index, $j = 1, 2, ..., N$.

The correctness (C) obtained from Eq.(13) determines how much U_f looks like U_g. It is generally in the range $[0, 1]$, and is 1 if $U_f = U_g$.

The cost of a circuit describes the level of ease with which this circuit is implemented; in order to make the implementation of a circuit easier and the circuits less error-prone, the cost of a circuit also needs to be optimized by minimizing the number of gates in the circuits. Hence, in addition to the correctness, the objective function should include the cost of the circuit.

The cost of a circuit is found by summing up the cost of each quantum gate in the circuit. The different types of quantum gates have different costs: The cost of a single gate is 1 and less than the cost of any type of the control gates since the implementation of a single gate on quantum computers is much simpler than the implementation of the control gates. To estimate the cost of the control gate, instead of placing a constraint on the target and control qubits by requiring them to be adjacent as done in [16], we take the cost of a gate with the closer target and control qubits; the number of qubits between the control and target qubits is multiplied by a constant. This constant is 2 for the regular control gates and 3 for the multi-control gates. For instance, while the cost of the CNOT gate whose control and target qubits are neighbors is $2 \times 1 = 2$ (2 is the constant and 1 is the number of qubits between the target and control qubits.), the cost for the Toffoli gate is $3 \times 2 = 6$ (3 is the constant for the multi-control gates, and 2 is the number of qubits between the first control and the target qubits.). The following expression finds the cost of the circuit in Fig.3a:

$$Cost = 2 \times 1 + 2 \times 1 + 2 \times 2 + 2 \times 1 = 12. \quad (14)$$

The general objective function is defined by including both the correctness and the cost of the circuit with some weights:

$$y = \left| 1 - (\alpha C + \frac{\beta}{Cost}) \right|, \quad (15)$$

where the constants α and β are the weights for the correctness and the cost, and defined as:

$$\begin{gathered} 0 \leq (\alpha, \beta) \leq 1; \\ \alpha + \beta = 1. \end{gathered} \quad (16)$$

The cost in the objective function is also scaled to the range $[0, 1]$ by taking β less than 1. And in general, taking α greater than β leads to more accurate results. In our experiments, we took $\alpha = 0.9$ and $\beta = 0.1$:

$$y = \left| 1 - (0.9C + \frac{0.1}{Cost}) \right|. \quad (17)$$

In this definition of the objective function, the effect of the cost can reach a maximum of 0.1 on the result of the objective function when the cost of a circuit is 1 (the circuit includes only one elementary gate). Therefore, because the constant β is small as opposed to α, the change in the value C is more significant than the cost. Hence, the optimization generally focuses on correctness more than cost.

It is important to note that the value of the objective function never becomes zero since all circuit designs with at least one gate have a cost value. For the test cases, in addition to the values of objective functions, we give the C values for the circuits to make the cost and correctness values in the objective function more coherent. Also note that on the results of the objective function there may be some computer round-off errors.

VI. GROUP LEADERS OPTIMIZATION ALGORITHM

Group leaders optimization algorithm (GLOA)[19] is a simple and effective global optimization algorithm that models the influence of leaders in social groups as an optimization tool. The general structure of the algorithm is made up by dividing the population into several disjunct groups each of which has its leader (the best member of the group) and members. The algorithm which is different from the earlier evolutionary algorithms and the pivot method algortihm [20–22] consists of two parts. In the first part, the member itself-the group leader with possible random part-and a new-created random solution are used to form a new member. If the formed new member is a better solution to the problem than the old member, it replaces the old one. This mutation is defined as:

$$\begin{aligned} new\ member = {} & r_1\ portion\ of\ old\ member \\ & \cup\ r_2\ portion\ of\ leader \\ & \cup\ r_3\ portion\ of\ random, \end{aligned} \quad (18)$$

where r_1, r_2, and r_3 determines the rates of the portions of the old member, the group leader, and the new-created random solution; which form the new member, and sum to 1.

In addition to the mutation, in each iteration for each group of the population one-way-crossover (also called the parameter transfer) is done between a chosen random member from the group and a random member from a different-random group. This operation is mainly replacing some random part of a member with the equivalent part of a random member from a different group. If new formed members give better solution to the problem, then they survive and replace the old members of the groups; otherwise, they do not. The amount of the transfer operation for each group is defined by a parameter called transfer rate. The values of the parameters of the algorithm used in the experiments are given in the next chapter. The flow chart of the algorithm for our optimization problem is drawn in Fig.10.

VII. TEST RESULTS

In the experiments, some parameter adjustments were done as follows: The set of gates in Table II were used as the default universal gate set. While in the case of the unitary propagator of the Hamiltonian of the hydrogen the angle is defined as a multiple of 0.005 and in the range $[0, 2\pi]$, in the other cases it is defined as $k\pi$, and k is multiple of 0.125 and in the range $[0, 2]$. All quantum gates, except the single quantum gates, were formed by using Algorithm-1 during the evaluation of the objective function in the optimization.

The parameters for the optimization algorithm are given in Table IV. The correctness (C) and the minimized objective function values and the number of iterations can be found in Table V with respect to all circuit design results that are given in the paper.

In the following subsections, the test cases are explained, and their circuit diagram results in related figures are drawn by using the output of the optimization program. As an example, the output of the program for the decomposition of the unitary matrix for the diffusion part of the Grover search algorithm is as follows:

$$\left\{\begin{array}{cccc} G & T & C & Q \\ \hline \text{Control X,} & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \text{Single V} & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \text{Single V} & 1 & 0 & 0 \\ \text{Control X} & 2 & 1 & 0 \\ \text{Single V} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right\}, \qquad (19)$$

where the column G represents the name of the gate, T is the target qubit, C is the control qubit, and Q determines the angle values for rotation gates. The number of rows is equal to the maximum number of gates. The zeros are the control gates with the equal control and target qubits, or a field the gate does not need to use such as the control qubit field for the single gates. Fig.4a shows the circuit diagram for this output.

A. Toffoli Gate

The Toffoli gate consists of two control and one target qubits, shown in Fig.1a. The single gate acts on the target qubit if and only if both control qubits are in state $|1\rangle$ [23, 24]. The unitary matrix representation of this gate is:

$$U_{toffoli} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}. \qquad (20)$$

And the operation of the Toffoli gate can be denoted as:

$$U_{toffoli}|\psi_0, \psi_1, \psi_2\rangle = |(\psi_0\psi_1) \otimes \psi_2\rangle. \qquad (21)$$

In the above equation, the symbol, $\otimes$, is the *exclusive-or* operator which simulates the function of a NOT gate.

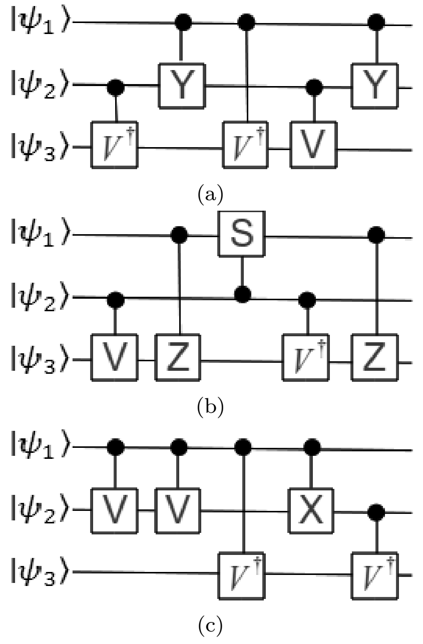

FIG. 3: The found circuit designs for the Toffoli gate.

The set of gates in Table II and the parameters in Table IV were the default sets used for the optimization program. From different runs of the algorithm, various circuit designs were found for the decomposition of the Toffoli gate, and the most efficient these are shown in Fig.3. For the parameters of the optimization algorithm, the correctness, and the values of the objective function, please, refer to Table IV and V.

B. Grover Search Algorithm

Grover search algorithm presented in [25] is one of the advances quantum computing has brought on classical computing. It reduces the computational time of a brute force search from $O(N)$ to $O(\sqrt{N})$. The main body of the algorithm is composed of two main operators: U_f and $U_{\psi^\perp}$ [10, 24, 26]. These two operators are applied after putting the n-qubit initial state $|\psi\rangle = |000...0\rangle$ into a superposition of quantum states with equal probabilities by applying Hadamard gates:

$$H^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=1}^{N}|x\rangle. \qquad (22)$$

The first operator U_f can be defined as:

$$U_f = I - |a\rangle\langle a|, \qquad (23)$$

where a is the element that is being searched, and I is the identity. The act of U_f is to mark the element x if and only if $x = a$; the function, $f(x)$, is equal to 1 for $x = a$. That can be denoted as:

$$U_f|x\rangle = -1^{f(x)}|x\rangle = \left\{ \begin{array}{cc} |x\rangle, & if\ x \neq a \\ -|x\rangle, & if\ x = a \end{array} \right\}. \qquad (24)$$

$U_{\psi^\perp}$ is the inversion about the average operator (or diffusion operator) which amplifies the amplitude of the marked state while reducing the amplitudes of the rest. And so the probability of seeing the marked element at the end of measurement gets higher. This operator is defined as:

$$U_{\psi^\perp} = 2|\psi\rangle\langle\psi| - I. \qquad (25)$$

The matrix representation of this diffusion operator D is found as follows [25]:

$$D_{ij} = \left\{ \begin{array}{cc} \frac{N}{2}, & if\ i \neq j \\ -1 + \frac{N}{2}, & if\ i = j \end{array} \right\}. \qquad (26)$$

If we sum up the algorithm, we can represent it in four steps [10]:

1. Start with an n-qubit initial state $|000..0\rangle$.
2. Put this initial state into the superposition by applying Hadamard (H) gates to the each qubit.
3. $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ times
   apply the first operator U_f.
   apply the second operator $U_{\psi^\perp}$.
4. Measure the result.

In our experiment, we have decomposed the matrix representation of the operator $U_{\psi^\perp}$ for two qubits, which is found by using Eq.(26). The matrix is defined as:

$$D = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}. \tag{27}$$

The circuit designs found for the above unitary matrix by the optimization are shown in Fig.4. The errors for the circuits in Fig.4 are zero; however, the objective function values are non-zero because of the cost of the circuits (see in Table V).

C. Quantum Fourier Transform

The quantum Fourier transform (QFT) functions exactly the same as the classical discrete Fourier transform (DFT) when applied to the amplitude of quantum state. While the computational time of the DFT is $O(N \log N)$, the quantum Fourier transform can be computed by using $O(\log^2 N)$ elementary operations [27]. Thus, the QFT exponentially speeds up the computation time and requires $O(n^2)$ quantum gates for the circuit implementation. Since the inverse QFT makes possible to estimate the phases of a unitary operator, it is the key element in many algorithms such as the factoring problem and the order finding problem [24]. Hence, it also advances the computation time of some of these problems.

The discrete and the quantum Fourier transforms can be described as follows [15]: Let w_n be a primitive N_{th} root of unity:

$$w_n = 2^{\frac{2\pi i}{N}},\ N = 2^n. \tag{28}$$

For an input vector of complex numbers $x_0, x_1, ..., x_{N-1}$, The DFT outputs the vector of complex numbers $y_0, ..y_{N-1}$. It is denoted as follows:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} x_j w_n^{jk}. \tag{29}$$

The quantum Fourier transform on an orthonormal basis $|0\rangle, ...|N-1\rangle$ maps the quantum states as:

$$|j\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} w_n^{jk} |k\rangle. \tag{30}$$

That means, the discrete transform of x_j becomes the amplitudes of y_k. The unitary matrix representation of the QFT (U_{QFT}) can be defined as:

$$\frac{1}{\sqrt{N}}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & 1 \\ 1 & w^2 & w^4 & \cdots & w^{(N-1)} \\ 1 & w^3 & w^6 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{(N-1)} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{pmatrix}. \tag{31}$$

For the optimization, we used the unitary matrices for the three and two-qubits QFT. The matrix for the two-qubit QFT is as follows:

$$U_{QFT_4} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. \tag{32}$$

Using the parameters in Table IV and V, the decompositions of the two-qubit QFT are found as in Fig.5 with the related objective function values in Table V. The unitary matrix for three-qubit QFT (U_{QFT_8}) can be represented concisely as:

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1+i}{\sqrt{2}} & i & \frac{-1+i}{\sqrt{2}} & -1 & \frac{-1-i}{\sqrt{2}} & -i & \frac{1-i}{\sqrt{2}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{-1+i}{\sqrt{2}} & -i & \frac{1+i}{\sqrt{2}} & -1 & \frac{1-i}{\sqrt{2}} & i & \frac{-1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1-i}{\sqrt{2}} & i & \frac{1-i}{\sqrt{2}} & -1 & \frac{1+i}{\sqrt{2}} & -i & \frac{-1+i}{\sqrt{2}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{1-i}{\sqrt{2}} & -i & \frac{-1-i}{\sqrt{2}} & -1 & \frac{-1+i}{\sqrt{2}} & i & \frac{1+i}{\sqrt{2}} \end{pmatrix}. \tag{33}$$

This matrix is decomposed into the quantum gates, and the exact circuit design in Fig.6 is found with the objective function value given in Table V.

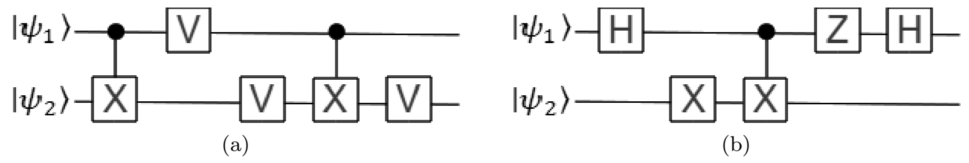

FIG. 4: The found circuit designs for the two-qubit amplification part of Grover search algorithm.

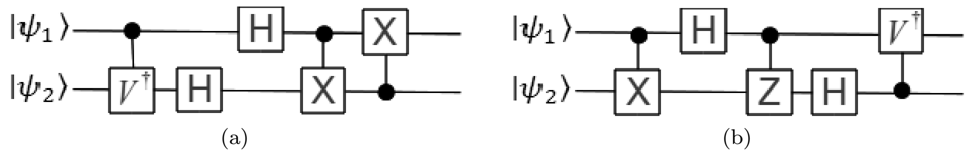

FIG. 5: Found circuit designs for the two-qubit quantum Fourier transform: (a) is the two-qubit QFT-1, and (b) is the two-qubit QFT-2 in Table V.

D. Quantum Teleportation

Suppose Alice who has the first qubit wants to send information to Bob who has the second qubit. Quantum teleportation is a protocol which allows Alice to communicate an unknown quantum state of a qubit by using two classical bits in a way that Bob is able to reproduce the exact original state from these two classical bits [10, 15]. In order to successfully transmit information between Alice and Bob, they must share an entangled qubit in the beginning. Quantum teleportation can be categorized in four parts [10, 15, 24, 28, 29]:

- Sender part and measurement: Alice encodes the information

- Classical channel that carries the information from Alice to Bob.

- Receiver part: Bob decodes the information and reproduce the original quantum state.

As a case for the optimization, we used the unitary matrix representation for the sender part of the teleportation (U_{sender}), given in [2, 18] as follows:

$$\frac{1}{2}\begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}. \quad (34)$$

The exact circuit designs for the unitary matrix of the sender part of the quantum teleportation are shown in Fig.7.

E. The Hamiltonian of hydrogen molecule

It has been shown that the ground and excited state energies of small molecules can be carried out on a quantum computer simulator using a recursive phase-estimation algorithm [30, 31]. Lanyon et. al. reported the application of photonic quantum computer technology to calculate properties of the hydrogen molecule in a minimal basis [32]. Here, we show how our method can be used to reduced the number of gates needed to perform the simulations.

Fermion model of quantum computation is defined through the spinless fermionic annihilation (a_j) and creation ($a_j^\dagger$) operators for each

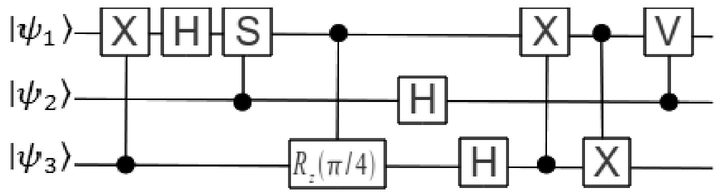

FIG. 6: Found circuit design for the three-qubit quantum Fourier transform.

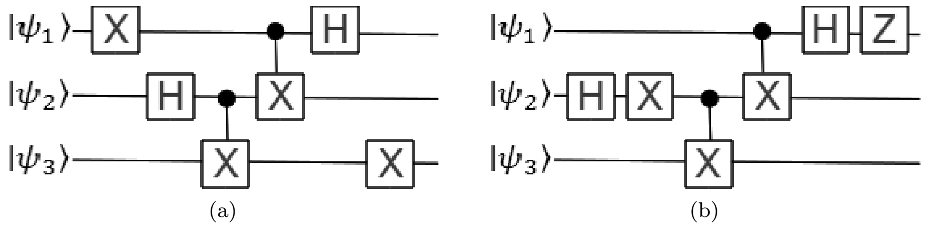

FIG. 7: The found circuit designs for the sender part of the quantum telportation.

qubit j (j=1, ... , n), where the algebra of 2n elements obey the fermionic anti-commutation rules [33]:

$$\{a_i, a_j\} = 0,\ \{a_i, a_j^\dagger\} = \delta_{ij}, \tag{35}$$

where $\{A, B\} = AB + BA$ defines the anti-commutator. Using the Jordan-Wigner transformation, the fermion operators is mapped to the standard quantum computation operators through the Pauli spin operators $(\sigma_x, \sigma_y, \sigma_z, I)$ [33]:

$$\begin{aligned} a_j &\to \left(\prod_{k=1}^{j-1} -\sigma_z^k\right)\sigma_-^j = (-1)^{j-1}\,\sigma_z^1\sigma_z^2....\sigma_z^{j-1}\sigma_-^j \\ a_j^\dagger &\to \left(\prod_{k=1}^{j-1} -\sigma_z^k\right)\sigma_+^j = (-1)^{j-1}\,\sigma_z^1\sigma_z^2....\sigma_z^{j-1}\sigma_+^j. \end{aligned} \tag{36}$$

Once the electronic Hamiltonian is defined in second quantized form, the state space can easily be mapped to qubits. The electronic Hamiltonian in second quantized form is described as [32, 34, 35]:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2}\sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s, \tag{37}$$

where the integrals h_{pq} and h_{pqrs} evaluated during the Hartree-Fock procedure are defined as follows:

$$h_{pq} = \int dx \chi_p^*(x)\left(-\frac{1}{2}\nabla^2 - \sum_\alpha \frac{Z_\alpha}{r_{\alpha x}}\right)\chi_q(x) \tag{38}$$

and

$$h_{pqrs} = \int dx_1 dx_2 \frac{\chi_p^*(x_1)\chi_q^*(x_2)\chi_r(x_2)\chi_s(x_1)}{r_{12}}, \tag{39}$$

where $r_{\alpha x}$ is the distance between the α^{th} nucleus and the electron, r_{12} is the distance between electrons, ∇^2 is the Laplacian of the electron spatial coordinates, and $\chi_p(x)$ is a selected single-particle basis. Whitfield et al.[34] considered H in Eq.(37) as $H^{(1)} + H^{(2)}$. Since $h_{pqrs} = h_{qprs}$, and so $h_{ijji} = h_{jiij} = -h_{ijij} = -h_{jiji}$, they noted the parts of the Hamiltonian as follows:

$$H^{(1)} = h_{11}a_1^\dagger a_1 + h_{22}a_2^\dagger a_2 + h_{33}a_3^\dagger a_3 + h_{44}a_4^\dagger a_4, \tag{40}$$

and

$$\begin{aligned}H^{(2)} = {} & h_{1221}a_1^\dagger a_2^\dagger a_2 a_1 + h_{3443}a_3^\dagger a_4^\dagger a_4 a_3 + h_{1441}a_1^\dagger a_4^\dagger a_4 a_1 + h_{2332}a_2^\dagger a_3^\dagger a_3 a_2 \\ & + (h_{1331} - h_{1313})a_1^\dagger a_3^\dagger a_3 a_1 + (h_{2442} - h_{2424})a_2^\dagger a_4^\dagger a_4 a_2 \\ & + Re(h_{1423})(a_1^\dagger a_4^\dagger a_2 a_3 + a_3^\dagger a_2^\dagger a_4 a_1) + Re(h_{1243})(a_1^\dagger a_2^\dagger a_4 a_3 + a_3^\dagger a_4^\dagger a_2 a_1) \\ & + Im(h_{1423})(a_1^\dagger a_4^\dagger a_2 a_3 + a_3^\dagger a_2^\dagger a_4 a_1) + Im(h_{1243})(a_1^\dagger a_2^\dagger a_4 a_3 + a_3^\dagger a_4^\dagger a_2 a_1).\end{aligned} \tag{41}$$

Using their findings [34] for the spatial integral values for atomic distance $1.401a.u.$ in Table III, we found the Hamiltonian, and then the unitary propagator, e^{-iHt} (t was taken as 1), as a matrix operator. And then we ran the GLOA with the parameter values in Table IV and Table V for this unitary matrix, and decomposed it as shown in Fig.8 with related objective function values and the correctness in Table V.

In addition to the exact unitary propagator, we also found the unitary operator at bond distance 1.401 atomic units by simulating the approximated-general circuit designs for the simulation of the Hamiltonian in the paper [34] (In [34], the non-commuting terms in the Hamiltonian are approximated by following the Trotter-Suzuki decomposition, and the given approximated circuit design is a general circuit for the unitary propagator; the values of the angles in the quantum gates depend on the spatial integral values.). The decomposition of this unitary propagator is found as in Fig.9 with the value of correctness 0.998. The circuit design in Fig.9 consists of 6 quantum gates while the complete circuit design in [34] consists of 87 quantum gates.

VIII. CONCLUSION

To be able to simulate Hamiltonians of atomic and molecular systems and also apply quantum algorithms to solve different kinds of problems on quantum computers, it is necessary to find implementable quantum circuit designs including the minimum cost and number of quantum gate sequences. Since deterministic-efficient quantum circuit design methodology is an open problem, we applied stochastic evolutionary optimization algorithm, GLOA, to search quantum circuit designs for given unitary matrices representing algorithms or the unitary propagator of a molecular Hamiltonian. In this paper, in addition to explaining the ways of the implementation and design of the optimization problem, we present two algorithmic methods with which to find the unitary matrix representation of different type of quantum control gates, and then we give some efficient circuit designs for the Grover search algorithm, the Toffoli gate, the quantum Fourier transform, and the quantum teleportation. Moreover, we give the two circuit designs for the simulation of the Hamiltonian of the hydrogen molecule by decomposing the unitary matrix operators found by following the fermionic model of quantum computation and simulating the circuits given in [34].

IX. ACKNOWLEDGMENTS

We would like to thank the Army Research Office (ARO) and the National Science Foundation (NSF) for financial support of this project.

BIBLIOGRAPHY

[1] C. P. Williams and E. G. Gray (Springer, 1999) pp. 113–125.

[2] T. Yabuki and H. Iba, in *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference* (Morgan Kauffman Publishers, 2000) pp. 421–425.

[3] F. Peng, G. jun Xie, and T. hao Wu, Convergence Information Technology, International Conference on **0**, 70 (2009).

[4] L. Spector, *Automatic Quantum Computer Programming: A Genetic Programming Approach (Genetic Programming)* (Springer-

Verlag New York, Inc., Secaucus, NJ, USA, 2006) ISBN 038736496X.

[5] R. Stadelhofer, W. Banzhaf, and D. Suter, AI EDAM **22**, 285 (2008).

[6] A. Leier, *Evolution of Quantum Algorithms using Genetic Programming*, Ph.D. thesis, Dortmund University, Germany (jul # 21 2004).

[7] M. Lukac and M. Perkowski, in *EH '02: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02)* (IEEE Computer Society, Washington, DC, USA, 2002) p. 177, ISBN 0-7695-1718-8.

[8] P. Massey, J. A. Clark, and S. Stepney, in *GECCO (2)* (2004) pp. 569–580.

[9] A. Gepp and P. Stocks, Genetic Programming and Evolvable Machines **10**, 181 (2009).

[10] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, Inc., New York, NY, USA, 2007) ISBN 0198570007.

[11] D. P. DiVincenzo, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 261 (1998).

[12] R. Kosloff, The Journal of Physical Chemistry **92**, 2087 (1988).

[13] D. P. DiVincenzo, Phys. Rev. A **51**, 1015 (Feb 1995).

[14] L. Arnaud and D. Braun, *Efficiency of Producing Random Unitary Matrices with Quantum Circuits*, Tech. Rep. arXiv:0807.0775 (2008).

[15] M. Nakahara and T. Ohmi, *Quantum Computing: From Linear Algebra to Physical Realizations* (CRC Press, Boca Raton, FL, 2008).

[16] S. Ding, Z. Jin, and Q. Yang, Soft Comput. **12**, 1059 (2008).

[17] J. F. Miller and S. L. Harding, in *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation* (ACM, New York, NY, USA, 2008) pp. 2701–2726, ISBN 978-1-60558-131-6.

[18] T. Reid, *On the evolutionary design of quantum circuits*, Master's thesis, Waterloo University, Ontario, Canada (2005).

[19] A. Daskin and S. Kais(2010), arXiv/1004.2242.

[20] P. Serra, A. F. Stanton, and S. Kais, Phys. Rev. E **55**, 1162 (Jan 1997).

[21] P. Nigra and S. Kais, Chemical Physics Letters **305**, 433 (1999).

[22] P. Serra, A. F. Stanton, S. Kais, and R. E. Bleil, The Journal of Chemical Physics **106**, 7170 (1997).

[23] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (Nov 1995).

[24] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 1st ed. (Cambridge University Press, 2000) ISBN 0521635039.

[25] L. K. Grover, in *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 1996) pp. 212–219, ISBN 0-89791-785-5.

[26] J. Zhu, Z. Huang, and S. Kais, Molecular Physics: An International Journal at the Interface Between Chemistry and Physics **107**, 2015 (2009).

[27] T. Beth and G. Leuchs, *Quantum Information Processing* (John Wiley & Sons, 2005) ISBN 3527405410.

[28] G. Brassard, S. L. Braunstein, and R. Cleve, in *PhysComp96: Proceedings of the fourth workshop on Physics and computation* (Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 1998) pp. 43–47, ISBN 0167-2789.

[29] S. Kais, Advances in Chemical Physics **134**, 493 (2007).

[30] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, Science **309**, 1704 (2005).

[31] H. Wang, S. Kais, A. Aspuru-Guzik, and M. R. Hoffmann, Phys. Chem. Chem. Phys. **10**, 5388 (September 2008).

[32] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, Nature Chemistry **2**, 106 (January 2010).

[33] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, Physical Review A **64**, 022319+ (Jul 2001).

[34] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, *Quantum Computing Resource Estimate of Molecular Energy Simulation*, Tech. Rep. arXiv:1001.3855 (2010).

[35] E. Ovrum and M. Hjorth-Jensen, *Quantum computation algorithm for many-body studies*, Tech. Rep. arXiv:0705.1928 (2007).

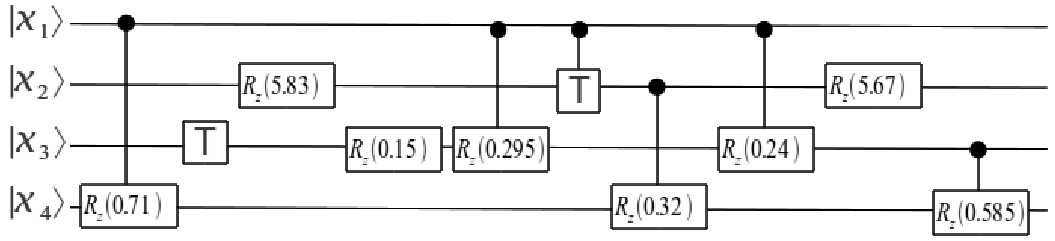

FIG. 8: The circuit design for the unitary propagator of the Hamiltonian of hydrogen molecule. The unitary propagator is found by using the spatial integral values in Table III and the definitions for the annihilation and creation operators in Eq.(36) into the Eq.(37), Eq.(40), and Eq.(41).

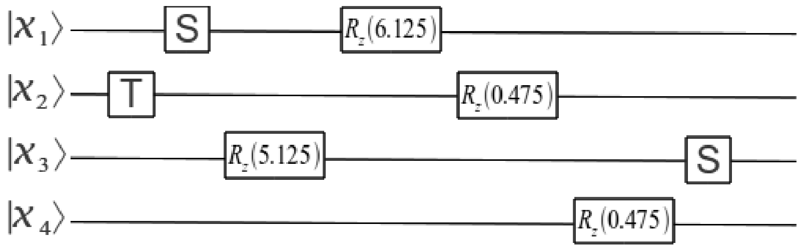

FIG. 9: The circuit design for the unitary propagator of the hydrogen Hamiltonian. The unitary propagator is found by simulating the circuits and pseudo-code given in [34].

TABLE I: The unitary matrices for some of the quantum gates used in the circuits [10, 23, 24]

The name of the gate	The unitary matrix
V (square root of X gate)	$\frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$
$V^\dagger$	$\frac{1}{2}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix}$
$R_x(\theta)$	$\begin{pmatrix} \cos(\frac{\theta}{2}) & i\sin(\frac{\theta}{2}) \\ i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$
$R_y(\theta)$	$\begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ -sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$
$R_z(\theta)$	$\begin{pmatrix} 1 & 0 \\ 0 & exp(i\theta) \end{pmatrix}$

.

TABLE II: Universal set of quantum gates used in the optimization

Single Gates	Control Gates
Single I	—
Single X	Control X
Single H	—
Single Z	Control Z
Single Y	Control Y
Single V	Control V
Single $V^\dagger$	Control $V^\dagger$
Single S	Control S
Single T	Control T
Single RX	Control RX
Single RY	Control RY
Single RZ	Control RZ

*Rotation gates are used only for three qubit QFT and the Hamiltonian of H_2

TABLE III: Spatial Integral Values[34]

Spatial Integral	Value
h_{11}, h_{22}	-1.25247729802
h_{33}, h_{44}	-0.475934479839
h_{1221}, h_{2112}	0.674493103326
h_{1331}, h_{1441}, h_{2332}, h_{2442}, h_{3113}, h_{3223}, h_{4114}, h_{4224}	0.663472044861
h_{3443}, h_{4334}	0.69739794982
h_{2424}, h_{3241}, h_{1423}, h_{1243}	0.181287535812

TABLE IV: The values of the parameters within the group leaders optimization algorithm

Parameter name	Number of groups	Number of population in each group	r_1	r_2	r_3	Number of transfer for each group
Value	15	25	0.8	0.1	0.1	$\frac{\text{Number of Variables}^\dagger}{2} - 1$

†Since each gate has four variables, the total number of variables is equal to four times maximum number of gates.

TABLE V: Optimization parameters and results for the test problems

The unitary matrix	Number of qubits	Maximum number of gates	Number of iteration	Objective function value	Correctness (C)
Toffoli Gate, Fig.3a	3	8	500	0.09167	1
Toffoli Gate, Fig.3b	3	8	500	0.09286	1
Toffoli Gate, Fig.3c	3	8	500	0.09286	1
Sender part of teleportation, Fig.7a	3	8	2500	0.08752	1
Sender part of teleportation, Fig.7b	3	8	2500	0.08752	1
Diffusion Operator of GSA, Fig.4a	2	8	500	0.08335	1
Diffusion Operator of GS, Fig.4b	2	8	500	0.08571	1
Two-qubit QFT, Fig.5a	2	8	500	0.08752	1
Two-qubit QFT, Fig.5b	2	8	500	0.08752	1
Three-qubit QFT, Fig.6	3	12	2500	0.09565	1
Simulation of the H_2 Hamiltonian, Fig.8	4	12	10000	0.12819	0.965
Simulation of the H_2 Hamiltonian, Fig.9	4	12	10000	0.09939	0.985

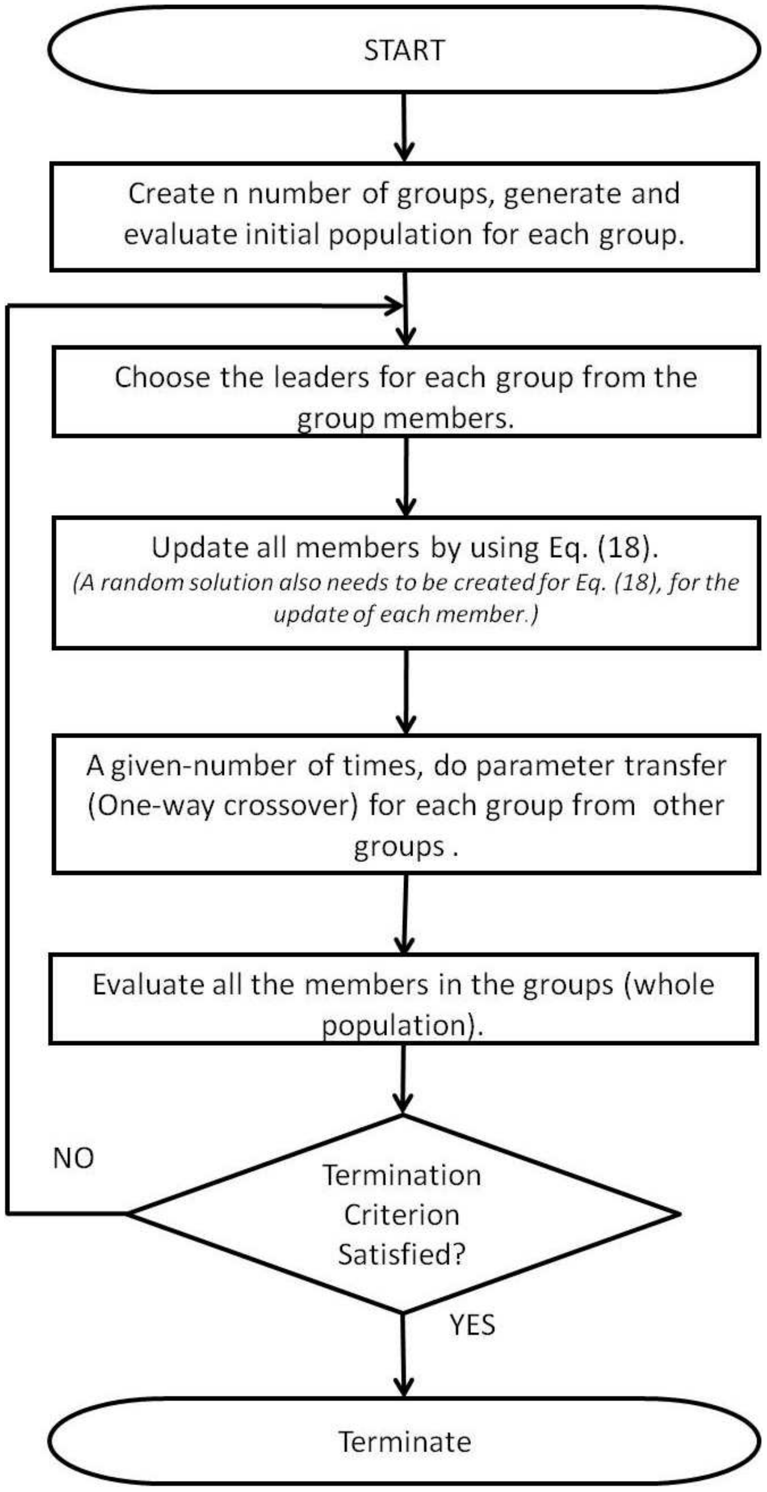
START
Create n number of groups, generate and evaluate initial population for each group.
Choose the leaders for each group from the group members.
Update all members by using Eq. (18).
(A random solution also needs to be created for Eq. (18), for the update of each member.)
A given-number of times, do parameter transfer (One-way crossover) for each group from other groups .
Evaluate all the members in the groups (whole population).
Termination Criterion Satisfied?
NO
YES
Terminate

FIG. 10: The flow chart of the group leaders optimization algorithm