Quantum Algorithms for Bio-molecular Solutions to the Satisfiability Problem on a Quantum Computer

Weng-Long Chang1, Ting-Ting Ren2, Jun Luo3, Mang Feng4 and Minyi Guo5

Abstract—We demonstrate that the logic computation performed by the DNA-based algorithm for solving general cases of the satisfiability problem can be implemented by our proposed quantum algorithm on the quantum machine proposed by Deutsch. Moreover, we also prove that the logic computation by the bio-molecular operations proposed by Adleman can be implemented by quantum gates (for example, the Hadamard gate, NOT, CNOT, and CCNOT) on the quantum machine. Furthermore, those NP-complete problems solved on a bio-molecular computer are also solvable on a quantum computer. To test our theory, we carry out a three-qubit NMR experiment for solving the simplest satisfiability problem.

Key Words—Quantum Algorithms, Quantum Circuits, Nuclear Magnetic Resonance, Molecular Algorithms.

I. INTRODUCTION

THE molecular computation was first proposed by Adleman in 1994 when he successfully solved an instance of the Hamiltonian path problem in a test tube by DNA strands [1]. On the other hand, Deutsch [2] has proposed a general model of quantum computation — the quantum Turing machine. The rest of the paper is organized as follows: Section II gives motivation for developing quantum algorithm of solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables. In Section III we will briefly review the development of molecular computing and quantum computing. Section IV is for our quantum algorithm solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables. In Section V we demonstrate that the bio-molecular operations proposed by Adleman [1] can be implemented on the quantum machine proposed by Deutsch [2], and those NP-complete problems solvable on a bio-molecular computer can be fully solved on a quantum computer. The time complexity and the space complexity of our quantum algorithm are analyzed in Section VI for solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables. Based on Cook’s theorem, the relation between the class P and class NP is described in Section VII. Section VIII is for the nuclear magnetic resonance (NMR) experiment to check our theory. We give the simplest example of three quantum bits for the satisfiability problem, and an experiment by NMR is carried out. We conclude with a brief discussion in Section IX.

II. MOTIVATION

In the DNA-based algorithm for the satisfiability problem, 2\(^n\) DNA strands are used to represent 2\(^n\) combinational states of \( n \) bits in computational subspace. So if the value of \( n \) is greater than 512, real bio-molecular experiments become very challenging. However, 2\(^n\) possible solutions for the satisfiability problem can be represented by means of arbitrary superposition of \( n \) quantum bits. This implies that if the corresponding relations between a bio-molecular computer [1] and a quantum computer [2] can be set up, we may transcribe the DNA-based algorithms to the corresponding quantum algorithms and work them out with reduced difficulty on the quantum computer.

III. BRIEF REVIEW OF MOLECULAR COMPUTING AND QUANTUM COMPUTING

The optimal solution of every NP-complete is figured out from its characteristic [3]. Potentially significant areas of application for DNA algorithms are the breaking of encryption schemes and solving the NP-Complete problems [4-8]. On the other hand, the Deutsch-Jozsa algorithm [9] and Coppersmith’s quantum algorithm for the fast Fourier transform [10] were proposed. So far, the most frequently cited quantum algorithms are Shor’s algorithms for solving factoring integers and discrete logarithm [11] and Grover’s search algorithm [12] for unsorted databases. Afterward, strengths and weaknesses of the quantum Turing machine for solving NP-complete problems were discussed [13].

IV. QUANTUM ALGORITHMS FOR BIO-MOLECULAR SOLUTIONS OF THE SATISFIABILITY PROBLEM

In this section, we will demonstrate that the logic computation performed by Lipton’s DNA-based algorithm [6] for solving the satisfiability problem can be implemented by our proposed quantum algorithms on a quantum machine.
A. THE DEFINITION OF THE SATISFIABILITY PROBLEM

A clause is a formula that is of the form \( u_k \lor u_{k-1} \ldots \lor u_2 \lor u_1 \), where each \( u_i \) for \( 1 \leq k \leq n \) is a Boolean variable or its negation. Definition 4-1 from [6] is introduced to denote the satisfiability problem.

**Definition 4-1:** In general, a satisfiability problem includes a Boolean formula of the form \( C_1 \land C_2 \ldots \land C_m \) where each \( C_j \) for \( 1 \leq j \leq m \) is a clause. Then, the question is to find values of the variables so that the whole formula has the value 1. This is equal to finding values to the variables that make each clause have the value 1.

B. ALL OF THE POSSIBLE SOLUTIONS TO THE SATISFIABILITY PROBLEM

Assume that \( U \) is a set of 2^n possible choices and equal to \( \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \), which implies that the length of each element in \( U \) is \( n \) bits and each element represents one of 2^n combinational states for \( n \) Boolean variables. For the sake of presentation, we suppose that \( u_k^0 \) is used to denote the value of \( u_k \) to be zero and \( u_k^1 \) means value of \( u_k \) to be one. The following definition will show how each element in \( U \) is represented to be a unique computational state vector with 2^n-tuples of binary numbers.

**Definition 4-2:** The \( j \)th element in \( U \) can be represented as a unique computational state vector \( \{ u_1^j \} \) where \( u_{1,t} \), \( u_{1,t+1} \ldots u_{1,2^{n-1}} \), \( u_{1,2^n}^T \), where \( u_{1,t} = 1 \), \( u_{j,t} = 1 \), \( \forall u_{1,h} = 0 \) and \( \forall u_{h+1} = 0 \) for \( 1 \leq h < j \leq 2^n \).

The corresponding computational state vector for the first element, \( u_0^0, u_{n-1}^0, \ldots u_2^0, u_1^0 \), in \( U \) is \( \{ 1, 0, \ldots, 0 \} \), and the corresponding computational state vector for the last element, \( u_0^1, u_{n-1}^1, \ldots u_2^1, u_1^1 \), in \( U \) is \( \{ 0, 0, \ldots, 1 \} \). For the sake of presentation, we assume that \( B \) is a set of the corresponding computational state vectors to the elements in \( U \) and \( B = \{ \{ 1, 0, \ldots, 0 \} \ldots \{ 0, 0, \ldots, 1 \} \} \). Because each component in \( B \) is a coordinated vector, we span \( B = C_{2^n} \), where \( C_{2^n} \) is a Hilbert space. This implies that each vector in the set \( B \) is an orthonormal basis in a Hilbert space.

C. COMPUTATIONAL SPACE OF MOLECULES FOR THE SATISFIABILITY PROBLEM

The following bio-molecular operations cited from [1] will be used to construct computational space of molecules for solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables.

**Definition 4-3:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \) and a Boolean variable \( u_t \), the bio-molecular operation, “Append-Head”, appends \( u_t \) onto the head of every element in the set \( U \). The formal representation is written as

\[
\text{Append-Head}(U, u_t) = \{ u_t, u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \text{ for } 1 \leq k \leq n \text{ and } u_t \in \{ 0, 1 \}.
\]

**Definition 4-4:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \) and a Boolean variable \( u_t \), the bio-molecular operation, “Append-Tail”, appends \( u_t \) onto the end of every element in the set \( U \). The formal representation is written as

\[
\text{Append-Tail}(U, u_t) = \{ u_n, u_{n-1}, \ldots u_2, u_1, u_t \} \forall u_i \in \{ 0, 1 \} \text{ for } 1 \leq k \leq n \text{ and } u_t \in \{ 0, 1 \}.
\]

**Definition 4-5:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \), the bio-molecular operation, “Discard(\( U \))” creates a number of identical copies, \( U \), of the set \( U \) and then discard(\( U \)).

**Definition 4-6:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \) and a Boolean variable \( u_t \), if the value of \( u_t \) is equal to one, then the bio-molecular extract operation creates two new sets, \( \ast(U, u_t) = \{ u_n, u_{n-1}, \ldots u_t, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \neq j \leq n \). Otherwise, it produces another two new sets, \( \ast(U, u_t) = \{ u_n, u_{n-1}, \ldots u_t, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \neq j \leq n \).

**Definition 4-7:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \) and a Boolean variable \( u_t \), if the value of \( u_t \) is equal to one, then the bio-molecular operation “Detect(\( U \))” returns true if \( U \neq \emptyset \). Otherwise, it returns false.

**Definition 4-8:** Given \( m \) sets \( U_1 \ldots U_m \), the bio-molecular merge operation, \( \cup(U_1, U_2) = U_1 \cup U_2 \ldots \cup U_m \).

**Definition 4-9:** Given a set \( U = \{ u_n, u_{n-1}, \ldots u_2, u_1 \} \forall u_i \in \{ 0, 1 \} \) for \( 1 \leq k \leq n \), the bio-molecular operation “Read(\( U \))” performs an arbitrary element in \( U \). Even if \( U \) contains many different elements, the bio-molecular operation can give an explicit description of exactly one of them.

For solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables, the following bio-molecular algorithm can be used to create all of the 2^n possible choices. A set \( U \) is an empty set and is regarded as the input set of the DNA-based algorithm. The second parameter \( n \) in **CombinationStates** (\( U, n \)) is to represent the number of Boolean variables.

**Procedure CombinationStates(U, n)**

(0a) Append-Tail(\( U_1, u_n^0 \)).
(0b) Append-Tail(\( U_2, u_n^0 \)).
(0c) \( U = \cup(U_1, U_2) \).

(1) For \( k = n - 1 \) downto 1

(1a) Amplify(\( U, U_1, U_2 \)).
(1b) Append-Tail(\( U_1, u_n^1 \)).
(1c) Append-Tail(\( U_2, u_n^1 \)).

(1d) \( U = \cup(U_1, U_2) \).

**End For**

**End Procedure**

**Lemma 4-1:** For solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables, 2^n possible choices created from the DNA-based algorithm, **CombinationStates(U, n)**, form an orthonormal basis of a Hilbert space (i.e., a complex vector space, \( C_{2^n} \)).
D. COMPUTATIONAL SPACE OF QUANTUM MECHANICAL SOLUTION FOR THE SATISFIABILITY PROBLEM

A qubit (quantum bit) has two ‘computational basis vectors’ \(|0\rangle\) and \(|1\rangle\) of the two-dimensional Hilbert space corresponding to the classical bit values 0 and 1 [14], and an arbitrary state \(|\varphi\rangle\) of a qubit is a linearly weighted combination of the computational basis vectors (4.1): \(|\varphi\rangle = l_1 \cdot |0\rangle + l_2 \cdot |1\rangle\), where the weighted factors \(l_1\) and \(l_2\) \(\in \mathbb{C}\) are the so-called probability amplitudes, with \(|l_1|^2 + |l_2|^2 = 1\). A collection of \(n\) qubits is called a qregister (quantum register) of size \(n\). It may include any of the \(2^n\)-dimensional computational basis vectors, \(n\) qubits of size, or arbitrary superposition of these vectors [14].

E. LIPTON’S DNA-BASED ALGORITHMS OF SOLVING THE SATISFIABILITY PROBLEM

Although Lipton did not explicitly present his operation set in [6], his solution may be phrased in terms of the eight operations defined in Subsection C. Lipton’s DNA-based algorithm for solving the satisfiability problem denoted in Definition 4–1 is described below. The symbol \(|C|\) in the following algorithm is introduced to represent the number of Boolean variables and their negations in the \(j\)th clause in a formula.

Algorithm 4–1: Lipton’s DNA-based algorithm for solving the satisfiability problem.
1. CombinationalStates\((U, n)\)
2. For \(j = 1\) to \(m\) do begin
3. For \(i = 1\) to \(|C|\) do begin
4. If the \(i\)th element in the \(j\)th clause is one of \(n\) Boolean variables \(u_i\), Then
5. \(U_i \leftarrow \oplus(U, u_i)\) and \(U \leftarrow (U, u_i)\)
6. Else
7. \(U_i \leftarrow \oplus(U, u_i)\) and \(U \leftarrow (U, u_i)\)
8. End If
9. End For
10. Discard\((U)\)
11. For \(i = 1\) to \(|C|\) do begin
12. \(U = \oplus(U, U)\)
13. End For
14. End For
15. If (Detect\((U) = \text{true}\)) Then
16. (15a) Read\((U)\)
End If
17. End Algorithm

Lemma 4–2: Algorithm 4–1. Lipton’s DNA-based algorithm, can be applied to the satisfiability problem with \(m\) clauses and \(n\) Boolean variables.

F. INTRODUCTION OF QUANTUM GATES FOR SOLVING THE SATISFIABILITY PROBLEM

The time evolution of the states of quantum registers can be modeled by means of unitary operators which are often referred to as quantum gates [14]. Therefore, a quantum gate can be regarded as an elementary quantum-computing device which performs a fixed unitary operation on selected qubits during a fixed period of time. One-qubit and two-qubit quantum gates are elementary quantum gates. The \textit{NOT} gate is a one-qubit gate to negate the only (target) bit [14]. The \textit{CNOT} (controlled \textit{NOT}) gate is a two-qubit gate to flip the second qubit (the target qubit) if and only if the first qubit (the control qubit) is one [14]. The \textit{CCNOT} (controlled-controlled-\textit{NOT}) gates is a three-qubit gate to flip the third qubit (the target qubit) if and only if the first qubit and second qubit (the two control qubits) are both one [14].

G. CONSTRUCTING QUANTUM NETWORKS FOR SOLVING THE SATISFIABILITY PROBLEM

The operations \textit{OR} and \textit{AND} are implemented by quantum circuits in Figures 4–1 and 4–2, respectively. For evaluating a clause with the form \(u_n \lor u_{n-1} \cdots \lor u_2 \lor u_1\), three quantum registers \([u_n \cdots u_1], [y_n \cdots y_1]\) and \([r_n \cdots r_1]\) are needed. Therefore, its evaluating computation is equal to

\[
(r_n \cdots r_1) \
(\begin{array}{c}
|y_n \cdots y_1\rangle \\
|u_n \cdots u_1\rangle
\end{array}) \
(\begin{array}{c}
[r_n \cdots r_1]\rangle \\
[u_n \cdots u_1]\rangle
\end{array})
\]

where \(\oplus\) denotes operation \textit{AND} of their negations of two Boolean variables \([\overline{y}_k, \overline{r}_k]\) for \(1 \leq k \leq n\). The first bit, \(r_0\), in the third quantum register is initially prepared in state \(|0\rangle\), and other \(n\) bits in the third quantum register are initially prepared in state \(|1\rangle\). The \((n + 1)\)th quantum bit, \(r_n\), in the third register is employed to store the final result of the evaluating computation.

Then, in order to evaluate \textit{AND} operation of the previous clause and the current clause, the fourth quantum register \([c_m c_{m-1} \cdots c_1 c_0]\) is needed. The first bit, \(c_0\), in the fourth quantum register is initially prepared in state \(|0\rangle\), and other \(m\) bits in the quantum register are initially in state \(|1\rangle\), and other \(m\) bits in the quantum register are initially in state \(|0\rangle\). The \((m + 1)\)th quantum bit, \(c_m\), in the fourth register is employed to store the final result of evaluating computation for all of the clauses. The full network, QEC (the abbreviation of quantum evaluating circuit for checking whether the current clause is true or not), is illustrated in Figure 4–3 in Appendix A and can be understood as follows:

- We compute the \((n + 1)\)th bit of the third quantum register to the final result of evaluating a clause. This step requires computing all the \textit{OR} operations through the relation \(r_k \leftarrow r_k \oplus (\overline{y}_k \cdot \overline{r}_k)\) for \(1 \leq k \leq n\). Then, we compute \textit{AND} operation of the previous clause and the
current clause through the relation $c_j \leftarrow c_j \oplus (c_{j-1} \bullet r_j)$ for $1 \leq j \leq m$.

- Subsequently we reverse all those OR operations in order to restore every quantum bit of each quantum register to its initial state. This enables us to reuse the same quantum registers, should the problem, for example, require repeated OR operation.

Subsequently, we reverse all those operations (NOT or CNOT) on the second quantum register to restore every quantum bit of the second quantum register to its initial state. This enables us to use the second quantum register repeatedly.

![Figure 4-1: OR operation of two Boolean variables.](image)

![Figure 4-2: AND operation of two Boolean variables.](image)

H. QUANTUM ALGORITHMS OF LIPTON’S DNA-BASED ALGORITHMS FOR SOLVING THE SATISFIABILITY PROBLEM

Based on Algorithm 4-1 in Subsection E, the following quantum algorithm is proposed to work on the physical quantum computer proposed by Deutsch [2]. For convenience of our following presentation, we suppose that $c_j^n$ denotes the value of $c_j$ to be 1 and $c_j^0$ defines the value of $c_j$ to be 0 for $1 \leq j \leq m$. We also assume that $r_j^1$ denotes the value of $r_j$ to be 1 and $r_j^0$ defines the value of $r_j$ to be 0 for $0 \leq k \leq n$. Similarly, $y_k^1$ denotes the value of $y_k$ to be 1 and $y_k^0$ defines the value of $y_k$ to be 0. Moreover, the notations used in Algorithm 4-2 below have been denoted in previous subsections.

Algorithm 4-2: Quantum algorithms of Lipton’s DNA-based algorithm for solving the satisfiability problem with $m$ clauses and $n$ Boolean variables.

1. For an input $|\Phi\rangle = (\bigotimes_{j=m}^{1} |c_j^0\rangle) \otimes (\bigotimes_{k=m}^{1} |c_k^0\rangle) \otimes (\bigotimes_{q=n}^{1} |r_q^1\rangle) \otimes (\bigotimes_{q=n}^{1} |r_q^0\rangle)$

2. $|\varphi_{1,0}\rangle = (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (I_{2\times 2}) \otimes (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (|\Phi\rangle)$

3. $|\varphi_{j,1-i}\rangle = \frac{1}{\sqrt{2^n}} (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle), \quad \text{where}$

4. $c_j = c_j \oplus (c_{j-1} \bullet r_j)$ for $j \leq 1 \leq s \leq 1$.

5. Else

6. $|\varphi_{j,1}\rangle = (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (I_{2\times 2}) \otimes (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (I_{2\times 2}) \otimes (|\Phi\rangle) \otimes (|\Phi\rangle)$

7. $|\varphi_{j,1-i}\rangle = \frac{1}{\sqrt{2^n}} (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (\bigotimes_{j=m}^{1} I_{2\times 2}) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle) \otimes (|y_k^0 \oplus (u_k^0 + u_k^1)\rangle), \quad \text{where}$

8. End If

2008 IEEE Congress on Evolutionary Computation (CEC 2008)
(9) End For

(10) \[ \varphi_{j+1,1} = \text{QEC} \times (\varphi_{j,n} I_{2n-2}^{q=0}) \times (\varphi_{j,n} I_{2n-2}^{q=1}) \times \left( c_j \oplus (c_{j+1} \cdot r_n) \right) \]

(11) \[ \varphi_{j+1,0} = (\varphi_{j,n} I_{2n-2}^{q=0}) \times (\varphi_{j,n} I_{2n-2}^{q=1}) \times \left( c_j \oplus (c_{j+1} \cdot r_n) \right) \]

Y_q = \begin{cases} 
Y_0^q + Y_1^q & \text{if it is a Boolean variable} \\
Y_0^q + Y_1^q & \text{if it is the negation of a Boolean variable} 
\end{cases}

Lemma 5-1: For any NP-complete problem, 2^n possible solutions on the bio-molecular computer proposed by Adleman [1] can be implemented on the quantum machine proposed by Deutsch [2].

Lemma 5-2: The bio-molecular operation, Append-Head(U, V) = {u_1 u_2 u_3 ... u_n} \forall u_k \in \{0, 1\} for 1 \leq k \leq n and u_n \in \{0, 1\}, denoted in Definition 4-3, can be implemented by the corresponding quantum operation, \( |\varphi\rangle = |0\rangle \otimes (H^n |000...0\rangle) \otimes (H^{n-1} |1\rangle) \).

Lemma 5-3: The bio-molecular operation, Append-Tail(U, V) = {u_1 u_2 u_3 ... u_n} \forall u_k \in \{0, 1\} for 1 \leq k \leq n and u_n \in \{0, 1\}, denoted in Definition 4-4, can be implemented by \( |\varphi\rangle = (H^n |000...0\rangle) \otimes |0\rangle \).

Lemma 5-4: The bio-molecular extract operation, + (U, u_k) = {u_1 u_2 u_3 ... u_k u_{k+1}} \forall u_k \in \{0, 1\} for 1 \leq k \leq n and \( |\varphi\rangle = (H^n |000...0\rangle) \otimes |0\rangle \).

Lemma 5-5: The bio-molecular merge operation and discard operation, i.e., \( \cup(U_1, U_2) = U_1 \cup \cdots \cup U_m \) and Discard(W) = \( \emptyset \), perform logic computation of the satisfiability problem that can be implemented by NOT, CNOT, CCNOT.

Lemma 5-6: Assume that U = {u_1 u_2 u_3 ... u_n} \forall u_k \in \{0, 1\} for 1 \leq k \leq n and W is a subset of U and U_k is also a subset of U for 1 \leq k \leq m. From Definition 4-5 and Definition 4-8, the bio-molecular merge operation and discard operation, i.e., \( \cup(U_1, U_2) = U_1 \cup \cdots \cup U_m \) and Discard(W) = \( \emptyset \), perform logic computation of the satisfiability problem that can be implemented by NOT, CNOT, CCNOT.

Lemma 5-7: Assume that U = {u_1 u_2 u_3 ... u_n} \forall u_k \in \{0, 1\} for 1 \leq k \leq n. From Definition 4-9 and Definition 4-10, the bio-molecular detect operation and read operation, i.e.,
Theorem 5–1: The bio-molecular computer proposed by Adleman [1] is a subset of the quantum machine proposed by Deutsch [2] and can be implemented on a quantum computer.

Theorem 5–2: For those famous NP-complete problems that had been solved on a bio-molecular computer, their corresponding DNA-based algorithms can be fully implemented on a quantum machine.

VI. COMPLEXITY ASSESSMENT

From [6] and Algorithm 4–1, we know that the satisfiability problem with \( m \) clauses and \( n \) Boolean variables is solvable by \( O(m \times n) \) biological operations, where the number of DNA sequences in computational space is \( O(2^n) \), the longest DNA sequence in computational space is \( O(n) \), and the number of tubes is \( O(1) \). The following lemmas could be used to show the time complexity and space complexity of Algorithm 4–2 for solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables.

Lemma 6–1: Time complexity of solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables is \( O(n) \) Hadamard gates, \( O(6 \times m \times n) \) CNOT gates, \( O(2 \times m \times n) \) CNOT gates, \( O(m \times n + m) \) CCNOT gates, and \( O(1) \) projective operators.

Lemma 6–2: Space complexity of solving the satisfiability problem with \( m \) clauses and \( n \) Boolean variables is \( O(m + 3 \times n + 2) \) quantum bits.

VII. APPLICATION OF COOK’S THEOREM

The classes \( P \) (the abbreviation of deterministic Polynomial) and NP (the abbreviation of Non-deterministic Polynomial) of problems are solvable in polynomial time by deterministic and non-deterministic Turing machines, respectively, and by the technique of polynomial-time reduction [15]. The following definition is applied to the definition of the class NP-complete of problems.

Definition 7–1: A problem \( L \) is called NP-complete if (a) \( L \in \text{NP} \); and (b) for every problem \( L' \in \text{NP} \), there is a polynomial-time reduction from \( L' \) to \( L \).

Cook’s Theorem cited from [15] is that if one algorithm of solving a NP-complete problem is developed, then other problems will be solved by means of reduction to that problem. The following theorem is used to describe application of Cook’s Theorem for the classes \( P \) and NP on a quantum computer and a molecular computer.

Theorem 7–1: Assume that \( L \) is the satisfiability problem that is NP-complete. Then \( P = \text{NP} \) if and only if \( L \in \text{P} \).

VIII. AN EXAMPLE OF THREE-QUBIT SOLUTION FOR THE SATISFIABILITY PROBLEM

Consider the formula: \( F = (u_1) \), the simplest case of the satisfiability problem. It contains one clause, “\((u_1)\)”, and one Boolean variable \( u_1 \). The following algorithm, Algorithm 8–1, is the reduced version of Algorithm 4–2 in Subsection H and is employed to find the answer of the satisfiability problem of the clause, \( F = (u_1) \).

Algorithm 8–1: Solving the satisfiability problem of the clause, \( F = (u_1) \).

1. For an input \( |\Phi\rangle = \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle \), two choices are \( \left| \varphi_{1,0} \right\rangle = (I_{2 \times 2}) \otimes (I_{2 \times 2}) \otimes (H) \left| \Phi \right\rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle + \left| u_1^1 \right\rangle \right) \).

2. \( \left| \varphi_{1,1} \right\rangle = (I_{2 \times 2}) \otimes \left( \left| y_1^0 \oplus (u_1^0 + u_1^1) \right\rangle \right) \otimes (I_{2 \times 2}) \left| \varphi_{1,0} \right\rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 \right\rangle \otimes \left| y_1^0 + y_1^1 \right\rangle \otimes \left| u_1^0 \right\rangle + \left| u_1^1 \right\rangle \right) \).

3. \( \left| \varphi_{2,1} \right\rangle = \left( \left| c_1^0 \right\rangle \otimes \left( \left| y_1^0 + y_1^1 \right\rangle \right) \right) \otimes (I_{2 \times 2}) \left| \varphi_{1,1} \right\rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 + c_1^1 \right\rangle \otimes \left| y_1^0 + y_1^1 \right\rangle \otimes \left( \left| u_1^0 \right\rangle + \left| u_1^1 \right\rangle \right) \right) \).

4. \( \left| \varphi_{2,0} \right\rangle = (I_{2 \times 2}) \otimes \left( \left| y_1^0 \oplus u_1^0 \right\rangle + \left( y_1^0 \oplus u_1^1 \right) \right) \otimes (I_{2 \times 2}) \left| \varphi_{2,1} \right\rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 + c_1^1 \right\rangle \otimes \left( \left| y_1^0 \right\rangle \right) \otimes \left( \left| u_1^0 \right\rangle + \left| u_1^1 \right\rangle \right) \right) \).

5. We obtain the final result, \( \left| \varphi_{2,0} \right\rangle \), after a measurement of \( \left| \varphi_{2,0} \right\rangle \).

End Algorithm

So, from Step (1) of Algorithm 8–1, for an input \( |\Phi\rangle = \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle \), 2\(^1\) possible choices are \( |\varphi_{1,0} \rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle + \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^1 \right\rangle \right) \).

Because only a clause and a Boolean variable are involved in the example, we could use CNOT gate to replace QEC in Figure 4–3 in Appendix A to evaluate whether the only clause with the only Boolean variable is true or not. Hence, the implementation of Step (3) and Step (4) in Algorithm 8–1 yields \( |\varphi_{2,1} \rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle + \right. \left| c_1^1 \right\rangle \otimes \left| y_1^1 \right\rangle \otimes \left. \left| u_1^1 \right\rangle \right) \) and \( |\varphi_{2,0} \rangle = \frac{1}{\sqrt{2}} \left( \left| c_1^0 \right\rangle \otimes \left| y_1^0 \right\rangle \otimes \left| u_1^0 \right\rangle + \right. \left| c_1^1 \right\rangle \otimes \left| y_1^1 \right\rangle \otimes \left. \left| u_1^1 \right\rangle \right) \).
Step (5) of Algorithm 8-1 is for a measurement on $|\varphi_{2,0}\rangle$, i.e., the answer to the satisfiability problem for the formula: $F = (u_1)$. The measurement yields $|\varphi_{2,1}\rangle = |c_1\rangle|y_1\rangle|u_1\rangle$, and the corresponding quantum circuit of the example above is shown in Figure 8-1.

NMR approach has been widely employed to quantum information processing over past years due to its mature and well-controllable technology [16]. Although the quantum information processing by NMR is made on ensembles of nuclear spins, instead of individual spins, NMR has remained to be the most convenient experimental tool to demonstrate quantum information processing. We here also adopt this technology to check our theory.

Our experiment is carried out on a Varian INOVA 600 NMR spectrometer. The sample is $^{13}$C-labelled alanine with formula $^{13}$CH$_2-^{13}$CH(NH$_2$)-$^{13}$COOH, where the three carbons $^{13}$C, $^{13}$C, $^{13}$C correspond to the qubits $I_1, I_2, I_3$, respectively. The J-coupling constants are $J_{12} = 34.79$ Hz, $J_{23} = 54.01$ Hz, $J_{13} = 1.20$ Hz. Selective excitation was achieved by using soft pulses.

The experiment has three main steps as follows:

Step 1 is for initialization. Before the algorithm is carried out, the initial state, i.e., the pseudo-pure state, must be well prepared. There have been many methods to do this job, among which the spatial averaging method proposed by Cory et al. is most commonly used [16]. So in our experiment, we have also employed this technique to prepare three-qubit pseudo-pure state and the detailed pulse sequence can be found in [16]. The states of the input qubits are prepared by following operation

$$E + I_{1z} + I_{2z} + I_{3z} + 2I_{1z}I_{2z} + 2I_{1z}I_{3z} + 2I_{2z}I_{3z} + 4I_{1z}I_{2z}I_{3z},$$

where $E$ is the unity operator with the form of

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and $\sigma_i$ is the $i$th spin angular momentum operator in the $z$ direction, and $\sigma_z$ is the Pauli matrix,

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Step 2 translates the quantum gates into NMR pulses, respectively. We had to connect and optimize the pulses to construct the total NMR pulse sequence. The Hadamard gate can be achieved by a single $\pi/2$ pulse with phase x. The CNOT gate can be implemented by NMR pulses [17] as follows,

$$[\pi/2]_x \rightarrow (1/4J) \rightarrow [\pi]_x^2 \rightarrow (1/4J) \rightarrow [\pi]_x^2 \rightarrow [\pi/2]_x,$$

where the flip angle of the pulse and the time of delay are written in the square brackets and in the round brackets, respectively. The subscripts are the phases (i.e., along the $x$ or $y$ axis) of the pulse, and the superscripts are the nuclei to which the pulses are applied. Then we could obtain the total pulse sequence by connecting and optimizing the above pulses according to the quantum circuit.

Step 3 is the measurement, where a readout pulse is applied to each qubit to obtain the spectra.

Note that in NMR measurements, the frequencies and phases of NMR signals could clearly indicate the state the system evolved to after the readout pulses have been applied. In our experiment, the phases of the reference of $^{13}$C spectra from a thermal equilibrium was adjusted to be in absorption (i.e., to be positive), and then the same phase corrections were used to determine the absolute phases of the experimental spectra of the $^{13}$C after the algorithm is accomplished. In our case, the final state was $(000)_y + |101>_y)/\sqrt{2} = (000)_y + |111>_y)/\sqrt{2}$, which means the first and the third qubits are entangled. As the readout by NMR is a weak measurement, we have no state collapse after the measurement. Besides, only single quantum coherence can be detected in NMR. As a result, we have to employ some additional operations for detecting the output state $(000>_y + |101>_y)/\sqrt{2}$. We may detect the second qubit directly by applying a $\pi/2$ readout pulse along the $x$ axis,

$$\begin{pmatrix} c_1^0 \\ y_1^0 \\ u_1^0 \end{pmatrix} \rightarrow H \begin{pmatrix} c_1^0 \\ y_1^0 \\ u_1^0 \end{pmatrix} \rightarrow \begin{pmatrix} |u_1^0\rangle = \frac{1}{\sqrt{2}}(000)_y + |101>_y \\ y_1^0 \\ c_1^0 \end{pmatrix},$$

Figure 8-1: The quantum circuit of the example above.
yielding Figure 8-2 (b). But for the first and third qubits, we need to disentangle them before measurement. To this end, we apply a CNOT gate, respectively, on the first and second qubits followed by another CNOT gate, respectively, on the second and first qubits to get the state \((|000> + |011>)/\sqrt{2}\). Then the first qubit can be read out by a single \(\pi/2\) pulse along the x axis, as shown in Figure 8-2 (a). Similar steps applied to the third qubit result in the spectrum in Figure 8-2 (c). It is evident that the experimental results are in good agreement with our theoretical prediction.

Therefore, due to the fact that NMR quantum operations are not made on individual nuclear spins, but on spin ensemble, there is a difference in the operation between Figure 8-1 and our experiment. Some remarks must be addressed. First of all, the three-qubit NMR experiment we have carried out suffices to make a comprehensive test for our theory, because we have achieved the key aspects of our theory. Although the simple case with three qubits did not reflect the efficiency of quantum computation for SAT problem, we argue that, with more variables and clauses involved, the quantum computing efficiency would be more and more evident, which could also be found in our later discussion about the computational complexity. Secondly, DNA computation does not involve entanglement, whereas entanglement does appear in our quantum treatment. The necessity of additional operations to disentangle the output qubits is not the intrinsic characteristic of our quantum mechanical treatment, but due to the unique feature of NMR technique. Anyway, those additional operations have not changed the essence of our implementation.

**IX. CONCLUSIONS**

We have tried in this work to set up a bridge between the physical quantum computer proposed by Deutsch and the biological molecular computer proposed by Adleman. We have shown in Theorem 5–1 that the biological molecular computer proposed by Adleman is a subset of the physical quantum computer proposed by Deutsch and can thereby be implemented on the quantum computer. Moreover, we have also proven in Theorem 5–2 that for those famous NP-complete problems solvable on a bio-molecular computer, their corresponding DNA-based algorithms can be all implemented on a quantum machine. Based on this new computing paradigm, we have also demonstrated in Theorem 7–1 that \(P = NP\). By using the mature technique of NMR, we have carried out a solution for the simplest satisfiability problem. The experimental results are in well agreement with the theoretical prediction. Although the current technology is only available for the simplest case of the satisfiability problem, we believe, with more advanced NMR technique and more experimental efforts, more complicated satisfiability solutions could be demonstrated by

![Figure 8-2 (a): The experimental spectra of the three-qubit solution for satisfiability problem after the readout on the first qubit.](image1)

![Figure 8-2 (b): The experimental spectra of the three-qubit solution for satisfiability problem after the readout on the second qubit.](image2)

![Figure 8-2 (c): The experimental spectra of the three-qubit solution for satisfiability problem after the readout on the third qubit.](image3)
NMl experiments. We also expect that our algorithm could be carried out by a real quantum machine in the future.

REFERENCES


Figure 4–3: The full network is the QEC (the abbreviation of quantum evaluating circuit) to check whether the current clause is true or not.