Quantum Algorithm for Multiple Alignment of Amino Acid Sequences

SATOSHI IRIYAMA
Tokyo University of Science
Department of Information Science
2641 Yamazaki, Noda City, Chiba
JAPAN
iriyama@is.noda.tus.ac.jp

MASANORI OHYA
Tokyo University of Science
Department of Information Science
2641 Yamazaki, Noda City, Chiba
JAPAN
ohya@is.noda.tus.ac.jp

Abstract: The alignment of genome sequences or amino acid sequences is one of fundamental operations for the study of life. Usual computational complexity for the multiple alignment of $N$ sequences with common length $L$ by dynamic programming is $O(L^N)$. This alignment is considered as one of the NP problems, so that it is desirable to find a nice algorithm of the multiple alignment. Thus in this paper we propose the quantum algorithm for the multiple alignment based on the works [7, 14] in which the NP complete problem was shown to be the P problem by means of quantum algorithm and chaos information dynamics.

Key–Words: Quantum algorithm, Multiple alignment, NP complete problem

1 Introduction

When we analyze life in gene level, we examine the homology of genome or amino acid sequences to compare these sequences, for which we have to align the sequences. The alignment of two sequences is called the pairwise alignment, and that for sequences more than three is called the multiple alignment. To align the sequences, we insert gap (*) into the position of a sequence where an base or amino acid is considered to deviate. Such alignment should be first done to analyze genome sequences or amino acid sequences, so that it is one of the fundamental operations for study of life.

At the present stage, the algorithms of the pairwise alignment are done by applying dynamic programming [8, 12, 13]. However, it is rather difficult to use the similar algorithm for the multiple alignment because the computational complexity of the $N$ sequences with their length $L$ by dynamic programming becomes $O(L^N)$, whose alignment will be very difficult as $N$ increases. Therefore, the various methods have been considered to reduce the computational complexity. Among those, the Simulated Annealing has been used in [7, 9, 10]. The simulated annealing is one of the methods solving some combinatorics optimization problems such as travelling salesman problem. Even if the simulated annealing effectively works, it is difficult to demonstrate the multiple alignment in polynomial time of $N$, so that the multiple alignment is considered as one of the NP problems.

We have studied quantum algorithm for many years, and we discovered that there exist quantum algorithms to solve NP complete problem in polynomial time[1, 2, 3, 4, 15, 16, 17].Ohya and Volovich could prove that the NP complete problem becomes P problem by quantum algorithm and chaos information dynamics.

In this paper, we discuss the quantum algorithm for the multiple alignment, that is, how we can construct quantum gate to make the sequences align. Though we are not able to implement the quantum algorithm on the present computer, our present work will be effective to study life once quantum computer is realized.

2 Quantum Algorithm

In this section, we explain the usual quantum algorithm that is represented by unitary operators. Its computational complexity is defined by the number of fundamental quantum gates in it. For mathematical expression of the problem, we construct the quantum algorithm in the following steps:

1. Define a Hilbert space for computation.
2. Construct an initial state.
3. Construct unitary operators to solve the problem.
4. Apply them for the initial state and obtain the result.
5. Measure an observable with the result state.

2.1 OMV SAT Algorithm

OMV SAT algorithm and an alternative algorithm using stochastic limit are discussed more precisely in the papers[1, 2, 3, 5, 15].

Let \( X \equiv \{x_1, \ldots, x_n\}, n \in \mathbb{N} \) be a set. \( x_k \) and its negation \( \overline{x}_k \) \((k = 1, \ldots, n)\) are called literals. Let \( \overline{X} \equiv \{\overline{x}_1, \ldots, \overline{x}_n\} \) be a set, then the set of all literals is denoted by \( X' \equiv X \cup \overline{X} = \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\} \). The set of all subsets of \( X' \) is denoted by \( \mathcal{F}(X') \) and an element \( C \in \mathcal{F}(X') \) is called a clause. We take a truth assignment \( t \) to all variables \( x_k \). If we can assign the truth value to at least one element of \( C \), then \( C \) is called satisfiable.

Let \( L = \{0, 1\} \) be a Boolean lattice with usual join \( \lor \) and meet \( \land \), and \( t(x) \) be the truth value of a literal \( x \) in \( X \). Then the truth value of a clause \( C \) is written as \( t(C) \equiv \lor x \in C \ t(x) \).

Moreover the set \( \mathcal{C} \) of all clauses \( C_j \ (j = 1, 2, \ldots, m) \) is called satisfiable if \( t(C) \equiv \land_{j=1}^m t(C_j) = 1 \). Thus the SAT problem is written as follows:

**Definition 1 (SAT problem)** Given a Boolean set \( X \equiv \{x_1, \ldots, x_n\} \) and a set \( \mathcal{C} = \{C_1, \ldots, C_m\} \) of clauses, determine whether \( \mathcal{C} \) is satisfiable or not.

That is, this problem is to ask whether there exists a truth assignment to make \( \mathcal{C} \) satisfiable. It is known in usual algorithm that it is polynomial time to check the satisfiability only when a specific truth assignment is given, but we can not determine the satisfiability in polynomial time when an assignment is not specified.

In [3] we discussed the quantum algorithm of the SAT problem, which was rewritten in [5] with showing that the OM SAT algorithm is combinatorial. In [1, 2] it is shown that the chaotic quantum algorithm can solve the SAT problem in polynomial time.

Ohya and Masuda pointed out [3] that the SAT problem, hence all other NP problems, can be solved in polynomial time by quantum computer if the superposition of two orthogonal vectors \( |0\rangle \) and \( |1\rangle \) is physically detected. However this detection is considered not to be possible in the present technology. The problem to be overcome is how to distinguish the pure vector \( |0\rangle \) from the superposed one \( \alpha |0\rangle + \beta |1\rangle \), obtained by the OM algorithm, if \( \beta \) is not zero but very small. If such a distinction is possible, then we can solve the NPC problem in the polynomial time. In [1, 2] it is shown that it can be possible by combining nonlinear chaos amplifier with the quantum algorithm, which implies the existence of a mathematical algorithm solving NP=P.

Let \( \mathcal{C} = \{C_1, \ldots, C_n\} \) be a set of clauses on \( X' \equiv \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\} \). The computational basis of this algorithm is on the Hilbert space \( \mathcal{H} = \mathbb{C}^{2n+\mu+1} \) where \( \mu \) is a number of dust qubits, it is shown that \( \mu \) is less than \( 2mn \) [15].

Let \( |v_{in}\rangle \equiv |0^n, 0^\mu, 0\rangle \), \( |v_{out}\rangle \)

be an initial state vector.

For \( X = \{x_1, \ldots, x_n\} \) and a truth assignment \( t \), we put

\[
t(x_1) = \varepsilon_1, t(x_2) = \varepsilon_2, \ldots, t(x_n) = \varepsilon_n,
\]

where \( \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \in \{0, 1\} \), and we write \( t \) as a sequence of binary symbols:

\[
t = \varepsilon_1 \varepsilon_2 \cdots \varepsilon_n.
\]

A unitary operator \( U_C : \mathcal{H} \rightarrow \mathcal{H} \) computes \( t(C) \) for all truth assignment as follows

\[
U_C |v_{in}\rangle = U_C |0^n, 0^\mu, 0\rangle = \frac{1}{\sqrt{2^{m-1}}} \sum_{i=0}^{2^{m-1}} |e_i, d^\mu, t(C)\rangle \equiv |v_{out}\rangle
\]

where \( |d^\mu\rangle \) is dust qubits denoted by \( \mu \) strings of binary symbols, and \( |e_i\rangle = |\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\rangle \) is a binary representation of \( t \).

**Theorem 2 ([15])** For a set of clauses \( \mathcal{C} = \{C_1, \ldots, C_m\} \) on \( X' \equiv \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\} \), the number \( \mu \) of dust qubits for algorithm of SAT problem is

\[
\mu \leq 2mn.
\]

**Theorem 3 ([3, 5, 15])** For a set of clauses \( \mathcal{C} = \{C_1, \ldots, C_m\} \), we can construct the unitary operator \( U_C \) to calculate the truth value of \( \mathcal{C} \) as

\[
U_C = \prod_{i=1}^{m-1} U_{AND}(i) \prod_{j=1}^{m} U_{OR}(j) \ H(n)
\]

where, \( H(k) \) is a unitary operator to apply Hadamard transformation to first \( k \) qubits, that is

\[
H(k) = H^\otimes k I^\otimes N-k
\]

The computational complexity of quantum computation depends on the number of unitary operator in
The quantum circuit. Let $U$ be the unitary operator, it is written as
\[ U = U_n U_{n-1} \cdots U_1 \]
where $U_n, \ldots, U_1$ are fundamental gates. The computational complexity $T(U)$ is considered as $n$.

We need to combine some fundamental gates such as $U_{NOT}, U_{C-N}$ and $U_{C-C-N}$ to construct the quantum circuit in fact. $U_{AND}$ and $U_{OR}$ can be written as a combination of fundamental gates. Here we obtain the computational complexity $T(U_C)$ of SAT algorithm by the number of $U_{NOT}, U_{AND}$ and $U_{OR}$.

**Theorem 4** ([15]) For a set of clauses $C = \{ C_1, \ldots, C_m \}$ and literal $X' = \{ x_1, \ldots, x_n, x'_1, \ldots, x'_n \}$, $T(U_C)$ is
\[
T(U_C) = m - 1 + \sum_{k=1}^{m} (|C_k| + 2|C_k| - 1) \\
\leq 4mn - 1
\]

Various aspects of classical and quantum chaos have been the subject of numerous studies [1, 2, 6]. Consider the so called logistic map which is given by the equation
\[ x_{n+1} = ax_n(1 - x_n) \equiv g_a(x), \quad x_n \in [0, 1]. \]
The properties of the map depend on the parameter $a$. If we take, for example, $a = 3.71$, then the Lyapunov exponent is positive, the trajectory is very sensitive to the initial value and one has the chaotic behavior [2]. It is important to notice that if the initial value $x_0 = 0$, then $x_n = 0$ for all $n$.

The state $|\psi\rangle$ of the previous subsection is transformed into the density matrix of the form
\[ \rho = q^2 P_1 + (1 - q^2) P_0 \]
where $P_1$ and $P_0$ are projectors to the state vectors $|1\rangle$ and $|0\rangle$. One has to notice that $P_1$ and $P_0$ generate an Abelian algebra which can be considered as a classical system. The following theorems are proven in [1, 2, 6].

**Theorem 5** For the logistic map $x_{n+1} = ax_n(1 - x_n)$ with $a \in [0, 4]$ and $x_0 \in [0, 1]$, let $x_0$ be $\frac{1}{2}$ and a set $J$ be \{ 0, 1, 2, \ldots, $n$, \ldots, 2n \}. If $a$ is 3.71, then there exists an integer $k$ in $J$ satisfying $x_k > \frac{1}{2}$.

**Theorem 6** Let $a$ and $n$ be the same in above theorem. If there exists $k$ in $J$ such that $x_k > \frac{1}{2}$, then $k > \frac{n-1}{\log_2 3.71 - 1}$.

**Theorem 7** Let $|t(C)|$ be the cardinality of these assignments, if $x_0 \equiv \frac{r}{2^n}$ with $r = |t(C)|$ and there exists $k$ in $J$ such that $x_k > \frac{1}{2}$, then there exists $k$ satisfying the following inequality if $C$ is SAT.
\[
\left[ \frac{n - 1 - \log_2 r}{\log_2 3.71 - 1} \right] \leq k \leq \left[ \frac{5}{4} (n - 1) \right].
\]

From these theorems, for all $k$, it holds
\[
g^k_{3.71} (q^2) \begin{cases} \neq 0 & \text{iff } C \text{ is not SAT} \\
> 0 & \text{iff } C \text{ is SAT}
\end{cases}
\]

### 3 Outline of Alignment

Let
\[
A : MNPWYSTWQYT \\
B : MNPQYTVWPY \\
C : MNWYSTQPYV
\]
be the amino acid sequences of three organisms or identical proteins.

These sequences $A$, $B$ and $C$ look not so close each other. It is considered that some amino acids are changed, deleted or inserted during the course of the biological evolution from a common origin of $A$, $B$ and $C$. Therefore it is important to align the sequences $A$, $B$ and $C$ to study similarity or difference of organisms properly. After the alignment, they become
\[
A : MNPWYST \ast WQYT \\
B : MNPQY \ast TVWPY \ast \\
C : MN \ast WYST \ast QPYV
\]
by which we can see the similarity of $A$, $B$ and $C$.

### 4 Quantum Algorithm

Let bit 0 and 1 be denoted by the vectors $|0\rangle \equiv (1, 0)$ and $|1\rangle \equiv (0, 1)$ in the Hilbert space $\mathbb{C}^2$, respectively. A vector $|\psi\rangle \equiv \alpha |0\rangle + \beta |1\rangle$ ($\alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1$) is called a quantum bit or qubit. A state vector of an input or a register is denoted by a vector (n qubits) $|\psi\rangle \equiv \otimes_{k=1}^{n} (\alpha_k |0\rangle + \beta_k |1\rangle)$ ($\forall k \in \{1, \cdots, n\}, \alpha_k, \beta_k \in \mathbb{C}, |\alpha_k|^2 + |\beta_k|^2 = 1$) in the tensor product Hilbert space $\otimes_1^n \mathbb{C}^2$. The CONS of this Hilbert space consists of $2^n$ vectors $|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle, |1\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle, \cdots, |1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle$.

Let us briefly explain the outline of quantum algorithm[?]. Quantum computation demonstrates a
parallel computation by means of coherence of quantum states. Quantum algorithms usually consists of the following three steps.

**Step1:** Prepare a superposition of states as an initial input state.

**Step2:** Compute an objective function by quantum gates (unitary operators).

**Step3:** Read a result (quantum measurement).

The Hilbert space $\mathcal{H}_{all}$ is denoted by

$$\mathcal{H}_{all} \equiv \otimes_{1}^{N} \mathbb{C}^{2} \otimes_{1}^{n} \mathbb{C}^{2},$$

where an input state is described in the Hilbert space $\otimes_{1}^{N} \mathbb{C}^{2}$ including the dust qubits, and quantum computation is done in this space and the output (result) is described in the Hilbert space $\otimes_{1}^{n} \mathbb{C}^{2}$.

**Step1:** Prepare an initial state in $\mathcal{H}_{all}$.

$$|\psi\rangle = \sum_{i} c_{i} |\psi_{i}\rangle \sum_{i} |c_{i}|^{2} = 1$$

**Step2:** Transform $|\psi\rangle$ by suitable unitary gates (specified by each problem), and obtain the final state $|\tilde{\psi}\rangle$ in $\mathcal{H}_{all}$.

$$|\tilde{\psi}\rangle = \sum_{k} c_{k} |\tilde{\psi}_{k}\rangle$$

**Step3:** Measure a proper result.

## 5 Quantum Algorithm of Multiple Alignment

We discuss the quantum algorithm of the multiple alignment by simulated annealing[10] in this section. Let us consider $N$ amino acid sequences $A_1, A_2, \ldots, A_N$.

$$A_1 : a_1^{1}a_2^{1}\cdots a_{m_1}^{1}$$

$$A_2 : a_1^{2}a_2^{2}\cdots a_{m_2}^{2}$$

$$\cdots$$

$$A_N : a_1^{N}a_2^{N}\cdots a_{m_N}^{N},$$

where $m_i (i = 1, \ldots, N)$ is the number of amino acids for each sequence.

In order to make the quantum algorithm of the multiple alignment, we need to modify the input data for the alignment. Let us explain this process for the following three sequences:

**GGIPG**

**GGQPIGA**

**GIPQIG**

First, we add some gaps at the end of the amino acid sequences to make all sequences have the same length and to be useful for the alignment such as

**GGIPG * * * * **

**GGQPIGA * * **

**GIPQIG * * * * **

Here, the maximum number of the gaps needed for the multiple alignment is due to the rules considered in [9]. Then let $L$ be the length of the arranged amino acid sequences, so that all amino acid sequences can be written by

$$A_1 : a_1^{1}a_2^{1}\cdots a_{L}^{1}$$

$$A_2 : a_1^{2}a_2^{2}\cdots a_{L}^{2}$$

$$\cdots$$

$$A_N : a_1^{N}a_2^{N}\cdots a_{L}^{N}.$$  

Further we set the total sequence

$$A = A_1 \cdot A_2 \cdots A_N$$

and define the objective function $f(A)$ to apply the simulated annealing by

$$f(A) = \sum_{k=1}^{L-1} \sum_{j=i+1}^{L} d(a_{k}^{i}, a_{k}^{j}) \frac{1}{NC_{2}},$$

where

$$d(a_{k}^{i}, a_{k}^{j}) = \begin{cases} 0 & \text{if } a_{k}^{i} = a_{k}^{j} \\ 1 & \text{if } a_{k}^{i} \neq a_{k}^{j} \text{ and } (a_{k}^{i} \neq \text{wanda}_{k}^{j} \neq \text{*}) \\ w & \text{if } a_{k}^{i} \neq a_{k}^{j} \text{ and } (a_{k}^{i} = \text{*or} a_{k}^{j} = \text{*}) \end{cases}$$

In the above definition, $w$ is called the weight having a value in $(0, 2]$, we take $w = 2$ in the sequel. This objective function $f$ is an averaged difference among all sequences.

In order to compute the minimum value of $f(A)$ by the simulated annealing[7], we have to replace any amino acid with 1 and the gap with 0, and the resulting sequence of 0 and 1 is called the labelled sequence.

For example,

$$A_1 : GGIP \ast \ast G \quad B_1 : 1111001$$

$$A_2 : GGQPIGA \ast \ast \Rightarrow B_2 : 1111110$$

$$A_3 : GI \ast PQIG \quad B_3 : 1101111$$

The perturbation in the simulated annealing means here to exchange some 0 and 1 in the labelled sequences, for instance,
For instance, let \( U \) then construct \( b \) as follows: Make a pair of \( m,n \) randomly. Then, repeat this operation until all \( b \) or \( g \) are considered as the vectors \( |0\rangle \) and \( |1\rangle \) of qubits. Let \( s \) be a threshold of the objective function, we define the Hilbert space of the algorithm as

\[
\mathcal{H} = (\mathbb{C}^2)^{\otimes NL+X+\log s+1}
\]

where \( X \) is a suitable number of qubit of work space.

Therefore an initial state vector for quantum algorithm is

\[
|\psi_0\rangle = |k_1,0^{g_1}\rangle \otimes |k_2,0^{g_2}\rangle \otimes \cdots \\
\otimes |k_N,0^{g_N}\rangle \otimes |0^X\rangle \otimes |s\rangle \otimes |0\rangle
\]

where \( k_i(g_i) \) is a number of amino acid (gap) of \( i \)-th amino acid sequence, respectively and the last qubit is called a check bit of the algorithm.

**Step 1**

This step is composed of the following three

S(1.1), S(1.2) and S(1.3).  

S(1.1)

Construct the unitary operator \( U_P \) for the perturbation as follows: Make a pair of \(|0\rangle\) and \(|1\rangle\) by choosing randomly. Then, repeat this operation until all \(|0\rangle\) or \(|1\rangle\) make pairs. Let \( m \) be the position of \(|1\rangle\) and \( n \) be the position of \(|0\rangle\). The Controlled-Not gate \( U_{m,n} \) on \( \mathcal{H} \) attached to the positions \( m \) and \( n \) is given by

\[
U_{m,n} \equiv \otimes^N \left\{ \otimes^{n-1} I \otimes |0\rangle \langle 0| \otimes^{L-n} I \\
+ \otimes^{m-1} I \otimes |0\rangle \langle 1| + |1\rangle \langle 0| \right\} \\
\otimes^{n-m-1} I \otimes |1\rangle \langle 1| \otimes^{L-n} I \\
\otimes^{X+\log s+1} I
\]

Then construct \( U_P \) as a combination of \( U_{m,n} \):

\[
U_P \equiv \prod_{\text{pair}} U_{m,n}
\]

For instance, let \( |v_{in}\rangle \) be an initial state such that amino acid gap

\[
|v_{in}\rangle \equiv |1\rangle \otimes |1\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle
\]

We make pairs of \(|0\rangle\) and \(|1\rangle\) as

- Pair1 \( \equiv (2, 7) \),
- Pair2 \( \equiv (5, 6) \),
- Pair3 \( \equiv (3, 8) \),

where the first number of the pair denotes the position of \(|1\rangle\) and the second number denotes the position of \(|0\rangle\).

Then \( U_P \) has the form as

\[
U_P = U_{2,7} \cdot U_{5,6} \cdot U_{3,8}
\]

S(1.2)

Apply the Hadamard transformation \( H \) to \(|0\rangle\) (the part of the gap). This operation is given by

\[
U_H = \otimes^N \left( \otimes^{k_1} I \otimes^{g_1} H \right) \otimes^{X+\log s+1} I
\]

where

\[
H |0\rangle \equiv \frac{1}{\sqrt{2}} \left( |0\rangle + |1\rangle \right)
\]

S(1.3)

The third step is to apply \( U_P \) to the state \( U_H |\psi_0\rangle \). The resulting state is called the perturbated state of the sequence.

**Step 2 (computation)**

S(2.1)

Compute the distance among the sequences by the unitary operator \( U_C \) which is constructed by fundamental gates. The result is represented in \(|f(A'_i)| \) \((i = 1 \cdots 2^{g_1+\cdots+g_N})\).

S(2.2)

Define the unitary operator \( U_A \) as the following

\[
U_A \otimes^L I \otimes |f_A(\phi(A))\rangle \otimes |s\rangle \otimes |0\rangle \\
\equiv \otimes^L I \otimes |f_A(\phi(A))\rangle \otimes |s\rangle \\
\otimes \left\{ \begin{array}{ll}
|1\rangle & (f(A) \geq f(A'_i)) \\
|0\rangle & (f(A) < f(A'_i))
\end{array} \right.
\]

Apply \( U_A \) to the state \( U_C U_P U_H |v_{in}\rangle \), the check bit of the sequences satisfying \( f(A) \geq f(A'_i) \), namely the sequences accepted, becomes \(|1\rangle\). If the acceptance probability is very small (about less than \( \frac{1}{2^{2N}} \)), we use the chaotic dynamics exposed in the above section to amplify the probability [14]. If the acceptance probability is not so small, calculate again with the same input state and smaller \( f(A) \). About 100 times of calculation is adequate.

**Step 3 (observation)**

We observe the check vector whether it is \(|1\rangle\) or not by a similar way given in [14]. Then we obtain the aligned sequences which is the result of the multiple alignment.
6 Conclusion

Finally, we discuss the difference between quantum algorithm and classical algorithm for the multiple alignment. Table 1 shows the computational costs of the classical algorithm, the classical algorithm of the simulated annealing and the quantum algorithm, from which we can easily see the advance of the quantum algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFT</td>
<td>$2^L$</td>
<td>$L$</td>
</tr>
<tr>
<td>perturbation</td>
<td>$2^L$</td>
<td>$N \times L$</td>
</tr>
<tr>
<td>calculation</td>
<td>$2^N \times L \times N \times C_2$</td>
<td>$L \times N \times C_2$</td>
</tr>
<tr>
<td>total</td>
<td>$(2 + L \times N) \times 2^L$</td>
<td>$(1 + N \times C_2) \times L$</td>
</tr>
</tbody>
</table>

Table 1. Computational cost

It is shown that this quantum algorithm is finished in polynomial order of the size $N$ of input sequences and the sequence length $L$.

References:


