On a Method of Solving SAT Efficiently Using the Quantum Turing Machine

(Preliminary Version)

Takashi Mihara Tetsuro Nishino

School of Information Science
Japan Advanced Institute of Science and Technology, Hokuriku
15 Asahidai Tatsunokuchi, Ishikawa 923-12, Japan

Abstract

In this paper, under an assumption that a superposed physical states can be observed without collapsing the superpositon, we show that the satisfiability problem (SAT, for short) can be solved by a quantum Turing machine in $O(2^{\frac{n}{4}})$ time. This assumption is not widely accepted among physicists, however, Aharonov et al. [1] conjecture that a physical state actually exists as a superposition and can be observed without collapsing the superposition.

1 Introduction

Current computing devices are based on Turing machines. Since their behavior can be deterministically described, they are based on the principles of *classical physics*. On the other hand, *quantum Turing machine* (*QTM* for short) due to Deutsch [4] is based on the principles of *quantum physics*, and involves *superpositions* of physical states. Recently, several researchers have proposed possible designs of computers based on Deutsch's QTM [7, 8, 12].

The possible advantages of the QTM are as follows :

- Every finitely realizable physical system can be perfectly simulated by a QTM.
- A computer based on the QTM may be capable of dissipating very small amount of energy per step.
- Several researchers pointed out the possibilities of faster computations via QTMs as follows :
  - Deutsch and Jozsa found a problem such that QTM can solve faster than any other classical models of computation [5].
  - Shor showed that a QTM can *factor integers* and *find discrete logarithms* in polynomial time with a bounded probability of error [10].

What are major characteristics of computations performed by QTMs ? Let us consider a simple example. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolen function with n variables, and $X, Y \in \{0,1\}^n$ be assignments for the variables in f. Our task is to evaluate the values $f(X)$ and $f(Y)$. In ordinary computation, we will evaluate $f(X)$ first, then evaluate $f(Y)$. On the other hand, in *quantum computation*, we take the following procedure :

1. Represent X and Y as the elements $\vec{X}$ and $\vec{Y}$ of a certain linear space, respectively.
2. Make a *quantum superposition* $\vec{X} + \vec{Y}$.
3. Execute an ordinary program to evaluate f *once* for the input $\vec{X} + \vec{Y}$. On QTMs, an execution of an ordinary program induces a linear transformation corresponding to the program. Let U_f be such a linear transformation corresponding to the program for f. Then, by the linearity of U_f, we have

$$U_f(\vec{X} + \vec{Y}) = U_f(\vec{X}) + U_f(\vec{Y}).$$

This type of computation is called *quantum parallel computation.*

We can use the same procedure even when we have to evaluate the values of f for exponentially many assignments. The above equation means that a QTM can compute a superposition of the values of f quickly. But, according to the current framework of quantum

0-8186-6715-X/94 $4.00

physics, it is not certain whether we can read each value in the obtained superpositon efficiently. This is because the following observation problem has not been solved yet.

> **Observation Problem in Quantum Physics** : Explain what will happen when we observe a quantum physical object using the terms of quantum physics.

To find a method to solve NP-complete problems efficiently is one of the central issues in theoretical computer science. We have been studying efficient methods to solve NP-complete problems using QTMs. Since the observation problem has not been completely solved, we have been studying relationships between the restrictions on the observation and the efficiency of the quantum computation. In [9], we obtained the following result.

Theorem [9] Under assumption A, a QTM can solve SAT in polynomial time.

> **Assumption A** : We can observe the existence of a specific physical state S in a superposition with certainty in polynomial time, if S exists in the superposition.

Unfortunately, assumption A is not supported in current quantum physics, however, we also showed in [9] that

$$\text{Assumption A} \Rightarrow \text{NP} \subseteq \text{EQP},$$

where EQP is a quantum analogue of the complexity class P (i.e. the class of polynomial time languages). Thus, we have

$$\text{NP} \subsetneq \text{EQP} \Rightarrow \neg\text{Assumption A}.$$

Namely, if $\text{NP} \subsetneq \text{EQP}$ is shown, it follows that the assumption A is not valid in quantum physics. In this sense, we established an interesting relationship between computational complexity theory and quantum physics.

Along these lines of research, we have tried to find more relationships between the restrictions on the observation and the efficiency of the quantum computation. Very recently, Aharonov et al. [1] have proposed a new interpretation of the *observation*. That is, they conjectured that a physical state actually exists as a superposition and can be observed without collapsing the superposition. In this paper, we propose the following restriction on the observation based on Aharonov's interpretation.

> **Assumption B**: A superposition of configurations is preserved after an observation, and all of the configurations in the superposition can be observed in time proportional to the number of the configurations in the superposition.

Then, we shall show the following theorem.

Theorem 3.1 Under assumption B, a UQTM solves SAT in $O(2^{n/4})$ time, where n is the length of the description of an instance of SAT.

In the proof of the theorem, we use a deterministic algorithm to find a maximum independent set in an n-vertex graph in $O(2^{\frac{n}{3}})$ time due to Tarjan and Trojanowski [11].

2 The Quantum Turing Machine

The Definition of the Quantum Turing Machine

Like an ordinary Turing machine, a quantum Turing machine M consists of a finite control, an infinite tape, and a tape head.

Definition 2.1 [3] A *Quantum Turing Machine* (QTM) is a 7-tuple $M = (Q, \Sigma, \Gamma, U, q_0, B, F)$, where

> Q is a finite set of *states*,
> Γ is a *tape alphabet*,
> $B \in \Gamma$ is a *blank symbol*,
> $\Sigma \subseteq \Gamma$ is an *input alphabet*,
> δ is a *state transition function* and is a mapping from $Q \times \Gamma \times \Gamma \times Q \times \{L, R\}$ to $\mathbf{C}$ (the set of complex numbers),
> $q_0 \in Q$ is a *initial state*,
> $F \subseteq Q$ is a set of *final states*.

An expression $\delta(p, a, b, q, d) = c$ represents the following: if M reads a symbol a in a state p (let c_1 be this configuration of M), M writes a symbol b on the square under the tape head, changes the state into q, and moves the head one square in the direction denoted by $d \in \{L, R\}$ (let c_2 be this configuration of M), and c is called an *amplitude* of this event. Then we define the probability that M changes its configuration from c_1 to c_2 to be $|c|^2$.

This state transition function δ defines a linear mapping in a linear space of superpositions of M's configurations. This linear mapping is specified by the following matrix M_δ. Each row and column of M_δ corresponds to a configuration of M. Let c_1 and c_2 be

two configurations of M, then the entry corresponding to c_2 row and c_1 column of M_δ is δ evaluated at the tuple which transforms c_1 into c_2 in a single step. If no such tuple exists, the corresponding entry is 0. We call this matrix M_δ a *time evolution matrix* of M.

Condition: For any QTM M, the time evolution matrix M_δ must be a *unitary matrix.*

Namely, if $M_\delta^\dagger$ is the transpose conjugate of M_δ and I is the unit matrix, then the relations $M_\delta^\dagger M_\delta = M_\delta M_\delta^\dagger = I$ must be satisfied by M_δ.

A Physical Representation of the Quantum Turing Machine

We can construct *1 bit*, a minimum unit of information, using a 2-state physical system (e.g. a spin-$\frac{1}{2}$ system, etc.). We represent the physical states corresponding to these two states as [0] and [1]. In the sequel, we define that

$$[0] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } [1] = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

For, $n \geq 2$, n *bits* can be realized as a composed system of simultaneous observable 2-state physical systems. Therefore, the physical state $[x_1, x_2, \ldots, x_n]$ corresponding to n bits is represented as tensor products of 1-bit physical states as follows:

$$[x_1, x_2, \ldots, x_n] \stackrel{\text{def.}}{=} [x_1] \otimes [x_2] \otimes \cdots \otimes [x_n],$$

where $x_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$.

A QTM M consists of a finite control, an infinite tape, and a tape head. Thus, it will be construced as a composed system of the physical systems corresponding to these three components. We denote a physical system corresponding to the finite control by $[C]$, a physical system corresponding to the tape by $[T]$, and a physical system corresponding to the tape head by $[H]$. Each of these physical systems are also constructed as a composed system of 1-bit physical systems. Namely, if $c_i \in \{0, 1\}$ for $i = 1, 2, \ldots, u = \lceil \log |Q| \rceil$, then $[C] = [c_1] \otimes [c_2] \otimes \ldots \otimes [c_u]$. And, if M is $S(n)$-space bounded, then $[T] = [t_1] \otimes [t_2] \otimes \ldots \otimes [t_v]$, where $t_i \in \{0, 1\}$ for $i = 1, 2, \ldots, v = S(n)$, and $[H] = [h_1] \otimes [h_2] \otimes \ldots \otimes [h_w]$, where $h_i \in \{0, 1\}$ for $i = 1, 2, \ldots, w = \lceil \log S(n) \rceil$.

A physical state $[M]$ corresponding to M is represented as a composed system of these physical systems as follows:

$$[M] = [C] \otimes [H] \otimes [T].$$

Thus, a *state* of M is represented by

$$[C] \otimes [H] \otimes [T] = [c_1, \ldots, c_u, h_1, \ldots, h_w, t_1, \ldots, t_v].$$

This is a vector of length 1 in the 2^{u+w+v}-dimensional Hilbert space. In general, a state of M corresponds to a superposition of configurations of M. If a state of M is not a superposed one, the state of M is equal to a configuration of M.

Finally, if M has k tapes $T_1, \ldots, T_k$, the physical state of M will be represented as follows:

$$[M] = [C] \otimes [H_1] \otimes \cdots \otimes [H_k] \otimes [T_1] \otimes \cdots \otimes [T_k],$$

where $H_1, \ldots, H_k$ are heads on $T_1, \ldots, T_k$, respectively.

Computation and Observation

Computation by M is an evolution process of a physical system defined by the unitary matrix M_δ. Let $[\psi(0)]$ be an initial state of M which is represented as follows:

$$[\psi(0)] = [q_0] \otimes [1] \otimes [T_0],$$

where T_0 denotes the tape contents before the execution, which is an input string written from the first square followed by an infinite sequence of B's. If we denote the state at time s by $[\psi(s)]$,

$$[\psi(\tau t)] = M_\delta^t [\psi(0)],$$

where τ is the time required by M to execute a single step.

In quantum mechanics, since observations from outside will change the state of the physical system, we can not observe the tape content of a QTM from outside before the computation has been completed. Therefore, as in [4], we must define one bit of the finite control as a *halting flag* so that M can signal actively that the computation has been completed. Only this bit can be observed from outside without changing the state of M. The halting flag is set to 0 initially, and will be changed to 1 when M halts.

When the halting flag becomes 1, the tape content will be *observed* as follows: if the vector (a superposition of configurations) $\psi = \sum_i \alpha_i c_i$ is written on the tape, for any vector ϕ, we can observe with probability $|(\phi, \psi)|^2$ that ψ is parallel to ϕ. Especially, we can observe with probability $|\alpha_i|^2$ that ψ is parallel to c_i.

The Universal Quantum Turing Machine

Deutsch's universal QTM U can executes all operations of ordinary reversible Turing machines, and unitary transformations for 1-bit state space[4]. Notice that the ordinary Turing machines can not execute these unitary transformations. Deutsch's universal

QTM can execute the following eight types of transformations:

$$V_0 = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \quad V_1 = \begin{pmatrix} \cos\alpha & i\sin\alpha \\ i\sin\alpha & \cos\alpha \end{pmatrix},$$

$$V_2 = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & 1 \end{pmatrix}, \quad V_3 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix},$$

$$V_4 = V_0^{-1}, \quad V_5 = V_1^{-1}, \quad V_6 = V_2^{-1}, \quad V_7 = V_3^{-1},$$

where α is any irrational multiple of π. In this paper, we define α as follows:

$$\alpha = \frac{\pi}{4}.$$

A QTM U_0 which simulates Bennett's reversible universal DTM M can be constructed in a way that Deutsch showed in [4]. This QTM U_0 runs in the same number of steps as M. The universal QTM which we will use in the sequel is the QTM that will be obtained from U_0 by adding the abilities to execute the above eight types of unitary transformations. Without loss of generality, we use the following convention when we evaluate the execution time of the universal QTM.

> **Convention**: The universal QTM executes a single step of the reversible DTM in a single step, and each one of the eight types of transformations above in a single step.

By the way, notice the following:

> A QTM can simulate a DTM M if and only if the time evolution matrix of the QTM constructed from the state transition function of M is a unitary matrix. $\cdots(*)$

Of cource, the reversible DTMs satisfy the condition $(*)$.

Deutsch's universal QTM (UQTM, for short) is defined to be the QTM which simulates the reversible universal DTM. Namely, Deutsch's UQTM directly simulates the reversible universal DTM in the case of operations of ordinary Turing machines. The existence of the reversible universal DTM has been shown in [2].

But according to the construction in [2], the UQTM must record its history. On the other hand, it is not known whether there exists a UQTM which satisfies $(*)$ and runs without history. Therefore, we assume that UQTM in this paper simulates such a reversible universal DTM which records its history.

The UQTM U consists of a finite control and the following three tapes: an input tape T_1, a work tape T_2, and a history tape T_3. Let H_1, H_2, and H_3 be heads on these three tapes, respectively.

Let us consider the case where U simulates a QTM Q. For brevity, we assume that Q has only one tape. Inputs for U are an input x for Q, and a program P for Q (i.e. a set of state transition functions of Q). The input x is written on the work tape T_2, and P is written on the input tape T_1. The machine U writes a state s, a head position h, and a tape content t of Q (initially, $t = x$) on the work tape T_2, and then by decoding P, simulates Q given the input x. On the history tape T_3, the numbers of the state transition rules of U, which are used during the simulation of Q, are written. The machine U behaves as follows: (1) U accepts x if and only if Q accepts x, (2) U rejects x if and only if Q rejects x, and (3) U does not halt for x if and only if Q does not halt for x.

If Q becomes a superposition of two configurations Q_1 and Q_2, U will also become a superposition of two configurations U_1 and U_2. The configuration U_1 is the configuration in which U simulates Q_1 and U_2 is the configuration in which U simulates Q_2. Every time the time evolution matrix of U is applied, both of U_1 and U_2 evolve one step at the same time. The cases when Q is a superposition of more than two configurations are explained in a similar fashon.

Notice the following points:

1. In general, the progress of the computation in U_1 is different from that in U_2. For example, even when an execution of P has been completed in U_1, it is possible that an execution of P has not been completed yet. This is because the number of steps required to execute P in U_1 is different from that in U_2, in general.

2. The state of a finite control, head positions, and tape contents in U_1 are different from those in U_2, in general. Especially, histories written on the history tape in U_1 are different from those in U_2.

3 Results

To find a method to solve NP-complete problems efficiently is very important problem in theoretical computer science. We have been studying efficient methods to solve NP-complete problems using QTMs. Since the observation problem in quantum physics has not been completely solved, we have been studying relationships between the restrictions on the observation and the efficiency of the quantum computation. In [9],

we obtained the following result.

Theorem [9] Under assumption A, a QTM can solve SAT in polynomial time.

> **Assumption A** : We can observe the existence of a specific physical state S in a superposition with certainty in polynomial time, if S exists in the superposition.

Unfortunately, assumption A is not supported in current quantum physics, however, Aharonov et al. [1] have proposed a new interpretation of the *observation*. That is, they conjectured that a physical state actually exists as a superposition and can be observed without collapsing the superposition. In this paper, we propose the following restriction on the observation based on Aharonov's interpretation.

> **Assumption B**: A superposition of configurations is preserved after an observation, and all of the configurations in the superposition can be observed in time proportional to the number of the configurations in the superposition.

In this section, we show a method of solving SAT in $O(2^{\frac{n}{4}})$ time using Deutsch's UQTM under the assumption B, where n is the total length of a description of a logical formula f whose satisfiability should be decided, and a description of m which is the number of variables in f.

As mentioned above, the UQTM U has an input tape T_1, a work tape T_2, and a history tape T_3. Let H_1, H_2, and H_3 be the heads on the tapes T_1, T_2, and T_3, respectively (see Figure 1). All of the heads H_1, H_2, and H_3 can read and write. As shown in Figure 1, a program P that U simulates is written on the input tape T_1. On the right-hand side of P, infinitely many blank symbols are written. Notice that the length of the description of P is a constant which is independent of the length of an input given on T_2.

The program P consists of state transition rules of one-tape standard DTMs and sentences corresponding to the eight unitary transformations above. Let

$$V(n, i)$$

be the sentence corresponding to these unitary transformations, where $0 \leq n \leq 7$. This sentence represents that

> "apply the transformation V_n on the ith bit of the work tape".

When U simulates an execution of a one-tape standard DTM M, it will move as follows. Let q be a state of M written on T_2, and a be a symbol that the head of M scans. Then, U will scan the set P_M of state transition rules of M written on T_1 from the leftmost square of P_M to right, and will find only one state transition rule of the form $\delta(q, a) = \cdots$ (we assume that P_M always has one such a rule). If U finds such a rule, it will memorize the rule using its finite control. And then, U moves H_1 to the right until the rightmost square of P_M is reached. If H_1 reaches to the rightmost square of P_M, then U moves H_1 to the leftmost square of P_M. After that, U will change the configuration of M written on T_2 according to the found state transition rule. From this, in all configurations in a superposition, U can simulate a single step of M in the same number of steps.

The UQTM U starts the execution given a logical formula f and the number m of variables in f on the work tape T_2, and the program P to simulate on the input tape T_1. The number m of variables in f need not be supplied as an input[1], however in order to simplify the following presentations, we assume that m is also supplied as input. U simulates the program P, and records the history of the simulation on T_3.

The *time complexity* of the UQTM U is the sum of the number of steps executed by U until it finally changes the halting flag to 1 and the number of steps needed to observe the output. The time complexity of U is represented as a function of the length of the input (in this case, the total length of the descriptions of f and m) given on T_2.

Theorem 3.1 Under assumption B, the UQTM U can solve SAT in $O(2^{\frac{n}{4}})$ time, where n is the total length of the description of a logical formula f whose satisfiability should be decided and the description of the number of variables in f.

proof In the sequel, we show a program P that U simulates in order to solve SAT in $O(2^{\frac{n}{4}})$ time. Let us consider the satisfiability of an m-variable logical formula $f(x_1, x_2, \ldots, x_m)$, where x_i, $i = 1, 2, \ldots, m$ are Boolean variables. Without loss of generality, we can assume that the variables $x_1, \ldots, x_m$ are named in such a way that if $i < j$ then the number of occurrences of x_j in f is not larger than that of x_i.

The machine U writes an assignments to the variables in the logical formula f together with the corresponding value of f on the tape T_2. We show the

[1]Initially, U can scan the description of f on T_2 from left to right, and count the number of variables appearing in that description.

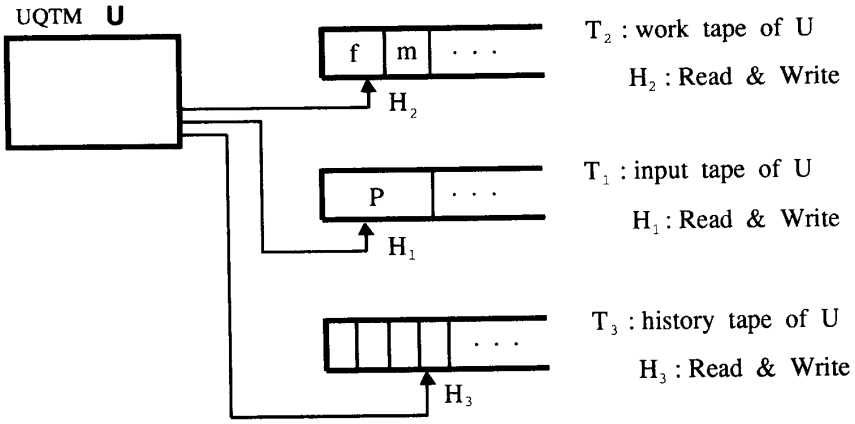

Figure 1: The UQTM U.

program that U will simulate in Figure 2. For example,

$$V(4,i)$$

means that "apply a matrix V_4 to $[x_i]$". In the sequel, we explain the behavior of U according to this program.

1. The initial configuration

   The descriptions of the logical formula f and the number m of the variables are given as input on the work tape T_2, and the program P that U simulates is given on the input tape T_1. In the sequel, we identify a configuration of U with a description of an assignment to the variables in f and the corresponding value of f, which are written on the tape T_2 (this description is written on the right-hand side of f and m on T_2). Let $[x_1],\ldots,[x_m]$, and $[x_{m+1}]$ be the bits corresponding to $x_1,\ldots,x_m$ and the value of f under the assignment in this configuration, respectively. In the sequel, let

   $$[x_1,\ldots,x_m,x_{m+1}] = [x_1]\otimes\cdots\otimes[x_m]\otimes[x_{m+1}]$$

   be a configuration of U. That is, actually,

   $$[U] = [C]\otimes[H_1]\otimes[H_2]\otimes[H_3]\otimes[T_1]\otimes[T_2]\otimes[T_3]$$

   holds, but we are indicating only $[x_1,\ldots,x_m,x_{m+1}]$ in $[T_2]$. Let

   $$[\underbrace{0,0,\ldots,0}_{m};0]$$

   be the initial configuration of U. In order to simplify the presentation, we separate the assignments for the variables from the value of f by semicolon(;).

2. Preparation of partial assignments

   There exist 2^m different assignments for m variables of f. Initially, U makes a superposition of all the configurations where $\lceil\frac{m}{4}\rceil$ variables, $x_1,\ldots,x_{\lceil\frac{m}{4}\rceil}$, are fixed. U will perform this by applying

   $$V_4 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\\ 1 & 1\end{pmatrix}$$

   to each bit corresponding to $\lceil\frac{m}{4}\rceil$ variables in order. U can execute the transformation by V_4 in a single step. In the execution of the **for** loop, the initial configuration is transformed as follows (see Figure 3):

   $$[0,0,\ldots,0;0] \overset{\mathsf{U}_{x_1}\cdots\mathsf{U}_{x_{\lceil\frac{m}{4}\rceil}}}{\longrightarrow} \frac{1}{\sqrt{2^{\lceil\frac{m}{4}\rceil}}}\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_{\lceil\frac{m}{4}\rceil}=0}^{1}[x_1,x_2,\ldots,x_{\lceil\frac{m}{4}\rceil},0\ldots,0;0],$$

   where the matrix M_{x_i} representing the unitary transformation U_{x_i} is

   $$\mathsf{M}_{x_i} = \underbrace{I\otimes\cdots\otimes I}_{i-1}\otimes V_4\otimes\underbrace{I\otimes\cdots\otimes I}_{m-i}.$$

   Because U can execute each transformation U_{x_i} in a single step, it can execute all the transformations $\mathsf{U}_{x_1},\cdots,\mathsf{U}_{x_{\lceil\frac{m}{4}\rceil}}$ in $\lceil\frac{m}{4}\rceil$ steps.

begin
%%% Preparation of partial assignments
1 **for** $i = 1$ **to** $\lceil \frac{m}{4} \rceil$ **do**
2 $V(4,i)$
od ;
%%% Computation of the values of f
3 Reduce an instance of SAT on $\lfloor \frac{3m}{4} \rfloor$-variable formula
to the corresponding maximum independent set problem.
4 Solve the obtained maximum independent set problem
using the algorithm of Tarjan & Trojanowski.
end.

Figure 2: The program that the UQTM U simulates.

Let us consider the unitary transformation U_{x_1} applied to the initial configuration $[0,0,\ldots,0;0]$ in detail. The initial configuration c_1 of U is transformed to a superposition of a configuration c_2 where $[0,0,\ldots,0;0]$ is written on T_2 and a configuration c_3 where $[1,0,\ldots,0;0]$ is written on T_2 as follows (notice that the following is an abbreviated notation):

$$[0,0,\ldots,0;0] \xrightarrow{\mathsf{U}_{x_1}} \frac{1}{\sqrt{2}}[0,0,\ldots,0;0] + \frac{1}{\sqrt{2}}[1,0,\ldots,0;0].$$

This transformation will be carried out immediately after the head H_1 of U reads $V(4,1)$ on T_1. Let q_1 be the state of U immediately after H_1 reads $V(4,1)$ on T_1, and j the position of H_1 at that time. Then each configuration of U is precisely represented as follows:

$$\begin{aligned} c_1 &= [q_1] \otimes [j] \otimes [k] \otimes [l] \otimes [P] \otimes \\ &[f, m, [0,0,\ldots,0;0]] \otimes [Z_1], \\ c_2 &= [q_2] \otimes [j] \otimes [k+1] \otimes [l] \otimes [P] \otimes \\ &[f, m, [0,0,\ldots,0;0]] \otimes [Z_1], \\ c_3 &= [q_3] \otimes [j] \otimes [k+1] \otimes [l] \otimes [P] \otimes \\ &[f, m, [1,0,\ldots,0;0]] \otimes [Z_1], \end{aligned}$$

where $[P]$ is the representation of T_1 including the program P, and $[f, m, [0,0,\ldots,0;0]]$ etc. are the representation of T_2. And, Z_1 appearing in $[T_3]$ is the history (i.e. a series of the numbers of the state transition rules used by U) of transitions made by U until it reaches to the configuration c_1. And, c_2 (or c_3) is a configuration obtained from c_1 by rewriting 0, which is written in the kth square on T_2, to 0 (or 1) and moving H_2 one square to the right.

By the way, c_2 and c_3 are the configurations obtained from c_1 by applying the unitary transformation V_4 to $[x_1]$. Namely, both of the c_2 row and c_1 column entry and c_3 row and c_1 column entry in the time evolution matrix M_δ are $\frac{1}{\sqrt{2}}$. Therefore, if U reaches to the configuration c_1 and then M_δ is applied, the U's configuration is transformed to the superposition of the configurations, $\frac{1}{\sqrt{2}}c_2 + \frac{1}{\sqrt{2}}c_3$.

Since U does not enter into c_2 (or c_3) from c_1 by using a state transition rule, the contents of T_3 (i.e. history) have not been changed in c_2 (or c_3). Notice that the same thing will always happen in the executions of the eight types of unitary transformations defined above.

So, when U completes the execution of the **for** loop in Figure 2, in all configurations contained in the obtained superposition of U's configurations, the history $Z_1 Z_2 \cdots Z_m$ is written on T_3. Here, Z_i $(2 \leq i \leq m)$ is the history of the U's transitions from the configuration immediately after the execution of $V(4, i-1)$ to the configuration immediately before the execution of $V(4,i)$.

3. Computation of the values of f

Now, in the logical formula f, the variables $x_1, \ldots, x_{\lceil \frac{m}{4} \rceil}$ are fixed and the other $\lfloor \frac{3m}{4} \rfloor$ variables are free. If we evaluate f as much as we can, the length of the description of f will be less than or equal to $\frac{3n}{4}$ by the assumption on the name of variables.

In the third line in Figure 2, U transforms this logical formula to the corresponding instance of the maximum independent set problem in the following way. Let $f = F_1 \wedge F_2 \wedge \cdots \wedge F_q$ be an

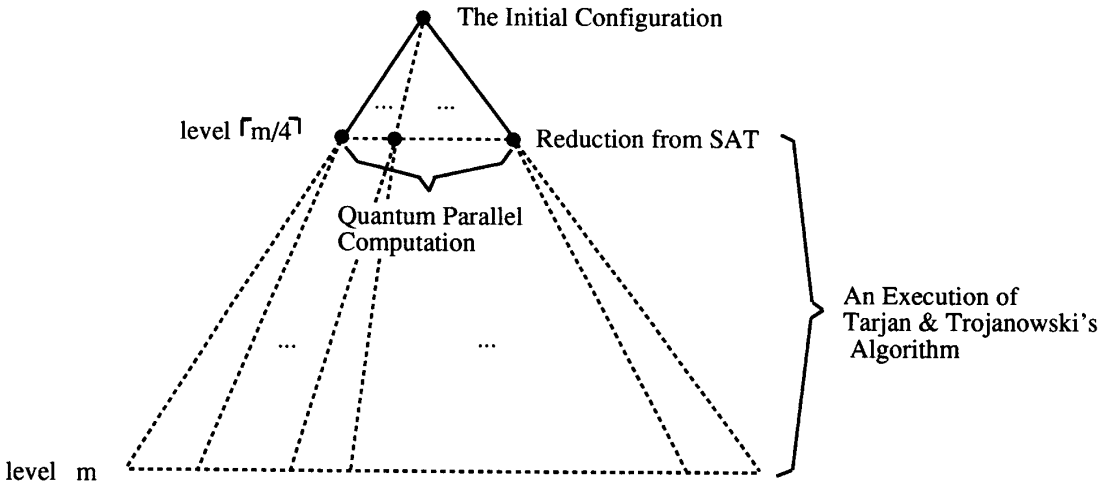

Figure 3: The computation executed by U.

expression in CNF, where each F_i is a clause of the form $(x_{i1} \vee x_{i2} \vee \cdots \vee x_{in_i})$, where each x_{ij} is a literal (i.e. either a variable or the negation of a variable) and n_i is the number of literals in clause F_i. U constructs an undirected graph $G = (V, E)$ whose vertices are pairs of integers $(i,j), 1 \leq i \leq q, 1 \leq j \leq n_i$. The vertex (i,j) represents the j-th literal of the i-th clause. The edges of the graph are $[(i,j),(k,l)]$ if $i = k$ or $x_{ij} = \bar{x}_{kl}$. Notice that the number of the vertices in the instance is less than or equal to $\frac{3n}{4}$, because the length of the description of f is less than or equal to $\frac{3n}{4}$. And this construction has the property that G has a maximum independent set of size q if and only if f is satisfiable. This transformation can be executed in polynomial time (for details, see [6] for example).

Next, in the fourth line in Figure 2, U solves this instance of the maximum independent set problem using the algorithm of Tarjan and Trojanowski[11]. This process can be performed in $O(2^{\frac{1}{3}\frac{3n}{4}}) = O(2^{\frac{n}{4}})$ time. Figure 3 illustrates the whole computation executed by U.

An Observation

When U completes all the computation of above and sets the halting flag to 1, we will observe all of the superposed $2^{\lceil \frac{m}{4} \rceil}$ configurations. If all the output bits of these configurations are zero, f will be unsatisfiable. On the other hand, if at least one output bit is one, f will be satisfiable. We can observe each of these configurations, because the superposed configurations are preserved after the each observation by assumption B. These observations can be executed in $O(2^{\frac{n}{4}})$ time again by the assumption.

The Time Complexity of UQTM U

Since the first line in Figure 2 is executed $\lceil \frac{m}{4} \rceil$ times in total, U can execute the **for** loop within $O(n)$ time. As mentioned above, U can execute the third line in polynomial time, and the fourth line in $O(2^{\frac{n}{4}})$ time. Finally, U can execute the observations of all the configurations in $O(2^{\frac{n}{4}})$ time. Therefore, U can execute the procedure in Figure 2 in $O(2^{\frac{n}{4}})$ time in total. This complete the proof of Theorem 3.1. □

Corollary 3.1 If there exists a deterministic algorithm to solve the maximum independent set problem in $O(2^{\varepsilon n})$ time $(0 < \varepsilon < 1)$, the QTM U can solve SAT in $O(2^{\frac{\varepsilon}{\varepsilon+1}n})$ time under assumption B.

proof The QTM U executes a quite similar procedure as shown in the proof of Theorem 3.1. In this case, U makes $2^{\lceil \frac{\varepsilon}{\varepsilon+1}m \rceil}$ different assignments for $x_1, \ldots, x_{\lceil \frac{\varepsilon}{\varepsilon+1}m \rceil}$ variables in the preparation step. Then U executes the same procedure shown in the proof of Theorem 3.1. Solving the maximum independent set problem and the observation can be executed

in $O(2^{\frac{\varepsilon}{\varepsilon+1}n})$ time in this case. □

4 Conclusions

In this paper, we showed that, if the maximum independent set problem can be solved in $O(2^{\varepsilon n})$ $(0 < \varepsilon < 1)$ time, the QTM U can solve SAT in $O(2^{\frac{\varepsilon}{\varepsilon+1}n})$ time under assumption B. But we cannot expect that U solves SAT faster than $O(2^{\varepsilon' n})$ time using the same method, because it is known that Tarjan & Trojanowski type algorithms for finding a maximum independent set must use $2^{\varepsilon n}$ time in the worst case, for some small positive ε[11].

It is to be expected that more relationships between the restriction on the observation and the efficiency of the quantum computation can be established. It is an important open question to show whether NP-complete problems can be efficiently solved on QTMs under only assumptions supported in current quantum physics.

References

[1] Y. Aharonov, J. Anandan and L. Vaidman, "Meaning of the Wave Function", *Phys. Rev.*, **A 47**, pp. 4616-4626, 1993.

[2] C. H. Bennett, "Logical Reversibility of Computation", *IBM J. Res. Dev.*, **17**, pp. 525-532, 1973.

[3] E. Bernstein and U. Vazirani, "Quantum Complexity Theory", *Proc. of 25th ACM Symposium on Theory of Computing*, pp. 11-20, 1993.

[4] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", *Proc. R. Soc. Lond.*, **A 400**, pp. 97-117, 1985.

[5] D. Deutsch and R. Jozsa "Rapid Solution of Problems by Quantum Computation", *Proc. R. Soc. Lond.*, **A 439**, pp. 553-558, 1992.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[7] S. Lloyd, "A Potentially Realizable Quantum Computer", *Science*, **261**, pp. 1569-1571, 1993.

[8] S. Lloyd, "Envisioning a Quantum Supercomputer", *Science*, **263**, p. 695, 1994.

[9] T. Mihara and T. Nishino, "Quantum Computation and NP-Complete Problems", In Proc. *Fifth Annual International Symposium on Algorithms and Computation*, 1994.

[10] P. W. Shor, *Algorithms for Quantum Computation : Discrete Log and Factoring*, DIMACS Technical Report 94-37, June 1994.

[11] R. E. Tarjan and A. E. Trojanowski, "Finding a Maximum Independent Set", *SIAM J. Compt.*, **6**, pp. 537-546, 1977.

[12] W. G. Teich, K. Obermayer and G. Mahler, "Structural basis of multistationary quantum systems II: Effective few-particle dynamics", *Phys. Rev.*, **B 37**, pp. 8111-8121, 1988.