Matrix-product-state simulation of an extended Brüschweiler bulk-ensemble database search

Akira SaiToh* and Masahiro Kitagawa
Division of Advanced Electronics and Optical Science, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
(Received 24 November 2005; published 26 June 2006)

Brüschweiler's database search in a spin Liouville space can be efficiently simulated on a conventional computer without error as long as the simulation cost of the internal circuit of an oracle function is polynomial, unlike the fact that in true NMR experiments, it suffers from an exponential decrease in the variation of a signal intensity. With the simulation method using the matrix-product-state proposed by Vidal [G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003)], we perform such a simulation. We also show the extensions of the algorithm without utilizing the *J*-coupling or *DD*-coupling splitting of frequency peaks in observation: searching can be completed with a single query in polynomial postoracle circuit complexities in an extension; multiple solutions of an oracle can be found in another extension whose query complexity is linear in the key length and in the number of solutions (this extension is to find all of marked keys). These extended algorithms are also simulated with the same simulation method.

DOI: 10.1103/PhysRevA.73.062332 PACS number(s): 03.67.Lx, 05.20.Gg, 89.20.Ff

I. INTRODUCTION

Since the time Deutsch [1] introduced quantum bits (qubits) and quantum circuits, there have been notable advances in quantum information theories, leading to the current status of quantum computation. Among quantum algorithms for quantum computation, there are two famous algorithms published from 1994 to 1997: Shor's [2] algorithm solves the problem of prime factorization within a time increasing polynomially in the number n of qubits; Grover's [3] algorithm solves the problem of unsorted-database searching (or solution searching using an unsorted database) in $O(2^{n/2})$ queries to find a solution (a key to find) from a key table consisting of n-bit-length keys of a database that contains $N=O(2^n)$ different records. This database search problem is concerned in this paper. We will see the definition of the problem in Sec. III. The query complexity $O(2^{n/2})$ is a quadratic speedup over standard classical methods that need $O(2^n)$ trials to find a single string (a key of a record) in the key table of an unsorted database. It was shown to be optimal [4], in the standard model of quantum computation using pure-state inputs and unitary transformations, as written in Chap. 6 of Ref. [5]. It should be noted that a search in a present context is a search using an oracle and it is conceptually different from a usual search (see, e.g., Ref. [6] for the traditional definitions of databases and searching problems). Moreover, there is another fact that should be mentioned: a quantum oracle-based database search is a searching for a particular key in a superposition of states by using a quantum oracle function, while a classical oracle-based database search is a searching in a table of record keys using a classical oracle function.

The speedup made by the algorithm had been considered due to a quantum nature. Recently, however, it was shown by Viamontes *et al.* [7] that Grover's algorithm can be simulated with the original query complexity unchanged by using their special decision diagram called quantum information decision diagram (QuIDD). In their method, the total complexity of the simulation is comparable to that of an ideal quantum computer for a construction of an oracle such that its internal simulation cost grows polynomially in n. Kawaguchi *et al.* [8] also showed that Grover's algorithm can be simulated on a conventional computer with a trivial increase in the complexity when one uses the density matrix renormalization group (DMRG) method [9] in the simulation of quantum circuits, which is equivalent to the simulation scheme using the matrix-product-state (MPS) decomposition proposed by Vidal [10]. This is because the amount of entanglement involved in the algorithm is small as long as the internal circuit of an oracle function is not considered. As shown by Vidal, if the amount of entanglement [Eq. (44) in the present paper] increases polynomially in n in a quantum algorithm for pure states, such an algorithm can be efficiently simulated on a classical computer. Thus, apart from applications of Grover's search (e.g., quantum counting [11]), we cannot strongly insist on the benefit of quantum computation in solving the problem of unsorted-database searching. In particular for solving practical *NP*-complete search problems, Grover's search is often inferior to conventional specialized algorithms according to Viamontes *et al.* [12].

There is another algorithm to solve the problem of unsorted-database searching in a different approach: Brüschweiler [13] proposed an algorithm with which one finds a solution in a linear query complexity by utilizing classical parallelism of a bulk ensemble quantum computer [14,15]. This algorithm, however, suffers from an exponential decrease in the variation of a signal intensity to detect, in increasing the number of qubits; one has to aggregate an exponentially increasing number of output signals to find a correct signal variation hidden in random noise. (The role of measurement precision in a similar algorithm was discussed by Okubo *et al.* [16].) This is why it has been not so much highlighted. Nevertheless, with the MPS simulation, this algorithm can be regarded as a quite efficient algorithm with respect to query complexity. The MPS simulation method can be used for an efficient simulation of time-evolving

*Electronic address: saitoh@qc.ee.es.osaka-u.ac.jp

mixed states under the condition that the amount of site-site correlation is small as shown by Zwolak and Vidal [17]. Because Brüschweiler's algorithm only processes slightly unmixed states, it meets this special condition.

Thus, in this paper, we argue the MPS simulation for the original Brüschweiler's search that we will see in Sec. III and that for its two variants that we construct in Sec. V: Explicit processes and computational complexities are described in Sec. VI; results from simulations are shown in Sec. VII. Here is a brief list of results.

(1) We find that the original Brüschweiler's database search is terminated with a good solution in polynomial time and polynomial memory space in n in the MPS simulation on a conventional computer as long as an oracle function is constructed as a quantum circuit whose cost in a simulation is small, such as one comprising a c^nNOT gate and NOT gates. Here, the term "memory space" is used in the meaning of a length of computer memory.

(2) We extend the algorithm so that a single query and $O(n)$ postoracle quantum gates are sufficient. Although this is similar to the Xiao-Long [18] algorithm, it is different because we do not use the split of frequency peaks caused by J or DD couplings in an observation stage but we use standard ensemble average measurements of polarizations. Simulation cost of this extension is also dependent on an oracle structure.

(3) Another extension is made so that one can use Brüschweiler's algorithm for a database search with multiple solutions involved (the J-coupling or DD-coupling split of frequency peaks is not used in observation); this extension also keeps the complexity of the simulation polynomial when an internal oracle structure is easy to simulate. One can find all of r solutions (their bit length is n) in $O(nr)$ queries.

These facts show that if we accept to regard quantum oracle functions in MPS simulations as a special sort of classical oracles, the cost of a classical simulation is, occasionally, exponentially smaller than that of quantum and/or ensemble computations for the problem of so-called unsorted-database searching owing to such small query complexities. We will discuss the meaning of the problem itself later in Sec. VIII because this result shows that the simulation is fast as if it were a sorted database or a database with a structure that we search in.

In this paper, we will first see the basics of a spin Liouville space in Sec. II because Brüschweiler's algorithm was originally described in a spin Liouville space. Brüschweiler's algorithm will be briefly introduced in Sec. III. Second, we will write about Vidal's MPS simulation method from the viewpoint of computer programming in Sec. IV. It is followed by Sec. V in which we will describe the extensions of Brüschweiler's algorithm. The details of our simulation and simulation complexities will be described in Sec. VI. Simulation results will be shown in Sec. VII. The discussion and the conclusion in Secs. VIII and IX will terminate this paper.

II. SPIN LIOUVILLE SPACE FOR A QUANTUM COMPUTATION

In this section, we will see a small part of the versatile formulations of quantum computing in a Liouville space proposed by Tarasov [19]. We omit the description of the formulation using generalized computational states here although it is the main part of his formulation. This is because it is not relevant to the description of Brüschweiler's database search which is of our interest.

A. Representation of quantum states under the terms of Hilbert spaces

As is well known, a pure state of a qubit is a unit vector of a Hilbert space $\mathcal{H}_2 \cong \mathbb{C}^2$. A pure state of n qubits is a unit vector of a product space $\mathcal{H}^{(n)}=\mathcal{H}_2^{\otimes n}=\mathcal{H}_2\otimes\cdots\otimes\mathcal{H}_2$. We usually use the Z basis for a single qubit, the basis under which the Pauli Z matrix is written as $|0\rangle\langle 0|-|1\rangle\langle 1|$ using basis vectors $|0\rangle$ and $|1\rangle$. For n-qubit pure states, we usually use a computational basis, with basis vectors $|0\cdots 0\rangle,\ldots,|1\cdots 1\rangle$; this is a product basis of basis $\{|0\rangle,|1\rangle\}$. We also express the basis by $\{|k\rangle\}_k=\{|0\rangle,|1\rangle,\ldots,|2^n-1\rangle\}$ in the decimal system instead of the binary system. A pure state $|\Psi\rangle\in\mathcal{H}^{(n)}$ may be represented as $|\Psi\rangle=\Sigma_{k=0}^{2^n-1}a_k|k\rangle$, where coefficients a_k are complex numbers that satisfy $\Sigma_k|a_k|^2=1$.

A mixed state of n qubits is generally represented as a density operator $\rho=\Sigma_{k,l=0,0}^{2^n-1,2^n-1}c_{kl}|k\rangle\langle l|$. ρ is an Hermitian ($\rho^\dagger=\rho$), positive operator ($\langle v|\rho|v\rangle\geqslant 0$ for vector $|v\rangle\in\mathcal{H}^{(n)}$) with unit trace ($\mathrm{Tr}\,\rho=1$). When it is idempotent ($\rho^2=\rho$), ρ is a pure state, i.e., $\rho=|\Psi\rangle\langle\Psi|$.

The expectation value of a measurement with an observable M (M is an Hermitian operator) is given by $\langle M\rangle=\mathrm{Tr}\,\rho M$. When a system is a bipartite of systems A and B with state spaces $\mathcal{H}^A$ and $\mathcal{H}^B$, respectively, for an Hermitian operator M^A acting on A, we have $\langle M^A\rangle=\mathrm{Tr}\,\rho^{AB}M^A\otimes I^B=\mathrm{Tr}\,\rho^A M^A$, where $\rho^A=\Sigma_{|\varphi^B\rangle}\langle\varphi^B|\rho^{AB}|\varphi^B\rangle$ is the reduced density operator of A ($\{|\varphi^B\rangle\}$ is a complete orthonormal system of $\mathcal{H}^B$). In addition, an expectation value of a measurement on a particular qubit i with an observable M^i is given by $\langle M^i\rangle=\mathrm{Tr}\,\rho I^{1,\ldots,i-1}\otimes M^i\otimes I^{i+1,\ldots,n}=\mathrm{Tr}\,\rho^i M^i$ where ρ^i is the reduced density operator of the ith qubit.

B. Liouville space

Consider a linear operator A acting on a state in a Hilbert space $\mathcal{H}^{(n)}$:

$$A=\sum_{k,l=0,0}^{2^n-1,2^n-1}A_{kl}|k\rangle\langle l|. \tag{1}$$

The dimension of such operators is $(2^n)^2=4^n$. We denote this operator as a *superket* $|A)$.

Let us define a Liouville space (operator Hilbert space) $\mathcal{L}^{(n)}$ of n qubits as a 4^n-dimensional complex linear space such that the inner product of two elements $|\sigma)$ and $|\eta)$ is defined as

$$(\sigma|\eta)=\mathrm{Tr}(\sigma^\dagger\eta), \tag{2}$$

and the norm of an element $|\sigma)$ is defined, with the positive square root, as

$$\|\sigma\|=\sqrt{(\sigma|\sigma)}. \tag{3}$$

Consider a complete orthonormal system $\{|k\rangle\}_k$ of a Hilbert space $\mathcal{H}^{(n)}$. Then $\{|k,l)\}_{k,l}=\{||k\rangle,\langle l|)\}_{k,l}$ is a complete orthonormal system of a Liouville space $\mathcal{L}^{(n)}$, i.e., for $\{|k,l)\}=\{|k_1,l_1)\otimes\cdots\otimes|k_n,l_n)\}=\{||k_1\rangle\otimes\cdots\otimes|k_n\rangle,\langle l_1|\otimes\cdots\otimes\langle l_n|)\}$,

$$(k,l|k',l') = \delta_{kk'}\delta_{ll'}, \tag{4}$$

$$\sum_{k=0}^{2^n-1}\sum_{l=0}^{2^n-1}|k,l)(k,l| = I. \tag{5}$$

Note that $|a,b)\otimes|c,d)=||a\rangle|c\rangle,\langle b|\langle d|)$.

For an arbitrary element $|\sigma)\in\mathcal{L}^{(n)}$,

$$|\sigma) = \sum_{k=0}^{2^n-1}\sum_{l=0}^{2^n-1}|k,l)(k,l|\sigma). \tag{6}$$

Here, $(k,l|\sigma)=\mathrm{Tr}(|l\rangle\langle k|\sigma)=\langle k|\sigma|l\rangle=\sigma_{kl}$ is a matrix element of σ.

C. Superoperators

An operator that acts on a superket in a Liouville space is called a superoperator. A superoperator $\mathcal{E}=\Sigma_{k,l,k',l'}\mathcal{E}_{klk'l'}|k,l)(k',l'|$ transforms $|\sigma)=\Sigma_{k,l}\sigma_{kl}|k,l)$ to the state

$$\mathcal{E}|\sigma) = \sum_{k,l=0,0}^{2^n-1,2^n-1}\sum_{k',l'=0,0}^{2^n-1,2^n-1}\mathcal{E}_{klk'l'}\sigma_{k'l'}|k,l). \tag{7}$$

1. Quantum gates

A quantum gate written as a unitary transformation $U=\Sigma_{ij}u_{ij}|i\rangle\langle j|$ acting on a state $|\psi\rangle$ in $\mathcal{H}^{(n)}$ is also represented as a superoperator

$$\mathcal{E} = \sum_{iljk}\mathcal{E}_{il,jk}|i,l)(j,k| = \sum_{iljk}u_{ij}u_{lk}^*|i,l)(j,k| \tag{8}$$

acting on $|\sigma)$ in $\mathcal{L}^{(n)}$. This is because

$$U\rho U^\dagger = \sum_{ijabkl}u_{ij}\rho_{ab}u_{lk}^*|i\rangle\langle j|a\rangle\langle b|k\rangle\langle l| = \sum_{ijkl}u_{ij}\rho_{jk}u_{lk}^*|i\rangle\langle l|. \tag{9}$$

For example, the NOT gate (X gate) is represented as a superoperator

$$\mathcal{E}_X = |1,1)(0,0| + |1,0)(0,1| + |0,1)(1,0| + |0,0)(1,1|. \tag{10}$$

III. BRÜSCHWEILER'S DATABASE SEARCH ALGORITHM

Brüschweiler's search is a search algorithm for an unsorted database by using NMR parallel computation. It offers an exponential speedup in query complexity for the search although the reduction of signal variation to detect is exponential in the number of qubits.

A database of present interest is a set of $N=2^n$ records each of which has an n-bit-length key. A particular key is marked but it is unknown in public. A table of keys is open to the public except for the information about marking. One intends to search for a marked key by sending a query to the database. The database will return some answer that will be used to determine a marked key. This is quite different from a usual database search. Usually, one knows a marked key that should be searched in a database. In contrast, one aims to find a marked key by sending queries to a database in the present context. We employ three different names to call this database according to its style of a query-and-answer behavior:

(a) It is called an unsorted database when it returns yes or no (i.e., 1 or 0) when one queries it as to whether a key that one shows to it is marked one.

(b) It is called a sorted database when it answers a query as to whether a key is equal to, larger than, or smaller than a marked key.

(c) It is called a partition-structured database when one can query as to whether a part of a key is equal to that of a marked key. An example of a partition-structured database is a database to which one can send a query as to whether a symbol $l\in\{0,1\}$ in a particular place of a key (in binary notation) is equal to a symbol in the same place of the marked key.

The problem of unsorted-database searching is to find a marked key in a database described in (a). This problem can be defined mathematically as written below, which is the definition used in many papers including Brüschweiler's one.

Definition 1: The problem of unsorted-database searching is defined as follows.

Suppose that there is a given binary function $f:\{0,1\}^n\mapsto\{0,1\}$ such that $f(\mathbf{x})=0$ for $\forall\mathbf{x}\neq\mathbf{w}$ and $f(\mathbf{w})=1$, where strings $\mathbf{x}$ and a string $\mathbf{w}$ are n-bit strings. The problem is to find $\mathbf{w}=w_1,\ldots,w_n$.

This function f can be realized as a unitary transformation U_f that performs a transposition of the two ket vectors $|\mathbf{w}\rangle|0\rangle_o$ and $|\mathbf{w}\rangle|1\rangle_o$ in the Hilbert space $\mathcal{H}^{(n+1)}$ of $n+1$ qubits. Here, a ket vector with the subscript o is one in the state space of the oracle bit. Let the superscripts $1,\ldots,n$ and o denote the state space of the qubits $1,\ldots,n$ and that of the oracle qubit, respectively. We have

$$U_f = I^{1,\ldots,n,o} - |\mathbf{w}^{1,\ldots,n}0^o\rangle\langle\mathbf{w}^{1,\ldots,n}0^o| - |\mathbf{w}^{1,\ldots,n}1^o\rangle\langle\mathbf{w}^{1,\ldots,n}1^o| + |\mathbf{w}^{1,\ldots,n}1^o\rangle\langle\mathbf{w}^{1,\ldots,n}0^o| + |\mathbf{w}^{1,\ldots,n}0^o\rangle\langle\mathbf{w}^{1,\ldots,n}1^o|. \tag{11}$$

This oracle operation is also written as the superoperator

$$\mathcal{E}_f = \sum_{ijkl}\langle i|U_f|k\rangle(\langle j|U_f|l\rangle)^*|i,j)(k,l|. \tag{12}$$

In Brüschweiler's search, this problem is solved with an ensemble of molecules in a highly mixed state of $n+1$ spins 1/2. The superket of an original state is well approximated by

$$|\rho_0) = |\gamma)^{\otimes n+1} = [p|0,0) + (1-p)|1,1)]^{\otimes n+1}, \tag{13}$$

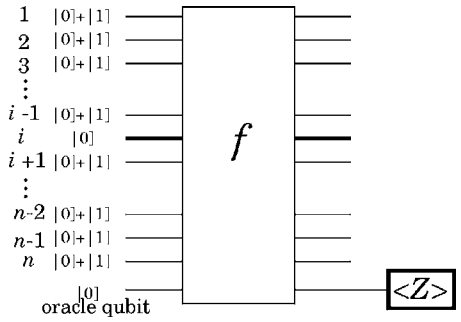

FIG. 1. The quantum circuit of the procedures (3) and (4) in the description of Brüschweiler's algorithm. $|0]\equiv|0,0)$ and $|1]\equiv|1,1)$.

where $|0,0)=||0\rangle,\langle 0|)$, $|1,1)=||1\rangle,\langle 1|)$, and $|\gamma)=p|0,0)+(1-p)|1,1)$; we also impose the condition $0\lessapprox p\lessapprox 1$. The searching is conducted in the following scheme.

For $s=1$ to n, do the procedures 1 to 4.

1. Initialize the qubit $n+1$ for the oracle bit by using a proper method, such as the effective pure state [20]. This creates the superket

$$|\rho_1)=|\gamma)^{\otimes n}\otimes|0,0)_o=[p|0,0)+(1-p)|1,1)]^{\otimes n}\otimes|0,0)_o. \tag{14}$$

2. Initialize the qubit s. The state at this point is the superket

$$|\rho_s)=|\gamma)^{\otimes s-1}\otimes|0,0)_s\otimes|\gamma)^{\otimes n-s}\otimes|0,0)_o. \tag{15}$$

3. Apply the oracle operation $\mathcal{E}_f$ to $|\rho_s)$:

$$|\rho_s')=\mathcal{E}_f|\rho_s). \tag{16}$$

In this transformation, the element $|\mathbf{w}^{1,\ldots,n}0^o,\mathbf{w}^{1,\ldots,n}0^o)$ is mapped to $|\mathbf{w}^{1,\ldots,n}1^o,\mathbf{w}^{1,\ldots,n}1^o)$ in the case where $w_s=0$.

4. Evaluate the function

$$F=1-(I^{\otimes n}\otimes Z|\rho_s')=1-\mathrm{Tr}\,\rho_s' I^{1,\ldots,n}\otimes Z=1-\langle Z_{\mathrm{oracle}}\rangle'. \tag{17}$$

F is a variation in the polarization $\langle Z_{\mathrm{oracle}}\rangle'$ of the oracle qubit in other words. If and only if the sth bit of $\mathbf{w}$, w_s, is 0, then we have $F>0$; otherwise we have $F=0$. Accordingly, we find the value of w_s. In particular when $p=0.5$, we have $F=2^{-n+2}$ if and only if w_s is 0.

After the iteration, we know the values $w_1,\ldots,w_n$. n queries are used in total. In addition, the procedures 3 and 4 may be illustrated as a quantum circuit in Fig. 1.

The variation V of a signal intensity, proportional to F, decreases as $V=O(2^{-n})$. Once V is lower than the mean noise level of an apparatus, we cannot get the signal in a single measurement even though we increase the number of bits in anolog-to-digital converters. Indeed, if we add additional m bits in anolog-to-digital converters, the precision of them is $\Omega(2^{-m})$, but a signal is hidden in noise before the input wires of them. It requires a number of measurements increasing exponentially in n to cancel noise by aggregating measurement results.

In contrast, in an MPS simulation, it is easy to add bits to a register to improve the precision. To refine the precision by the factor 2^{-m}, we simply add m bits to each register. Therefore, in a classical emulation of an NMR parallel computation, what they call the unsorted-database search can be exponentially faster than conventional schemes without any deficit in resolution, in terms of query complexity. The method of the MPS simulation will be described in detail in the next section from the viewpoint of computer programming.

IV. INTRODUCTION TO VIDAL'S MPS SIMULATION

From the viewpoint of computer programming, we will see the process of the MPS (matrix product state) simulation of a quantum computation. This method was first proposed by Vidal [10] in 2003. General simulation complexities shown by Vidal are also revisited in Eqs. (44)–(49), which will be especially used in calculating complexities of simulations of Brüschweiler's algorithm and its variants in Sec. VI. Some simulation techniques are also introduced in this section.

A. MPS decomposition of a quantum state

In the MPS simulation of time evolutions of quantum pure states, a state $|\Psi\rangle$ of n qubits at a particular time step is kept in the form of tensors $\{Q_s\}_{s=1}^n$ with parameters i_s,v_{s-1},v_s (v_0 and v_n are excluded), and $\{V_s\}_{s=1}^{n-1}$ with parameter v_s in the MPS decomposition shown in Eq. (18). $|\Psi\rangle=\Sigma_{i_1,\ldots,i_n}c_{i_1,\ldots,i_n}|i_1,\ldots,i_n\rangle$ is decomposed as [26]

$$\begin{aligned}|\Psi\rangle=\sum_{i_1=0}^{1}\cdots\sum_{i_n=0}^{1}\Bigg[&\sum_{v_1=1}^{m_1}\sum_{v_2=1}^{m_2}\cdots\sum_{v_{n-1}=1}^{m_{n-1}}Q_1(i_1,v_1)V_1(v_1)\\&\times Q_2(i_2,v_1,v_2)V_2(v_2)\cdots Q_s(i_s,v_{s-1},v_s)V_s(v_s)\cdots\\&V_{n-1}(v_{n-1})Q_n(i_n,v_{n-1})\Bigg]|i_1,\ldots,i_n\rangle.\end{aligned} \tag{18}$$

Here, m_s is a suitable number of values assigned to v_s with which the state is represented precisely or well approximated. That is to say, the coefficient $c_{i_1,\ldots,i_n}$ of $|i_1,\ldots,i_n\rangle$ is expanded as

$$\begin{aligned}&\sum_{v_1=1}^{m_1}\sum_{v_2=1}^{m_2}\cdots\sum_{v_{n-1}=1}^{m_{n-1}}Q_1(i_1,v_1)V_1(v_1)Q_2(i_2,v_1,v_2)V_2(v_2)\cdots\\&Q_s(i_s,v_{s-1},v_s)V_s(v_s)\cdots V_{n-1}(v_{n-1})Q_n(i_n,v_{n-1}).\end{aligned} \tag{19}$$

$Q_s(i_s,v_{s-1},v_s)$ is a tensor (namely, an array) with $2\times m_{s-1}\times m_s$ elements; $V_s(v_s)$ is an array in which the Schmidt coefficients for the splitting between the sth site and the $(s+1)$th site (i.e., the square roots of nonzero eigenvalues of the reduced density operator of qubits $1,\ldots,s$) are stored. By using V_s and eigenvectors $|\Phi_{v_s}^{1,\ldots,s}\rangle$ ($|\Phi_{v_s}^{s+1,\ldots,n}\rangle$) of the reduced density operator $\rho^{1,\ldots,s}$ ($\rho^{s+1,\ldots,n}$) of qubits $1,\ldots,s$ ($s+1,\ldots,n$), the state can be also written in the form of Schmidt decomposition

$$|\Psi\rangle = \sum_{v_s=1}^{m_s} V_s(v_s)|\Phi_{v_s}^{1,\ldots,s}\rangle|\Phi_{v_s}^{s+1,\ldots,n}\rangle. \tag{20}$$

In an MPS simulation, some small coefficients and corresponding eigenvectors are truncated out unlike a usual Schmidt decomposition that keeps all the nonzero coefficients. The number of kept coefficients is important to find a complexity of a simulation as we will see in Eqs. (44)–(49). In a computer program, we may regard Q_s as a function that returns a complex number for some inputs to its parameters i_s, v_{s-1}, v_s; $V_s(v_s)$ as a function of v_s that returns a real number larger than zero.

The MPS decomposition of $|\Psi\rangle$ can be accomplished by using the Schmidt decompositions sequentially although we seldom use this method in a usual simulation. The sequential Schmidt decomposition is accomplished in the following process:

First, the system is split into two parts, the first qubit and the rest qubits. The reduced density operators for individual parts, ρ^1 and $\rho^{2,\ldots,n}$, have the same set $\{\lambda_{v_1}\}$ of nonzero eigenvalues labeled by v_1 (the number of these eigenvalues is written as m_1); the set of eigenvectors for the former is $\{|\Phi_{v_1}^1\rangle\}$ and that for the latter is $\{|\Phi_{v_1}^{2,\ldots,n}\rangle\}$. Here, ρ^1 is computed by

$$\rho^1 = \sum_{\phi^{2,\ldots,n}} \langle\phi^{2,\ldots,n}|\Psi\rangle\langle\Psi|\phi^{2,\ldots,n}\rangle, \tag{21}$$

where $\{|\phi^{2,\ldots,n}\rangle\}$ is a complete orthonormal system of the state space of the qubits $2,\ldots,n$. $\rho^{2,\ldots,n}$ is also computed in a similar way. Setting $V_1(v_1)=\sqrt{\lambda_{v_1}}$, we have the Schmidt decomposition

$$\begin{aligned}|\Psi\rangle &= \sum_{v_1=1}^{m_1} V_1(v_1)|\Phi_{v_1}^1\rangle|\Phi_{v_1}^{2,\ldots,n}\rangle \\ &= \sum_{i_1=0}^{1}\sum_{v_1=1}^{m_1} V_1(v_1)Q_1(i_1,v_1)|i_1\rangle|\Phi_{v_1}^{2,\ldots,n}\rangle,\end{aligned} \tag{22}$$

where $Q_1(i_1,v_1)=\langle i_1|\Phi_{v_1}^1\rangle$. $|\Phi_{v_1}^{2,\ldots,n}\rangle$ can be written as

$$|\Phi_{v_1}^{2,\ldots,n}\rangle = \sum_{i_2=0}^{1} |i_2\rangle|\tau_{i_2,v_1}^{3,\ldots,n}\rangle, \tag{23}$$

where $|\tau_{i_2,v_1}^{3,\ldots,n}\rangle=\langle i_2|\Phi_{v_1}^{2,\ldots,n}\rangle$. Second, we compute $\rho^{3,\ldots,n}$ to get m_2 nonzero eigenvalues $\{\lambda_{v_2}\}$ and corresponding eigenvectors $\{|\Phi_{v_2}^{3,\ldots,n}\rangle\}$. Setting $V_2(v_2)=\sqrt{\lambda_{v_2}}$, $|\tau_{i_2,v_1}^{3,\ldots,n}\rangle$ is decomposed as

$$|\tau_{i_2,v_1}^{3,\ldots,n}\rangle = \sum_{v_2=1}^{m_2} V_2(v_2)Q_2(i_2,v_1,v_2)|\Phi_{v_2}^{3,\ldots,n}\rangle, \tag{24}$$

where $Q_2(i_2,v_1,v_2)=\frac{1}{V_2(v_2)}\langle\Phi_{v_2}^{3,\ldots,n}|\tau_{i_2,v_1}^{3,\ldots,n}\rangle$. Using this routine sequentially for the decompositions of eigenvectors from $|\Phi_{v_2}^{3,\ldots,n}\rangle$ to $|\Phi_{v_{n-1}}^{n}\rangle$, we determine the tensors $V_3(v_3),\ldots,V_{n-1}(v_{n-1})$ and $Q_3(i_3,v_2,v_3),\ldots,Q_n(i_n,v_{n-1})$.

As we mentioned, this method is expensive and seldom used to prepare the initial MPS in the simulation of a quantum computation.

B. Main process of the MPS simulation

In a usual MPS simulation of a quantum computation, we use the following three processes.

(i) *Preparation of the initial MPS*.

The initial state $|\Psi_{ini}\rangle$ for the input of a quantum circuit is often very easy to decompose into the MPS form: in the case of $(a|0\rangle+b|1\rangle)^{\otimes n}$ ($a,b\in\mathbb{C}$ and $|a|^2+|b|^2=1$), we have only to set $m_1=\cdots=m_{n-1}=1$ and $V_1(1)=\cdots=V_{n-1}(v_{n-1})=1$; $Q_1(0,1)=Q_2(0,1,1)=\cdots=Q_{n-1}(0,1,1)=Q_n(0,1)=a$ and $Q_1(1,1)=Q_2(1,1,1)=\cdots=Q_{n-1}(1,1,1)=Q_n(1,1)=b$. We need not to use any floating-point multiplication to store values of elements in tensors $V_1,\ldots,V_{n-1}$ and $Q_1,\ldots,Q_n$ for such a simple initial state. Once the MPS decomposition is done for the initial state, we use the following procedures to perform the simulation of a time evolution under unitary transformations.

(ii) *Procedure* 1 *for single-qubit quantum gates*. Let us regard the state $|\Psi\rangle$ in Eq. (18) as a state before some particular operation is applied. First, let us suppose that a single-qubit unitary transformation $U_s=\Sigma_{i_sk_s=00}^{11}u_{i_sk_s}|i_s\rangle\langle k_s|$ operates on the qubit l. This transformation is realized in the simulation by altering $Q_l(i_l,v_{l-1},v_l)$ to

$$\tilde{Q}_l(i_l,v_{l-1},v_l) = \sum_{k_l=0}^{1} u_{i_lk_l}Q(k_l,v_{l-1},v_l). \tag{25}$$

(iii) *Procedure* 2 *for two-qubit quantum gates*. Second, let us suppose that we apply the two-qubit unitary transformation $U_{s,s+1}=\Sigma u_{i_si_{s+1}k_sk_{s+1}}|i_si_{s+1}\rangle\langle k_sk_{s+1}|$ on some consecutive qubits $l,l+1$. We first represent the state $|\Psi\rangle$ before the transformation in two different expressions

$$|\Psi\rangle = \sum_{v_{l-1}=1}^{m_{l-1}} V_{l-1}(v_{l-1})|\Phi_{v_{l-1}}^{1,\ldots,l-1}\rangle|\Phi_{v_{l-1}}^{l,\ldots,n}\rangle, \tag{26}$$

$$|\Psi\rangle = \sum_{v_{l+1}=1}^{m_{l+1}} V_{l+1}(v_{l+1})|\Phi_{v_{l+1}}^{1,\ldots,l+1}\rangle|\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle. \tag{27}$$

Then we define the vectors

$$|v_{l-1}\rangle = V_{l-1}(v_{l-1})|\Phi_{v_{l-1}}^{1,\ldots,l-1}\rangle, \tag{28}$$

$$|v_{l+1}\rangle = V_{l+1}(v_{l+1})|\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle. \tag{29}$$

We do not need to know how they can be represented under the computational basis. In the simulation program, we have only to know the values

$$\langle v_{l-1}|v_{l-1}\rangle = [V_{l-1}(v_{l-1})]^2, \tag{30}$$

$$\langle v_{l+1}|v_{l+1}\rangle = [V_{l+1}(v_{l+1})]^2. \tag{31}$$

Next, utilizing Eqs. (28) and (29), we rewrite Eq. (18):

$$|\Psi\rangle = \sum_{i_l i_{l+1}=00}^{11} \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \sum_{v_{l+1}=1}^{m_{l+1}} [Q_l(i_l, v_{l-1}, v_l) V_l(v_l) \times Q_{l+1}(i_{l+1}, v_l, v_{l+1}) |v_{l-1}\rangle |i_l i_{l+1}\rangle |v_{l+1}\rangle]. \quad (32)$$

To this state, $U_{l,l+1} = \Sigma u_{i_l i_{l+1} k_l k_{l+1}} |i_l i_{l+1}\rangle\langle k_l k_{l+1}|$ is applied. In order to write the state after the application of $U_{l,l+1}$ in a succinct style, the tensor

$$\Theta(i_l, i_{l+1}, v_{l-1}, v_{l+1}) = \sum_{v_l=1}^{m_l} \sum_{k_l k_{l+1}=00}^{11} [u_{i_l i_{l+1} k_l k_{l+1}} \times Q_l(k_l, v_{l-1}, v_l) V_l(v_l) Q_{l+1}(k_{l+1}, v_l, v_{l+1})] \quad (33)$$

is introduced. With this new tensor, the evolved state $|\tilde{\Psi}\rangle = U_{l,l+1}|\Psi\rangle$ is given by

$$|\tilde{\Psi}\rangle = \sum_{i_l i_{l+1}=00}^{11} \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_{l+1}=1}^{m_{l+1}} \Theta(i_l, i_{l+1}, v_{l-1}, v_{l+1}) |v_{l-1}\rangle |i_l i_{l+1}\rangle |v_{l+1}\rangle. \quad (34)$$

The simulation should go on running, reshaping this state into that in the style of Eq. (18). This process will update Q_l, V_l, and Q_{l+1} to $\tilde{Q}_l$, $\tilde{V}_l$, and $\tilde{Q}_{l+1}$. For this purpose, we shall find the reduced density operator of the block of bits $l+1, \ldots, n$ from $|\tilde{\Psi}\rangle$. In this calculation, we use the orthonormal basis $\{|i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle\} = \{|01\rangle, \ldots, |1 m_{l+1}\rangle\}$ (this is a basis for a subset of states among the states of qubits $l+1, \ldots, n$):

$$\begin{aligned}
\rho^{l+1,\ldots,n} = \mathrm{Tr}_{1,\ldots,l} U_{l,l+1} |\tilde{\Psi}\rangle\langle\tilde{\Psi}| U_{l,l+1}^\dagger &= \sum_{i_{l+1}=0}^{1} \sum_{v_{l+1}=1}^{m_{l+1}} \sum_{i'_{l+1}=0}^{1} \sum_{v'_{l+1}=1}^{m_{l+1}} \left[\sum_{v_{l-1}=1}^{m_{l-1}} \sum_{i_l=0}^{1} \langle v_{l-1}|v_{l-1}\rangle \Theta(i_l, i_{l+1}, v_{l-1}, v_{l+1}) \right. \\
&\left. \times \Theta^*(i_l, i'_{l+1}, v_{l-1}, v'_{l+1}) \right] |i_{l+1} v_{l+1}\rangle\langle i'_{l+1} v'_{l+1}| \\
&= \sum_{i_{l+1}=0}^{1} \sum_{v_{l+1}=1}^{m_{l+1}} \sum_{i'_{l+1}=0}^{1} \sum_{v'_{l+1}=1}^{m_{l+1}} \left[\sum_{v_{l-1}=1}^{m_{l-1}} \sum_{i_l=0}^{1} [V_{l-1}(v_{l-1})]^2 \Theta(i_l, i_{l+1}, v_{l-1}, v_{l+1}) \right. \\
&\left. \times \Theta^*(i_l, i'_{l+1}, v_{l-1}, v'_{l+1}) V_{l+1}(v_{l+1}) V_{l+1}(v'_{l+1}) \right] |i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle\langle i'_{l+1}\Phi_{v'_{l+1}}^{l+2,\ldots,n}| \\
&= \sum_{i_{l+1} v_{l+1} i'_{l+1} v'_{l+1}} a_{i_{l+1} v_{l+1} i'_{l+1} v'_{l+1}} |i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle\langle i'_{l+1}\Phi_{v'_{l+1}}^{l+2,\ldots,n}|. \quad (35)
\end{aligned}$$

Elements $a_{i_{l+1} v_{l+1} i'_{l+1} v'_{l+1}} = [\Sigma_{v_{l-1}} \Sigma_{i_l} \cdots]$ of $\rho^{l+1,\ldots,n}$ are calculated in this way. Under the current basis $\{|i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle\}$, $\rho^{l+1,\ldots,n}$ is a $(2m_{l+1}, 2m_{l+1})$ matrix. This is then diagonalized by some numerical diagonalization method, such as the Hermitian Jacobi method, the Householder transformation, etc. (see, e.g., Ref. [21]). A unitary transformation U_d can be numerically found to obtain

$$U_d \rho^{l+1,\ldots,n} U_d^\dagger = \begin{pmatrix} \lambda_{01} & & \\ & \ddots & \\ & & \lambda_{1 m_{l+1}} \end{pmatrix}. \quad (36)$$

Note that the elements should be rearranged in the order of values. Among them, nonzero eigenvalues are labeled by v_l. We take the square roots of them to have

$$\tilde{V}_l(v_l) = \sqrt{\lambda_{v_l}}. \quad (37)$$

The corresponding eigenvectors $|\tilde{\Phi}_{v_l}^{l+1,\ldots,n}\rangle$, i.e., the column vectors of $U_d^\dagger$, are also obtained in this process. We set $\tilde{m}_l$ to the number of the nonzero eigenvalues obtained above. If we do not need so much accuracy, very small eigenvalues may be omitted. After omitting a particular eigenvalue λ_x out, the other eigenvalues λ_i are replaced with $\lambda_i/(1-\lambda_x) = \lambda_i/\Sigma_{i\neq x}\lambda_i$.

Because the basis is $\{|i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle\}$, the result of the process provides

$$|\tilde{\Phi}_{v_l}^{l+1,\ldots,n}\rangle = \sum_{i_{l+1}=0}^{1} \sum_{v_{l+1}=1}^{m_{l+1}} C_{l+1}(i_{l+1}, v_l, v_{l+1}) |i_{l+1}\Phi_{v_{l+1}}^{l+2,\ldots,n}\rangle, \quad (38)$$

where elements of tensor $C_{l+1}(i_{l+1}, v_l, v_{l+1})$ come from elements of a column vector of $U_d^\dagger$. By using the relation

$\tilde{Q}_{l+1}(i_{l+1},v_l,v_{l+1})=C_{l+1}(i_{l+1},v_l,v_{l+1})/V_{l+1}(v_{l+1})$, we have

$$|\tilde{\Phi}_{v_l}^{l+1,\ldots,n}\rangle=\sum_{i_{l+1}=0}^{1}\sum_{v_{l+1}=1}^{m_{l+1}}\tilde{Q}_{l+1}(i_{l+1},v_l,v_{l+1})|i_{l+1}v_{l+1}\rangle. \quad (39)$$

In the next process, we use the relation

$$\tilde{V}_l(v_l)|\tilde{\Phi}_{v_l}^{1,\ldots,l}\rangle=\langle\tilde{\Phi}_{v_l}^{l+1,\ldots,n}|\tilde{\Psi}\rangle. \quad (40)$$

This leads to that

$$\begin{aligned}\tilde{V}_l(v_l)|\tilde{\Phi}_{v_l}^{1,\ldots,l}\rangle&=\sum_{i_li_{l+1}=00}^{11}\sum_{v_{l-1}=1}^{m_{l-1}}\sum_{v_{l+1}=1}^{m_{l+1}}\big[\tilde{Q}_{l+1}^{*}(i_{l+1},v_l,v_{l+1})\\&\quad\times\Theta(i_l,i_{l+1},v_{l-1},v_{l+1})\langle v_{l+1}|v_{l+1}\rangle\big]|v_{l-1}i_l\rangle\\&=\sum_{i_l=0}^{1}\sum_{v_{l-1}=1}^{m_{l-1}}C_l(i_l,v_{l-1},v_l)|v_{l-1}i_l\rangle, \quad (41)\end{aligned}$$

where $C_l(i_l,v_{l-1},v_l)=\Sigma_{i_{l+1}}\Sigma_{v_{l+1}}[\cdots]$. Thereby we have

$$|\tilde{\Phi}_{v_l}^{1,\ldots,l}\rangle=\sum_{i_l=0}^{1}\sum_{v_{l-1}=1}^{m_{l-1}}\tilde{Q}_l(i_l,v_{l-1},v_l)|v_{l-1}i_l\rangle. \quad (42)$$

Consequently, the calculations that we have seen provides $\tilde{Q}_l(i_l,v_{l-1},v_l)$, $\tilde{V}_l(v_l)$, and $\tilde{Q}_{l+1}(i_{l+1},v_l,v_{l+1})$. Now we can write the state after $U_{l,l+1}$ acts on as

$$\begin{aligned}|\tilde{\Psi}\rangle=\sum_{i_1=0}^{1}\cdots\sum_{i_n=0}^{1}\Bigg[&\sum_{v_1=1}^{m_1}\cdots\sum_{v_l=1}^{\tilde{m}_l}\cdots\sum_{v_{n-1}=1}^{m_{n-1}}Q_1(i_1,v_1)V_1(v_1)\\&\times Q_2(i_2,v_1,v_2)V_2(v_2)\cdots\tilde{Q}_l(i_l,v_{l-1},v_l)\tilde{V}_l(v_l)\\&\times\tilde{Q}_{l+1}(i_{l+1},v_l,v_{l+1})\cdots V_{n-1}(v_{n-1})Q_n(i_n,v_{n-1})\Bigg]\\&\times|i_1,\ldots,i_n\rangle. \quad (43)\end{aligned}$$

When a unitary transformation U is applied to dispersed (nonconsecutive) qubits a and b, we use the bit-swapping gate, SWAP$=|00\rangle\langle 00|+|01\rangle\langle 10|+|10\rangle\langle 01|+|11\rangle\langle 11|$, $|a-b|-1$ times to gather them in consecutive locations; then we apply U, and again use $|a-b|-1$ SWAP gates to restore the locations of the qubits.

C. Complexity of the simulation

Let us consider the complexity of the MPS simulation using the above processes. Here, a variable to represent a largest one among $m_s=m_s(t)$ $(s=1,2,\ldots,n-1)$ at a time step t is utilized for convenience:

$$m_{\max}(t)=\max_s m_s(t). \quad (44)$$

This was introduced by Vidal as χ in the Eq. (2) of Ref. [10] (he also showed that χ is a measure of an amount of entanglement.). Let us also use the variable

$$m_{\max,\max}=\max_t m_{\max}(t). \quad (45)$$

It is tacitly assumed that an initial state is a product state; otherwise the decomposition for the initial state possibly requires a resource increasing exponentially in n and spoil the estimation below. The decomposition of an initial state needs $O(n)$ time to store values and $\Theta(n)$ memory space on that assumption.

First, we estimate the required memory space. The tensor $Q_s(i_s,v_{s-1},v_s)$ needs a stack of $2m_{s-1}m_s$ floating-point memory blocks. Each memory block has the real part and the imaginary part of a complex number, each of which comprises n_{prec} binary digits to have the precision (namely, the smallest distinguishable number in a signed mantissa portion) of $\Omega(2^{-n_{\mathrm{prec}}})$. $V_s(v_s)$ needs a stack of m_s floating-point memory blocks. In addition, the calculation of Θ, $\rho^{l+1\ldots n}$, etc. uses a memory space proportional to at most $2m_{\max}^2$. Therefore, the simulation of n-qubit quantum computation using the MPS decomposition requires

$$O(m_{\max,\max}^2 n n_{\mathrm{prec}}) \quad (46)$$

memory space. In general, we need the precision less than $1/2^n$ because the elements of the tensor V_s are square roots of the eigenvalues of the reduced density operator $\rho^{s+1,\ldots,n}$. How small precision is required is dependent on a quantum algorithm and input states. In the simulation of fast unsorted-database searching, $n_{\mathrm{prec}}=\Omega(n)$.

Second, we estimate the required number of time steps. This is dependent on the number g of quantum gates in a quantum circuit after it is decomposed into one-qubit or two-qubit quantum gates. We first evaluate how many arithmetic operations are used for simulations of individual quantum gates: The simulation of the time evolution for a single-qubit quantum gate on qubit l requires $O(m_{l-1}m_l)$ floating-point basic operations. The simulation for a unitary transformation acting on two consecutive qubits requires $O(m_{\max}^3)$ floating-point basic operations. This is due to the computation of Eq. (35) and the diagonalization. [The diagonalization of $\rho^{l+1,\ldots,n}$ needs $O(m_{l+1}^3)$ operations.] Next, it is known that $O(n)$ quantum SWAP gates can make any two qubits consecutive. Therefore, the required number of floating-point basic operations is

$$N_{\mathrm{fp}}=O(gm_{\max,\max}^3 n). \quad (47)$$

A single floating-point arithmetic operation is often regarded as an $O(1)$ time step in computer programs because most CPUs can do it in a few clocks. Nevertheless, this is not correct when n_{prec} is larger than the length that a CPU can read and write in $O(1)$ clocks. It is more accurate to write that the number of required time steps is

$$N_{\mathrm{time}}=O[gm_{\max,\max}^3 n\times\mathrm{poly}(n_{\mathrm{prec}})]. \quad (48)$$

Of course, the factor n is omitted when the number of SWAP gates for making nonconsecutive qubits consecutive is included in the factor g.

The value of $m_{\max,\max}$ is dependent on a quantum circuit that should be simulated. Thus the above estimation of complexities is incomplete in this sense. Complexities of a particular simulation can be calculated by tracking the time evolution of $m_{\max}$ to find its largest value.

It is thus important to find an increase of the value of m_s through a single simulation time step that involves a single

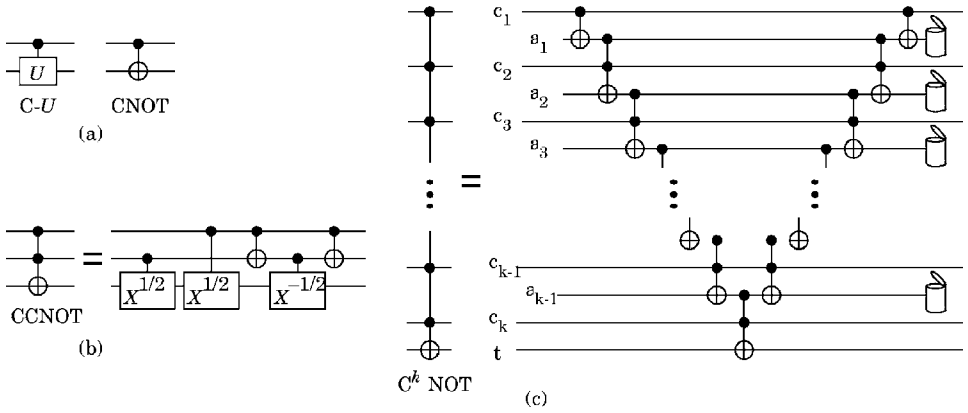

FIG. 2. (a) The illustration of the controlled-U gate and the CNOT gate. (b) The CCNOT gate composed of the CNOT gate, the controlled-$X^{1/2}$ gate, and the controlled-$X^{-1/2}$ gate. (c) The c^kNOT gate composed of the CNOT gates and CCNOT gates (see Ref. [22]). $k-1$ ancillary qubits are used. The garbages (the qubits thrown into garbage cans) are collected later.

gate so as to calculate complexities of a particular simulation of one's interest. Let us consider $m_l(t)$, the number of elements of tensor V_l at time step t. First, it is obvious that $m_l(t+1)=m_l(t)$ when a gate is a single-qubit gate. Second, when a gate is a two-qubit gate acting on the lth and $(l+1)$th qubits

$$m_l(t+1) \leqslant 2m_{l+1}(t). \tag{49}$$

This is directly derived from Eq. (36) in the diagonalization of a density operator under the basis $\{|i_{l+1}\Phi^{l+2,\ldots,n}_{v_{l+1}}\rangle\}$. m_{l+1} is unchanged because the updated tensors are Q_l, V_l, and Q_{l+1}.

D. Some techniques of the MPS simulation

1. c^kNOT gates in the simulation

CNOT gate. Controlled-U gates are gates with two input wires, with which (2,2) matrix U is applied to the lower qubit of states in which the upper is in $|1\rangle$,

$$\text{controlled-}U = \begin{pmatrix} 1 & & \\ & 1 & \\ & & U \end{pmatrix}. \tag{50}$$

U may be any (2,2) unitary transformation. When U is set to $X=\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, the gate is called the CNOT gate. The graphical symbols of them used in quantum circuits are illustrated in Fig. 2(a). The time evolution caused by controlled-U gates is simulated in procedure 2 as we have seen.

CCNOT *gate*. The CCNOT gate is a gate with three inputs, in which X is applied to the last input qubit of states that have the upper two qubits in $|11\rangle$. This can be realized by the quantum circuit composed of two-qubit quantum gates as shown in Fig. 2(b). In the figure, $X^{1/2}=\frac{1}{2}\left(\begin{smallmatrix} 1+i & 1-i \\ 1-i & 1+i \end{smallmatrix}\right)$ and $X^{-1/2}=(X^{1/2})^\dagger$. The simulation of this circuit is realized by the procedure 2.

c^kNOT for $k \geqslant 3$. c^kNOT for $k \geqslant 3$ is a quantum gate with k control qubits $c_1, \ldots, c_k$ and a target qubit t. This gate applies X to the target qubit of states in which all the control qubits are in $|1\rangle$. This can be realized by the quantum circuit composed of the CNOT gates and CCNOT gates, as was written by Kawaguchi *et al.* [22], using $k-1$ ancillary qubits $a_1, \ldots, a_{k-1}$ as shown in Fig. 2(c). The method to use ancillary qubits in the simulation is described below.

2. How to use ancillary qubits in a simulation

It is possible to use dynamic memory allocations to add and delete ancillary qubits in the following simulation scheme. We impose a condition: ancillary qubits must be disentangled with the other qubits before and after use; otherwise they should not be called ancillary qubits.

Before adding ancillary qubits, the state is represented in the MPS form

$$|\Psi\rangle = \sum_{i_1,\ldots,i_n} \sum_{v_1,\ldots,v_{n-1}} Q_1 V_1, \ldots, Q_{n-1} V_{n-1} Q_n |i_1, \ldots, i_n\rangle. \tag{51}$$

To add ancillary qubits, we first allocate a new memory to keep the state of ancillary qubits in the style of the MPS

$$|\phi_a\rangle = \sum_{a_1,\ldots,a_l=0\cdots0}^{1,\ldots,1} \left[\sum_{w_1,\ldots,w_{l-1}} P_1(a_1,w_1) W_1(w_1) \times P_2(a_2,w_1,w_2) \cdots W_{l-1}(w_{l-1}) P_l(a_l,w_{l-1}) \right] \times |a_1, \ldots, a_l\rangle. \tag{52}$$

For ancillary qubits, we write tensors and variables as P_s, W_s, a_s, and w_s instead of Q_s, V_s, i_s, and v_s respectively. It is usual to have all the ancillary qubits in $|0\rangle$ initially. Then $P_s(0,1,1)=1$, $P_s(1,1,1)=0$, and $W_s(1)=1$ for all s. We use the tensor $V_a(v_a)$ [it contains $V_a(1)=1$] to concatenate $|\phi_a\rangle$ and $|\Psi\rangle$. The state $|T\rangle$ of the total of qubits after adding ancillary qubits is written as

$$|T\rangle = \sum_{v_a=1}^{1} V_a(v_a) |\phi_{a,v_a}\rangle |\Psi_{v_a}\rangle, \tag{53}$$

where $|\phi_{a,v_a}\rangle$ has the same tensors as $|\phi_a\rangle$ except for $P_l(a_l,w_{l-1},v_a)$ and $|\Psi_{v_a}\rangle$ has the same tensors as $|\Psi\rangle$ except for $Q_1(i_1,v_a,v_1)$. In short, we concatenate the two MPS $|\phi_a\rangle$ and $|\Psi\rangle$ by inserting the intermediate layer $P_l(a_l,w_{l-1},v_a)V_a(v_a)Q_1(i_1,v_a,v_1)$. Because the element of V_a is $V_a(1)=1$ at this point, the values of $P_l(a_l,w_{l-1})$ and $Q_1(i_1,v_1)$ are kept unchanged in $P_l(a_l,w_{l-1},v_a)$ and $Q_1(i_1,v_a,v_1)$, respectively.

At the next step, SWAP gates are used to put ancillary qubits to some locations where they are needed. Then the time evolution under a quantum circuit using ancillary qubits

is simulated. After this simulation, SWAP gates are used again to gather ancillary qubits to the original position. There is no entanglement between the block of ancillary qubits and the block of the rest qubits if the imposed condition holds. Hence, the state after the time evolution is

$$|T'\rangle = \sum_{v_a=1}^{1} V_a(v_a)|\phi'_{a,v_a}\rangle|\Psi'_{v_a}\rangle. \quad (54)$$

Finally, we delete the memory allocated to the tensors for ancillary qubits after the time evolution, $P'_1, W'_1, \ldots, W'_{l-1}, P'_l$, and the tensor V_a. By replacing $Q'_1(i_1, v_a, v_1)$ with $Q'_1(i_1, v_1)$, we have the state after truncating ancillary qubits out

$$|\Psi'\rangle = \sum_{i_1,\ldots,i_n} \sum_{v_1,\ldots,v_{n-1}} Q'_1 V'_1, \ldots, Q'_{n-1} V'_{n-1} Q'_n |i_1, \ldots, i_n\rangle. \quad (55)$$

3. The reduced density operator of a single qubit

It is easy to obtain the reduced density operator of a single qubit from the state in the MPS representation. Let the state for the whole system be $|\Psi\rangle$. Let the set of kept eigenvectors for the qubits $1, \ldots, l-1$ be $\{|\Phi^{1,\ldots,l-1}_{v_{l-1}}\rangle\}$ and the set of those for the qubits $l+1, \ldots, n$ be $\{|\Phi^{l+1,\ldots,n}_{v_l}\rangle\}$. Then, $|\Psi\rangle$ may be written as

$$|\Psi\rangle = \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \sum_{i_l=0}^{1} V_{l-1}(v_{l-1}) Q_l(i_l, v_{l-1}, v_l) V_l(v_l) \times |\Phi^{1,\ldots,l-1}_{v_{l-1}}\rangle|i_l\rangle|\Phi^{l+1,\ldots,n}_{v_l}\rangle. \quad (56)$$

The density operator for the whole system is

$$\begin{aligned}|\Psi\rangle\langle\Psi| = \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \sum_{i_l=0}^{1} \sum_{v'_{l-1}=1}^{m_{l-1}} \sum_{v'_l=1}^{m_l} \sum_{i'_l=0}^{1} [V_{l-1}(v_{l-1}) \\ \times Q_l(i_l, v_{l-1}, v_l) V_l(v_l) V_{l-1}(v'_{l-1}) \\ \times Q^*_l(i'_l, v'_{l-1}, v'_l) V_l(v'_l) |\Phi^{1,\ldots,l-1}_{v_{l-1}}\rangle|i_l\rangle|\Phi^{l+1,\ldots,n}_{v_l}\rangle \\ \times \langle\Phi^{1,\ldots,l-1}_{v'_{l-1}}|\langle i'_l|\langle\Phi^{l+1,\ldots,n}_{v'_l}|]. \end{aligned} \quad (57)$$

The reduced density operator of the lth qubit is given by

$$\rho^l = \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \langle\Phi^{1,\ldots,l-1}_{v_{l-1}}|\langle\Phi^{l+1,\ldots,n}_{v_l}|\Psi\rangle\langle\Psi|\Phi^{1,\ldots,l-1}_{v_{l-1}}\rangle|\Phi^{l+1,\ldots,n}_{v_l}\rangle. \quad (58)$$

This leads to that

$$\begin{aligned}\rho^l &= \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \sum_{i_l=0}^{1} \sum_{i'_l=0}^{1} [V_{l-1}(v_{l-1}) Q_l(i_l, v_{l-1}, v_l) V_l(v_l) \\ &\quad \times V_{l-1}(v_{l-1}) Q^*_l(i'_l, v_{l-1}, v_l) V_l(v_l) |i_l\rangle\langle i'_l|] \\ &= \sum_{v_{l-1}=1}^{m_{l-1}} \sum_{v_l=1}^{m_l} \sum_{i_l=0}^{1} \sum_{i'_l=0}^{1} \{[V_{l-1}(v_{l-1})]^2 [V_l(v_l)]^2 \\ &\quad \times Q_l(i_l, v_{l-1}, v_l) Q^*_l(i'_l, v_{l-1}, v_l) |i_l\rangle\langle i'_l|\}. \end{aligned} \quad (59)$$

Therefore, the calculation of ρ^l is accomplished within $\Theta(m_{l-1}m_l)$ floating-point basic operations.

4. Simulation of a projective measurement on a single qubit

It is also possible to simulate the time evolution of a state in the process of a projective measurement on a single qubit. This is not used in simulations that we will see in Sec. VI because we do not consider a standard model of quantum computation but a bulk-ensemble model. Nevertheless, it is necessary if one intends to simulate a standard quantum computation that involves projective measurements. We will easily find that the number of floating-point operations used for the simulation of a projective measurement on a single qubit is $O(m^3_{\max,\max} n)$. The details are available in a supplementary electronic file [27].

5. A simple example of the MPS simulation

To understand the process of the MPS simulation, an example is helpful. A simple example is available as an additional electronic file [27]. In this electronic file, we will see the simulation process of a time evolution of elements of tensors for a simple quantum circuit. The simulation process of an projective measurement on a single qubit is also written in the example.

V. EXTENSIONS OF BRÜSCHWEILER'S ALGORITHM

We will show two extensions of Brüschweiler's algorithm.

A. Single-query algorithm

In the following algorithm, one has only to send a single query to an oracle with a single solution to find it with $O(n)$ postoracle quantum gates. It is an algorithm similar to one that Xiao and Long [18] published. The different part is that we do not use the frequencies of peaks caused by J or DD couplings in observation but standard ensemble average measurements with the observable Z.

The problem is the same one as we have had before: Consider the function $f:\{0,1\}^n \mapsto \{0,1\}$ such that $f(\mathbf{x})=0$ for all inputs $\mathbf{x}$ except for the input $\mathbf{w}$ for which it returns 1. The problem is to find $\mathbf{w}$.

Algorithm. We utilize $2n+1$ qubits in the state

$$\frac{1}{2^n}[(|0,0\rangle + |1,1\rangle)^{\otimes n}_A |0,0\rangle_o (p|0,0\rangle + q|1,1\rangle)^{\otimes n}_B], \quad (60)$$

where the subscripts A, o, and B denote the block A with n qubits (with the initial polarization of 0), the oracle qubit, and the block B with n qubits (with the initial polarization of $p-q$), respectively. The block A is input to the oracle function [Eq. (11)] that has a solution $\mathbf{w}=w_1, \ldots, w_n$. In the output of it, we have the state

$$\begin{aligned}&\frac{1}{2^n}|\mathbf{w},\mathbf{w}\rangle_A |1,1\rangle_o (p|0,0\rangle + q|1,1\rangle)^{\otimes n}_B \\ &+ \frac{1}{2^n} \sum_{\mathbf{k}\neq\mathbf{w}} [|\mathbf{k},\mathbf{k}\rangle_A |0,0\rangle_o (p|0,0\rangle + q|1,1\rangle)^{\otimes n}_B]. \end{aligned} \quad (61)$$

The C-SWAP gate is used n times, with the oracle qubit as the control qubit, to swap A and B for the states with the oracle qubit in $|1,1)$. The state becomes

$$\frac{1}{2^n}(p|0,0) + q|1,1))_A^{\otimes n}|1,1)_o|\mathbf{w},\mathbf{w})_B + \frac{1}{2^n}\sum_{\mathbf{k}\neq\mathbf{w}}[|\mathbf{k},\mathbf{k})_A|0,0)_o(p|0,0) + q|1,1))_B^{\otimes n}]. \quad (62)$$

An ensemble measurement is performed on each of n qubits of the block B with the observable Z to get signals $s_1,\ldots,s_n$. s_i is the polarization of the ith qubit, $(p-q)(1-2^{-n})\pm 2^{-n}$. This measurement can be done simultaneously if we can use different frequencies for n qubits originated from chemical shifts and/or different gyromagnetic ratios in an experiment (it requires at most n different NMR probes). To get the bit w_i, we check s_i. If $s_i > (p-q)(1-2^{-n})$, w_i is found to be 0; otherwise it is found to be 1. Consequently, $\mathbf{w}=w_1,\ldots,w_n$ is found by using $O(n)$ quantum gates subsequent to the oracle if the variation of signals, $\pm 2^{-n}$, is large enough to detect. In NMR experiments, this leads to an exponential increase in the number of experiments to find the sign of the variation disturbed by random noise.

B. Algorithm for the oracle with multiple solutions—How to find all of them

It is possible to extend Brüschweiler's algorithm to work with an oracle with multiple solutions. It is tacitly assumed that the number r of solutions may be unknown in advance.

A database of present interest is similar to the one that we have had in Sec. III. It is a set of $N=2^n$ records each of which has an n-bit-length key. r keys are marked but they are unknown in public. A table of keys is open to the public, except for marking information. We call this database in three different ways according to its style of a query-and-answer behavior.

(a′) It is called an unsorted database when it returns yes or no when one queries it as to whether a key is one of marked keys.

(b′) It is called a sorted database when it answers a query as to whether a key is equal to, larger than, or smaller than the nearest one of marked keys.

(c′) It is called a partition-structured database when one can query as to whether a part of a key is equal to that of some of marked keys.

The problem of unsorted-database searching in the present context is to find all the marked keys in a database (a'). The mathematical definition is as follows.

Definition 2: The problem of unsorted-database searching for an oracle with multiple solutions is defined as follows.

Consider the function $f:\{0,1\}^n \mapsto \{0,1\}$ such that $f(\mathbf{x})=0$ for all inputs $\mathbf{x}$ except for r different inputs $\mathbf{w}_1,\ldots,\mathbf{w}_r$ with which $f(\mathbf{w}_1)=\cdots=f(\mathbf{w}_r)=1$. The problem is to find the all of $\mathbf{w}_1,\ldots,\mathbf{w}_r$.

Outline of the algorithm. To solve this problem, we add the wire of an ancillary qubit a beneath the wires of n qubits and an oracle qubit. We will use subscripts o and a to represent the state space of an oracle qubit and that of an ancillary qubit, respectively. The oracle is realized as a quantum gate that takes the input string $x_1,\ldots,x_n 0_o$ and outputs the string $x_1,\ldots,x_n f(\mathbf{x})_o$ (here, $\mathbf{x}=x_1,\ldots,x_n$ and $x_i\in\{0,1\}$). Here is an outline: In every step to survey the ith bit of solutions, we prepare the input state such that the ith qubit, the oracle qubit, and the ancillary qubit are in $|0,0)$ and the rest in maximally mixed states. A signal intensity measured after the oracle function is proportional to the polarization $\langle Z_{\text{oracle}}\rangle$ of the oracle qubit in starting steps and in later steps the polarization $\langle Z_a\rangle$ of the ancillary qubit, which is one of $1,1-2^{-n+2},1-2\times 2^{-n+2},\ldots,1-r\times 2^{-n+2}$. Using measured signals, all the solutions will be found. The details are as follows.

Algorithm. In the first step, the initial state is set to $2^{-n+1}|0,0)[|0,0)+|1,1)]^{\otimes n-1}|0,0)_o|0,0)_a$. The oracle function is applied to this state. This operation inverts the oracle qubit of the states that have a solution with 0 in the first bit. $\langle Z_{\text{oracle}}\rangle$ is measured, which is then equal to $s_1=1-c_1 2^{-n+2}$. We find that c_1 solutions among r solutions have 0 in their first bits. When the value of r is unknown in advance, the oracle function is also applied to the initial state $2^{-n+1}|1,1)[|0,0)+|1,1)]^{\otimes n-1}|0,0)_o|0,0)_a$ to get the value of $\langle Z_{\text{oracle}}\rangle$, $s_1'=1-c_1' 2^{-n+2}$. $r=c_1+c_1'$ is immediately found in this way. We label the solutions that have 0 in the first bit by $\mathbf{w}_1,\ldots,\mathbf{w}_{c_1}$. The solutions with the first bit 1 are $\mathbf{w}_{c_1+1},\ldots,\mathbf{w}_r$.

After the first step, we continue to use the following subroutine. This subroutine is used at every node except for the root node, bottom nodes, and nodes from which no solution will be found in the tree illustrated in Fig. 4 [an example is shown in Fig. 4(a)].

Subroutine. Suppose that we are trying to determine a solution $\mathbf{w}_j$ and we have already determined its first k bits, $w_{j,1},\ldots,w_{j,k}$. Suppose we have found that m solutions $\mathbf{w}_j,\ldots,\mathbf{w}_{j+m-1}$ have the same sequence $w_{j,1},\ldots,w_{j,k}$ in their first k bits, i.e., $w_{j,1},\ldots,w_{j,k}=w_{j+1,1},\ldots,w_{j+1,k}=\cdots=w_{j+m-1,1},\ldots,w_{j+m-1,k}$. Then we use the following method to determine the $(k+1)$th bit in each of those solutions.

The initial state is set to $2^{-n+1}[|0,0)+|1,1)]^{\otimes k}|0,0)[|0,0)+|1,1)]^{\otimes n-k-1}|0,0)_o|0,0)_a$. After the oracle function is used, the state is

$$2^{-n+1}\left[\sum_{\mathbf{x}}|\mathbf{x},\mathbf{x})|1,1)_o|0,0)_a + \sum_{\mathbf{y}\notin\{\mathbf{x}\}}|\mathbf{y},\mathbf{y})|0,0)_o|0,0)_a\right], \quad (63)$$

where $\mathbf{x}$ are solutions with 0 in the $(k+1)$th bit among r solutions and $\mathbf{y}$ are other strings having 0 in the $(k+1)$th bit. Then we operate with the transposition

$$(w_{j,1},\ldots,w_{j,k}1_o 0_a,\ w_{j,1},\ldots,w_{j,k}1_o 1_a) \quad (64)$$

on the qubits $1,\ldots,k$, the oracle qubit, and the ancillary qubit. This can be composed of the C^{k+1}NOT gate and NOT gates. The state becomes

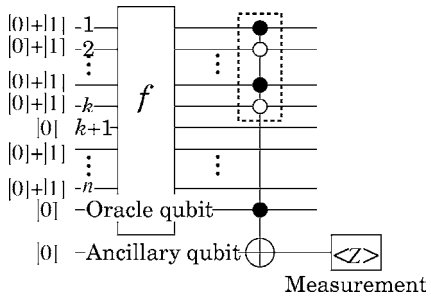

FIG. 3. The subroutine illustrated as a quantum circuit. The dashed box surrounds control qubits each of which may be zero control (open circle) or one control (solid circle) according to the value of $w_{j,s}$ ($s=1,\ldots,k$) in Eq. (64). $|0]\equiv|0,0)$ and $|1]\equiv|1,1)$.

$$2^{-n+1}\Bigg[\sum_{\mathbf{x}'}|\mathbf{x}',\mathbf{x}')|1,1)_o|1,1)_a+\sum_{\mathbf{x}''}|\mathbf{x}'',\mathbf{x}'')|1,1)_o|0,0)_a + \sum_{\mathbf{y}\notin\{\mathbf{x}\}}|\mathbf{y},\mathbf{y})|0,0)_o|0,0)_a\Bigg], \tag{65}$$

where $\mathbf{x}'$ are those with the sequence $w_{j,1},\ldots,w_{j,k}$ in the first k qubits among $\mathbf{x}$ and $\mathbf{x}''$ are those without the sequence among $\mathbf{x}$. We conduct a measurement on a with Z to get $\langle Z_a\rangle=1-h2^{-n+2}$. h is the number of those represented as $\mathbf{x}'$. We find that $w_{j,k+1}=\cdots=w_{j+h-1,k+1}=0$ and $w_{j+h,k+1}=\cdots=w_{j+m-1,k+1}=1$. In this way, the $(k+1)$th bit is found in each of solutions $\mathbf{w}_j,\ldots,\mathbf{w}_{j+m-1}$. By this subroutine, two children are created from a node as illustrated in Fig. 4(b).

This subroutine may be illustrated as a quantum circuit of Fig. 3. In the figure, the dashed box surrounds control qubits each of which may be zero control (open circle) or one control (solid circle) according to the value of $w_{j,s}$ ($s=1,\ldots,k$) in Eq. (64). As we have mentioned, this subroutine is used in every node except for the root node, bottom nodes, and nodes from which no solution is located, from upper nodes to lower nodes. The whole of this scheme is done with $O(nr)$ queries.

Let g^{oracle} be a number of one-qubit or two-qubit quantum gates composing an oracle quantum circuit. Then $O(\max(g^{\text{oracle}},n)nr)$ quantum gates are involved in total. For example of an oracle structure, consider the oracle function internally decomposed into r C^kNOT gates and $O(nr)$ NOT gates. Then $O(n^2r^2)$ quantum gates are involved in total. One cannot find a computational time and/or space complexity correctly as long as the internal circuit of the oracle function is unknown and/or ignored.

An example of the use of the algorithm is shown in Fig. 4(a). In the example, the oracle has solutions $\mathbf{w}_1=0100,\mathbf{w}_2=0101,\mathbf{w}_3=1011,\mathbf{w}_4=1100$. At every node k from the root ($k=0$) in the tree, we use the above algorithm with that subroutine to find how many solutions have 0 or 1 in the bit next to the already-determined sequence $w_{j,1},\ldots,w_{j,k}$.

The above problem is to find all the solutions. If it is enough to find a single solution among $\mathbf{w}_1,\ldots,\mathbf{w}_r$, a single path from the root to a final node in the tree is taken. Therefore, it is terminated in $O(n)$ queries to find a single solution. The reason why we impose the task of finding all the solutions on that problem is because it is a natural situation where one needs to find all the solutions.

There is a benefit of this algorithm in comparison with the Xiao-Long algorithm that can find solutions with a single query. The Xiao-Long algorithm uses the split of frequency peaks caused by J or DD couplings in finding solutions—the inversion of peaks owing to an oracle function are used for the purpose. Therefore, it is possible that one cannot find some of solutions if one does not know some labels of peaks and/or if some peaks overlap each other. In contrast, in the above algorithm, one can find all the solutions by using standard average measurements of the polarization of an oracle qubit. It is more robust against the overlap of frequency peaks, which is a common phenomenon for a large number of qubits. In addition, it has another benefit: the simulation of polarization measurements can be performed efficiently in the process of the MPS simulation because the reduced density operator of a single qubit can be achieved quickly as was described in Sec. IV D 3.

VI. SIMULATION OF BRÜSCHWEILER'S DATABASE SEARCH AND COMPLEXITIES

This section is to describe explicit methods to conduct simulations for the three algorithms that we have seen. It is tacitly assumed that the quantum circuit of an oracle function f is composed of NOT gates and r C^nNOT gates (r is the number of solutions) unless another construction is stated by sentences. Computational complexities are dependent on a circuit construction of an oracle $f:(x,0)\rightarrow(x,f(x))$; a consumed resource can be larger when a more complicated

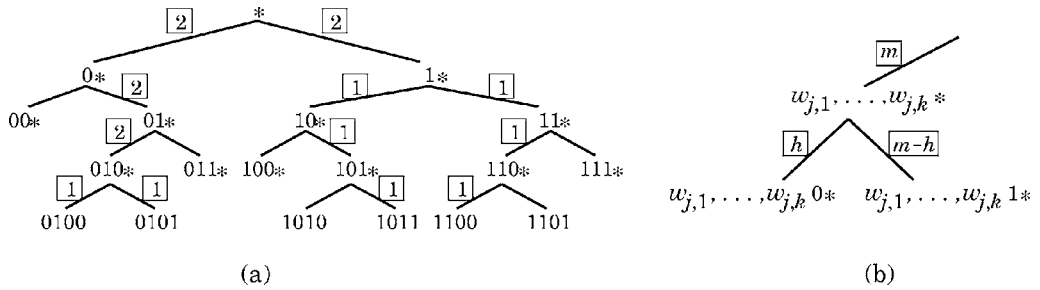

FIG. 4. A tree in which we track the number of solutions from the root in order to determine all solutions. A number in a solid box is the number of solutions. See the text for details. (a) An example where solutions are 0100, 0101, 1011, and 1100. (b) Illustration of how node's children are generated by a call of the subroutine.

construction is chosen. If it is ignored, one cannot quantify the computational complexity of a simulation except for the query complexity that has already been found $O(nr)$. When we speak of a complexity of an MPS simulation, $m_{\max}=m_{\max}(t)$ defined in Eq. (44) is the main source with which a complexity is estimated. t is a time step and it points a horizontal position in a quantum circuit. We also use Eqs. (45)–(49) for deriving complexities. (Note: Although the symbol n was used to denote the number of qubits in those equations, n is the bit length of a key in the present context.) Hereafter, the largest value of $m_{\max}$ in the internal oracle circuit is expressed as $m_{\max,\max}^{\text{oracle}}$ and that in the circuits outside the oracle is expressed as $m_{\max,\max}^{\text{circuit}}$ [hence, $m_{\max,\max}=\max(m_{\max,\max}^{\text{oracle}},m_{\max,\max}^{\text{circuit}})$ holds].

A. Simulation of the Brüschweiler's original algorithm

The Brüschweiler's original algorithm that we have seen in Sec. III can be simulated in an MPS simulation. Let us set $p=0.5$, i.e., $|\gamma)=0.5|0,0)+0.5|1,1)$, to simplify the simulation. As shown by Zwolak and Vidal [17], a time evolution of a mixed state may also be simulated by Vidal's MPS method with some technique in the choice of a basis. Nevertheless, we use a simpler technique to perform a simulation because Brüschweiler's algorithm uses operations only for exchanging populations of bit strings. We use the notations: $|0]=|0,0)=|0\rangle\langle 0|$ and $|1]=|1,1)=|1\rangle\langle 1|$. In Brüschweiler's algorithm, $|0,1)$ and $|1,0)$ are not used. Therefore, we consider the subspace spanned by basis vectors $|0\cdots 0],\ldots,|1\cdots 1]$. In this notation, the maximally mixed state of a qubit is written as $\frac{|0]+|1]}{2}$, and the maximally mixed state of τ qubits is $\frac{1}{\sqrt{2}^{\tau}}\left(\frac{|0]+|1]}{\sqrt{2}}\right)^{\otimes\tau}$. A τ-qubit state of an ensemble that is represented as a diagonal density matrix can also be written in the above notation

$$|\rho(t)]=\sum_{i_1,\ldots,i_\tau=0\cdots 0}^{1\cdots 1}c_{i_1,\ldots,i_\tau}(t)|i_1,\ldots,i_\tau], \tag{66}$$

where t is time step and $c_{i_1,\ldots,i_\tau}$ is the population of $|i_1,\ldots,i_\tau]$. Unless a time evolution of this vector involves a vector out of the subspace, such a time evolution can be simulated with the MPS method because the subspace spanned by $\{|k]\}_{k=0}^{2^\tau-1}$ is equivalent to the space spanned by $\{|k\rangle\}_{k=0}^{2^\tau-1}$. One can use the same computer-program library of MPS as that handling pure states. This is clear from the following explanation: A time evolution of $|\rho(t)]$ caused by a permutation operation U_p with respect to computational basis vectors (i.e., a conditional logical single/multiple-bit-flip operation usually consisting of the NOT gate, the CNOT gate, the CCNOT gate, etc.) can be simulated by the time evolution of

$$|\Psi_\rho(t)\rangle=\sum_{i_1,\ldots,i_\tau=0\cdots 0}^{1\cdots 1}\sqrt{c_{i_1,\ldots,i_\tau}(t)}|i_1,\ldots,i_\tau\rangle \tag{67}$$

using U_p. There is no difficulty in simulating exchanges of values among $\{c_{i_1,\ldots,i_\tau}(t)\}$; one has only to regard superkets as ket vectors and simulate a time evolution. For example, $p|0]+(1-p)|1]$ $(0\leqslant p\leqslant 1)$ is handled as if they were $\sqrt{p}|0\rangle+\sqrt{1-p}|1\rangle$ internally in a computer simulation program. Therefore, the MPS decomposition

$$\begin{aligned}|\Psi_\rho\rangle=\sum_{i_1,\ldots,i_\tau=0\cdots 0}^{1\cdots 1}\Bigg[&\sum_{v_1,\ldots,v_{\tau-1}=1\cdots 1}^{m_1,\ldots,m_{\tau-1}}Q_1(i_1,v_1)V_1(v_1)\\&\times Q_2(i_2,v_1,v_2)\cdots Q_{\tau-1}(i_{\tau-1},v_{\tau-2},v_{\tau-1})\\&\times V_{\tau-1}(v_{\tau-1})Q_\tau(i_\tau,v_{\tau-1})\Bigg]|i_1,\ldots,i_\tau\rangle\end{aligned} \tag{68}$$

can be used for the simulation of the time evolution of $|\rho]$. Using the tensors $Q_1,\ldots,Q_n,V_1,\ldots,V_{n-1}$ of Eq. (68), $|\rho]$ can be expressed as

$$\begin{aligned}|\rho]=\sum_{i_1,\ldots,i_\tau=0\cdots 0}^{1\cdots 1}\Bigg[&\sum_{v_1,\ldots,v_{\tau-1}=1,\ldots,1}^{m_1,\ldots,m_{\tau-1}}Q_1(i_1,v_1)V_1(v_1)\\&\times Q_2(i_2,v_1,v_2)\cdots Q_{\tau-1}(i_{\tau-1},v_{\tau-2},v_{\tau-1})\\&\times V_{\tau-1}(v_{\tau-1})Q_\tau(i_\tau,v_{\tau-1})\Bigg]^2|i_1,\ldots,i_\tau].\end{aligned} \tag{69}$$

In this way, a time evolution of $|\rho]$ may be calculated from the MPS simulation of a time evolution of $|\Psi_\rho\rangle$ without changing computational procedures. For the search algorithm of present interest, $\tau=\Theta(n)$ when τ is the total number of qubits.

The quantum circuit that should be simulated is one illustrated in Fig. 1 in Sec. III. In the circuit, the ith qubit is set to $|0]$ in the input state at the ith step in the Brüschweiler scheme. Then an oracle function comprising the NOT and C^nNOT gates is used. There is an assumption that the oracle has a single solution in his original algorithm. Then, measuring the polarization $\langle Z_{\text{oracle}}\rangle$ of the oracle qubit, we find that the ith bit w_i of the solution $\mathbf{w}$ is 0 when $\langle Z_{\text{oracle}}\rangle<1$ is apparent; otherwise it is 1. This step is easy because the density operator of the oracle qubit is calculated in $O(m_{\max,\max}^{\text{circuit}\ 2})$ basic floating-point operations in the simulation.

The reduced density operator ρ_s of a single qubit s is obtained using the same routine as we have seen in Sec. IV D 3. First, we make the reduced density operator $\hat{\rho}_s$ of the sth qubit from the MPS decomposition of $|\Psi_\rho(t)\rangle$. Second, we delete off-diagonal elements of $\hat{\rho}_s$. Third, we normalize it to avoid a propagation of a numerical error. Then we have ρ_s. This is, of course, because there is no term that comprises $|0,1)$ and/or $|1,0)$ in the process of Brüschweiler's search.

Let us look at the amount of computational resources required in the simulation step by step. For a moment, we omit coefficients and write superkets only. From $i=1$ to n, the iteration of the following routine is used: First, the initial state is $(|0]_1+|1]_1)\cdots(|0]_{i-1}+|1]_{i-1})|0]_i(|0]_{i+1}+|1]_{i+1})\cdots(|0]_n+|1]_n)|0]_o$ (o represents the oracle qubit) stored as an MPS with $m_{\max}=1$. This goes into the oracle function. In the oracle function, $n-1$ ancillary bits are inserted: $(|0]_1+|1]_1)|0]_a\cdots|0]_a(|0]_{i-1}+|1]_{i-1})|0]_a|0]_i|0]_a(|0]_{i+1}+|1]_{i+1})|0]_a\cdots|0]_a(|0]_n+|1]_n)|0]_o$ by using SWAP gates with $m_{\max}=1$ unchanged (or indeed, it is

possible to prepare this state as the initial state). We assume that an oracle with a single solution is realized by NOT gates and a single C^nNOT gate. Among them, the two NOT gates acting on the ith qubit and the C^nNOT gate do effect on the state. A NOT gate does not change the value of $m_{\max}$. Let us use the decomposition of the C^nNOT gate illustrated in Fig. 2. Then, in the top of the decomposed circuit of the C^nNOT gate, a CNOT gate acting on the first qubit and the first ancillary qubit will change the value of $m_{\max}$ to 2. In considering the other parts in the C^nNOT gate [see Fig. 2(c)], let us take the first CCNOT gate pertaining to the qubits $a_{s-1}c_s a_s$, namely, the $(s-1)$th ancillary qubit, the sth control qubit, and the sth ancillary qubit. Before the CCNOT, $m_{a_{s-1}}=2$ and $m_{c_s}=m_{a_s}=1$. In the CCNOT, realized as the combination of two-qubit operations, a_{s-1} is used as a control bit. Thus, for every value of $v_{a_{s-1}}$, some (4,4) unitary transformation acts on $c_s a_s$ in the simulation. The values of $m_{a_{s-1}}$, m_{c_s}, and m_{a_s} become 2 after the CCNOT gate. In the inside of the CCNOT gate, it is at most 4 [this is calculated from Eq. (49)]. The second CCNOT gate acting on $a_{s-1}c_s a_s$ is to restore the ancillary bits independent. Before the CCNOT gate, the value of m_{a_s} is already restored to 1. In the inside of the CCNOT gate, the value of $m_{a_{s-1}}$ is not more than 4 and the value of m_{c_s} is not more than 2. This is again calculated according to Eq. (49). After the CCNOT gate, the value of $m_{a_{s-1}}$ is restored to 1, but the value of m_{c_s} can be remained 2. In consequence, $m_{\max,\max}^{\text{oracle}}$ is not more than 4, and the C^nNOT gate changes the value of $m_{\max}$ from 1 to 2, which is the value of $m_{\max,\max}^{\text{circuit}}$. Thus the simulation time is $O(n^2\text{poly}(n_{\text{prec}}))$ and the memory space consumed is $O(nn_{\text{prec}})$.

For convenience, the largest value of $m_{\max}(t)$ for t in the external (internal) circuit of CCNOT gates is represented as $m_{\max,\max}^{\text{ext}}$ ($m_{\max,\max}^{\text{int}}$) hereafter. Here, the internal circuit of CCNOT gates is one illustrated in Fig. 2(b). Then the above result may be stated in other words: the value of $m_{\max,\max}^{\text{ext}}$ is 2 and that of $m_{\max,\max}^{\text{int}}$ is not more than 4.

When an oracle is constructed in a different way, the value of $m_{\max,\max}^{\text{circuit}}$ is unchanged while the value of $m_{\max,\max}^{\text{oracle}}$ may be altered. Let g^{oracle} be the number of one-qubit or two-qubit quantum gates composing the quantum circuit of an oracle. Let $n^{\text{oracle}} \geqslant n$ be the number of qubits used in the oracle quantum circuit [hence, $O(n^{\text{oracle}})$ SWAP gates can make a pair of nonconsecutive qubits consecutive]. Then the simulation time is $O(m_{\max,\max}^{\text{circuit}\ 3} g^{\text{oracle}} n^{\text{oracle}} n\text{poly}(n_{\text{prec}}))$ and the memory space consumed is $O(m_{\max,\max}^{\text{circuit}\ 2} n^{\text{oracle}} n_{\text{prec}})$; these are dependent on $m_{\max,\max}^{\text{oracle}}$ and n^{oracle}. In simulations that we performed, such a different construction was not examined.

It should be noted that a threshold should be set properly in truncation of eigenvalues. Because the size of every register is limited, it is inevitable that rounding operations are accumulated and we cannot judge whether an eigenvalue is actually zero or not. To truncate out eigenvalues of zero, the threshold should be more than the possible largest rounding error. In the present simulation, we can truncate out eigenvalues less than 2^{-n}. (Considering the errors of internal floating-point operations, a practical threshold is $c\times 2^{-n}$ with $c\simeq 0.01$.) This needs the size of each floating-point register more than n.

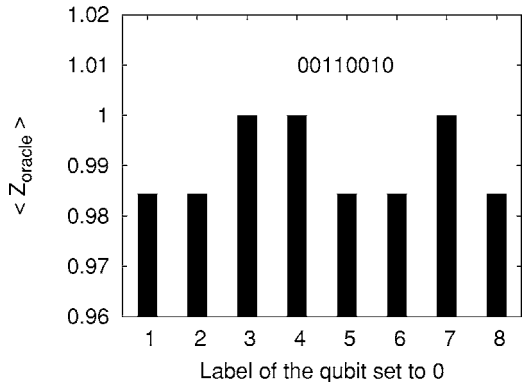

FIG. 6. A simulation result for the example of the solution 00110010. Every bar shows the value of $\langle Z_{\text{oracle}}\rangle$ against the label of a particular qubit that was set to $|0]$ in the input. The original Brüschweiler's algorithm was used.

B. Simulation of the single-query algorithm

Similarly, the algorithm described in Sec. V A is also simulated. The controlled-SWAP (CSWAP) gate used in this algorithm is decomposed into the CNOT and CCNOT gates as shown in Fig. 5. Although the number of quantum gates used in the algorithm is $O(n)$, $O(n^2)$ quantum gates are used in the MPS simulation. This is because the MPS simulation is conducted after decomposing the whole quantum circuit into two-qubit quantum gates acting on consecutive qubits. In addition to the oracle circuit assumed to be composed of the CNOT and the CCNOT gates here, $O(n)$ SWAP gates acting on consecutive qubits are used to decompose every CSWAP gate on dispersed qubits.

In this simulation, first the C^nNOT gate in the oracle changes the value of $m_{\max}$ from 1 to 2 (it may be larger in the inside of each CCNOT gate), which is equal to the value of $m_{\max}$ in its output. Second, CSWAP gates change it to at most 3 (it may be also larger in the inside of each CCNOT gate). According to Eq. (49), the value of $m_{\max}$ in the internal circuit of a CCNOT gate is not more than $3\times 2\times 2=12$. Thus $m_{\max,\max}^{\text{ext}}<4$ and $m_{\max,\max}^{\text{int}}<13$ for this simulation. The time consumed in the simulation is $O(n^2\text{poly}(n_{\text{prec}}))$ and the memory space consumed is $O(nn_{\text{prec}})$. In general for different constructions of an oracle, the running time is $O(m_{\max,\max}^{\text{oracle}\ 3}\max(g^{\text{oracle}},n)n^{\text{oracle}}\text{poly}(n_{\text{prec}}))$ and the consumed memory space is $O(m_{\max,\max}^{\text{oracle}\ 2} n^{\text{oracle}} n_{\text{prec}})$.

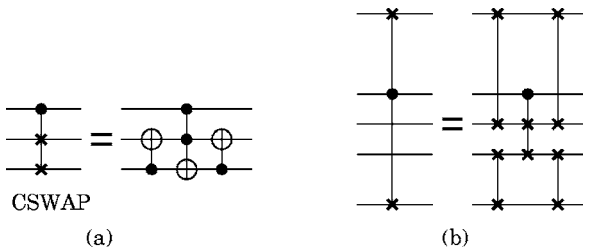

FIG. 5. (a) The CSWAP gate to swap the second and the third qubits under the control of the first qubit. (b) The CSWAP gate for dispersed qubits. The solid circle is the control qubit; when it is 1, SWAP of the other two qubits occurs.

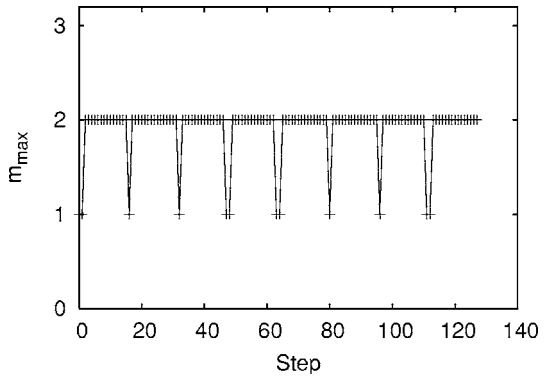

FIG. 7. Another simulation result for the example of the solution 00110010. Data points show the values of $m_{\max}$ in the outside of CCNOT gates against steps in the simulation. We took 16 steps in the decomposition of the C^nNOT gate in a single call of the oracle to find w_i (thus 128 steps in total). The original Brüschweiler's algorithm was used.

C. Simulation of the algorithm for the oracle with multiple solutions

In the simulation of the algorithm for finding multiple solutions, we use the array $l_s[\ ]$ of already-found strings and the array $n_s[\ ]$ of corresponding numbers of solutions, i.e., $n_s[i]$ is the number of solutions that have $l_s[i]$ in their heads. $l_s[\ ]$ looks like $l_s[0]=010, l_s[1]=011, \ldots$ and $n_s[\ ]$ looks like $n_s[0]=3, n_s[1]=2, \ldots$. A pair $(l_s[i], n_s[i])$ is located in the ith location of a location table. The operations of deletion and insertion of a pair in the ith location are equivalent to the operations in the ith elements of both the arrays. We use the algorithm described in Sec. V B in the MPS simulation to update $l_s[\ ]$ and $n_s[\ ]$ step by step. In a call of the subroutine in the algorithm, a pair $(l_s[i], n_s[i])$ makes two pairs (l_1, n_1) and (l_2, n_2) as its children where $l_1=l_s[i]0$ and $l_2=l_s[i]1$; n_1 (n_2) is the number of solutions that have l_1 (l_2) in their heads (hence, $n_1+n_2=n_s[i]$). The parent $(l_s[i], n_s[i])$ is deleted from the location of i and one or both of children with a non-zero value in n_1 (n_2) are inserted in this location. This updates the values of $l_s[i], l_s[i+1], \ldots$ and $n_s[i], n_s[i+1], \ldots$.

Each step involves a single call of an oracle (here, it is assumed to comprise r C^nNOT gates and some NOT gates) and a single C^{k+1}NOT gate. r different computational basis vectors should be separated out from the maximally mixed state in the MPS decomposition of the output state, i.e., we have $r+1$ different combinations of nonzero values of $v_1, v_2, \ldots$ in the MPS decomposition. Therefore, the value of $m^{\text{ext}}_{\max,\max}$ in the simulation is at most $r+1$. In the internal circuit of the CCNOT gate, the value of $m^{\text{oracle}}_{\max,\max}$ (equal to $m^{\text{int}}_{\max,\max}$) is not more than $4r+4$. This is easily calculated by using Eq. (49), although this is possibly an overestimation. This leads to the results: The simulation time is $O(r^5 n^2 \text{poly}(n_{\text{prec}}))$ because $O(m^3_{\max,\max})=O(r^3)$ and $O(r^2 n^2)$ two-qubit quantum gates are used in total in the tree. In addition, the consumed memory space is $O(r^2 n n_{\text{prec}})$.

In general for an unspecified internal construction of an oracle, the running time is

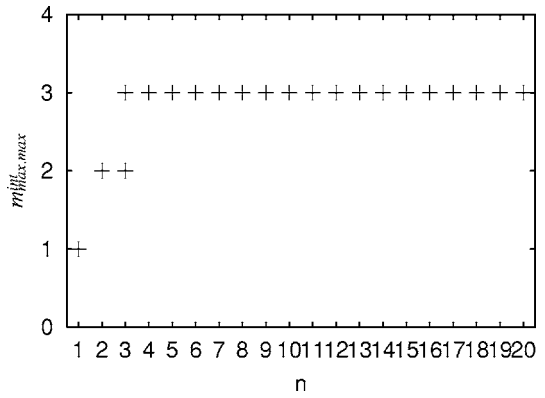

FIG. 8. Plots of the values of $m^{\text{int}}_{\max,\max}$ against $n=\log_2 N$. For each value of n, ten trials were performed, each of which involved a randomly chosen solution of an oracle. Thus multiple data points may appear for a single value of n.

$$O([\max(m^{\text{oracle}}_{\max,\max}, r)]^3 \max(g^{\text{oracle}} n^{\text{oracle}}, n) \times nr \times \text{poly}(n_{\text{prec}})) \tag{70}$$

and the memory space consumed is

$$O([\max(m^{\text{oracle}}_{\max,\max}, r)]^2 n^{\text{oracle}} n_{\text{prec}}), \tag{71}$$

where the function $\max(a,b)$ returns the larger one of integers a and b, g^{oracle} is the number of one-qubit or two-qubit quantum gates composing the quantum circuit of an oracle, and $n^{\text{oracle}} \geqslant n$ is the number of qubits used in the oracle quantum circuit [hence, $O(n^{\text{oracle}})$ SWAP gates can make a pair of nonconsecutive qubits consecutive]. The above running time is easily derived from Eq. (48) by considering compositions of the oracle and postoracle quantum circuits and the number of queries $O(nr)$. The memory space is also easily derived from Eq. (46). Note that n is the bit length of a key in the current context while the symbol n was used to denote the number of qubits in Eqs. (46) and (48).

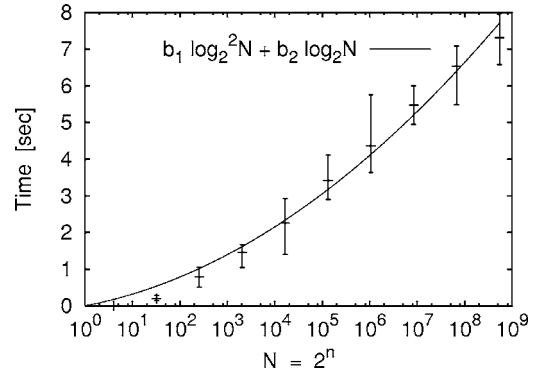

FIG. 9. The plots of the simulation time against the number N of keys in a database. The oracle used here is one with a single solution chosen randomly. The original Brüschweiler's algorithm was used. The solid curve is a least-squares fitting ($b_1=0.00660$ and $b_2=0.0746$) to the data points of average values. The error bars are drawn from the shortest time to the longest time in five different trials.

VII. SIMULATION RESULTS

A simulation library for a general MPS simulation has been made as a class of C++ language. With this library, a C++ class of the oracle function has been written, in which solutions are chosen randomly at the moment that a computational object of the oracle is constructed. This class is separated from the classes for the simulation of other quantum circuits. The computer used for the simulation is the Alpha ES45 Server (EV68 dual CPUs with 32 GB main memory, produced by Compaq Computer Corporation) with a customized Linux™ operating system. The compiler was GCC 4.0.0 with binutils 2.1.5.90 (produced by Free Software Foundation, Inc.). In addition, a simulation time in figures is a real time including a CPU processing time, a memory access time, and I/O times. It is assumed that an oracle is simply constructed by NOT gates and r C^nNOT gates as we stated in the previous section.

A. Simulation results for the Brüschweiler's original algorithm

The simulation described in Sec. VI A was conducted to simulate the Brüschweiler's original algorithm. For every particular value of n, the simulation found the single solution of the function f (randomly chosen) from the values of $\langle Z_{\text{oracle}} \rangle$ obtained in individual steps in the iteration. For an example, the values of $\langle Z_{\text{oracle}} \rangle$ gotten in the simulation to find the eight-bit solution 00110010 are shown in Fig. 6. As we have seen in Sec. VI A, the value of $m^{\text{ext}}_{\text{max,max}}$ is at most 2 irrespectively of n. The time evolution of m_{max} in the out side of CCNOT gates in the same simulation of the example is shown in Fig. 7. In addition, $m^{\text{int}}_{\text{max,max}}$ is at most 4 according to the theory. The simulation results are consistent with it and the value seems 3 for $n \geqslant 4$ as shown in Fig. 8. Thus the value of 4 is probably an overestimate as long as the value of n is in a range for practical use.

The simulation time is expected to be on the curve $b_1 \log_2^2 N + b_2 \log_2 N$ (here, $N=2^n$ and coefficients $b_1, b_2 \in \mathbb{R}$) because it is $O(n^2 \text{poly}(n_{\text{prec}}))$ and n_{prec} is a constant when n is small ($n \lesssim 30$). This is found to be correct and well depicted in Fig. 9.

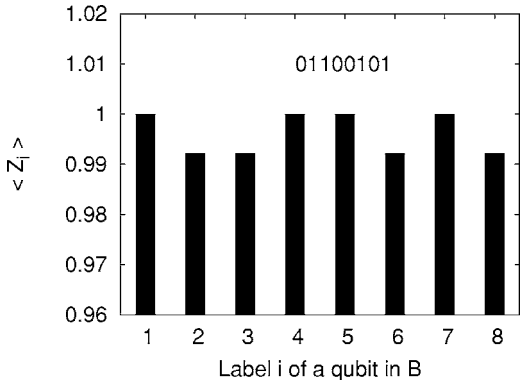

FIG. 10. A simulation result for the eight-bit solution 01100101 in the single-query algorithm. $\langle Z_i \rangle$ for $i=1,\ldots,8$ in the block B are shown.

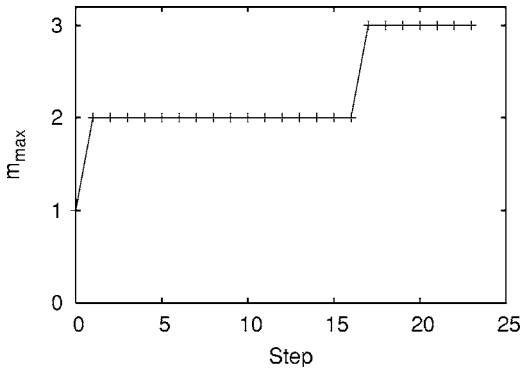

FIG. 11. A simulation result for the eight-bit solution 01100101 in the single-query algorithm. The time evolution of m_{max} in the outside of CCNOT gates is shown against time steps. The steps 0 to 15 are for the simulation of the C^nNOT gate and the rest steps are for the simulation of CSWAP gates.

B. Simulation results for the single-query algorithm

An extension of Brüschweiler's search, the single-query algorithm described in Sec. V A, was also simulated. The value of p in B (introduced in that section) was initially set to 1 in a simulation. First, let us look at the results of a simple example in which we find the single solution 01100101. The polarizations of qubits in B in the output were obtained as illustrated in Fig. 10. The time evolution of m_{max} in the outside of CCNOT gates is shown in Fig. 11. As we have seen in Sec. VI, the value of $m^{\text{ext}}_{\text{max,max}}$ is at most 3 irrespectively of n. This is well depicted. Second, we show some data plotted against n. The value of $m^{\text{int}}_{\text{max,max}}$ is found to be 3 as long as n is in a practical range from simulation results as illustrated in Fig. 12. This is smaller than the calculated upper bound of 12 that we have seen in Sec. VI B and, hence, these are consistent.

For the simulation of the single-query algorithm, we also expect the data points of the consumed time to be on the curve $c_1 \log_2^2 N + c_2 \log_2 N$ (coefficients $c_1, c_2 \in \mathbb{R}$) owing to the decomposition of CSWAP gates as we have seen in Sec. VI B. This is found to be approximately valid as shown in

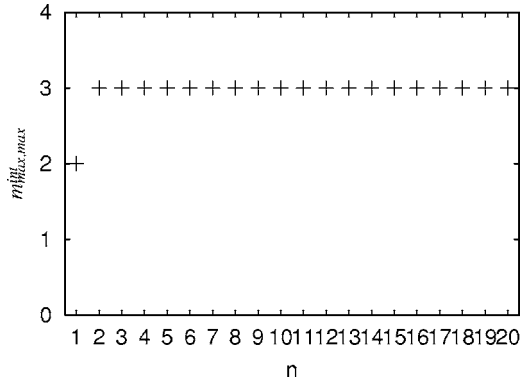

FIG. 12. Plots of the values of $m^{\text{int}}_{\text{max,max}}$ against $n=\log_2 N$. For each value of n, ten trials were performed, each of which involved a randomly chosen solution of an oracle. Thus multiple data points may appear for a single value of n. The single-query algorithm was used.

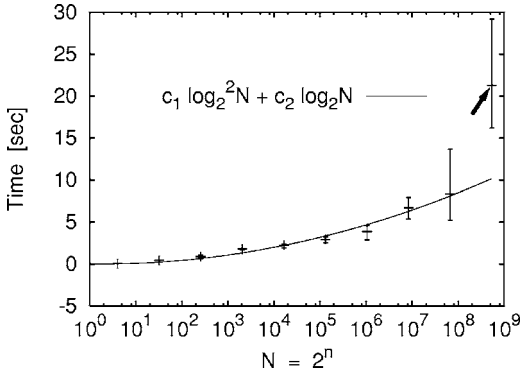

FIG. 13. The plots of the simulation time against the number N of keys in a database. The oracle used here is one with a single solution chosen randomly. The algorithm used here was the single-query algorithm that we have seen in Sec. V A. The solid curve is a least-squares fitting ($c_1=0.0128$ and $c_2=-0.0211$) to the data points (average values) except for the one pointed by the arrow. The error bars are drawn from the shortest time to the longest time in five different trials.

Fig. 13. In the figure, there is an abrupt increase in the simulation time as pointed by the arrow. This is originated from the limitation in the range of floating-point registers to which digits are correctly computed in trigonometric operations for the diagonalization of a density matrix; this is a trouble of a program library irrelevant to computational complexities. Hence, we ignored that data point in fitting.

C. Simulation results for the algorithm to find multiple solutions

Simulations of the algorithm to find all solutions of an oracle function with multiple solutions were also performed. It takes two algorithmic steps to find the value of the number r of solutions as we have seen in Sec. V B. The simulation time shown in this section is a consumed time to find all solutions after the value of r is found.

The running time of simulations is plotted against $N=2^n$ with the values of r fixed, in Fig. 14, and it is plotted against r with the values of n fixed, in Fig. 15. These figures show that an average running time is in accord with the theory in which the running time has been found to be $O(r^5n^2\mathrm{poly}(n_{\mathrm{prec}}))$. Because floating-point basic operations were terminated in particular clock times in the used CPU, the factor of $\mathrm{poly}(n_{\mathrm{prec}})$ is not clear. In contrast, the factors of r^2 and n^2 are well depicted in the figures. These are due to the number of two-qubit quantum gates, $O(r^2n^2)$. Although it is certain that $m^{\mathrm{ext}}_{\mathrm{max,max}} \leqq r+1$ (see the next paragraph), the factor r^3 seems to be hidden. This is because only a few of the values of $m_1,\ldots,m_{n-1}$ reaches the value of $m_{\mathrm{max,max}}$ in a single run of a simulation, and partly because matrix calculations, such as a diagonalization, are implemented in a more effective manner than that for other operations; the number of conditional jump operations for matrix calculations is relatively small in an executable binary program. Because the running time was smaller than expected from the factors, one can state that the fittings to the data points of the average running time is in accord with the theoretical asymptotic upper bound on the running time.

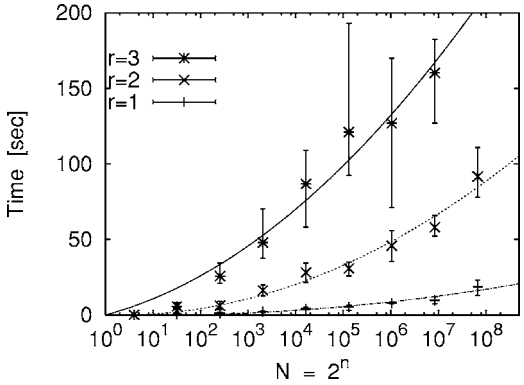

FIG. 14. The time spent in finding all the solutions in each simulation is plotted against the number N of keys in a database. The number r of solutions was fixed to 1, 2, or 3. The curves are least-squares fittings with the function $a\log_2^2 N+b\log_2 N$ by finding coefficients a and b. The error bars are drawn from the shortest time to the longest time in five different trials.

The value of $m^{\mathrm{ext}}_{\mathrm{max,max}}$ was demonstrated to be less than or equal to $r+1$ in simulations, which agrees with the prediction that we have seen in Sec. VI C. For an example, the values of $m^{\mathrm{ext}}_{\mathrm{max,max}}$ in individual simulations are plotted against r particularly when $n=7$ in Fig. 16. This is true for other values of n. In addition, as mentioned in Sec. VI C, the upper bound to the value of $m^{\mathrm{int}}_{\mathrm{max,max}}$ is $4r+4$. This is probably an overestimate according to a simulation result. As shown in Fig. 17, it seems not more than $r+2$ when $n=7$ although this is not certain for large n and/or large r. The achieved data in the figure are still justified in the sense that they are consistent with the bound.

VIII. DISCUSSION

We have obtained simulation results in accord with the theory of the MPS simulation method: the three algorithms—the original Brüschweiler's search algorithm, the single-query algorithm, and the algorithm for an oracle with multiple solutions—were simulated in a polynomial

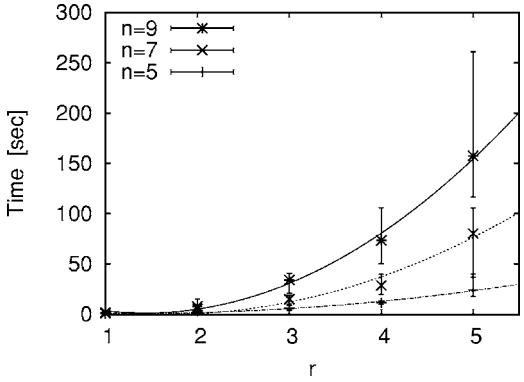

FIG. 15. Plots of the time spent in simulations to find all solutions against the number r of solutions. The key length n was fixed to 5, 7, or 9. The curves are least-squares fittings with the function ar^2+br by finding coefficients a and b. The error bars are drawn from the shortest time to the longest time in five different trials.

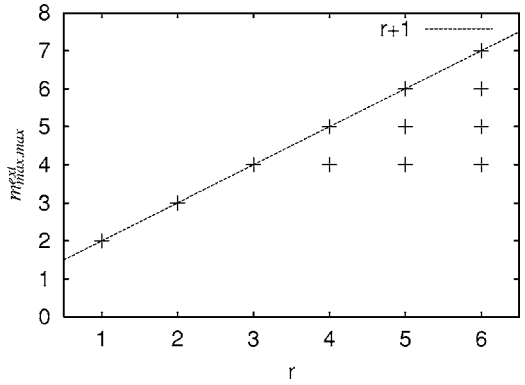

FIG. 16. The values of $m^{\mathrm{ext}}_{\mathrm{max,max}}$ in individual simulations for finding all the solutions are displayed as crosses. r is the number of solutions and data points for each value of r are those taken in ten runs with different sets of solutions randomly chosen.

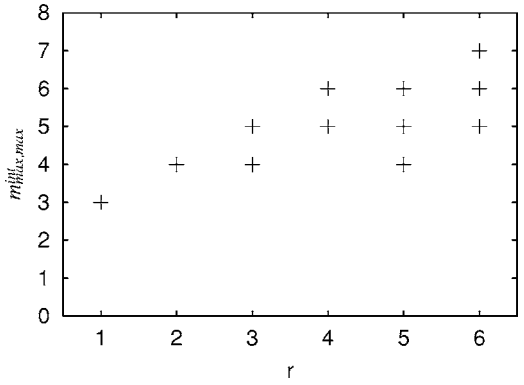

FIG. 17. The values of $m^{\mathrm{int}}_{\mathrm{max,max}}$ in individual simulations for finding all the solutions are displayed as crosses. r is the number of solutions and data points for each value of r are those taken in ten runs with different sets of solutions randomly chosen.

time and a polynomial memory space for some explicit oracle structure. A simulation with a polynomial cost is generally possible as long as a simple oracle that keeps $m^{\mathrm{oracle}}_{\mathrm{max,max}}=\mathrm{poly}(n,r)$ and $n^{\mathrm{oracle}}=\mathrm{poly}(n)$ is used. Moreover, the MPS simulation of Brüschweiler's search does not suffer from the error in a detection as long as the precision 2^{-n} is kept, unlike the fact that an experiment of it suffers from an exponential increase in the number of accumulations to truncate noise out. This is a typical example that an emulation of some machine is faster and more accurate for a particular problem. There is no advantage for an NMR computer for those algorithms when simple oracles are involved.

Nevertheless, if it were a true unsorted database that we use together with those algorithms, the fast simulations might be regarded as the proof for the proposition: "An unsorted oracle-based search is a **P** problem under the condition that a database answers a query by using an MPS simulation of an oracle quantum circuit for which the database chooses a simple circuit construction such that $m^{\mathrm{oracle}}_{\mathrm{max,max}}=\mathrm{poly}(n,r)$ and $n^{\mathrm{oracle}}=\mathrm{poly}(n)$." This is incorrect because of the abuse of the word "unsorted" as we will see below.

What is the meaning of "unsorted database"? An unsorted database is a one-dimensional array in which $O(N)=O(2^n)$ records are stored, to which a table consisting of record keys is attached; there is no arrangement of the ordering in record keys in the table that is open to the public. An oracle attached to the database can check whether a key is a marked one (the mark is not written in the table), but it does not have another functionality. In contrast, a vector in a Liouville space in the form of $|v)=\Sigma_k c_k|k,k)$ should be considered a key table for a database with a special arrangement of key positions in a natural sense. Indeed, it is a custom in quantum computation theories to regard it as a key table of an unsorted database, but we believe that it is nothing but a gadget used for an exaggeration of the power of a quantum computer and that of a bulk-ensemble computer. Although one can dare to state that there is no order among keys $|k,k)$, all the keys are placed in the head of a table simultaneously when a superposition is utilized; one can send an oracle the whole key table in a single query. This is in contrast to a classical query in which a single key placed in some location of a table is sent. In this context, an MPS simulation to solve the search problem does not involve a classical query because it uses a matrix product state in which all the keys may be placed; this is sent in a single query. The difference in these three types of a query is illustrated in Fig. 18. It is apparent that the original meaning of unsorted is abused in quantum computation theories. Taking it into account, we will consider the speed-up effect in simulations that looks as if we had a sorted database or a database with a structure.

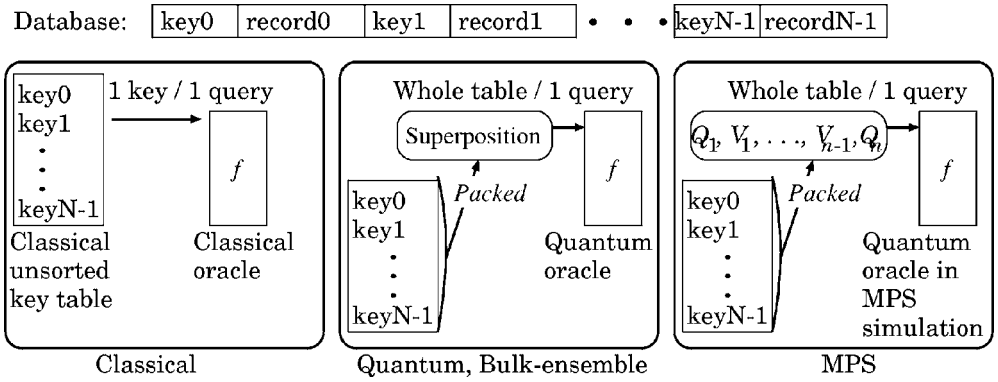

FIG. 18. A comparison of unsorted database searches: classical one, one in a quantum or bulk-ensemble model, one in a MPS simulation. $N=2^n$ is a number of keys.

Let us consider the speed-up effect of a conventional "sorted database." Sorting is to order data-record keys so that it is convenient to find a key that satisfies some condition. In an oracle-based search, sorting is to have an oracle function that can answer whether a key is equal to, smaller than, or larger than the nearest one of solutions as we have seen in (b′) of Sec. V B. This enables a single query to eliminate approximately a half of candidate keys, and thus an exponential speed-up is possible. The speed-up achieved by sorting, which is an exponential speed-up, had been understood as the factorization of a search process according to Patel [23]. Nevertheless, his definition is not common; a factorization of an oracle is relevant to a "structured database" as we employ the description (c′) of Sec. V B.

If we can use a factorized oracle $f(\mathbf{x})=\Sigma_{k=1}^{r}\Pi_{i=1}^{n}f_i^k(x_i)$ where $\mathbf{x}=x_1,\ldots,x_n$ is a string of letters x_i and f_i^k are separately usable functions, a search with the oracle is regarded as a search by using a speed-up owing to structuring. A factorized oracle is a sum of products of functions $f_1^k,\ldots,f_n^k$, each of which maps $\{0,1\}$ to $\{0,1\}$ and is used to find each bit of the kth solution. In this case, the number of queries is $O(nr)=O(r\log_2 N)$ to find all solutions. Although we do not have a factorized oracle in the problem of searching in a so-called unsorted database, searches that we have seen in the descriptions of Brüschweiler's algorithm and its extensions are equivalent to what Patel called searches using a factorized oracle. Thus it has been shown that a simple oracle whose structure is feasible to keep a simulation cost polynomial (this structure may be hidden to a user), especially one implemented as a set of r C^nNOT gates and some NOT gates, can be factorized in a polynomial time with an extended Brüschweiler's algorithm on a conventional computer with an MPS simulation as well as on a bulk-ensemble computer. This is, of course, due to the fact that an input query for an oracle function is a superposition (or its MPS decomposition) of all record keys.

Usually, it is unfair to compare a search in a raw key table and a search in a compressed key table. It is, however, fair to use a compressed key table when one has to compare a classical search with a quantum search or a bulk-ensemble parallel search because the use of the powerful machines, quantum computers or bulk-ensemble parallel computers, packs $O(2^\tau)$ strings in a τ-bit length sequence of sites in principle. By using the MPS method, we can use the similar effect: $O(2^\tau)$ strings are packed in the tensors $Q_1,V_1,\ldots,V_{\tau-1},Q_\tau$. In addition, there is another point to suggest the fairness of this comparison: A state space of n qubits is spanned by 2^n basis vectors, and the space of the states $\Sigma_{i_1,\ldots,i_n}(\Sigma_{v_1,\ldots,v_{n-1}}Q_1,V_1,\ldots,V_{n-1},Q_n)^2|i_1,\ldots,i_n]$ is also spanned by 2^n basis vectors $|0_1\cdots0_n],\ldots,|1_1\cdots1_n]$. The MPS simulation of Brüschweiler's search scans in the space with the same dimension as a real n-qubit state space.

A mechanism has been revealed; there is a context behind the problem of database searching using quantum machines or bulk-ensemble machines: one can use a superposition for an input and unitary transformations for an oracle function. The search algorithms for quantum computers and those for NMR parallel computers search in a sort of parallel key tables, namely superpositions of vectors. When it is allowed to use precise ensemble measurements, a superposition may be regarded as a key table for a highly structured database if the internal cost of an oracle grows polynomially in n and r because the factorization of an oracle function can be done in a polynomial time. Of course a search in such a database is fast, in terms of query complexity, also on conventional computers if it is allowed to use a compressed key table produced by the MPS decomposition, and classical simulations are actually practical owing to the lack of noise. It is easy to let n_{prec} scale linearly in n to perform an error-free simulation.

It has been found that the problems of database searching for the extensions and the original of Brüschweiler's algorithm can be solved with a poly(n,r) cost on conventional computers as long as $m_{\mathrm{max,max}}^{\mathrm{oracle}}=\mathrm{poly}(n,r)$ and $n^{\mathrm{oracle}}=\mathrm{poly}(n)$. Nevertheless, it is still a controversial issue if the problem for Grover's search is equivalent to those problems. If an ensemble measurement is not allowed to use, it is the optimal search; if one does not mention the style of measurements, the problem is occasionally efficiently solved in a classical manner. It is obvious that the difficulty of the problem is dependent on the context.

In addition, it is curious to find a practical application of those simulations, particularly in *NP* problems. A quantum oracle may be constructed [5] to tell whether an input number is a factor of some number to solve the problem of prime factorization. SAT problems are also of public interest (see, e.g., Ref. [24] for recent developments), which may be solved by a database search using oracles. Although it was shown by Viamontes *et al.* [12] that Grover's search is often inferior to classical specialized search algorithms for SAT problems with respect to query complexity, Brüschweiler's search is different. An extension of Brüschweiler's search is terminated in $O(nr)$ query complexity, which is exponentially smaller than that of Grover's search. It was shown by Hogg [25] that a SAT problem under a certain strong constraint can be solved in a linear time on a classical computer and in a single step on a quantum computer in the limit of large n. It is usual that constraints and conditions for an oracle structure affect computational complexities very much. Thus a motivation will hopefully be raised to find some condition for the construction of an oracle under which $m_{\mathrm{max,max}}^{\mathrm{oracle}}=\mathrm{poly}(n,r)$ and $n^{\mathrm{oracle}}=\mathrm{poly}(n)$ is satisfied in an MPS simulation.

IX. CONCLUSION

Two extensions of the Brüschweiler's database search have been shown: a single-query algorithm and an algorithm for an oracle with multiple solutions; all of r solutions can be found in $O(nr)$ queries, where n is the key length asymptotically proportional to the number of qubits. By using the Vidal's MPS decomposition method, numerical simulations of these algorithms as well as that of the original Brüschweiler's algorithm have been demonstrated. The theory of the simulation has shown that those simulations finish in a polynomial time within a polynomial memory space with respect to n and r as long as the internal circuit of an oracle does not prevent us from truncating many eigenvalues out and poly(n) qubits are used in the oracle quantum circuit [see

Eqs. (70) and (71)]. The floating-point register length to simulate a precise ensemble measurement scales linearly in n. These facts were consistent with the results from simulations. Because of those fast classical simulations, one should be sceptical about the power of quantum computation andthat of bulk-ensemble computation for the problem of database searching of an unsorted database. Our discussion has led to the suggestion that what they called an unsorted database should be regarded as a highly structured database when a precise ensemble average measurement or its simulation is available and the internal cost of an oracle or that of a simulated oracle grows polynomially in n and r.

ACKNOWLEDGMENTS

The authors are thankful to Robabeh Rahimi for helpful comments and discussions. They would also like to thank Masanao Ozawa for thoughtful comments and discussions. This work is supported by the CREST grant from the Japan Science and Technology Agency.

[1] D. Deutsch, Proc. R. Soc. London, Ser. A **425**, 73 (1989).
[2] P. W. Shor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE Press, Los Alamitos, CA, 1994), pp. 124–134; SIAM J. Comput. **26**, 1484 (1997).
[3] L. K. Grover, *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation* (ACM Press, New York, 1996), pp. 212–219; Phys. Rev. Lett. **79**, 325 (1997).
[4] C. Zalka, Phys. Rev. A **60**, 2746 (1999).
[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
[6] D. E. Knuth, *Sorting and Searching*, Art of Computer Programming Vol. 3 (Addison-Wesley, Reading, MA, 1973).
[7] G. F. Viamontes, I. L. Markov, and J. P. Hayes, Quantum Inf. Comput. **2**, 347 (2003).
[8] A. Kawaguchi, K. Shimizu, Y. Tokura, and N. Imoto, e-print quant-ph/0411205.
[9] S. R. White, Phys. Rev. B **48**, 10345 (1993).
[10] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).
[11] G. Brassard, P. Høyer, and A. Tapp, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP)*, Aalborg, Denmark, 1998, Lecture Notes in Computer Science, edited by K. G. Larsen, S. Skyum, and G. Winskel (Springer-Verlag, New York, 1998), Vol. 1443, pp. 820–831, e-print quant-ph/9805082.
[12] G. F. Viamontes, I. L. Markov, and J. P. Hayes, Comput. Sci. Eng. **7**(3), 62 (2005).
[13] R. Brüschweiler, Phys. Rev. Lett. **85**, 4815 (2000).
[14] D. G. Cory, A. F. Fahmy, and T. F. Havel, Proc. Natl. Acad. Sci. U.S.A. **94**, 1634 (1997).
[15] I. L. Chuang, L. M. K. Vandersypen, X. Zhou, D. W. Leung, and S. Lloyd, Nature (London) **393**, 143 (1998).
[16] S. Okubo, T. Nishino, K. Ota, and N. Kunihiro, IPSJ Journal **46**, 1416 (2005) (in Japanese).
[17] M. Zwolak and G. Vidal, Phys. Rev. Lett. **93**, 207205 (2004).
[18] L. Xiao and G. L. Long, Phys. Rev. A **66**, 052320 (2002).
[19] V. E. Tarasov, J. Phys. A **35**, 5207 (2002); see also the conventional usage of a Liouville space in the theory of NMR in A. Abragam and M. Goldman, *Nuclear Magnetism. Order and Disorder* (Clarendon Press, Oxford, 1982), Chap. 1B.
[20] E. Knill, I. Chuang, and R. Laflamme, Phys. Rev. A **57**, 3348 (1998).
[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press, Cambridge, 1992).
[22] A. Kawaguchi, K. Shimizu, Y. Tokura, and N. Imoto, in *The Extended Abstracts of the Conference QIT2004*, Kyoto, Japan, 2004 (in Japanese), pp. 1–6 (unpublished).
[23] A. Patel, Phys. Rev. A **64**, 034303 (2001).
[24] L. Accardi and M. Ohya, e-print quant-ph/0401110.
[25] T. Hogg, Phys. Rev. Lett. **80**, 2473 (1998).
[26] This is easy when the state is $(a|0\rangle+b|1\rangle)^{\otimes n}$. In general, there is nothing but a given Hamiltonian; it is often required that the initial state should be determined numerically by DMRG or some method.
[27] See EPAPS Document No. E-PLRAAN-73-102606 for the MPS simulation of a projective measurement on a single qubit and a simple example of MPS simulation processes. For more information on EPAPS, see http://www.aip.org/pubservs/epaps.html.