Image Computation for Quantum Transition Systems

Xin Hong[1], Dingchao Gao[1], Sanjiang Li[2], Shenggang Ying[1] and Mingsheng Ying[2]

[1]*Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science Institute of Software, Chinese Academy of Sciences, Beijing, China*

[2]*Centre for Quantum Software and Information, University of Technology Sydney, Sydney, Australia*

Emails: hongxin@ios.ac.cn, gaodc@ios.ac.cn, sanjiang.li@uts.edu.au, yingsg@ios.ac.cn, mingsheng.ying@uts.edu.au

***Abstract*—With the rapid progress in quantum hardware and software, the need for verification of quantum systems becomes increasingly crucial. While model checking is a dominant and very successful technique for verifying classical systems, its application to quantum systems is still an underdeveloped research area. This paper advances the development of model checking quantum systems by providing efficient image computation algorithms for quantum transition systems, which play a fundamental role in model checking. In our approach, we represent quantum circuits as tensor networks and design algorithms by leveraging the properties of tensor networks and tensor decision diagrams. Our experiments demonstrate that our contraction partition-based algorithm can greatly improve the efficiency of image computation for quantum transition systems.**

***Index Terms*—quantum transition systems, image computation, model checking**

I. Introduction

The past several years have seen rapid progress in quantum hardware and software, which makes the need for verification of quantum systems increasingly pressing. Indeed, a series of testing and verification techniques have been developed for quantum circuits in the past 15 years. Several previous works have studied the equivalence checking [1]–[4] and testing [5] of quantum circuits.

Model checking is an important formal method for verifying finite-state systems. It models the system under examination as a transition system and specifies (safety or liveness) properties in a temporal logic. Given a model M and a property φ, it then checks if M satisfies φ. This is usually done by computing all its reachable states through the repeated use of an image computation procedure.

Image computation calculates the output (image) of a set of states under a transition system. Through the repeated calling of this procedure, we can obtain all the reachable states of this system and check the corresponding properties. Many efficient image computation algorithms have been developed for classical model checking. In particular, image computation can be significantly sped up by taking advantage of the symbolic representation of initial states and transition relations as BDDs [6] and by exploiting state space partition [7] and circuit partition [8]. Image computation has also been implemented in various classical verification tools (e.g., VIS [9]).

Work partially supported by the National Science Foundation of China (12471437), Beijing Nova Program 20220484128, and the Australian Research Council (DP220102059).

Model checking techniques have also been extended to quantum systems. Some earlier works focus on verification of quantum communication protocols [10], and some others were motivated by termination analysis of quantum programs [11]. We refer the reader to the recent book [12] for a systematic introduction to the principle and algorithms for model checking quantum systems. Notwithstanding these achievements, applications of model checking to quantum hardware verification are still an underdeveloped research area. Recently, a temporal extension of the Birkhoff-von Neumann quantum logic [13] was proposed in [14], in which atomic propositions are represented as (closed) subspaces of the quantum state space. While very simple, some important temporal properties of quantum circuits can be specified in this logic and checked by simulation on classical computers [14]. However, one challenge is that we are still lacking efficient implementations of the algorithms for checking quantum circuits.

This paper aims to advance the development of model checking quantum systems by providing efficient image computation algorithms for quantum systems. Similar to classical model checking, we represent each quantum system as a quantum transition system, where sets of states are replaced by subspaces and transition relations are replaced by quantum operations. In our approach, we regard quantum circuits as tensor networks and design algorithms by leveraging the properties of tensor networks and tensor decision diagrams (TDDs) [15], which are data structures that, similar to BDDs, represent tensors in a canonical and compact way. More precisely, we first introduce a basic algorithm for conducting image computation for quantum systems and then provide several partition-based optimisation schemes, which play similar roles as the disjunctive and conjunctive partitions for classical image computation [8]. Our experiments demonstrate that our tensor network partition-based algorithms can significantly improve the efficiency of image computation for quantum transition systems when compared with the basic algorithm.

The remainder of this paper proceeds as follows. In section II we recall relevant background such as quantum states, subspaces, quantum operations, and tensor decision diagrams. The notion of quantum transition systems is then introduced in section III. After describing the basic procedure for conducting the image computation in section IV, we introduce two tensor network partition-based schemes in section V. Empirical evaluations of these schemes are presented in VI. The last section concludes the paper.

II. Background

A. Quantum State Space and Quantum Operations

Quantum states are unit vectors in a Hilbert space $\mathcal{H}$. For a single-qubit system, we write $\mathcal{H}_2$ for its (2-dim) state space. A state in $\mathcal{H}_2$ has form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. We write $|\pm\rangle$ for the superposition states $\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. The Hilbert space $\mathcal{H}$ of an n-qubit system is the tensor product of n copies of $\mathcal{H}_2$, i.e., $\mathcal{H} = \mathcal{H}_2^{\otimes n}$. A *mixed state* in $\mathcal{H}$ is a density operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ on $\mathcal{H}$, which is a positive semi-definite operator on $\mathcal{H}$ with trace 1. The *support* of a density operator ρ, denoted as $supp(\rho)$, is the subspace of $\mathcal{H}$ spanned by the eigenvectors corresponding to non-zero eigenvalues of ρ.

Pure quantum states can be transformed by unitary matrices, also called quantum gates, and mixed states are transformed by super-operators. A *super-operator* on $\mathcal{H}$ is a linear operator $\mathcal{E}$ and a *quantum operation* on $\mathcal{H}$ is a completely positive super-operator on $\mathcal{H}$ such that $tr(\mathcal{E}(\rho)) \leq tr(\rho)$ for any density operator ρ in $\mathcal{H}$. Using the Kraus operator-sum representation, a super-operator can be represented as a set of operators $\mathcal{E} = \{E_i\}$ such that $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$.

Definition 1. Given a subspace S of $\mathcal{H}$. Its *image* under a set of quantum operations $\mathcal{T} = (\mathcal{T}_\sigma)_{\sigma\in\Sigma}$ is defined as $\mathcal{T}(S) = \bigvee_{\sigma\in\Sigma} \mathcal{T}_\sigma(S)$, where for each $\sigma \in \Sigma$:

$$\mathcal{T}_\sigma(S) = \bigvee_{|\psi\rangle\in S} supp(\mathcal{T}_\sigma(|\psi\rangle\langle\psi|)).$$

Recall that the *join* of a family of subspaces $\{X_i\}$ of $\mathcal{H}$ is defined as $\bigvee_i X_i = span(\bigcup_i X_i)$. Thus, $\mathcal{T}(S)$ is still a subspace.

The following results are useful in our work.

Proposition 1 ([12]). *Let $\mathcal{T}$ be a quantum operation. Then*

1) $\mathcal{T}(\bigvee_i S_i) = \bigvee_i \mathcal{T}(S_i)$.
2) *If $\mathcal{T} = (\mathcal{T}_\sigma)_{\sigma\in\Sigma}$ and each $\mathcal{T}_\sigma$ has the Kraus operator-sum representation $\mathcal{T}_\sigma = \{E_{\sigma j_\sigma}\}$, then*

$$\mathcal{T}(S) = span\Big(\bigcup_{\sigma, j_\sigma} \{E_{\sigma j_\sigma}|\psi\rangle : |\psi\rangle \in S\}\Big).$$

Combining 1) and 2), $\mathcal{T}(S)$ can also be written as:

$$\mathcal{T}(S) = span\Big(\bigcup_{\sigma, j_\sigma} \{E_{\sigma j_\sigma}|\psi\rangle : |\psi\rangle \in B\}\Big),$$

where $B = \{|\psi_1\rangle, \cdots, |\psi_k\rangle\}$ is a basis of the subspace S.

B. Tensor Decision Diagram

A tensor $\phi_{x_1,\cdots,x_n} : \{0,1\}^I \to \mathbb{C}$ is a multi-dimensional map, where $I = \{x_1, \cdots, x_n\}$ is the set of indices and n is called its rank. Matrices and, thus, quantum operations are examples of tensors. In this work, we assume that every index can take two different values 0 and 1 and will only consider $2^n \times 2^n$ matrices, which are regarded as rank $2n$ tensors.

$$P = \frac{1}{6}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

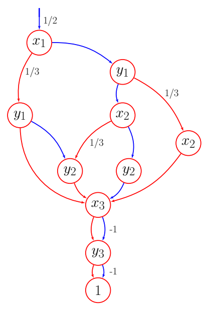

Fig. 1. A matrix P and its TDD representation. Note that edges with weight 0 are omitted in this diagram.

Take the matrix P in the left of Fig. 1 as an example. P can be seen as a rank 6 tensor $\phi_{x_1x_2x_3y_1y_2y_3}$, where the indices x_i and y_j encode the column and row numbers, respectively. For examples, the 7-th column is specified as $(x_1, x_2, x_3) = (1, 1, 0)$ and the 8-th row is specified as $(y_1, y_2, y_3) = (1, 1, 1)$. As for the entries, we have $\phi_{x_1x_2x_3y_1y_2y_3}(110111) = -3/6$.

Tensor decision diagrams (TDDs) [15] are designed to represent tensors compactly and canonically. For a fixed index order, each tensor $\phi_{x_1,\cdots,x_n}$ has a unique TDD representation. In a TDD representation, each node (except the terminal node) represents an index, and we use blue and red lines to indicate two different values of the index. Furthermore, each path starting from the root node to the terminal node corresponds to an assignment $a_1, \ldots, a_n$ of the indices $x_1, \ldots, x_n$, and the multiplication of weights along the path corresponds to the tensor value when $x_1 = a_1, \ldots, x_n = a_n$. For example, in the TDD shown in Fig. 1 (right), the value $\phi_{x_1x_2x_3y_1y_2y_3}(110111) = -1/2 = 1/2 \times 1 \times 1 \times 1 \times 1 \times 1 \times (-1)$ is obtained by multiplying the weight on the incoming edge of the root node and the weights on the path whose edges are coloured blue, blue, blue, blue, red, blue.

Quantum circuits can be naturally regarded as tensor networks, i.e., sets of tensors connected by shared indices. The most basic operation of a tensor network is the contraction, which sums up two connected tensors over shared indices. Take two tensors ϕ_{xy} and ψ_{yz} as an example; their contraction is the rank 2 tensor $\xi_{xz} = \sum_{y=0}^{1} \phi_{xy} \cdot \psi_{yz}$. The contraction as well as the addition of two tensors can be calculated using TDDs directly. Another useful tensor operation is *slicing*: The slicing of a tensor $\phi_{x,y_1,\ldots,y_k}$ with respect to $x = c$ with $c \in \{0, 1\}$ is a tensor with index set $\{y_1, \ldots, y_k\}$ such that $\phi|_{x=c}(a_1, \ldots, a_k) = \phi(c, a_1, \ldots, a_k)$.

This paper will use TDD to represent quantum states, quantum operations as well as subspaces. This makes it convenient for leveraging the advantages of both decision diagrams and tensor networks.

III. Quantum Transition Systems

A transition system M is a 4-tuple (S, S_0, Σ, R), where S is a set of states, $S_0 \subseteq S$ is the set of initial states, $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is a set of symbols, and $R \subseteq S \times \Sigma \times S$ is a transition relation. Given a set of states $S' \subseteq S$, its *image* under the transition relation R is defined as

$$R(S') := \{t \in S \mid (s, \sigma, t) \in R \text{ and } s \in S' \text{ and } \sigma \in \Sigma\}. \quad (1)$$

Quantum transition systems are the quantum generalisation of transition systems and have been introduced in the quantum computing and communication literature with several different names (e.g., quantum automata and (discrete-time) quantum Markov systems). It can be conveniently used to model quantum communication channels and quantum cryptographic protocols [16], [17], quantum circuits [14] and semantics of quantum programs [12].

Definition 2. Let $\mathcal{H}$ be a Hilbert space. A quantum transition system M on $\mathcal{H}$ is a 4-tuple $(\mathcal{H}, S_0, \Sigma, \mathcal{T})$, where S_0 is a (closed) subspace of $\mathcal{H}$, called the initial space, $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is a set of (classical) symbols, and $\mathcal{T} = (\mathcal{T}_\sigma)_{\sigma\in\Sigma}$ is a family of quantum operations on $\mathcal{H}$.

For each symbol $\sigma \in \Sigma$ a quantum operation $\mathcal{T}_\sigma$ is enabled, which maps a mixed state ρ to another mixed state $\mathcal{T}_\sigma(\rho)$. If the system is closed (i.e., without interaction with its environment), then each $\mathcal{T}_\sigma$ is a unitary transformation U_σ, which maps a state $|\psi\rangle \in \mathcal{H}$ to another state $U_\sigma |\psi\rangle \in \mathcal{H}$.

Quantum circuits are now the standard model for quantum computing. We now show how the behaviour of quantum circuits can be modelled as quantum transition systems and how the functionality can be checked by calculating their image. Our modelling covers combinational, dynamic, and noisy quantum circuits.

A. Modelling Quantum Circuits

1) Combinational Quantum Circuits: In a combinational quantum circuit, each gate represents a unitary operation.

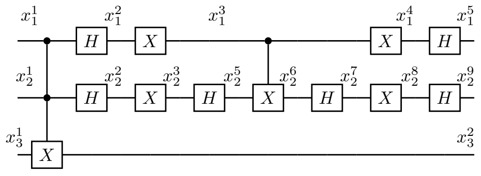

Fig. 2. The circuit for Grover iteration. It is also a tensor network with indices x_i^j, which denotes the j-th index on qubit i.

For example, Fig. 2 gives the circuit for implementing two-qubit Grover iteration [18], a basic procedure of Grover's algorithm for finding a solution of a Boolean function $f(x) = 1$. For this circuit, the first CCX gate represents an oracle $O|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle$ and the other gates represent a reflection $2|\psi\rangle\langle\psi| - I$, where $f(x) = x_1 \wedge x_2$ and $|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle$. Given an input state $|{+}{+}{-}\rangle = \frac{1}{2}\sum_{i=0}^{3}|i\rangle|-\rangle$, the circuit first changes the state to $\frac{1}{2}\sum_{i=0}^{2}|i\rangle|-\rangle - \frac{1}{2}|11\rangle|-\rangle$, and then to $|11\rangle|-\rangle$. In general, let $S = span\{|{+}{+}{-}\rangle, |11-\rangle\}$. Then, for any input state $|\varphi\rangle \in S$, the output state will always be in S.

Then, the system can be modelled by a quantum transition system $(\mathcal{H}_8, S, \{1\}, \mathcal{T})$, where $\mathcal{T}_1 = (2|\psi\rangle\langle\psi| - I)O$, and the property can be checked by calculating $\mathcal{T}_1(S) = S$.

2) Dynamic Quantum Circuits: A dynamic quantum circuit is a quantum circuit in which measurements can happen in the middle and the subsequent circuit can depend on the measurement outcomes [19].

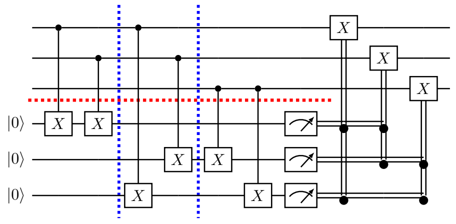

Fig. 3. The circuit for correcting one qubit bit-flip error.

Fig. 3 gives an example of a dynamic quantum circuit which can be used to correct the input quantum state if at most one bit-flip error may occur. For this circuit, the input state will first experience a period of syndrome detection (the six CX gates), denoted by U, then a set of measurements will be conducted to identify the error, and lastly, a set of operations will be applied according to the measurement results to correct the error. There are four different measurement results that can happen for this circuit, viz. 000, 101, 110, and 011, corresponding to the cases of no error or a bit-flip error on the first, second, and third qubits, respectively.

Thus, the system can be modelled by a transition system with four quantum operations $\mathcal{T}_{000} = \{(I_1 \otimes I_2 \otimes I_3 \otimes |000\rangle\langle 000|)U\}$, $\mathcal{T}_{101} = \{(X_1 \otimes I_2 \otimes I_3 \otimes |101\rangle\langle 101|)U\}$, $\mathcal{T}_{110} = \{(I_1 \otimes X_2 \otimes I_3 \otimes |110\rangle\langle 110|)U\}$, $\mathcal{T}_{011} = \{(I_1 \otimes I_2 \otimes X_3 \otimes |011\rangle\langle 011|)U\}$, where I_i and X_i are, respectively, the identity operation and X_i on the i-th qubit. The correctness of this circuit can be partly checked by $\mathcal{T}(span\{|100\rangle, |010\rangle, |001\rangle\}) = span\{|000\rangle\}$ which means that the bit-flip error has been corrected.

3) Noisy quantum circuits: Noises may happen during the execution of quantum circuits. In quantum computing, noises are often represented as super-operators in the Kraus operator sum form.

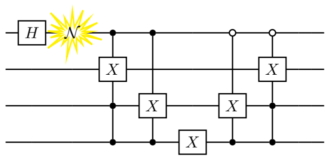

Fig. 4. A noisy version of a quantum walk along a 8-length cycle.

Take a quantum walk along a 8-length cycle [20], [21] as an example. Here, the coin operation is implemented by a Hadamard gate $H = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$. A state will be shifted to an adjacent state and, depending on the value of the coin, it could be either higher or lower. The shift operations can be described by two operators $S_0 = \sum_{i=0}^{7}|(i-1) \bmod 8\rangle\langle i|$ and $S_1 = \sum_{i=0}^{7}|(i+1) \bmod 8\rangle\langle i|$ and implemented using a series of $C^n(X)$ gates, see Fig. 4.

To simplify the discussion, we only consider the case where a noise occurs once here. Suppose a bit-flip noise may happen after the coin operator. Then the transition system can be represented as $(\mathcal{H}_2^{\otimes 4}, S, \{1, 2\}, \mathcal{T})$ where $\mathcal{T}_1 = \mathcal{S} \circ (\mathcal{E}_c \otimes \mathcal{I})$ and $\mathcal{T}_2 = \mathcal{S} \circ (\mathcal{E}_b \otimes \mathcal{I}) \circ (\mathcal{E}_c \otimes \mathcal{I})$. Here, we use $\mathcal{E}_c$ and $\mathcal{I}$ to represent the super-operator with only one Kraus operator H and I, $\mathcal{E}_b = \{\sqrt{p}I, \sqrt{1-p}X\}$ to represent the bit-flip error and

$\mathcal{S}$ to represent the super-operator with only one Kraus operator $S_0 \oplus S_1$. Both $\mathcal{E}_c$, $\mathcal{E}_b$ are applied on the first qubit; $\mathcal{I}$ is applied on the last three qubits, and $\mathcal{S}$ is applied on all four qubits. In other words, $\mathcal{S}$ applies the operator S_t to the last three-qubit if the state of the control qubit is $|t\rangle$ ($t = 0, 1$). The properties of this system can be checked by calculating $\mathcal{T}(span\{|0\rangle |i\rangle\}) = span\{|0\rangle |(i-1) \bmod 8\rangle, |1\rangle |(i+1) \bmod 8\rangle\}$, which means that sometimes a bit-flip error will not influence the reachable subspace significantly.

IV. Basic Algorithms

For classical systems, image computation is normally conducted by representing both the initial set of states and the transition relation as BDDs and then calculating their conjunction and eliminating the existential quantifier [6], [22].

In this section, we describe a basic approach for image computation for quantum systems. We represent Kraus operators E_i TDDs and give algorithms for computing the join of subspaces.

A. Representation of Subspaces

A nonempty Hilbert subspace contains infinitely many possible states. The most efficient method for representing a subspace is to represent it as a set of basis states. Given the projector P of a subspace S, we now show how to find a basis for S. Let $P = [|u_1\rangle, \cdots, |u_{2^n}\rangle]$ be the column representation of the projector. Then $P|u_i\rangle = |u_i\rangle$ for all $i \in \{1, \cdots 2^n\}$. Thus, if $|u_i\rangle \neq 0$, then $\frac{|u_i\rangle}{\||u_i\rangle\|}$ can be served as an element in the basis of the subspace, we denote it as $|v_i\rangle$ and update $P = P - |v_i\rangle\langle v_i|$. Then, we can recursively find a set of orthogonal basis vectors for the original subspace. Although this process may experience a high complexity when traversing all columns using the matrix representation, it can be achieved easily using the TDD representation since the first non-zero column can be found by locating the leftmost non-zero path of the TDD representation of P.

Example 1. *Take the subspace $S = span\{|++-\rangle, |11-\rangle\}$ used in Grover iteration (cf. Sec. III-A) as an example. The matrix and TDD representations of the corresponding projector P are shown in Fig. 1. The leftmost path corresponds to $(x_1, x_2, x_3, y_1, y_2, y_3) = (0, 0, 0, 0, 0, 0)$ while means that the first column of the projector is non-zero. Traverse all paths with $(x_0, x_1, x_2) = (0, 0, 0)$, we can obtain the vector $|v_1\rangle = \frac{1}{6}[1, -1, 1, -1, 1, -1, 0, 0]$, which can be normalised as $|v_1\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)|-\rangle$. Let $P' = P - |v_1\rangle\langle v_1|$. Then P' equals to $|11-\rangle\langle 11-|$. Repeating the process for P', we obtain $|v_2\rangle = |11\rangle|-\rangle$. Then $\{|v_1\rangle, |v_2\rangle\}$ is a basis of the subspace S.*

Note that we do not need to represent the corresponding vectors in the explicit form during the calculation process but to represent them as TDDs, thus reducing the complexity.

B. Computing the Join of Two Subspaces

Let $S = S_1 \vee S_2$ be the join of two subspaces $S_, S_2$. Suppose $B_1 = \{|\psi_{11}\rangle, \cdots, |\psi_{1k}\rangle\}$ and $B_2 = \{|\psi_{21}\rangle, \cdots, |\psi_{2l}\rangle\}$ are orthonormal bases for S_1 and S_2. We show how to compute a basis B for S.

Set $B = B_1$ and let $P = \sum_{j=1}^{k} |\psi_{1j}\rangle\langle\psi_{1j}|$. We complete B into a basis of S step by step by following the Gram-Schmidt procedure. That is, we consider basis vectors in B_2 one by one. Suppose the current vector is $|\psi_{2j}\rangle$. We calculate $|u_j\rangle = |\psi_{2j}\rangle - P|\psi_{2j}\rangle$ and normalise it as $|v_j\rangle = \frac{|u_j\rangle}{\||u_j\rangle\|}$. If $|v_j\rangle$ is 0, then we consider the next vector; otherwise, it is orthogonal to P and we add it to B. Meanwhile, we also update P as $P + |v_j\rangle\langle v_j|$. Repeat this process until all elements in B_2 have been considered. Then, B will be a basis of S and P will be the projector to S.

Example 2. *Consider Grover iteration again. Let P_1, P_2 be the projectors of two one-dimensional subspaces S_1, S_2, which are generated by $B_1 = \{|++-\rangle\}$ and $B_2 = \{|11-\rangle\}$ respectively. Clearly, $P_1 = |++-\rangle\langle ++-|$. We complete B_1 into a basis of $S = S_1 \vee S_2$. A simple calculation shows that $|u\rangle = |11-\rangle - P_1|11-\rangle = |11-\rangle - \frac{1}{4}|++-\rangle = [-\frac{1}{4}, \frac{1}{4}, -\frac{1}{4}, \frac{1}{4}, -\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, -\frac{3}{4}]^T$, which can be normalised as $|v\rangle = -\frac{1}{2\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle - 3|11\rangle)|-\rangle$. Then $B = \{|++-\rangle, |v\rangle\}$ is an orthonormal basis, and $P = P_1 + |v\rangle\langle v|$ is the corresponding projector, which is exactly the one shown in Example 1.*

C. A Basic Image Computation Algorithm

Based on the above algorithms, we now give the basic image computation algorithm for quantum transition systems.

Algorithm 1 *Basic_Image*$(\mathcal{H}, S, \Sigma, \mathcal{T})$

```
Input: A quantum transition system $(\mathcal{H}, S, \Sigma, \mathcal{T})$, where $\mathcal{T} = \{\mathcal{T}_\sigma \mid \sigma \in \Sigma\}$ and $\mathcal{T}_\sigma = \{E_{\sigma,j_\sigma}\}$
Output: the projector $P$ of $\mathcal{T}(S)$
1: $P \leftarrow 0$
2: $B \leftarrow Basis\_Decompose(S)$
3: $K \leftarrow \cup_{\sigma,j_\sigma}\{E_{\sigma,j_\sigma}\}$
4: for $|\psi\rangle$ in $B$, $E$ in $K$ do
5:     $|\phi\rangle \leftarrow cont(|\psi\rangle, E)$
6:     $P = P \vee span\{|\phi\rangle\}$
7: end for
8: return $P$
```

The basic procedure is to find a basis of the initial subspace and then calculate $E|\psi\rangle$ for every Kraus operator E and basis state $|\psi\rangle$ (both are represented as TDDs) and then calculate the join of all these states. In this paper, we use $cont(\phi_1, \cdots, \phi_k)$ to represent the contraction of k tensors $\phi_1, \cdots, \phi_k$.

V. Partition-Based Optimisations

Representing a transition relation as a monolithic BDD usually needs huge memory resources. In classical model checking, partition-based methods are introduced [8] to optimise image computation by partitioning the transition relation as the disjunction or conjunction of smaller transition relations. Analogously, by leveraging the properties of tensor networks and TDDs, we can partition a quantum circuit (regarded as a tensor network or a TDD) into many parts, which can be added or contracted to recover the functionality of the quantum circuit. The two methods are addressed as, respectively, *addition* and *contraction* partitions.

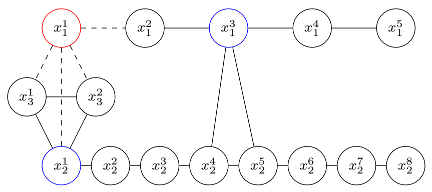

Fig. 5. The undirected graph for Grover iteration (cf. Fig. 2), where x_i^j represents the j-th index of the i-th qubit of the circuit.

A. Addition Partition

Regarding quantum circuits as tensor networks, we first transform (as in [23]) a quantum circuit C as an undirected graph G and then partition the circuit according to some vertices with the highest degrees. More specifically, every node in G represents an index of the quantum circuit and two nodes are connected in G if they are the input or output indices of the same gate. In this work, we regard two indices as the same if (i) they are the input and output indices of a diagonal quantum gate; or (ii) they are the input and output indices of a control qubit of a controlled gate. That is, hyper-edges [24] can appear in G. This graph captures the connectivity of the quantum circuit and facilitates the partition of the circuit by slicing some indices with the highest degrees. We call this method *addition partition*.

Consider the circuit for Grover iteration (see Fig. 2) again. The corresponding undirected graph is shown in Fig. 5. In this circuit, we examine each qubit wire from left to right one by one and use x_i^j to represent the j-th index of the i-th qubit. From Fig. 5, we can see that the nodes labelled with x_1^1, x_2^1, and x_1^3 have the highest degree. We choose any one of the indices and slice the circuit into two simpler ones by giving a value of 0 or 1 to the index. We can also choose two or all of the indices to partition the circuit into four or eight simpler parts.

Suppose a circuit with tensor representation ϕ has been partitioned into several parts as above. Let $\phi_1, \cdots, \phi_k$ be their tensor representations. To calculate the image of a state $|\psi\rangle$ under the tensor ϕ, we need only calculate the contractions $cont(|\psi\rangle, \phi_i)$ and then add them up. This is because $cont(|\psi\rangle, \phi) = \sum_i cont(|\psi\rangle, \phi_i)$.

The advantage of addition partition lies in that calculations of large TDD, as in the basic method, can be avoided. Moreover, contractions of different parts $cont(|\psi\rangle, \phi_i)$ can be done in parallel.

B. Contraction Partition

In contraction partition, we cut the quantum circuit into several smaller parts whose contraction equals the original one. For two preset integer parameters k_1 and k_2, we partition the circuit into small parts such that every part involves at most k_1 qubits and connects with at most k_2 multi-qubit gates that across different parts (cf. [15]).

Take the bit-flip code circuit (cf. Fig. 3) as an example. Setting $k_1 = 3$ and $k_2 = 2$, the circuit is cut into six blocks as shown in Fig. 3. The method first cuts the circuit *horizontally* into $\lceil n/k_1 \rceil$ parts and then cuts the circuit *vertically* whenever k_2 multi-qubit gates have been cut, where n is the total number of qubits of the circuit. For this example, $n = 6$ and the circuit is cut horizontally into two parts. Whenever the horizontal line cuts two CX gates, we add a vertical line. As a consequence, the circuit is cut into six parts.

Suppose ϕ and $\phi_1, \cdots, \phi_k$ are the tensor representations of the circuit and its k blocks. Then $\phi = cont(\phi_1, \cdots, \phi_k)$. To compute the image of a state $|\psi\rangle$ under the transition specified by ϕ, we need only contract the tensor network connected by these ϕ_i and $|\psi\rangle$. By using this method, we can avoid calculating the TDD that represents the whole circuit directly. In most cases, it takes much less memory space to store these smaller TDDs and it takes less time contracting these ϕ_i with $|\psi\rangle$ as they have small ranks.

VI. Empirical Evaluation

In this section, we evaluate the effectiveness and efficiency of our algorithms. All experiments were conducted on a server with an Intel Xeon-Gold-5215 CPU and 512GB RAM.

A. Comparison Among Three Methods

We compare and demonstrate the scalability of our methods on a set of well-known benchmarks. The transition relations of our experiments cover a range of quantum circuits, including circuits for preparing GHZ states and circuits for algorithms such as Grover's algorithm, Bernstein-Vazirani algorithm (BV), quantum Fourier transform (QFT), and quantum random walk (QRW). For the quantum random walk, we add a bit-flip error at the coin qubit after the Hadamard gate. We use the commonly used input states to form the initial subspace of these algorithms.

Table I presents the experimental results for three image computation methods: the basic algorithm, addition partition, and contraction partition. The first column gives the names of the circuits, which also indicate the qubit number. Each of the rest columns corresponds to one of these methods, displaying the time (in seconds) and the maximum node number of TDDs generated during the image computation of the corresponding circuit. For addition partition, we assign a parameter value of $k = 1$, which corresponds to dividing the circuit into two parts. For the contraction partition, we set $k_1 = k_2 = 4$, indicating a horizontal circuit cut every four qubits and a vertical cut whenever four multi-qubit gates which have been horizontally cut are encountered.

From Table I we can see that all three methods can calculate the images of the 500-qubit BV in 5 minutes and 500-qubit GHZ in 4 seconds. For Grover's algorithm, QFT and QRW, the basic algorithm and addition partition are difficult to go beyond 20 qubits. In general, addition partition is better than the basic algorithm. For example, for 'QRW_20', the basic algorithm takes about 341 seconds to finish while addition partition only takes 218 seconds. The maximal node numbers show the same trend. Compared with the basic algorithm and addition partition, our contraction partition method is way more efficient. For example, it takes only 14 seconds for 'QRW_20'. Moreover, it can go far beyond 20 qubits for Grover, QFT, and QRW circuits. More importantly, the maximum TDD size of contraction partition increases at most linearly for QFT, BV,

TABLE I
EXPERIMENT DATA

Benchmark	basic		addition		contraction	
	time	max #node	time	max #node	time	max #node
Grover_15	19.33	15785	17.35	15099	1.61	597
Grover_18	76.47	61694	66.02	60332	2.41	516
Grover_20	294.65	243946	259.87	241240	4.39	1036
Grover_40	-		-		2953.57	851973
QFT_15	34.64	65536	18.88	32770	0.08	63
QFT_18	282.12	524288	148.13	262146	0.10	31
QFT_20	1199.21	2097152	655.19	1048578	0.12	63
QFT_30	-		-		0.29	31
QFT_50	-		-		1.02	51
QFT_100	-		-		7.14	101
BV_100	7.36	596	7.43	596	0.41	102
BV_200	31.57	1196	30.03	1196	1.70	202
BV_300	75.66	1796	75.56	1796	4.28	302
BV_400	146.47	2396	145.40	2396	9.18	402
BV_500	244.15	2996	223.90	2996	16.31	502
GHZ_100	0.38	595	0.13	301	0.18	200
GHZ_200	0.72	1195	0.37	601	0.48	400
GHZ_300	1.29	1795	0.62	901	0.80	600
GHZ_400	2.03	2395	1.00	1201	1.26	800
GHZ_500	2.96	2995	1.45	1501	1.72	1000
QRW_15	36.86	13122	24.59	10882	7.16	222
QRW_18	139.76	90538	84.69	37064	11.23	226
QRW_20	341.05	265614	218.29	107714	14.31	404
QRW_30	-		-		36.82	404
QRW_50	-		-		118.08	404
QRW_100	-		-		692.08	436

TABLE II
TIME CONSUMPTION (SECONDS) FOR CONTRACTION PARTITION

k1 \ k2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.95	1.58	1.80	1.67	1.62	1.89	2.57	2.54	3.64	3.82	6.02	3.07	3.28	4.88	6.88
2	1.48	1.50	1.74	1.52	1.58	1.90	2.08	2.18	3.01	2.25	4.89	2.68	2.57	4.12	5.63
3	1.51	1.49	1.63	1.54	1.48	2.04	2.29	2.05	3.04	2.17	3.79	2.38	2.59	3.57	3.96
4	1.42	1.50	1.63	1.61	1.62	2.20	2.44	2.29	3.04	2.39	3.85	2.91	2.93	3.87	4.86
5	1.65	1.65	1.75	1.64	1.90	2.14	2.33	2.34	2.86	2.20	3.83	2.58	2.55	3.79	4.29
6	1.69	1.74	1.71	1.75	1.60	2.02	2.27	1.91	2.78	2.14	3.27	2.38	2.93	3.76	3.76
7	1.42	1.31	1.74	1.78	1.70	3.10	3.41	2.48	4.21	2.36	3.59	3.35	3.43	5.91	3.98
8	1.37	1.51	1.87	2.60	2.15	3.45	3.84	3.03	4.60	3.33	4.56	3.70	4.14	6.43	5.61
9	1.31	1.43	1.48	2.91	2.78	2.83	3.53	4.33	2.86	4.48	3.54	3.12	2.91	4.11	6.03
10	1.35	1.40	2.04	2.60	3.63	3.24	3.19	2.65	5.14	3.37	3.21	3.49	4.14	5.93	6.71
11	1.23	1.46	1.48	3.09	2.81	2.70	4.37	2.53	4.85	3.74	5.23	5.11	3.30	6.78	8.13
12	1.32	2.06	2.84	3.05	3.57	3.91	3.90	2.04	6.03	5.99	6.55	3.02	4.76	6.95	32.33
13	1.79	1.97	3.63	4.48	4.01	5.92	6.73	2.72	3.79	27.38	6.28	9.22	12.57	72.23	27.10
14	2.72	2.24	5.94	4.43	24.81	8.52	25.18	4.85	5.30	55.04	58.81	109.59	43.82	43.84	48.41
15	2.96	1.85	6.42	2.43	8.67	14.61	37.10	4.97	10.09	7.39	24.11	22.31	24.19	24.04	36.50

GHZ, and QRW. In the table, we use '-' to indicate that the time exceeds the timeout of 3600s. It should be noted that, while our method—particularly the contraction partition—has demonstrated good performance in the examples above, the algorithm's overall complexity remains exponential in the worst case.

B. Impact of Parameters for Contraction Partition

In this subsection, we investigate the influence of the parameters k_1 and k_2 on the performance of contraction partition. We use the circuit 'Grover_15' as an example and calculate images for different k_1 and k_2 ranging from 1 to 15. The experimental results are shown in Table II, where the value at entry (k_1, k_2) represents the image computation time for the corresponding parameters k_1, k_2 in contraction partition. In this table, times under 2 seconds are highlighted in purple, while times exceeding 5 seconds are marked in blue, with the colour intensity increasing as the duration lengthens. From Table II we can see that the contraction partition method is efficient as long as we do not set the parameters k_1, k_2 too large. This means that there is a wide range of values that can be chosen for the parameters.

VII. CONCLUSION

Image computation plays a key role in model checking classical and quantum transition systems. This paper focuses on advancing the development of model checking for quantum systems by introducing efficient image computation algorithms. We represent quantum circuits as tensor networks and leverage the properties of tensor networks and tensor decision diagrams to design efficient image computation algorithms. Empirical evaluation demonstrates that the contraction partition-based algorithm can greatly improve the efficiency of image computation for quantum transition systems.

REFERENCES

[1] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Checking equivalence of quantum circuits and states," in *International Conference on Computer-Aided Design, ICCAD*, G. G. E. Gielen, Ed., 2007, pp. 69–74.

[2] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 40, no. 9, pp. 1810–1824, 2021.

[3] X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, "Approximate equivalence checking of noisy quantum circuits," in *58th ACM/IEEE Design Automation Conference, DAC*, 2021, pp. 637–642.

[4] L. Burgholzer and R. Wille, "Handling non-unitaries in quantum circuit equivalence checking," in *DAC '22: 59th ACM/IEEE Design Automation Conference, San Francisco*, R. Oshana, Ed., 2022, pp. 529–534.

[5] L. Burgholzer, R. Kueng, and R. Wille, "Random stimuli generation for the verification of quantum circuits," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 767–772.

[6] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 4, pp. 401–424, 1994.

[7] H. Cho, G. D. Hachtel, E. Macii, B. Plessier, and F. Somenzi, "Algorithms for approximate FSM traversal based on state space decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1996.

[8] J. R. Burch, E. M. Clarke, and D. E. Long, "Symbolic model checking with partitioned transition relations," Carnegie Mellon University, Tech. Rep. CMU-CS-91-195, 1991.

[9] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, Somenzi *et al.*, "VIS: A system for verification and synthesis," in *International conference on computer aided verification*. Springer, 1996, pp. 428–432.

[10] S. Gay, R. Nagarajan, and N. Papanikolaou, "Probabilistic model–checking of quantum protocols," *arXiv preprint quant-ph/0504007*, 2005.

[11] Y. Feng, E. M. Hahn, A. Turrini, and L. Zhang, "QPMC: A model checker for quantum programs and protocols," in *FM 2015: Formal Methods: 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings 20*. Springer, 2015, pp. 265–272.

[12] M. Ying and Y. Feng, *Model Checking Quantum Systems: Principles and Algorithms*. Cambridge University Press, 2021.

[13] G. Birkhoff and J. Von Neumann, "The logic of quantum mechanics," *Annals of mathematics*, pp. 823–843, 1936.

[14] M. Ying, "Model checking for verification of quantum circuits," in *International Symposium on Formal Methods*. Springer, 2021, pp. 23–39.

[15] X. Hong, X. Zhou, S. Li, Y. Feng, and M. Ying, "A tensor network based decision diagram for representation of quantum circuits," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 6, pp. 1–30, 2022.

[16] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan, "Verification of concurrent quantum protocols by equivalence checking," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014, pp. 500–514.

[17] S. J. Gay, R. Nagarajan, and N. Papanikolaou, "QMC: A model checker for quantum systems," in *International Conference on Computer Aided Verification*. Springer, 2008, pp. 543–547.

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[19] A. D. Córcoles, M. Takita, K. Inoue, S. Lekuch, Z. K. Minev, J. M. Chow, and J. M. Gambetta, "Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits," *Physical Review Letters*, vol. 127, no. 10, p. 100501, 2021.

[20] B. Douglas and J. Wang, "Efficient quantum circuit implementation of quantum walks," *Physical Review A*, vol. 79, no. 5, p. 052335, 2009.

[21] F. Acasiete, F. P. Agostini, J. K. Moqadam, and R. Portugal, "Implementation of quantum walks on IBM quantum computers," *Quantum Information Processing*, vol. 19, pp. 1–20, 2020.

[22] K. L. McMillan, *Symbolic Model Checking*. Kluwer, 1993. [Online]. Available: https://doi.org/10.1007/978-1-4615-3190-6

[23] J. Chen, F. Zhang, C. Huang, M. Newman, and Y. Shi, "Classical simulation of intermediate-size quantum circuits," *arXiv preprint arXiv:1805.01450*, 2018.

[24] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, E. W. Draeger, E. T. Holland, and R. Wisnieff, "Pareto-efficient quantum circuit simulation using tensor contraction deferral," 2020.