Approximate Equivalence Checking of Noisy Quantum Circuits

Xin Hong[1], Mingsheng Ying[1,2,3*], Yuan Feng[1*], Xiangzhen Zhou[1,4] and Sanjiang Li[1*]

[1]*Centre for Quantum Software and Information, University of Technology Sydney, Australia*
[2]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China*
[3]*Department of Computer Science and Technology, Tsinghua University, China*
[4]*State Key Lab of Millimeter Waves, Southeast University, China*

***Abstract*—We study the fundamental design automation problem of equivalence checking in the NISQ (Noisy Intermediate-Scale Quantum) computing realm where quantum noise is present inevitably. The notion of approximate equivalence of (possibly noisy) quantum circuits is defined based on the Jamiolkowski fidelity which measures the average distance between output states of two super-operators when the input is chosen at random. By employing tensor network contraction, we present two algorithms, aiming at different situations where the number of noises varies, for computing the fidelity between an ideal quantum circuit and its noisy implementation. The effectiveness of our algorithms is demonstrated by experimenting on benchmarks of real NISQ circuits. When compared with the state-of-the-art implementation incorporated in Qiskit, experimental results show that the proposed algorithms outperform in both efficiency and scalability.**

***Index Terms*—Quantum computing, quantum circuits, noise, equivalence checking**

I. Introduction

Equivalence checking techniques have been widely employed in the EDA (Electronic Design Automation) industry to check whether two different circuit designs exhibit exactly the same behaviour. This is very important to ensure that a design is error-free. Important techniques such as BDD [1, 2] and SAT [2, 3] have been successful in such a task in the classical case.

Nowadays, quantum computing has attracted great attention from both industry and academia due to its potential speedup in solving problems such as integer factorisation [4], database search [5], and many machine learning tasks [6]. Experiments have been proposed to demonstrate quantum supremacy [7] and to explore the possibilities of simulating a chemical system [8]. International leading IT companies such as IBM and Google have invested enormous resources to develop quantum hardware and software. It is expected that quantum devices with several hundreds valid qubits will appear very soon.

As quantum circuits become larger and larger, they are more and more error-prone. Equivalence checking of quantum circuits has been discussed in [9–14], where it is assumed that each gate in either circuit is represented as a unitary operation and the aim is to check if the two unitary matrices representing the two circuits are equal (up to a global phase). Often, canonical decision diagram representations for the two circuits are constructed and the two circuits are equivalent if and only if they have the same representation.

On the other hand, quantum computers we have at present or in the near future are all Noisy Intermediate-Scale Quantum (NISQ) devices and quantum circuits executed on them are all plagued by more or less noise. For such kinds of circuits, the proposed approaches for equivalence checking would fail as noisy gates cannot be represented by unitary matrices, and a qualitative answer (being equivalent or not) does not make sense. Thus, it is necessary and very important to develop new methods for checking the approximate equivalence, or computing the distance, of noisy circuits.

*Corresponding authors:{mingsheng.ying,yuan.feng,sanjiang.li}@uts.edu.au.

This paper aims at proposing a proper definition as well as efficient algorithms for the problem of approximate equivalence between one ideal (unitary) circuit and its noisy implementation. The key notion is the Jamiolkowski fidelity [15] which measures the average distance between output states of two super-operators when the input is chosen at random. In addition to the clear physical interpretation, Jamiolkowski fidelity enjoys some nice properties such as stability and chaining [16] which guarantee that the error scales at most linearly when smaller noisy quantum circuits are concatenated into a larger one. More importantly, computing this measure of distance can be reduced to the calculation of traces of some (non-unitary) matrices obtained from the original circuits, a task which can be done efficiently by employing tensor network contraction. With these observations, we present two algorithms, aiming at different situations where the number of noises varies, and implement them by using a recently developed software package — Tensor Decision Diagram (TDD) [17] which represents a tensor as a decision diagram and supports efficient contraction of tensor networks. The effectiveness of our algorithms and implementation is demonstrated by experimenting on real NISQ circuits. Experimental results show that our algorithms outperform in both efficiency and scalability over the state-of-the-art implementation incorporated in Qiskit [18].

Related works. Existing works in equivalence checking of quantum circuits are based on Decision Diagrams (DDs) or SAT or a combination of them [9–14]. In particular, QMDD [19], which provides a more economical way to represent unitary operators when compared with array-based representations, is used in [11] as the underlying data structure for equivalence checking. To our best knowledge, there are no attempts in extending these methods to deal with approximate equivalence of noisy quantum circuits.

Distance measures of quantum processes have been considered in [16, 20]. In particular, the Jamiolkowski fidelity is identified in [16] as one of the best measures for comparing quantum processes. Biamonte [21] describes a method for calculating the average gate fidelity based on tensor networks, where quantum processes are expressed in terms of Choi-matrices. However, these matrices are not convenient to compute from the classical description of quantum noisy circuits.

The important notion of reversible miter is introduced in [10] for equivalence checking of (unitary) quantum circuits. For two circuits C, D, a miter is constructed by concatenating C with the reverse of D, and the two circuits are equivalent if and only if the miter implements the identity operator. When combined with local optimisations like cancelling two consecutive gates, this technique can significantly simplify the circuits and thus the process of equivalence checking. Such a miter is also used in [11], where a more efficient method for equivalence checking is designed by carefully choosing the gates to be calculated every time. Our algorithms also rely on calculating the traces of similar miter constructions.

Classical simulation of noisy quantum circuits are considered in

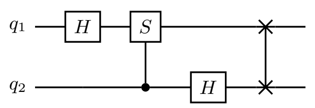


Fig. 1. Circuit for 2-qubit quantum Fourier transform (QFT).

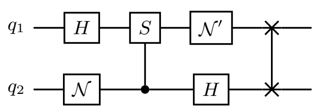


Fig. 2. A noisy circuit for 2-qubit QFT with two noises inserted.

[22] and [23]. Gao and Duan [22] develop a tensor network tool to represent the ensemble of noisy quantum circuits, while Grurl et al. [23] represent density matrices as decision diagrams and show that considering decoherence errors does not necessarily affect the compactness of the decision diagram representations. Li et al. [24] propose an optimisation technology for the Monte Carlo simulation of noisy quantum circuits, where they exploit the structure similarity of an ensemble of noisy circuits to be simulated and try to maximally reuse precomputed states. Such a technology is inherently used by decision diagrams, where a single shared Hash table, called computed table, is used to store all nodes of the decision diagrams generated in the whole calculation process.

II. Quantum Circuits and Noise

In this section, we review some basic concepts from quantum computing to help understand noisy quantum circuits.

A. Quantum Circuits

In classical digital computation, data are represented by a string of bits. When sending through a classical circuit, the state of the input will be transformed by a sequence of logical gates. In quantum computing, the counterpart of bit is called *qubit*. The state of a qubit is often represented in Dirac notation $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0 and α_1 are complex numbers, called the amplitudes of $|\psi\rangle$, and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. We also use the vector $[\alpha_0, \alpha_1]^T$ to represent a single-qubit state. In general, an n-qubit quantum state is represented as a 2^n-dimensional complex vector $[\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}]^T$.

The evolution of a quantum system is described by a unitary transformation, which is usually called a *quantum gate* in quantum computing. Generally, an n-qubit quantum gate is represented as a $2^n \times 2^n$-dimensional unitary transformation matrix. Typical 1-qubit gates used in this paper include Pauli gates

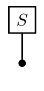
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

Hadamard gate $H = (X + Z)/\sqrt{2}$, and phase gate $S = \sqrt{Z}$. The quantum state after applying a gate can be obtained by multiplying the corresponding unitary matrix and the vector that represents the input state. For example, the output state resulted from applying a Hadamard gate to $|\psi\rangle$ defined above is calculated as follows

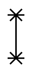
$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha_0 + \alpha_1 \\ \alpha_0 - \alpha_1 \end{bmatrix}.$$

A *quantum circuit* consists of a set of qubits and a sequence of quantum gates. Given an input state, quantum gates in the circuit are applied to the corresponding qubits in a sequential manner. Clearly, an n-qubit quantum circuit also describes a functionality represented as a $2^n \times 2^n$-dimensional unitary matrix.

Example 1. *Depicted in Fig. 1 is a circuit which implements the 2-qubit quantum Fourier transform (QFT) [25], where*



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix},$$



$$= SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For simplicity, we often use the same symbol, say U, to denote both a quantum gate (or circuit) and its corresponding unitary matrix.

B. Noisy quantum circuits

The pure state-unitary transformation framework presented above works perfectly well for ideal (noiseless) quantum computing. However, to verify noisy implementation of ideal quantum circuits, which is the main purpose of the current paper, we have to extend this framework to accommodate mixed states and super-operators.

Mathematically, a *mixed state* of an n-qubit system is described by a $2^n \times 2^n$ density matrix, i.e., a positive semi-definite matrix ρ with $\mathrm{tr}(\rho) = 1$. Obviously, for a pure state $|\psi\rangle$, the outer product $|\psi\rangle\langle\psi|$, denoted by ψ henceforth, of $|\psi\rangle$ with its complex conjugate $\langle\psi|$ is a mixed state.

Dynamics of an n-qubit system whose states are given as mixed states is modelled by a *super-operator* which is a linear map $\mathcal{E}$ between $2^n \times 2^n$ density matrices. A convenient way of representing an n-qubit super-operator $\mathcal{E}$ is the *Kraus operator-sum form*: there exist a set of $2^n \times 2^n$ matrices $\{E_i\}$ satisfying the normalisation condition $\sum_i E_i^\dagger E_i = I_{2^n}$, such that for each mixed state ρ, $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$. In this case, we write $\mathcal{E} = \{E_i\}$. Note that unitary operator U can also be seen as a super-operator $\mathcal{U} : \rho \mapsto U\rho U^\dagger$.

Noises in quantum computing can be naturally represented as super-operators (in the Kraus operator-sum form).

Example 2. *Given $p \in [0, 1]$, several canonical noises on a single qubit are:*

1) *Bit flip. This kind of noise flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa with probability $1 - p$. It is modelled by the super-operator $\mathcal{N}_{bf}(\rho) = p\rho + (1 - p)X\rho X$.*
2) *Phase flip. This noise applies a phase operator Z on the target qubit with probability $1 - p$. It is given as the super-operator $\mathcal{N}_{pf}(\rho) = p\rho + (1 - p)Z\rho Z$.*
3) *Bit-phase flip. This noise applies Y on the target qubit with probability $1-p$: $\mathcal{N}_{bpf}(\rho) = p\rho+(1-p)Y\rho Y$. Obviously, it is a combination of a bit-flip and a phase flip because $Y = iXZ$.*
4) *Depolarisation. The depolarisation of a qubit with parameter p is modelled by $\mathcal{N}_{dep}(\rho) = p\rho + \frac{1-p}{3}(X\rho X + Y\rho Y + Z\rho Z)$.*

Now by allowing occurrence of noises, we can easily extend the notions of quantum gates and circuits to the noisy ones. Specifically, a noisy quantum circuit is composed of noisy quantum gates which are represented as super-operators instead of only unitary operators. Since unitary operators can be regarded as super-operators, noiseless gates and circuits are special cases of their noisy counterparts. As a simple example, Fig. 2 shows a noisy implementation of the 2-qubit QFT where two 1-qubit noises $\mathcal{N}$ and $\mathcal{N}'$ are introduced.

Again, we abuse the notation slightly to use the same symbol to denote both a noisy quantum gate (or circuit) and its corresponding super-operator. However, to avoid confusion, we usually use $\mathcal{N}$, $\mathcal{N}'$, etc., for noisy gates and $\mathcal{E}$, $\mathcal{E}'$, etc., for noisy circuits.

III. Approximate Equivalence of Quantum Circuits

This section aims at formalising the problem of approximate equivalence checking of noisy quantum circuits. The key notion is an appropriate definition of distance between the ideal circuit, mathematically represented as a unitary operator U, and its noisy implementation, mathematically represented as a super-operator $\mathcal{E}$. Our main references for this section are [25, Chapter 9] and [16].

To this end, we first recall that the *fidelity* between two density operators ρ and σ on a Hilbert space $\mathcal{H}$ is defined by

$$F(\rho,\sigma) = (\mathrm{tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}})^2.$$

When $\rho = |\psi\rangle\langle\psi|$ is a pure state, $F(\psi,\sigma) = \langle\psi|\sigma|\psi\rangle$.

The fidelity between density operators can be extended to measure distance between super-operators with the help of *Jamiolkowski isomorphism* that maps each super-operator $\mathcal{E}$ on $\mathcal{H}$ to a density operator $\rho_{\mathcal{E}} = (\mathcal{I}\otimes\mathcal{E})(|\Psi\rangle\langle\Psi|)$ on $\mathcal{H}\otimes\mathcal{H}$, where $\mathcal{I}$ is the identity super-operator on $\mathcal{H}$, $|\Psi\rangle = \frac{1}{\sqrt{d}}\sum_i |ii\rangle$ is the maximally entangled state, $d = \dim\mathcal{H}$, and $\{|i\rangle\}$ is an orthonormal basis of $\mathcal{H}$. The *Jamiolkowski fidelity* between super-operators $\mathcal{E}$ and $\mathcal{F}$ is defined as

$$F_J(\mathcal{E},\mathcal{F}) = F(\rho_{\mathcal{E}},\rho_{\mathcal{F}}).$$

In the special case when $\mathcal{F} = \{U\}$ is a unitary operation, which is exactly what we are concerned with in this paper, we slightly abuse the notation to write $F_J(\mathcal{E},U)$ instead of $F_J(\mathcal{E},\mathcal{F})$.

Note that there are a wide range of distance measures for density operators and super-operators presented in the literature [16], each having applications in different quantum information tasks. Our design decision to choose Jamiolkowski fidelity to characterise the approximate equivalence of an ideal circuit and its noisy implementation is based on the following observations.

Physical interpretation. The Jamiolkowski fidelity F_J can serve as the average-case error measure in computation of a function or sampling distribution [16], which is particularly useful in quantum simulation and quantum machine learning. To be specific, for super-operator $\mathcal{E}$ and unitary operator U, we have

$$\int d\psi F(\mathcal{E}(\psi), U|\psi\rangle) = \frac{d\cdot F_J(\mathcal{E},U)+1}{d+1},$$

where the integral is over the Haar measure on $\mathcal{H}$. In other words, $F_J(\mathcal{E},U)$ characterises the *average fidelity* of the output $\mathcal{E}(\psi)$ of the noisy circuit $\mathcal{E}$ and that of the ideal one $U|\psi\rangle$, when the input pure state $|\psi\rangle$ is chosen at random.

Nice properties. It is easy to prove that the Jamiolkowski fidelity enjoys the following properties (see [16]):

1) *Stability*: $F_J(\mathcal{E}\otimes\mathcal{I},\mathcal{F}\otimes\mathcal{I}) = F_J(\mathcal{E},\mathcal{F})$, where $\mathcal{I}$ stands for the identity operation on an arbitrary ancillary system;
2) *Chaining*: $C_J(\mathcal{E}_1\circ\mathcal{E}_2, U_1\circ U_2) \le C_J(\mathcal{E}_1,U_1) + C_J(\mathcal{E}_2,U_2)$, where $C_J(\mathcal{E},\mathcal{F}) = \sqrt{1-F_J(\mathcal{E},\mathcal{F})}$ is a metric between super-operators induced by F_J.

In the scenario of approximate equivalence checking, these properties guarantee that the error scales at most linearly when smaller noisy quantum circuits are concatenated into a larger one.

Ease of calculation. Given an ideal circuit U and a noisy implementation $\mathcal{E} = \{E_i\}$, we have

$$\begin{aligned} F_J(\mathcal{E},U) &= \sum_i \langle\Psi|\, I\otimes U^\dagger E_i\,|\Psi\rangle\,\langle\Psi|\, I\otimes E_i^\dagger U\,|\Psi\rangle \\ &= \sum_i |\,\langle\Psi|\, I\otimes U^\dagger E_i\,|\Psi\rangle\,|^2 = \frac{1}{d^2}\sum_i |\mathrm{tr}(U^\dagger E_i)|^2. \end{aligned}$$

Thus the calculation of Jamiolkowski fidelity $F_J(\mathcal{E},U)$ boils down to computing the traces of some matrices. As we are going to show in the following sections, this can be done quite easily by employing the recently proposed Tensor Decision Diagram (TDD) techniques [17].

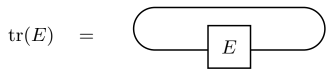


Fig. 3. The tensor network of computing the trace of the functionality matrix E of a (noiseless or noisy) circuit.

To conclude this section, we propose the notion of approximate equivalence between quantum circuits.

Definition 1. *Let $\epsilon \in [0,1]$, and $\mathcal{C}_1,\mathcal{C}_2$ be two (noisy) quantum circuits represented by super-operators $\mathcal{E}_1,\mathcal{E}_2$, respectively. Then $\mathcal{C}_1$ and $\mathcal{C}_2$ are ϵ-equivalent, written $\mathcal{C}_1 \approx_\epsilon \mathcal{C}_2$, if $F_J(\mathcal{E}_1,\mathcal{E}_2) > 1-\epsilon$.*

The problem of approximate equivalence checking of noisy quantum circuits can then be formalised as follows:

Problem 1. *Let $\epsilon \in [0,1]$ be an error threshold. Given (the classical description of) an ideal circuit $\mathcal{C}$ and a noisy implementation $\mathcal{N}$, determine if $\mathcal{C} \approx_\epsilon \mathcal{N}$.*

IV. Algorithms for Approximate Equivalence Checking

In the previous section, we formalise the problem of approximate equivalence checking for quantum circuits. This section is devoted to two algorithms solving this problem in different situations.

Before diving into the details, we first note that when regarding a quantum circuit as a tensor network, the trace of the functionality matrix represented by the circuit can be calculated by connecting the input qubits with the corresponding output ones and contracting the obtained network. Fig. 3 shows a general scheme of computing the trace of the functionality matrix E of a (noiseless or noisy) circuit.

A. Algorithm I: Calculate Traces Individually

Recall that $F_J(\mathcal{E},U) = \sum_i |\mathrm{tr}(U^\dagger E_i)|^2/d^2$ where $\mathcal{E} = \{E_i\}$. Our first algorithm calculates $\mathrm{tr}(U^\dagger E_i)$ for each E_i and adds up the squares of their norms. Specifically, suppose there are m noisy gates $\mathcal{N}_1,\cdots,\mathcal{N}_m$ in the circuit and $\mathcal{N}_k = \{N_1^k,\cdots,N_{n_k}^k\}$ for each k. Then every choice of $j_k \in [1,n_k]$, $1\le k\le m$, determines a Kraus operator E_i of $\mathcal{E}$ by replacing each $\mathcal{N}_k$ in the noisy circuit with the non-unitary gate $N_{j_k}^k$. Note that the total number of different choices is $\prod_k n_k$. Furthermore, as the ideal circuit U represents a unitary operator, we obtain a circuit representing $U^\dagger$ by replacing each gate in U with its Hermitian conjugate and reversing the order of gates. Then $\mathrm{tr}(U^\dagger E_i)$ can be computed by concatenating the circuits of E_i and $U^\dagger$ (the order is irrelevant), connecting the corresponding input and output qubits, and then contracting the obtained tensor network.

Example 3. *Consider the noisy circuit $\mathcal{E}$ shown in Fig. 2 which implements the 2-qubit quantum Fourier transform QFT_2. Suppose the noisy gate $\mathcal{N} = \{N_1,N_2\}$ represents a bit flip, with $N_1 = \sqrt{p}I$ and $N_2 = \sqrt{1-p}X$, and $\mathcal{N}' = \{N_1',N_2'\}$ represents a phase flip, with $N_1' = \sqrt{p}I$ and $N_2' = \sqrt{1-p}Z$. Then, we have four tensor networks that need to be contracted, all having form shown in Fig. 4. Contracting these tensor networks gives $\mathrm{tr}(U^\dagger E_{1,1}) = 4p$, and $\mathrm{tr}(U^\dagger E_{1,2}) = \mathrm{tr}(U^\dagger E_{2,1}) = \mathrm{tr}(U^\dagger E_{2,2}) = 0$, where $U = QFT_2$, $E_{i,j}$ is the functionality matrix of the circuit in Fig. 2 with $\mathcal{N}$ being replaced by N_i and $\mathcal{N}'$ replaced by N_j'. Thus $F_J(\mathcal{E},U) = p^2$.*

Note that the number of circuits to be contracted increases exponentially w.r.t. the number of noisy gates in the circuit. This method will be efficient only when there are very few noisy gates. However, we note that distributed computing techniques can help in contracting these circuits in parallel, since they are completely independent. Furthermore, for the purpose of approximate equivalence checking, it

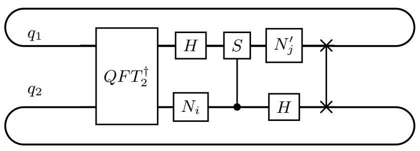


Fig. 4. Tensor networks to be contracted in Example 3 using Alg. I.

is often sufficient to calculate only a small part of these trace terms, as an affirmative answer is obtained once a partial sum is greater than $1-\epsilon$. Consider the above example again. Suppose $p = 0.95$ and our aim is to check if $\mathcal{E} \approx_{0.1} \mathcal{U}$. Clearly, computing $\mathrm{tr}(U^\dagger E_{1,1})$ already suffices as $F_J(\mathcal{E}, U) \geq (4p)^2/4^2 = 0.9025 > 0.9 = 1 - 0.1$. In practice, this can partially alleviate the problem caused by the exponential number of trace calculations.

B. Algorithm II: Calculate Traces Collectively

We now introduce the second algorithm for approximate equivalence checking, which calculates $\sum_i |\mathrm{tr}(U^\dagger E_i)|^2$ collectively by contracting a single (in contrast to the exponentially many in Alg. I) tensor network, although it is twice bigger than the tensor networks contracted in Alg. I.

To see how it works, we note that

$$\begin{aligned}\sum_i |\mathrm{tr}(U^\dagger E_i)|^2 &= \sum_i \mathrm{tr}(U^\dagger E_i)\cdot \mathrm{tr}(U^T E_i^*)\\ &= \sum_i \mathrm{tr}(U^\dagger E_i \otimes U^T E_i^*) = \mathrm{tr}\big((U^\dagger \otimes U^T) M_{\mathcal{E}}\big),\end{aligned}$$

where U^T is the transpose of U and E_i^* the complex conjugate of E_i, both with respect to a fixed orthonormal basis of the state space, and for each super-operator $\mathcal{E} = \{E_i\}$, $M_{\mathcal{E}} := \sum_i E_i \otimes E_i^*$ is the *matrix representation* of $\mathcal{E}$ [26]. In other words, the quantity $\sum_i |\mathrm{tr}(U^\dagger E_i)|^2$ as a whole can be computed by the following steps:

1) introduce for each qubit q_j in the ideal circuit an auxiliary one q'_j, and do the same for the noisy circuit;
2) for each (unitary) gate V acting on $q_{j_1}, \ldots, q_{j_\ell}$ in the ideal circuit, add the complex conjugate gate V^* acting on $q'_{j_1}, \ldots, q'_{j_\ell}$. Denote by U_E the obtained circuit. Note also that U_E consists of two completely separated circuits: the original one U and the newly added one U^*;
3) for each unitary gate in the noisy circuit, do the same as for the ideal circuit; for each noisy gate $\mathcal{N} = \{N_k\}$ acting on $q_{j_1}, \ldots, q_{j_\ell}$, replace it with the corresponding $M_{\mathcal{N}} = \sum_k N_k \otimes N_k^*$, operating on $q_{j_1}, \ldots, q_{j_\ell}, q'_{j_1}, \ldots, q'_{j_\ell}$. Denote by $\mathcal{E}_E$ the obtained circuit;
4) concatenate $U_E^\dagger$ and $\mathcal{E}_E$ (again, the order is irrelevant), connect the corresponding input and output qubits, and contract the obtained tensor network, as in Alg. I.

Example 4. *Back to the noisy circuit shown in Fig. 2 implementing QFT*$_2$. *Let $\mathcal{N}$ and $\mathcal{N}'$ be defined as in Example 3. Alg. II requires to contract the (single) tensor network in Fig. 5. Note that two auxiliary qubits q'_1 and q'_2 are added, and the noisy gate $\mathcal{N}$ applied on q_2 is replaced by a two-qubit gate $M_{\mathcal{N}} = pI\otimes I + (1-p)X\otimes X$ applied on q_2 and q'_2. Similarly, $\mathcal{N}'$ is replaced by $M_{\mathcal{N}'} = pI\otimes I + (1-p)Z\otimes Z$. Contracting this tensor network then gives the quantity $16p^2$ directly, and the desired Jamiolkowski fidelity p^2, coinciding with Example 3.*

Compared to Alg. I, the benefit of this new one is obvious: we have only to contract one single tensor network instead of exponentially many. When there are a large number of noisy gates, which is always the case in actual quantum devices since every gate suffers

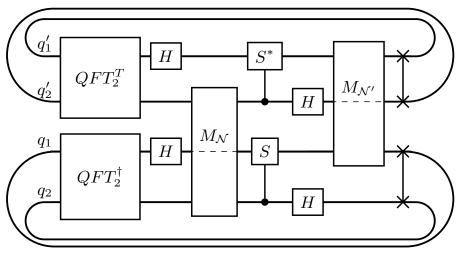


Fig. 5. The tensor network to be contracted in Example 4 using Alg. II.

some degree of noise, this approach will be definitely more efficient. However, the cost we pay here is that the number of qubits doubles, and the circuit to be contracted is more complicated. This will offset the benefit brought by the single tensor network when the number of noisy gates is quite small. This time-space trade off will be clearly demonstrated with experimental results shown in Sec. V.

C. Data Structure and Optimisation

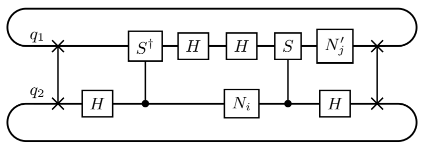
Our algorithms for computing $F_J(\mathcal{E}, U)$ presented above are both based on tensor network contraction. To implement these algorithms, we have to find an efficient data structure to help represent and manipulate tensor networks. To this end, we employ the recently proposed Tensor Decision Diagram (TDD) [17]. This software package represents a tensor as a decision diagram which optimises the memory consumption. Note that the final TDD of the contracted tensor network in our algorithms has only one node and the weight of its incoming edge is exactly the trace needed. Our algorithms leave more room for optimisation techniques in contracting tensor networks. We briefly discuss four of them in the following.

Tree decomposition. We have employed the tree decomposition algorithm proposed in [27] to determine the contraction order for tensor networks. A tree decomposition is a mapping of a graph into a tree, while in our circumstances, the graph is the corresponding line graph of the tensor network and each tree node consists of indices of the tensor network. An optimal contraction order can be obtained in a standard way from a tree decomposition [27].

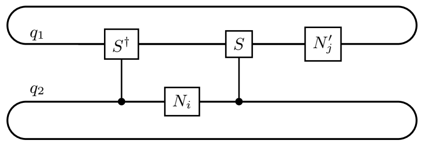
Computed table. In the TDD package [17], a computed table is used to store all nodes of the decision diagrams generated in each trace calculation process. Recall that Alg. I computes traces for many miter-like tensor networks with the same structure. It is therefore natural to keep only one computed table in the whole process and maximally reuse the computed results. This idea was first introduced in [24] in the simulation of noisy quantum circuits, but implemented with different (more involved) techniques.

Local optimisations. Note that a simple optimisation technique for tensor network contraction is to eliminate adjacent gates whose product is the identity operator, for example, a pair of mutually inverse gates [10, 11]. This is particularly useful in approximate equivalence checking because, in many cases, the noisy circuit shares most of the unitary gates with the ideal one except for a few noisy gates. Furthermore, as the corresponding input and output qubits are connected to compute the trace, this technique can be used on both ends of the composed circuit as well.

SWAP elimination. Note that in quantum Fourier transform, a series of SWAP gates are used to reverse the order of output qubits. For this type of circuits, we may simply omit these SWAP gates but instead connect the input qubits to the corresponding outputs determined by the SWAP gates. Obviously this does not change the computed trace value.



(a)



(b)

Fig. 6. (a) The tensor network that needs to be contracted in approximate checking the noisy circuit in Fig. 2 against the ideal one in Fig. 1, and (b) the simplified one using the optimisation techniques in Sec. IV-C.

Example 5. *Back to the noisy circuit shown in Fig. 2, but this time we assume that the ideal circuit is given explicitly in Fig. 1. Then according to Alg. I, we have to contract tensor networks with the form shown in Fig. 6 (a). Using the optimisation techniques presented above, the two SWAP gate can be eliminated by reconnecting the inputs and outputs and the four H gates can be cancelled locally. Consequently, we only have to contract the much simplified tensor network shown in Fig. 6 (b).*

V. Empirical Evaluations

In this section, we evaluate the effectiveness of our algorithms by comparing with the Qiskit $process\_fidelity$ method. We also illustrate how our algorithms perform as the number of noises increases. Finally, the utility of the computed table in saving computations for Alg. I is demonstrated. We implement the algorithms using Python3 and conduct experiment on a laptop with Intel i7-1065G7 CPU and 8GB RAM. The optimisation technologies of tree decomposition and computed table (cf. Sec. IV-C) are incorporated. However, in order to make a fair comparison with Qiskit's corresponding method, local optimisations and SWAP elimination are excluded.

A. Scalability With the Number of Qubits

Baseline We compare the time and memory consumption of our algorithms with the corresponding Qiskit implementation. To this end, we first transform the ideal circuit and the noisy one into Qiskit classes $Operator$ and $SuperOp$, respectively. Then the method $process\_fidelity$ is called to calculate the fidelity.

Benchmarks We experiment on some well-known algorithms, such as Bernstein-Vazirani algorithm (bv) [28], Quantum Fourier Transform (qft) [25], Grover algorithm [5], as well as benchmarks appeared in real quantum tasks, such as Quantum Volume (qv) [29], Modular Multiplication (7x1mod15) [18], and Randomised Benchmarking (rb) [30]. All of our benchmarks are from [24].

For noisy implementations, we randomly insert some depolarisation noises (see Example 2) to model the realistic errors that occur in real NISQ devices. The numbers of noises range from 1 to 14 and the probability parameter of the noisy gate is set to be 0.001 (i.e., $p = 0.999$ in Example 2), representing the state-of-the-art design technology [31]. It is worth noting that the selection of p does not affect the performance of the algorithms.

Results Table I summarises the experimental results of the Qiskit $process\_fidelity$ method and our algorithms. In the experiment, the time-out and memory bound are set as 3600 seconds and 8GB,

TABLE I
EXPERIMENT RESULTS

Circuit	n	$\|G\|$	k	Qiskit	Alg.II		Alg. I	
				time (s)	time (s)	nodes	time (s)	nodes
rb	2	7	6	0.03	0.10	16	24.34	6
qft2	2	7	2	0.05	0.05	12	0.14	8
grover	3	96	4	0.15	3.56	283	23.58	38
qft3	3	18	7	0.16	0.53	116	230.82	13
qv_n3d5	3	50	2	0.21	0.91	32	1.55	16
bv4	4	11	7	0.24	0.17	30	210.48	8
7x1mod15	5	14	3	4.29	0.18	14	1.39	10
bv5	5	14	6	5.13	0.26	22	64.01	8
qft5	5	55	3	31.90	2.44	270	4.07	62
qv_n5d5	5	100	3	20.12	10.82	272	33.82	256
bv6	6	17	14	258.15	0.48	30	TO	TO
qv_n6d5	6	150	1	1158.10	18.54	256	8.94	256
qft7	7	112	6	MO	33.72	1092	391.74	345
qv_n7d5	7	150	2	MO	72.17	928	151.04	1024
bv9	9	26	6	MO	0.52	46	117.18	8
qv_n9d5	9	200	3	MO	82.98	962	361.29	1673
qft9	9	189	2	MO	131.41	2820	27.32	1216
qft10	10	235	2	MO	595.30	15753	194.08	3012
bv13	13	38	4	MO	0.81	16	11.48	8
bv14	14	41	4	MO	0.86	14	14.31	8
bv16	16	47	9	MO	1.61	22	TO	TO

* n, $|G|$, k are, respectively, the numbers of qubits, gates, noises. The 'nodes' columns record the maximum numbers of the nodes of the TDDs constructed in the calculation process.
* 'TO' stands for time out, 'MO' stands for out of memory.

respectively. Note that in Qiskit class $SuperOp$, a super-operator in an n-qubit system is stored as a $2^{2n} \times 2^{2n}$ complex matrix, which is extremely space-consuming. For example, at least 64GB memory space is needed to describe an 8-qubit super-operator using the data type $complex128$. Consequently, the baseline algorithm can only process circuits with at most 7 qubits on our computer with 8GB RAM. Even for the two 7-qubit circuits $qft7$ and qv_n7d5, $process\_fidelity$ fails due to memory overflow.

From Table I we can see that, when the number of qubits is small (≤ 5), the time consumptions of Qiskit and Alg. II are very close, but when the number of qubits becomes 6, Alg. II runs faster than Qiskit in several orders of magnitude. The table further shows that our algorithms scale well when the number of qubits increases from 6 to 16. This is partially due to that we adopt the decision diagram representation, where the minimum number of complex numbers needed to store for a TDD with m nodes is only $2m - 1$.

From the table we also observe that, when the number of noises is small, Alg. I could be more efficient than Alg. II, but when the number of noises becomes greater this will be changed. The following subsection illustrates this in more details.

B. Scalability With the Number of Noises

In this subsection we examine the performance of Algs. I and II when the number of noises increases.

Benchmarks We select Bernstein-Vazirani algorithm and Quantum Fourier Transform as our benchmark circuits of this part. The numbers of qubits in these circuits range from 3 to 5.

Results The results are summarised in Fig. 7, where the polyline 'bv3' represents the logarithm of the time consumption ratio $\log(t_1/t_2)$ of the two algorithms on the 3-qubit Bernstein-Vazirani algorithm, and the meaning of other polylines is similar. From the figure, we can see that when there is only one noise, for most of the circuits, we have $\log(t_1/t_2) < 0$, which means $t_2 > t_1$, i.e., the time consumption of Alg. II is bigger than that of Alg. I. But when the number of noises increases, $\log(t_1/t_2)$ increases linearly, meaning that the running time of Alg. I increases exponentially compared to that of Alg. II. This suggests that Alg. II is more suitable when the circuit contains many noisy gates.

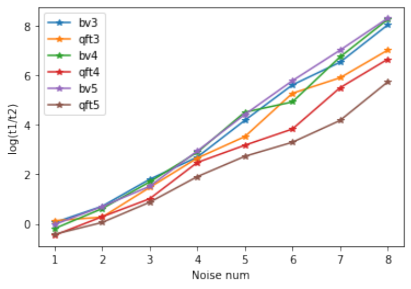


Fig. 7. The comparison of the two algorithms as the number of noisy gates increases. The horizontal axis is the number of noises and the vertical axis is $\log(t_1/t_2)$, where t_1 and t_2 are the time for Alg. I and Alg. II respectively.

TABLE II
UTILITY OF THE COMPUTED TABLE

Noise	bv3			bv4			bv5		
num	Opt.	Ori.	Rate	Opt.	Ori.	rate	Opt.	Ori.	Rate
1	0.05	0.10	0.50	0.11	0.13	0.82	0.10	0.17	0.56
2	0.17	0.44	0.38	0.35	0.53	0.66	0.30	0.67	0.45
3	0.63	1.84	0.34	1.06	2.20	0.48	1.00	2.63	0.38
4	2.26	7.52	0.30	3.87	8.76	0.44	3.89	10.83	0.36
5	8.55	34.03	0.25	15.91	34.09	0.47	15.13	44.07	0.34
6	37.24	133.59	0.28	57.87	142.44	0.41	62.77	182.12	0.34
7	142.92	535.77	0.27	225.86	584.75	0.39	254.34	735.31	0.35
8	593.56	2119.26	0.28	913.23	2445.28	0.37	1090.03	3051.05	0.36
SUM	785.39	2832.56	0.28	1218.27	3218.18	0.38	1427.56	4026.87	0.35

* Running time (in seconds) of Alg. I w/ and w/o computed tables.
* The rate is calculated by Opt./Ori.

C. The Utility of the Computed Table

To test the optimisation technique of introducing computed tables (see Sec. IV-C), we check the noisy implementation of Bernstein-Vazirani algorithm using Alg. I, with the numbers of qubits and noises ranging from 3 to 5 and from 1 to 8, respectively. Table II shows the experimental results where the columns 'Ori.' and 'Opt.' record, respectively, the running time of Alg. I without and with the shared computed table. It can be observed that an average of 72%, 62%, and 65% of time consumption can be saved for these circuits by reusing the computed table, which is comparable to the result of [24].

VI. Conclusion and Future works

Although the equivalence checking of quantum (noiseless) circuits has been studied for more than ten years, little attention was paid to noisy circuits. In this paper, we defined the approximate equivalence of noisy quantum circuits and proposed two algorithms to check it. Our algorithms are based on calculating the Jamiolkowski fidelity, which is reduced to calculating the traces of miter-like tensor networks. We implemented our algorithms by using the newly proposed data structure TDD. When compared with the current Qiskit method for calculating the Jamiolkowski fidelity, experiments on various real benchmark circuits show that our algorithms outperform in both efficiency and scalability, especially when the circuits have five or more qubits. When comparing our algorithms, Alg. II is more efficient when many errors occur in the noisy circuit and Alg. I works better when errors are rare. In addition, for the purpose of approximate equivalence checking, as we often only need to calculate a small part of the involved trace terms, Alg. I may be more attractive.

Future work will incorporate more optimisation techniques like local optimisations and SWAP elimination (cf. Sec. IV-C) in the implementation of our algorithms. We will also consider how to select a small subset of trace terms to efficiently approximate the fidelity computation in Alg. I.

References

[1] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[2] P. Molitor and J. Mohnke, *Equivalence Checking of Digital Circuits: Fundamentals, Principles, Methods*. Springer, 2007.

[3] A. Mishchenko *et al.*, "Improvements to combinational equivalence checking," in *ICCAD*. ACM, 2006, pp. 836–843.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *STOC*. ACM, 1996, pp. 212–219.

[6] J. Biamonte *et al.*, "Quantum machine learning," *Nat.*, vol. 549, no. 7671, pp. 195–202, 2017.

[7] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nat.*, vol. 574, no. 7779, pp. 505–510, 2019.

[8] Y. Cao *et al.*, "Quantum chemistry in the age of quantum computing," *Chem. Rev.*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[9] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Checking equivalence of quantum circuits and states," in *ICCAD*. IEEE, 2007, pp. 69–74.

[10] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," in *NANOARCH*. IEEE, 2010, pp. 23–28.

[11] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *arXiv preprint arXiv:2004.08420*, 2020.

[12] R. Wille *et al.*, "Equivalence checking of reversible circuits," in *ISMVL*. IEEE Computer Society, 2009, pp. 324–330.

[13] L. Burgholzer and R. Wille, "The power of simulation for equivalence checking in quantum computing," in *DAC*. IEEE, 2020, pp. 1–6.

[14] R. Wille, N. Przigoda, and R. Drechsler, "A compact and efficient sat encoding for quantum circuits," in *2013 Africon*. IEEE, 2013, pp. 1–6.

[15] M. Raginsky, "A fidelity measure for quantum channels," *PHYS REV A*, vol. 290, no. 1-2, pp. 11–18, 2001.

[16] A. Gilchrist, N. K. Langford, and M. A. Nielsen, "Distance measures to compare real and ideal quantum processes," *PHYS REV A*, vol. 71, no. 6, p. 062310, 2005.

[17] X. Hong *et al.*, "A tensor network based decision diagram for representation of quantum circuits," *arXiv preprint arXiv:2009.02618*, 2020.

[18] G. Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

[19] P. Niemann *et al.*, "Qmdds: Efficient quantum function representation and manipulation," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 35, no. 1, pp. 86–99, 2015.

[20] V. P. Belavkin, G. M. D'ariano, and M. Raginsky, "Operational distance and fidelity for quantum channels," *J Math Phys*, vol. 46, no. 6, p. 062106, 2005.

[21] J. Biamonte, "Lectures on quantum tensor networks," *arXiv preprint arXiv:1912.10049*, 2019.

[22] X. Gao and L. Duan, "Efficient classical simulation of noisy quantum computation," *arXiv preprint arXiv:1810.03176*, 2018.

[23] T. Grurl, J. Fuß, and R. Wille, "Considering decoherence errors in the simulation of quantum circuits using decision diagrams," in *ICCAD*. IEEE, 2020.

[24] G. Li, Y. Ding, and Y. Xie, "Eliminating redundant computation in noisy quantum computing simulation," in *DAC*. IEEE, 2020, pp. 1–6.

[25] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[26] R. R. Ernst *et al.*, *Principles of Nuclear Magnetic Resonance in One and Two Dimensions*. Clarendon Press Oxford, 1987, vol. 14.

[27] I. L. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM J. Comput.*, vol. 38, no. 3, pp. 963–981, 2008.

[28] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1411–1473, 1997.

[29] N. Moll *et al.*, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Sci. Technol.*, vol. 3, no. 3, p. 030503, 2018.

[30] E. Knill *et al.*, "Randomized benchmarking of quantum gates," *PHYS REV A*, vol. 77, no. 1, p. 012307, 2008.

[31] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers," in *ASPLOS*. ACM, 2019, pp. 987–999.