

# Quantum Circuit Synthesis Based on LimTDD

Xin Hong<sup>1</sup>, Chenjian Li<sup>1</sup>, Aochu Dai<sup>2</sup>, Runhong He<sup>1</sup> and Shenggang Ying<sup>1\*</sup>

<sup>1</sup>Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software, Chinese Academy of Sciences, Beijing, China

<sup>2</sup>Department of Computer Science and Technology, Tsinghua University, Beijing, China

Emails: {hongxin, licj, runhong93, yingsg}@ios.ac.cn, dac22@mails.tsinghua.edu.cn

**Abstract**—Quantum circuit synthesis is a crucial task in quantum computing, aiming to transform a given high-level logic quantum operation into a sequence of elementary quantum gates. Traditional synthesis methods often rely on special characteristics of quantum operations or complex mathematical operations. While effective, they tend to incur high computational costs because they do not fully exploit the underlying structure of the quantum operation. In this work, we introduce a novel synthesis approach leveraging the LimTDD (Local Invertible Map Tensor Decision Diagram) data structure, known for its high compression efficiency and ability to identify isomorphic structures within tensors.

By utilising LimTDD, our algorithm achieves efficient synthesis for specific types of quantum circuits, significantly reducing computational overhead. Moreover, the ability to extract isomorphic operators allows for reducing the entanglements, making our method particularly effective in accelerating other synthesis algorithms. We demonstrate the efficacy of our approach through experiments, showing substantial improvements in gate count and synthesis time compared to existing methods. Our work not only provides a powerful tool for quantum circuit synthesis but also highlights the potential of LimTDD in advancing the field of quantum computing.

## I. INTRODUCTION

Quantum computing has emerged as a promising technology with the potential to solve complex problems more efficiently than classical computing. At the heart of quantum computing lies the quantum circuit, which is a sequence of quantum gates that manipulate qubits to perform computations. The synthesis of quantum circuits, i.e., the process of transforming a given quantum operation into a sequence of implementable elementary quantum gates, is a fundamental task in quantum computing. Effective and efficient synthesis methods are essential for realising practical quantum algorithms and reducing the resource requirements for quantum computations.

The common methods for quantum circuit synthesis typically rely on certain mathematical tools, such as Quantum Shannon Decomposition (QSD) [1], Cosine-Sine Decomposition (CSD) [2], and QR Decomposition (QRD) [3]. QSD employs quantum multiplexors to decompose unitary operators, CSD provides a systematic approach to decomposing multi-qubit operations into sequences of elementary gates, and QRD decomposes quantum operations column-by-column. These methods are applicable to any unitary operator and have clearly

defined execution procedures, enabling the decomposition of small-scale unitary operators to be carried out efficiently.

There are also several other popular approaches based on circuit structure search, including QFAST [4], QGo [5], and QSearch [6]. These methods start from a specific circuit structure to search for a circuit that closely approximates the target unitary operator. For small-scale cases (e.g., fewer than 3 qubits), they can carry out the circuit synthesis process more efficiently.

In the context of reversible logic, which shares similarities with quantum computing, Refs. [7], [8] demonstrated the potential for optimising reversible circuits, and can be adapted for quantum circuit synthesis. These methods focus on minimising the number of gates required to implement a given reversible operation, providing valuable insights for quantum circuit design. The realization of permutation groups with quantum circuits is studied in [9], and applied to superconducting quantum processors [10]. The synthesis of Clifford group operations has also been extensively studied, with methods like those in [11], [12] providing efficient solutions for this class of quantum operations. These techniques exploit the structure of Clifford group operations to achieve optimal or near-optimal gate counts, significantly improving the efficiency of quantum circuit synthesis for these operations. The decomposition of multi-control qubit gates and the optimal synthesis for two-qubit gates are studied in [13], [14], [15], etc.

Recent advancements also include the use of machine learning techniques for quantum circuit synthesis [16], as well as methods based on simulated annealing [17] and diffusion models [18]. These approaches leverage modern computational techniques to explore new possibilities for optimising quantum circuit synthesis. Other notable works include the application of ZX-calculus for graph-theoretic simplification of quantum circuits [19], and SAT-based methods for synthesising quantum circuits using the {CNOT, T} gate set [20]. These methods provide alternative approaches to quantum circuit synthesis, each with its own advantages and applications. The synthesis of quantum circuits for noisy intermediate-scale quantum devices, focusing on practical implementations, has also been studied [21].

Although significant progress has been made in the field of quantum circuit synthesis, existing methods can only handle relatively small scales or are restricted to specific types of quantum circuits, limiting their practical application. Moreover, these methods do not fully exploit the underlying structure of the quantum operation, leading to redundant gate counts

Work partially supported by Quantum Science and Technology-National Science and Technology Major Project under Grant No. 2024ZD0300502, Beijing Nova Program Grant No. 20240484652.

\* Corresponding author: Shenggang Ying (yingsg@ios.ac.cn).

and circuit depths. Even with the adoption of compression-oriented data structures such as Quantum Decision Diagrams (QDDs) [22] and Quantum Multiple-Valued Decision Diagrams (QMDDs) [23], this fundamental challenge remains largely unresolved.

In this paper, we present a new quantum-circuit synthesis framework that exploits decision diagrams enhanced with Local Invertible Maps. Vinkhuijzen et al. introduced the Local Invertible Map Decision Diagram (LIMDD)—a highly efficient decision diagram for quantum-circuit simulation—in [24], while Hong et al. recently extended this idea to tensor computation with the Local Invertible Map Tensor Decision Diagram (LimTDD) [25]. Since LimTDD uses a richer class of local transformations known as XP-operators and shows superior compression potential, we mainly focus on LimTDD in this paper. LimTDD offers the ability to identify isomorphic operators within matrices, which is the key feature for optimising quantum circuit synthesis. By leveraging these properties, our method can synthesise specific types of quantum circuits efficiently and significantly reduce the computational overhead compared to traditional methods. Moreover, the algorithm’s ability to extract isomorphic operators allows for further optimisation, making our approach particularly effective in accelerating other synthesis algorithms, such as those based on matrix decompositions.

The contributions of this paper are as follows:

- We propose a new quantum circuit synthesis algorithm using LimTDD, which efficiently handles specific types of quantum circuits.
- We demonstrate the efficacy of our approach through experiments, showing substantial improvements in synthesis time and gate count compared to existing methods.

## II. BACKGROUND

### A. Quantum Computing

1) *Quantum States*: The basic unit of quantum computing is the quantum bits, or qubits. Unlike classical bits that are restricted to *either* 0 or 1, qubits can exist in *both* of them, namely in superpositions of states. Formally, a qubit is described by a complex vector in a two-dimensional vector space with orthonormal basis states  $|0\rangle$  and  $|1\rangle$ , satisfying:

$$\langle 0|0\rangle = \langle 1|1\rangle = 1 \quad \text{and} \quad \langle 0|1\rangle = \langle 1|0\rangle = 0.$$

In general, the state of a single qubit state is a linear combination of them:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle,$$

where  $\alpha$  and  $\beta$  are complex numbers with  $|\alpha|^2 + |\beta|^2 = 1$ . Moreover, an  $n$ -qubit quantum state can be represented as a  $2^n$ -dimensional quantum state. This superposition capability allows qubits to represent multiple states simultaneously, providing a fundamental parallelism advantage over classical bits.

2) *Quantum Gates*: Quantum gates are unitary operators that map a quantum state to another state, thus acting as the elementary building block of quantum computation. They can be represented by unitary matrices mathematically and

are abstractedly represented as small blocks (nodes) in quantum circuits graphically. Some common quantum gates are Hadamard gate ( $H$  gate), Pauli- $Z$  gate ( $Z$  gate), Controlled- $X$  gate ( $CX$  gate).

3) *Quantum Circuits*: Quantum circuits are sequences of quantum gates that implement quantum algorithms. They are represented graphically, with qubits represented as lines and gates represented as symbols. The order in which gates are applied determines the overall transformation. Quantum circuits are designed to perform specific tasks, such as quantum teleportation, quantum error correction, and quantum algorithms like Shor’s algorithm for factoring and Grover’s algorithm for search.

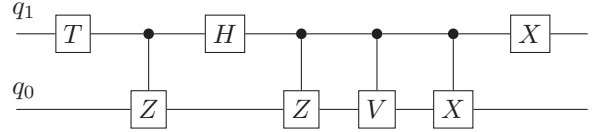


Fig. 1: An example of a quantum circuit consisting of 2 qubits and 5 gates.

Fig. 1 shows an example of a quantum circuit with 2 qubits and 7 gates. Here,  $V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ . The functionality of this circuit can be represented using the unitary matrix:

$$U = \begin{bmatrix} \frac{1}{2} & \frac{e^{-3\pi i/4}}{2} & -\frac{1}{2} & \frac{e^{-3\pi i/4}}{2} \\ \frac{1}{\sqrt{2}} & \frac{e^{\pi i/4}}{\sqrt{2}} & 0 & 0 \\ \frac{1}{2} & \frac{e^{-3\pi i/4}}{2} & \frac{1}{2} & \frac{e^{\pi i/4}}{2} \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{e^{-3\pi i/4}}{\sqrt{2}} \end{bmatrix}. \quad (1)$$

Quantum circuits are essential for implementing quantum algorithms on physical quantum devices. The design and optimisation of quantum circuits are critical for achieving efficient and practical quantum computations.

### B. Tensor Network

A tensor is a multidimensional array indexed by a finite set of indices, where the number of indices  $n$  is called the rank of the tensor. Scalars, vectors, and matrices are tensors of rank 0, 1, 2, respectively. In the context of quantum computing the indices usually take values in  $\{0, 1\}$ , so a tensor with index set  $S = \{x_n, \dots, x_1\}$  is essentially a map  $\phi: \{0, 1\}^S \rightarrow \mathbb{C}$ . We write  $\phi_{x_n \dots x_1}$  or  $\phi_{\vec{x}}$  for such tensor and  $\phi(\vec{a})$  for its value at assignment  $x_i = a_i$ . For instance, an  $n$ -qubit state vector is a rank- $n$  tensor, and the functionality of a quantum circuit, which transforms a quantum state to another one, can be represented as a rank- $2n$  tensor.

The fundamental operation on tensors is contraction. Given two tensors  $\phi_{\vec{x}, z}$  and  $\psi_{\vec{y}, z}$ , their contraction on variable  $z$  is the tensor

$$\xi_{\vec{x}, \vec{y}}(\vec{a}, \vec{b}) = \sum_c \psi(\vec{a}, c) \phi(\vec{b}, c).$$

Another useful operation is slicing: fixing one index to a certain value and focusing on the remaining tensor. For a

tensor  $\phi$  with label set  $S = \{x_n, \dots, x_1\}$ ,

$$\phi|_{x_1=c}(\vec{a}) = \phi(c, \vec{a}), \quad c \in \{0, 1\},$$

yields a rank- $n$  tensor over  $S' = \{x_n, \dots, x_2\}$ . When the sliced variable  $x$  is not in the label set,  $\phi|_{x=0} = \phi|_{x=1} = \phi$ .

A tensor network is an undirected graph  $G = (V, E)$  representing tensors and their contractions. In the network, tensors are represented as vertices and their indices are represented as edges attached to them. When edges are connected, it means the corresponding indices are contracted. Finally, contracting all adjacent tensors in any order produces a rank  $m$  tensor, where  $m$  is the number of open (unconnected) edges, and the resulted tensor is called the value of the tensor network. Quantum circuits are natural examples of tensor networks. For a more detailed introduction to tensor networks, you can refer to the tutorials [26], [27].

### C. Quantum Circuit Synthesis

1) *Formal Definition:* Quantum circuit synthesis is the process of transforming a given quantum operation, typically represented as a unitary matrix  $U$ , into a sequence of elementary quantum gates. The goal is to decompose the unitary matrix into a sequence of basic quantum gates that can be physically realized on a quantum device. It is an essential step in implementing quantum algorithms and ensuring their efficient execution on physical devices.

Formally, given a unitary matrix  $U \in \mathbb{C}^{2^n \times 2^n}$  representing an  $n$ -qubit quantum operation, the task of quantum circuit synthesis is to find a sequence of elementary quantum gates  $G_1, G_2, \dots, G_m$  such that:

$$U = G_m \cdot G_{m-1} \cdot \dots \cdot G_1,$$

where each  $G_i$  is an elementary quantum gate from a pre-defined gate set (e.g.,  $\{H, X, Z, CX, T\}$ ). The sequence of gates forms a quantum circuit that can be implemented on a quantum device.

## III. LIMTDD

LimTDD, or Local Invertible Map Tensor Decision Diagram, is an advanced decision diagram designed to efficiently represent and manipulate tensors and tensor networks. When representing the functionality of quantum circuits, it can extract the isomorphism relations between sub-matrices, thus enabling the exploitation of these structural symmetries to reduce the complexity of synthesis.

**Definition 1 (LIMs):** A local invertible map (LIM) of length  $n$  is an operator of the form

$$O = \lambda \cdot O_n \otimes \dots \otimes O_1, \quad \lambda \in \mathbb{C},$$

where each  $O_i$  is an invertible  $2 \times 2$  matrix. We abbreviate such operators as  $n$ -LIMs and write  $\mathcal{M}$  for the set of all LIMs.

Let  $\psi_{x_n \dots x_1}$  be a rank- $n$  tensor and  $O = \lambda \cdot O_n \otimes \dots \otimes O_1$  an  $n$ -LIM. The action  $O \cdot \psi_{x_n \dots x_1}$  denotes the tensor obtained by contracting each index  $x_i$  of  $\psi$  with the corresponding matrix  $O_i$  and scaling by  $\lambda$ . (After contraction, the open index is relabelled  $x_i$ .) Two rank- $n$  tensors  $\phi_{x_n \dots x_1}$  and  $\psi_{x_n \dots x_1}$  are

called *isomorphic*, denoted as  $\phi \cong \psi$ , if there exists an  $n$ -LIM  $O$  such that

$$\phi_{x_n \dots x_1} = O \cdot \psi_{x_n \dots x_1}.$$

**Definition 2 (LimTDD):** An  $n$ -LimTDD  $\mathcal{F}$  over a set of indices  $S$  is a rooted, weighted, and directed acyclic graph representing a rank- $n$  tensor. Formally, it is a 7-tuple  $\mathcal{F} = (V, E, \text{idx}, \text{rank}, \text{low}, \text{high}, \text{wt})$  defined as follows:

- $V$  is a finite set of nodes which consists of non-terminal nodes  $V_{NT}$  and a terminal node  $v_T$  labelled with integer 1.
- $\text{idx} : V_{NT} \rightarrow S$  assigns each non-terminal node an index in  $S$ .
- $\text{rank} : V \rightarrow \{0, \dots, n\}$  assigns the rank of the tensor represented by the node.
- Both  $\text{low}$  and  $\text{high}$  are mappings in  $V_{NT} \rightarrow V$ , which map each non-terminal node to its 0- and 1-successors, respectively;  $\text{rank}(\text{low}(v)) = \text{rank}(\text{high}(v)) = \text{rank}(v) - 1$ .
- $E = \{(v, \text{low}(v)), (v, \text{high}(v)) : v \in V_{NT}\}$  is the set of edges, where  $(v, \text{low}(v))$  and  $(v, \text{high}(v))$  are called the low- and high-edges of  $v$ , respectively. For simplicity, we also assume the root node  $r_{\mathcal{F}}$  has a unique incoming edge, denoted  $e_r$ , which has no source node;
- $\text{wt} : E \rightarrow \mathcal{M}$  assigns each edge  $e$  a weight of  $k$ -LIMs, where  $k$  is the rank of the node directed by  $e$ .  $\text{wt}(e_r)$  is called the weight of  $\mathcal{F}$ , and denoted  $w_{\mathcal{F}}$ .

The tensor represented by the terminal node is defined to be  $\Phi_{v_T} = 1$ , i.e., integer 1, which is a rank 0 tensor (scalar). The tensor represented by an edge  $e$ , directing to a node  $v$ , is defined to be

$$\Phi_e = \text{wt}(e) \cdot \Phi_v.$$

Let  $v$  be a non-terminal node and  $\text{idx}(v) = x$ , then the tensor represented by  $v$  is a tensor such that  $\Phi_v|_{x=0} = \Phi_{(v, \text{low}(v))}$ , and  $\Phi_v|_{x=1} = \Phi_{(v, \text{high}(v))}$ . The tensor  $\Phi_{\mathcal{F}}$  represented by the LimTDD  $\mathcal{F}$ , is defined to be the tensor represented by the incoming edge of the root node  $\Phi_{e_r}$ .

In this paper, we use the following graphical notation to represent an internal node of a LimTDD:

$$\textcircled{v_0} \xleftarrow{w_0} \textcircled{v} \xrightarrow{w_1} \textcircled{v_1}.$$

Each node within a LimTDD is uniquely characterised by three key elements: its index, the two nodes it points to (successors), and the weights assigned to the edges leading to these successors.

Note that, for a LimTDD, when normalisation has been applied, every non-terminal node will have the following form:

$$\textcircled{v_0} \xleftarrow{c \cdot I} \textcircled{v} \xrightarrow{\lambda \cdot O} \textcircled{v_1}.$$

Here,  $c = 0$  or  $1$  and if  $c = 0$ , then  $\lambda = 0$  and the whole tensor will represent constant 0.

When representing the functionality of a quantum circuit using a LimTDD, each qubit  $q_i$  is associated with two indices:  $x_i$  for the input and  $y_i$  for the output, and the  $2^n$ -dimensional

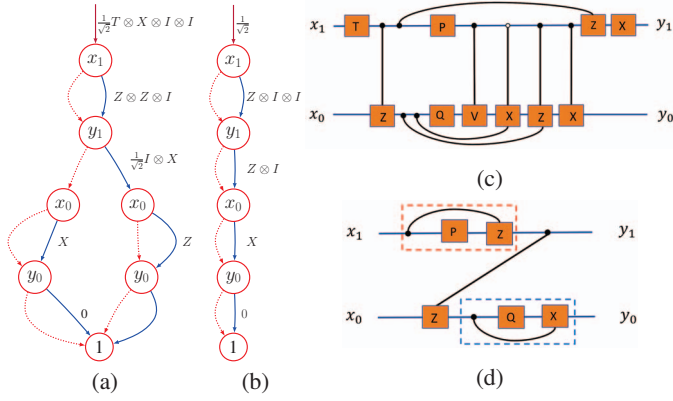


Fig. 2: LimTDDs and the equivalent tensor networks. Red dotted lines represent low edges and blue solid lines represent high edges. In this figure,  $P = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ ,  $Q = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ ,  $V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ , and other operators are standard quantum gates. The tensor in the red dashed line box equals a  $H$  gate, and the tensor in the blue dashed line box equals an identity matrix.

matrix is seen as a rank  $2n$  tensor. In this paper, we adhere to the following order:  $x_{n-1} \prec y_{n-1} \prec \dots \prec x_0 \prec y_0$ . This means that the index  $x_{n-1}$  appears at the top of the LimTDD, while the index  $y_0$  is at the bottom.

*Example 1:* The LimTDD encoding of the unitary matrix  $U$  (Eq. 1) corresponding to the quantum circuit in Fig. 1 is displayed in Fig. 2(a). As the circuit comprises two qubits, the diagram uses four indices ordered by  $x_1 \prec y_1 \prec x_0 \prec y_0$ . Matrix elements are extracted by traversing paths: for instance, the entry  $\frac{1}{2}$  in the top-left corner is obtained by following the red-blue-red-blue path and multiplying the complex weights encountered. Blue edges are selected for indices  $y_1$  and  $y_0$  because two  $X$  operators are applied to the corresponding nodes along this path.

#### IV. TRANSFORM LIMTDD TO A TENSOR NETWORK

The synthesis pipeline begins by converting the LimTDD into an equivalent tensor network. This LimTDD may be created by users according to the desired functions. Here, we place no restrictions on its specific origin; instead, we focus solely on how to process it. The conversion is done in an iterative manner, and at each iteration, we focus on the leftmost two reduced paths, thereby isolating the upper-left sub-matrix of the original unitary operator. Here, reduced paths are defined as paths that differ in at least one internal node. For example, the LimTDD in Fig. 2(a) contains exactly two reduced paths, while Fig. 2(b) contains only one. The isolated sub-matrix is simplified by mapping the two reduced paths to a small tensor network, then computing and applying the inverse operator that collapses them into a single effective edge. The size of the sub-matrix depends on the position of the branched index. Once these two paths have been merged, the procedure advances to the new leftmost pairs and repeats.

The following paragraphs describe, step by step, how the leftmost two reduced paths of a LimTDD are translated into a

tensor network; the result of the transformation for Fig. 2(a) is shown in Fig. 2(c).

##### A. Incoming edge operators

Recall that for an edge  $e$  pointing to node  $v$ , its associated tensor is

$$\Phi_e = \text{wt}(e) \cdot \Phi_v.$$

Suppose  $\text{rank}(v) = n$  and  $\text{wt}(e) = \lambda \cdot O_n \otimes \dots \otimes O_1$ , then the action  $\text{wt}(e) \cdot \Phi_v$  contracts  $O_i$  onto the  $i$ -th index of  $\Phi_v$  and multiplies by  $\lambda$  under the hood. Hence, for a LimTDD  $\mathcal{F}$  with root  $r_{\mathcal{F}}$ , the overall tensor  $\Phi_{\mathcal{F}}$  (complex coefficient omitted) is obtained by contracting all operators in  $\text{wt}(\mathcal{F})$  into  $\Phi_{r_{\mathcal{F}}}$ .

*Example 2:* In Fig. 2(a), the incoming edge of the root carries weight

$$\frac{1}{\sqrt{2}} T \otimes X \otimes I \otimes I.$$

Neglecting the prefactor  $\frac{1}{\sqrt{2}}$ , we first derive  $\Phi_{r_{\mathcal{F}}}$  and then apply  $T$  to index  $x_1$  and  $X$  to index  $y_1$ . Consequently,  $T$  and  $X$  appear as the outermost operators on indices  $x_1$  and  $y_1$  in Fig. 2(c).

##### B. High-edge operators

Next, we handle the operators carried by the high-edges. Consider a node  $v$  with index  $\text{id}_x(v) = x$  and the local pattern

$$\textcircled{v_0} \xleftarrow{c \cdot I} \textcircled{v} \xrightarrow{\lambda \cdot O} \textcircled{v_1}.$$

Evaluating the two branches gives

$$\Phi_v|_{x=0} = c \cdot \Phi_{v_0}, \quad \Phi_v|_{x=1} = \lambda \cdot O \cdot \Phi_{v_1}.$$

To construct  $\Phi_v$ , we first construct a tensor  $\psi$  satisfying

$$\psi|_{x=0} = c \cdot \Phi_{v_0}, \quad \psi|_{x=1} = \lambda \cdot \Phi_{v_1}$$

and subsequently contract the operators inside  $O$  only on the  $x = 1$  slice. In the resulting tensor (or quantum-circuit), these operators appear as controlled- $O_i$  gates conditioned on index  $x$ .

*Example 3:* Consider the operator  $Z \otimes Z \otimes I$  on the high edge of the root node of the LimTDD given in Fig. 2 (a). Two controlled- $Z$  gates are added to index  $x_1, y_1$  and  $x_1, x_0$  in the corresponding tensor. Similarly, consider the operator  $I \otimes X$  on the high-edge of the  $y_1$  node; we add a controlled- $X$  gate on indices  $y_1$  and  $y_0$ .

For the root node of Fig. 2(a) the high-edge operator is  $Z \otimes Z \otimes I$ . We thus insert two controlled- $Z$  gates: one on indices  $(x_1, y_1)$  and one on  $(x_1, x_0)$ . Similarly, the operator  $I \otimes X$  on the  $y_1$ -node high-edge becomes a controlled- $X$  gate on  $(y_1, y_0)$ .

When a high-edge operator lies on a branch, it is applied only if the corresponding branch condition is satisfied; hence we use multi-controlled gates.

*Example 4:* In Fig. 2(a) the two  $x_0$  nodes carry high-edge operators  $X$  and  $Z$ . We add a controlled-controlled- $X$  on  $(y_1, x_0, y_0)$  with the  $y_1$ -control set to 0, and a controlled-controlled- $Z$  on  $(y_1, x_0, y_0)$  with the  $y_1$ -control set to 1.

### C. Complex weights in different branches

After the two preceding steps, every edge operator of the LimTDD has been translated into (controlled) operators placed on the outermost layer of the tensor network. We now turn to the residual tensor in the inner layer.

First, we absorb the complex amplitudes that appear on different branches. Observe that the unitary matrix

$$M = \frac{1}{\sqrt{(1+|\lambda_1|^2)(1+|\lambda_2|^2)}} \begin{bmatrix} 1 & -\lambda_2^* \\ \lambda_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & \lambda_1^* \\ -\lambda_1 & 1 \end{bmatrix}$$

maps the vector  $[1, \lambda_1]^T$  to  $[1, \lambda_2]^T$ . Hence, whenever two sibling nodes share the same index but carry distinct outgoing weights  $[1, \lambda_1]^T$  and  $[1, \lambda_2]^T$ , we:

- 1) build the tensor for the first branch,
- 2) apply a controlled- $M$  gate conditioned on the splitting index.

*Example 5:* In Fig. 2(a) the two  $y_0$ -nodes emit weights  $[1, 0]^T$  and  $[1, 1]^T$ , respectively. We therefore insert a controlled- $V$  gate on qubits  $(y_1, y_0)$ , where

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

transforms  $[1, 0]^T$  into  $[1, 1]^T$ . The gate is controlled by  $y_1$  and targets  $y_0$ , reflecting the branch structure of the LimTDD.

### D. Kronecker Product and Tower forms

Finally, only the tensor corresponding to the leftmost path (stripped of all edge operators) remains to be converted into a tensor network. Fortunately, this tensor is simple enough to be written as the Kronecker product of a few small matrices.

*Example 6:* Consider again the LimTDD of Fig. 2(a). Ignoring the high-edge  $X$  operator, the leftmost  $x_0$  node encodes the rank-2 tensor

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

Similarly, after discarding all operators, the outgoing weights of the  $x_1$  and  $y_1$  nodes yield

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

(The factor  $\frac{1}{\sqrt{2}}$  that appeared on the  $y_1$  high-edge has already been absorbed into the controlled- $V$  gate added earlier.)

Now that the LimTDD can be transformed into a tensor network, we proceed to present our synthesis algorithm.

## V. LIMTDD BASED SYNTHESIS

In this section, we present our approach for quantum circuit synthesis leveraging the LimTDD data structure. The main procedure of our algorithm is as follows:

- 1) Convert the LimTDD to a tensor network;
- 2) Recognise elementary quantum gates, apply their inverses, and thereby simplify the LimTDD;
- 3) Update the LimTDD and repeat the previous stages until either the identity is reached or a preset iteration limit is exceeded.

Stage 2) gradually injects simple quantum gates from *both* sides of the unitary  $U$  (left via  $y$ -indices, right via  $x$ -indices) so that the LimTDD evolves from  $U$  towards the identity. If the procedure discovers gate sequences

$$G_{\ell m} \cdots G_{\ell 1} U G_{r 1} \cdots G_{r m'} = I,$$

then  $U$  is immediately decomposed as

$$U = G_{\ell 1}^\dagger \cdots G_{\ell m}^\dagger G_{r m'}^\dagger \cdots G_{r 1}^\dagger.$$

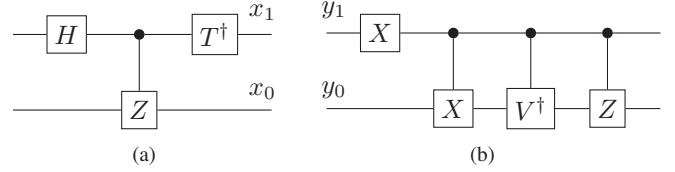


Fig. 3: The circuits that needed to be added from  $x$ -indices side and  $y$ -indices side to transform the unitary represented by the LimTDD given in Fig. 2 (a) to the identity matrix.

*Example 7:* We again use the LimTDD of Fig. 2(a) as a running example. After converting it to a tensor network (Fig. 2(c)), the normal quantum gates contains:

- a  $T$  gate on  $x_1$ ,
- a  $CZ$  gate on  $(x_1, x_0)$ ,
- an  $X$  gate on  $y_1$ ,
- a  $CX$  and a controlled- $V$  gate on  $(y_1, y_0)$ .

We therefore apply the inverses of these gates from the  $x$ -side and the  $y$ -side, accordingly, and update the LimTDD. The updated LimTDD is shown in Fig. 2(b), which can be transformed to a tensor network as shown in Fig. 2(d).

Since  $Z$  commutes with the control qubits, we move the  $Z$  operator to the inner layer. The tensors inside the red dashed box contract to an  $H$  gate, while those inside the blue dashed box contract to the identity. Hence, an  $H$  gate is applied on the  $x$ -side. Similarly, as the blue-box is equal to identity, the  $CZ$  gate on  $(y_1, x_0)$  is reduced to a  $CZ$  on  $(y_1, y_0)$ ; we thus apply a  $CZ$  gate on the  $y$ -side.

All these gates are collected to form two simple circuits (Fig. 3), which map the original LimTDD to the identity matrix. Reversing and concatenating them yields the final quantum circuit shown in Fig. 1, which equals exactly the unitary represented by the initial LimTDD.

## VI. EXPERIMENTS

In this section, we present the results of our experiments, which aim to evaluate the success rate of our proposed method and compare its performance with that of the QSD-based method implemented in Qiskit.

### A. Successful Rate

It is important to note that our method is not complete. Specifically, we do not account for the commutation relations of the operators, which may lead to the introduction of more complex operators during the elimination process.

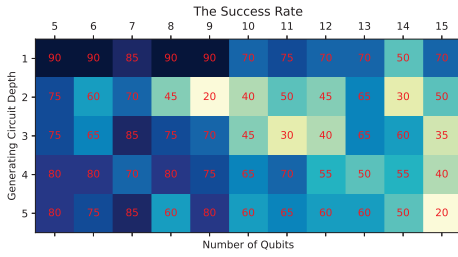


Fig. 4: The success rate (%) of our method.



Fig. 5: The reducible rate (%) by our method.

Consequently, our method can only synthesize a subset of tensors represented by LimTDD.

To assess the success rate of our method, we tested our method on random Clifford + T circuits. The circuits were generated with each gate randomly selected from the set  $\{X, Y, Z, H, S, CX, T\}$  with equal probability. The circuits were composed of  $n = \{5, \dots, 15\}$  qubits, and the number of gates was set to  $l \cdot n$  for  $l \in \{1, \dots, 5\}$ .

For each circuit configuration, we generated the corresponding LimTDD (or unitary matrix for the other method) and attempted to synthesize the functionality using our method. For each configuration, 20 trials are conducted, and the number of successful cases is recorded. The success rate is shown in Fig. 4. Our experiments indicate that the success rate is relatively high when the number of qubits and gates is small, as the LimTDD representation tends to be more compact in these cases. As the number of qubits and gates increases, the success rate decreases.

We also observed that, even when our method fails to complete the synthesis, it can still un-entangle some qubits from the other ones, thereby, it can be used to simplify the task for other methods. When we included these cases, the reducible rate, shown in Fig. 5, was found to be higher than the success rate. For instance, for functionality generated with 7 qubits and 35 gates, the success rate was 85%, while the reducible rate reached 100%.

### B. Compared with Qiskit

In this subsection, we compare the performance of our proposed method with the `UnitarySynthesis` interface provided in Qiskit, which is based on the Quantum Shannon Decomposition (QSD)[1] under the hood.

We evaluate and compare the time required to complete the synthesis tasks and the number of gates generated by each

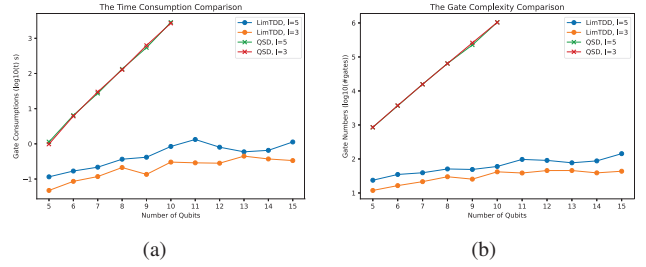


Fig. 6: The time consumption comparison and gate complexity comparison.

method. Since Qiskit decomposes gates into single-qubit gates and  $CX$  gates, we also decompose the gates generated by our method into this gate set using the transpiler provided in Qiskit. The comparison results are shown in Fig. 6 (a) and Fig. 6 (b). We demonstrate the results for qubit numbers  $n \in \{5, \dots, 15\}$ , with generating circuit gates set to  $3n$  and  $5n$ . The data is transformed using  $\log_{10}(\cdot)$  for better visualization. The timeout for each synthesis task was set to 3600 seconds. From the figures, it is evident that our method offers an exponential advantage in both time and gate complexity compared to Qiskit. For instance, when the number of qubits is 10, and the generating circuit consists of 50 gates, our method requires only 0.85 seconds and the synthesised circuits have an average of 61 gates, whereas Qiskit requires 2850 seconds and 1,043,970 gates, respectively.

## VII. CONCLUSION

In this work, we have presented a novel approach to quantum circuit synthesis using the LimTDD data structure. Our method leverages the high compression efficiency and the ability of LimTDD to identify isomorphic operators within functionalities, enabling efficient synthesis of specific types of quantum circuits. This approach not only reduces computational overhead compared to traditional methods but also provides a foundation for accelerating other synthesis algorithms. Through experiments, we demonstrated the effectiveness of our method in both synthesis time and gate count compared to existing techniques.

Future work will focus on extending this approach to more general quantum operations, improving the success rate and exploring additional optimisations enabled by the unique properties of LimTDD.

## REFERENCES

- [1] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum logic circuits," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, 2005, pp. 272–275.
- [2] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Quantum circuits for general multiqubit gates," *Physical review letters*, vol. 93, no. 13, p. 130502, 2004.
- [3] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, "Efficient decomposition of quantum gates," *Physical review letters*, vol. 92, no. 17, p. 177902, 2004.
- [4] E. Younis, K. Sen, K. Yelick, and C. Iancu, "Qfast: Conflating search and numerical optimization for scalable quantum circuit synthesis," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2021, pp. 232–243.
- [5] X.-C. Wu, M. G. Davis, F. T. Chong, and C. Iancu, "Qgo: Scalable quantum circuit optimization using automated synthesis," *arXiv preprint arXiv:2012.09835*, 2020.
- [6] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, and C. Iancu, "Towards optimal topology aware quantum circuit synthesis," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2020, pp. 223–234.
- [7] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.
- [8] J. Jiang, X. Sun, S.-H. Teng, B. Wu, K. Wu, and J. Zhang, "Optimal space-depth trade-off of cnot circuits in quantum logic synthesis," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 213–229.
- [9] J. Liu, Y. Ren, Y. Cao, H. Sun, and L. Chen, "Realization of permutation groups by quantum circuit," *arXiv preprint arXiv:2406.01350*, 2024.
- [10] M. Soeken, F. Mozafari, B. Schmitt, and G. De Micheli, "Compiling permutations for superconducting qpus," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1349–1354.
- [11] P. Niemann, R. Wille, and R. Drechsler, "Efficient synthesis of quantum circuits implementing clifford group operations," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2014, pp. 483–488.
- [12] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 70, no. 5, p. 052328, 2004.
- [13] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 69, no. 3, p. 032315, 2004.
- [14] A. N. Glaudell, N. J. Ross, and J. M. Taylor, "Optimal two-qubit circuits for universal fault-tolerant quantum computation," *npj Quantum Information*, vol. 7, no. 1, p. 103, 2021.
- [15] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, N. Margolus, P. W. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, 1995.
- [16] Y. Ge, W. Wenjie, C. Yuheng, P. Kaisen, L. Xudong, Z. Zixiang, W. Yuhan, W. Ruo Cheng, and Y. Junchi, "Quantum circuit synthesis and compilation optimization: Overview and prospects," *arXiv preprint arXiv:2407.00736*, 2024.
- [17] A. Paradis, J. Dekoninck, B. Bichsel, and M. Vechev, "Synthetiq: Fast and versatile quantum circuit synthesis," *Proceedings of the ACM on Programming Languages*, vol. 8, no. OOPSLA1, pp. 55–82, 2024.
- [18] F. Furrutter, G. Muñoz-Gil, and H. J. Briegel, "Quantum circuit synthesis with diffusion models," *Nature Machine Intelligence*, pp. 1–10, 2024.
- [19] R. Duncan, A. Kissinger, S. Perdrix, and J. Van De Wetering, "Graph-theoretic simplification of quantum circuits with the zx-calculus," *Quantum*, vol. 4, p. 279, 2020.
- [20] G. Meuli, M. Soeken, and G. De Micheli, "Sat-based {CNOT, T} quantum circuit synthesis," in *Reversible Computation: 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings 10*. Springer, 2018, pp. 175–188.
- [21] S. Yang, G. Tian, J. Zhang, and X. Sun, "Quantum circuit synthesis on noisy intermediate-scale quantum devices," *Physical Review A*, vol. 109, no. 1, p. 012602, 2024.
- [22] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Proceedings of the Design Automation & Test in Europe Conference*, vol. 1. IEEE, 2006, pp. 1–6.
- [23] P. Niemann, R. Wille, and R. Drechsler, "Advanced exact synthesis of Clifford+T circuits," *Quantum Information Processing*, vol. 19, no. 9, p. 317, sep 2020. [Online]. Available: <https://link.springer.com/10.1007/s11128-020-02816-0>
- [24] L. Vinkhuijzen, T. Coopmans, D. Elkouss, V. Dunjko, and A. Laarman, "LIMDD: A decision diagram for simulation of quantum computing including stabilizer states," *Quantum*, vol. 7, p. 1108, 2023.
- [25] X. Hong, A. Dai, D. Gao, S. Li, Z. Ji, and M. Ying, "LimTDD: A Compact Decision Diagram Integrating Tensor and Local Invertible Map Representations," Apr. 2025, arXiv:2504.01168 [cs]. [Online]. Available: <http://arxiv.org/abs/2504.01168>
- [26] I. L. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, 2008.
- [27] J. Biamonte, "Lectures on quantum tensor networks," *arXiv preprint arXiv:1912.10049*, 2019.