Simulating large quantum circuits on a small quantum computer

Aram W. Harrow * Maris Ozols † Tianyi Peng ‡ Xiaodi Wu §

November 19, 2016

Abstract

We explore various strategies for simulating large quantum circuits on a classical computer that has access to a small quantum device. By representing the circuit as a tensor network, we can cut it into smaller parts that can be executed independently by contracting each of the smaller tensor networks. Assuming a partition with not too many edges between different parts can be found, we provide efficient algorithms for simulating such circuits. While in general the simulation cost scales exponentially in the total number of edges between different parts, the size of the quantum memory required scales only linearly in the degree of each part.

Contents

*Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
†Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK.
‡Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China.
§Department of Computer and Information Science, University of Oregon, Eugene, OR 97403, USA.

1 Introduction

While the first small-scale general purpose quantum computers are already available [IBM, BSL+16] and can perform non-trivial tasks such as factoring small numbers [MNM+16], it remains a major experimental challenge to scale these devices up to the point where classical simulation would no longer be feasible. This is not only an important technological challenge but also a fundamental problem that could benefit from theoretical insights. Precisely, we ask

> *How can we (efficiently) simulate large-scale quantum machines, with limited quantum resources, such as small-scale (in time or space) quantum machines, and classical computational resources?*

We hope that in addition to experimental advancement, a theoretical solution to this problem would help bring large-scale quantum computation closer to reality. Several existing theoretical studies can be deemed as efforts towards this end. The first one is *circuit optimization*. Devising efficient ways of compiling high-level quantum algorithms down to actual quantum gates can provide significant savings in terms of quantum memory and time. Another approach is devising *hybrid quantum models* that rely on classical resources, such as classical post-processing or adaptive interaction with a classical machine (e.g., measurement-based quantum computing). An extreme form of classical assistance is full *classical simulation* where the entire computation is carried out on a classical computer.

We are inspired by [BSS16] which explores how an $(n+k)$-qubit quantum computation can be simulated on an n-qubit machine, with the remaining k qubits provided "virtually" through a classical simulation. We propose an alternative approach to this problem based on tensor network formalism to describe quantum circuits. This leads naturally to a *decomposition-combination* strategy: we decompose the given large quantum circuit into smaller pieces that can be simulated independently on a much smaller device; we then combine the simulation results using classical post-processing.

1.1 Tensor network approach

It is natural to represent quantum circuits by tensor networks—graphs with vertices carrying small-order tensors of quantum gates and edges referring to qubit wires and indicating which indices of the two adjacent tensors must be contracted. This alternative perspective of quantum circuits has been very fruitful for discovering both new quantum [AAEL07, FKW02, FLW02, AJL09, AL10] and classical algorithms [Vid03, Joz06, MS08] for simulating quantum circuits and many-body quantum systems [Vid04, SDV06, Vid08]. Tensor network formalism also has close connections with matrix product states, measurement-based quantum computation [GE07, GESPG07], and the study of multi-partite entanglement [PGVWC07, VMC08].

Tensor networks are also the natural mathematical object for studying the decomposition of large circuits, since viewing a large circuit as a graph naturally leads to the idea of cutting it into smaller pieces, each of which can be executed separately on a much smaller device. More importantly, tensor networks blend the two core features of quantum circuits, the unitarity of the gates and the notion of the time (i.e., that gates are applied in some order). In particular, there could be no particular temporal order in a tensor network. Comparing to merely splitting the qubit space in [BSS16], decompositions on tensor networks in some sense allow us to split both the space and the time in a much more flexible manner. The most non-trivial part of such approach is piecing together the simulation results, which is our main technical contribution.

To illustrate the flexibility of the tensor network approach, let us elaborate on a particularly interesting feature. While gates in a quantum circuit have a very rigid time-ordering, this in general is not a case for the corresponding tensor network. Indeed, a tensor network is evaluated by contracting indices, which can typically be done in many different ways (e.g., see the notion of "bubbling" introduced in [AL10]). This feature is particularly striking when the quantum circuit is compiled using a specially chosen universal gate set which does not require treating specific wires as inputs and specific wires as outputs, since any combination results in a unitary gate (see Appendix A). In such case, the tensor network does not have a specific temporal order and hence can be evaluated in a multitude of ways. Given the ability to apply gates in parallel, some choices of the ordering can thus lead to shorter gate sequences than others. An example of this phenomenon is illustrated in Fig. 1: while the original circuit might require executing the gates from left to right, in the tensor network framework we can instead proceed from top to bottom and still maintain the unitarity of each gate. While both execution orders would require the same amount of quantum memory, executing the circuit from top to bottom clearly is more advantageous in terms of the required time.

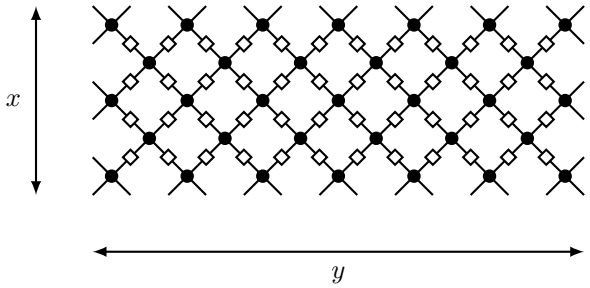

Figure 1: A qutrit tensor network on a 2D grid. Each white node is a random qutrit gate (order-2 tensor) while each black node is the two-qutrit gate (order-4 tensor) Ω defined in Eq. (A.3) (see Appendix A). Because of the properties of Ω, this circuit does not have a specific temporal order and thus can be simulated in many different ways. In particular, executing it from top to bottom takes less time-steps than when executing form left to right, since more two-qutrit gates can be executed in parallel.

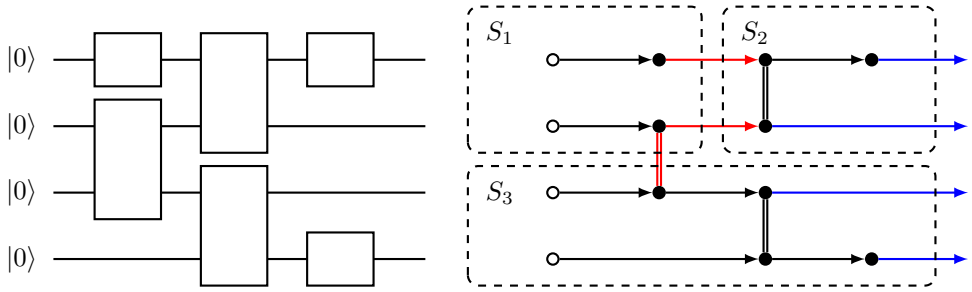

Figure 2: An illustration of our parameters. $T = 3$, represented in red. The undirected degrees of partitions are: $d(S_1) = d(S_3) = 1$, $d(S_2) = 0$ while directed degrees are: $\vec{d}(S_1) = \vec{d}(S_2) = \vec{d}(S_3) = 2$, which include also the free edges (blue). If $\mathcal{P} = \{S_1, S_2, S_3\}$, the maximal directed degree $\vec{d}(\mathcal{P})$ is the maximum of $\vec{d}(S_i)$ and hence equals to 2. The maximal total degree $d^*(\mathcal{P})$ is the maximum of $\vec{d}(S_i) + d(S_i)$ and hence equals to 3.

1.2 Simulation model

Informally, *tensors* are multilinear generalizations of vectors and matrices (they are order-1 and -2 tensors, respectively). An order-k tensor A has k indices, so its entries $A_{i_1,\ldots,i_k}$ can be encoded as amplitudes of a quantum state: $|A\rangle := \sum_{i_1,\ldots,i_k \in \{0,1\}} A_{i_1,\ldots,i_k} |i_1, \ldots, i_k\rangle$ (called *quantum encoding*, see Definition 2.15). A tensor network is a collection of tensors, together with a graph representing their connections. The value of a tensor network is the tensor obtained by contracting all pairs of indices corresponding to edges. For example, the probability of a quantum circuit with a given input to produce a given output is equal to the value of the tensor network representing the circuit.

Tensor network of quantum circuits. More specifically, we consider quantum circuits consisting of only 1-qubit and 2-qubit gates. We represent 1-qubit gates by nodes and 2-qubit gates by a pair of nodes, connected by an undirected edge, and we use directed edges to represent the flow of information from one gate to another (final output is free directed edge). All 1-qubit nodes have degree 2 and are assigned the order-2 tensor describing the corresponding 1-qubit unitary. Following [BSS16], we decompose each 2-qubit gate as $\sum_\alpha c_\alpha V_\alpha \otimes W_\alpha$, where $c_\alpha > 0$ and V_α and W_α are 1-qubit unitaries. We assign the order-3 tensors $\sqrt{c_\alpha} V_\alpha$ and $\sqrt{\alpha} W_\alpha$ to the two nodes representing the 2-qubit gate so that the index α corresponds to the undirected edge between the two nodes. We refer to this as the *tensor network* of the quantum circuit (see Definition 2.21). Our algorithms will be good at simulating circuits whose tensor networks can be broken into sufficiently small pieces by cutting just a few edges.

All our simulation strategies are based on a partition of the vertices of the tensor network into several disjoint parts $S_1, \ldots, S_k$ and we proceed by simulating each S_i separately. While each S_i is also a tensor network of its own, it has several loose edges that connect to other S_j, so we need to ensure that we can

still perform the simulation even if it requires transmitting a qubit from S_i to S_j (in case of a directed edge) or applying a 2-qubit gate between S_i and S_j (in case of an undirected edge). To quantify the number and type of edges that have been cut, we introduce several notions of *degree*. We write $\vec{d}(S_i)$ and $d(S_i)$ to denote the number of directed and undirected edges from S_i to any of the other parts (includes final output), and we define $d^*(S_i) := \vec{d}(S_i) + d(S_i)$. For a given partition, we denote by $\vec{d}$ and d^* the corresponding *maximal degree* (see Definition 2.11) of S_i over all $i \in \{1, \ldots, k\}$. Finally, another graph-theoretic quantity we require is T, the total number of edges between all pairs of distinct parts S_i and S_j. (see Fig. 2 as an example).

Simulation by small quantum computers. To model a small quantum device, we allow the following steps of quantum computation: (1) initialize all n qubits to $|0\rangle$, (2) apply some n-qubit quantum circuit C, (3) measure all qubits in the computational basis and get an n-bit string y, (4) compute $f(y)$ where $f : \{0,1\}^n \to [0,1]$ is some classical poly-time function. We refer to algorithms of this form as *QC (quantum-classical)* and parametrize them by (C, f). Note that all steps of a QC algorithm are well within the capabilities of currently available devices, such as IBM's 5-qubit computer [IBM]. As a slight extension, we also consider a model that can *recycle* qubits. Namely, individual qubits can be measured also at intermediate steps and afterwards initialized again to $|0\rangle$ so that more gates can be applied on the same qubit.

The simulator's goal is to approximate the expectation $\mathbb{E}[f(y)]$ within some accuracy ϵ. Motivated by currently available technology, our simulation model consists of a small quantum device assisted by a powerful classical computer. We treat the quantum device (m qubits) as an oracle that can execute a given circuit and output the random string obtained by measuring all qubits in the standard basis up to Q calls. If a classical algorithm that has access to such device can approximate the expectation of the function computed by the original circuit within error ϵ, we call it a (Q, m, ϵ)-*simulator* (see Definition 2.6, Definition 3.4).

1.3 Contributions

Let (C, f) be a QC algorithm that we need to simulate where C is an n-qubit quantum circuit. Assume we are provided with a partition $\{S_1, \ldots, S_k\}$ of the vertices of the tensor network corresponding to C into k parts, and this partition has parameters T and $\vec{d} \leq d^*$ as defined above.

Theorem 1.1 (Informal)**.** *(C, f) has simulators with the following parameters:*

- $(2^{O(T)} O(\frac{k}{\epsilon^2}), d^*, \epsilon)$*-simulator (Theorem 3.3),*
- $(2^{O(T)} O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$*-simulator with recycling (Corollary 3.5).*

Making additional assumptions about the partition of the tensor network and the classical post-processing function f, we can improve the simulation parameters further. More specifically, let G^* denote the graph obtained by contracting each S_i to a single node and denote the *contraction complexity* [MS08] of G^* by $\mathrm{cc}(G^*)$ (Definition 2.19). If we further assume that the function f has rank r (i.e., f can be expressed as a sum of at most r product terms), then we get the following result:

Theorem 1.2 (Informal)**.** *(C, f) has a $\left(\mathrm{poly}(n + L, r) 2^{O(\mathrm{cc}(G^*))} O(\frac{1}{\epsilon^2}), d^*, \epsilon\right)$-simulator for small enough ϵ, where n is the number of qubits and L is the number of gates in the original circuit (Theorem 3.9).*

We can readily apply these results to obtain important applications. In particular, we show the possibility of simulating a general tensor network (rather than the one from any circuit) in Corollary 4.1/4.2 and of simulating low-dimensional quantum circuits in Corollary 4.3/4.4 Finally, we also provide an alternative approach that generalizes ideas from [BSS16] with comparable parameters, however, without keeping the nice features of tensor networks.

Theorem 1.3 (Informal)**.** *(C, f) has a $(2^{O(T+k)} O(\frac{k}{\epsilon^2}), \vec{d} + 1, \epsilon)$-simulator (Theorem 5.3).*

1.4 Techniques and related work

Our approach is inspired by previous applications of tensor networks in quantum information theory, and in particular in quantum circuit simulation. For example, Arad and Landau [AL10] study the quantum simulation of general tensor networks by implementing the contraction operation on a quantum computer,

while Markov and Shi [MS08] provide a general framework for classical simulation of quantum circuits through tensor networks.

While the general idea of using tensor networks for simulating quantum circuits is central to our work, we have a different perspective since we are specifically interested in simulation methods that can run on a *small* quantum device, assisted by a powerful classical computer. From this perspective, the most relevant work to ours is [BSS16], which shows how $(n+k)$-qubit quantum circuits can be simulated on an n-qubit quantum machine. For this purpose, [BSS16] considers both the quantum circuit model as well as a Pauli-based model of computation. In contrast, our results are based on the tensor network formalism. Nevertheless, we provide a strict generalization of the circuit-based results from [BSS16].

One of our main technical ideas is to encode tensors by pure quantum states, and to use Pauli measurements and classical post-processing for tensor contractions. This allows us to combine several tensors together, even if the corresponding quantum states are not physically present at the same time. Indeed, we can separately prepare the quantum encoding of each piece of the tensor network. We use Pauli measurements and classical post-processing to combine the results.

In the case where the simulation of one part of the system depends on the output of another part, we use one half of a maximally entangled state the replace the unknown qubit. Effectively, this allows us to recover the same simulation result for any possible input up to some parameter loss.

Finally, we also provide a direct generalization of the [BSS16] approach. One key new observation is to use ancillary SWAP gates to make more flexible and powerful partitions, involving not only the qubit space but also the temporal order.

1.5 Conclusions

Our ultimate goal is to provide a flexible platform for simulating large quantum circuits on small devices based on the tensor network formalism. We expect that such framework would appeal both to theoreticians as well as experimentalists and engineers, and we hope that it would become a foundation onto which other features can be added to later on.

There are obviously many open questions of this work. One important question is how to find such good partitions of tensor networks efficiently. Another one is to apply this framework to specific quantum algorithms, such as Shor's algorithm and phase estimation, for real-world implementation.

Organization

Our paper organizes as following structure: In Section 2, we introduce the definition of tensor network of the quantum circuit and other necessary preliminaries for presenting our statements. In Section 3, we state our main results of simulation based on tensor network formalism. Then, we discuss some applications such as 2D circuits with low-connectivity in Section 4. Finally, we provide an alternative approach generalized from [BSS16] in Section 5.

2 Preliminaries

In this section we describe the model of quantum computation and the notion of simulation we are interested in. We also define concepts such as the graph and tensor network associated to a quantum circuit.

2.1 Useful lemmas

For convenience, we first provide some useful inequalities without proof.

Lemma 2.1 (Hoeffding's inequality[Hoe63])**.** *If $X_1, \ldots, X_L$ are independent real random variables satisfying $a_i \leq X_i \leq b_i$ and $\bar{X} := (X_1 + \cdots + X_L)/L$ then*

$$\Pr\left[\left|\bar{X} - \mathbb{E}[\bar{X}]\right| < \epsilon\right] > 1 - 2\exp\left(-\frac{2L^2\epsilon^2}{\sum_{i=1}^{L}(b_i - a_i)^2}\right). \tag{2.1}$$

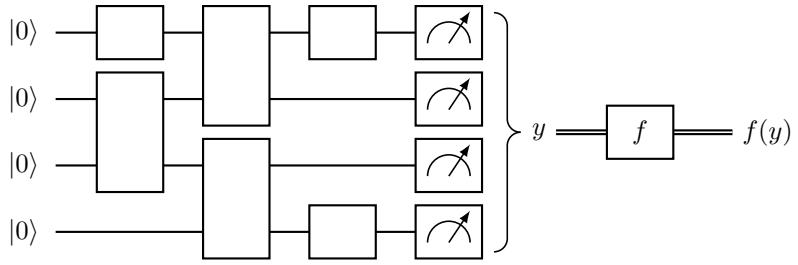

Figure 3: An algorithm in the QC model (see Definition 2.5). If the number of qubits is n then $y \in \{0,1\}^n$ and $f : \{0,1\}^n \to [0,1]$ is a function that is evaluated on a classical computer.

Corollary 2.2. *Suppose $X_1, \ldots, X_L$ are i.i.d. real random variables such that $|X_i| \leq a$ and define $\bar{X} := (X_1 + \cdots + X_L)/L$. If we set $L := 4a^2/\epsilon^2$ then*

$$\Pr\left[\left|\bar{X} - \mathbb{E}[\bar{X}]\right| < \epsilon\right] > 1 - 2\exp\left(-\frac{2L\epsilon^2}{4a^2}\right) > \frac{2}{3}. \tag{2.2}$$

2.2 Computational model: quantum-classical circuits

We begin by introducing the general notions of a quantum circuit and a quantum algorithm.

Definition 2.3 (Quantum gates and circuits)**.** *A k-qubit* quantum gate *is a unitary operator in $\mathrm{U}(2^k)$. An n-qubit* quantum circuit *is an ordered sequence of 1-qubit and 2-qubit quantum gates (along with a specification on which qubit(s) each gate is applied) and thus implements a unitary operation in $\mathrm{U}(2^n)$.*

Definition 2.4 (Quantum algorithm)**.** *A* quantum algorithm *based on an n-qubit quantum circuit C consists of the following steps:*

1. *initialize n qubits to $|0^n\rangle$;*
2. *apply the quantum circuit C on $|0^n\rangle$;*
3. *measure all qubits in the 0/1 basis and output the resulting n-bit string.*

While any other intuitive notion of a quantum algorithm can be cast in the form described above, due to practical considerations it is useful to explicitly include an extra classical post-processing step at the end of the algorithm. Such step, for example, can turn the n-bit output string into a real number whose expectation might be of interest (e.g., it could correspond to energy of some physical system). While such classical post-processing can in principle be done on a quantum computer and thus might seem superfluous, it is useful to single it out as a separate step that is performed classically since, in general, classical implementation should be less costly. To make this distinction a bit more formal, we define the following *quantum-classical* (QC) computational model.

Definition 2.5 (QC model)**.** *Algorithms in the* QC model *are denoted by (C, f), where C is an n-qubit quantum circuit and $f : \{0,1\}^n \to [0,1]$ is a classical polynomial-time function. The execution of a QC algorithm (C, f) consists of:*

1. *running the quantum algorithm based on circuit C;*
2. *classically computing a real number $f(y)$ from the n-bit output string y.*

We denote by $\pi(C, f) \in [0,1]$ the expectation of $f(y)$ over many runs of the algorithm.

An example of an algorithm in the QC model is illustrated in Fig. 3. It produces a classical description of a real number in the bounded interval $[0,1]$ whose expectation $\pi(C, f)$ will be our main interest.

Q	the number of times we can use the quantum device
n	the number of qubits in the original circuit C
m	the number of qubits available on the simulator ($m \leq n$)
L	the number of gates in the original circuit C (simulator can execute $O(L)$ gates)
ϵ	the desired accuracy of the simulation

Table 1: Simulator parameters used in Definition 2.6.

2.3 Simulator

Our goal is to execute computations of the type described above on a small quantum device assisted by a much larger classical computer. In other words, we would like to simulate such computations using less quantum memory but at the cost of more quantum time as well as more classical computation. To formalize this notion, we introduce a fairly general class of simulators (for convenience, we summarize the relevant parameters in Table 1). We do not explicitly specify what the input of a simulator is, but normally it is a classical description of some quantum circuit C to be simulated, along with some extra advice[1], such as how to break up the circuit into smaller pieces.

Definition 2.6 (Simulator)**.** *Let (C, f) be a QC algorithm (see Definition 2.5) where C is an n-qubit quantum circuit with L gates. Then a (Q, m, ϵ)-*simulator *of (C, f) is a classical $\mathrm{poly}(Q, n, L)$-time algorithm that has access to a quantum oracle. The oracle can be called at most Q times and, in each call, can execute any $O(L + m)$-gate quantum algorithm on an m-qubit device (see Definition 2.4). Such simulator has accuracy ϵ for simulating (C, f) if its output $\theta \in \mathbb{R}$ satisfies:*

$$\Pr[|\theta - \pi(C, f)| < \epsilon] > \frac{2}{3}. \tag{2.3}$$

Corollary 2.7. *From Corollary 2.2, there is a trivial $(O(1/\epsilon^2), n, \epsilon)$-simulator for any QC algorithm (C, f) where C is an n-qubit quantum circuit with L gates. Indeed, we can simply execute $O(1/\epsilon^2)$ runs of the original algorithm (C, f) and then compute θ as the average of all results obtained.*

Note that the parameter Q not only relates to the number of quantum queries but also to the classical running time. The following corollary illustrates this.

Corollary 2.8. *There is a trivial $(2^n, 0, \epsilon)$-simulator for any QC algorithm (C, f) where C is an n-qubit quantum circuit with L gates. Simply represent the state and gates as a matrix and use matrix product to simulate the circuit classically. The parameter 2^n is because the classical running time is $\mathrm{poly}(2^n, L, n)$.*

A problem of central interest in our work is the trade-off between parameters Q and m. Namely, can more repetitions (and classical post-processing) help with decreasing the amount of quantum memory m used in the computation? Note that results above provide two extreme situations. We are thus interested in intermediate cases when a non-trivial amount of quantum computation is still used and the overall runtime is polynomial.

2.4 Quantum circuits as graphs

It is natural to represent the interactions between individual qubits of a quantum circuit by a graph. In fact, such representation will be used—in one form or another—by all our simulation strategies.

To any quantum circuit we assign a graph obtained as follows. Its vertices correspond to quantum gates (one vertex for a single-qubit gate and two vertices for a two-qubit gate) and its edges are of two types: *directed* edges represent the flow of information from one gate to another while *undirected* edges represent two-qubit gates (see Fig. 4 for an example). In general, we refer to graphs that have both types of edges as *mixed graphs*. Below is a more formal definition of the mixed graph associated to a quantum circuit.

Definition 2.9 (Graph of a quantum circuit)**.** *To any quantum circuit C we associate the following graph:*

[1]For the purpose of this work, we are not concerned with the question of where this advice comes from. We simply assume that it is provided by some oracle.

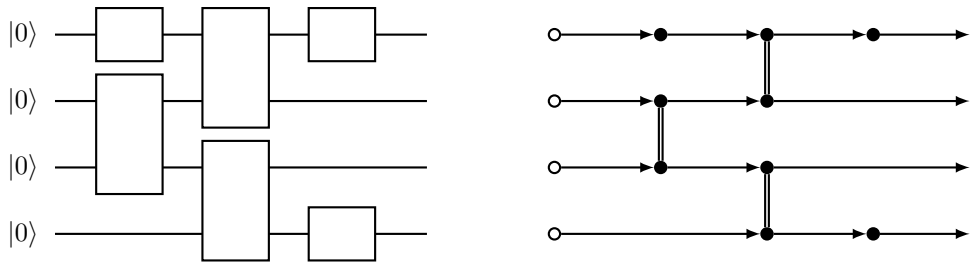

Figure 4: An example of a quantum circuit and the associated graph (see Definition 2.9).

1. *Let $L(i)$ be the number of gates applied to the i^{th} qubit in C. For the k^{th} gate applied to the i^{th} qubit, we create a node i_k, $1 \leq k \leq L(i)$ (in particular, for each two-qubit gate we create two nodes). For convenience, for each input qubit i we also create a node i_0.*
2. *For every pair of nodes (i_k, i_{k+1}), $0 \leq k < L(i)$, connect them by a directed edge from i_k to i_{k+1}.*
3. *For every node $i_{L(i)}$, create a directed edge from it without ending node to represent the output.*
4. *For each pair of nodes (i_k, j_t) such that these two nodes correspond to the same two-qubit gate, connect them by an undirected edge.*

Let V be the set of all nodes and E the set of all edges (E includes both directed and undirected edges). Then $G = (V, E)$ is the (mixed) graph associated to quantum circuit *C. Note that we create nodes for input qubits and single-node-edges for output qubits.*

Next, we introduce several quantities from graph theory that will later play important role in our results on simulation complexity of quantum circuits.

Definition 2.10 (Directed and undirected degree)**.** *Let $G = (V, E)$ be a mixed graph (such as one from Definition 2.9). For any $v \in V$, we write $\vec{d}(v)$ and $d(v)$, respectively, to denote the* directed *and* undirected degree *of v, which corresponds to the number of directed (outgoing) and undirected edges of v. Similarly, for any subset of vertices $S \subseteq V$, we write $\vec{d}(S)$ and $d(S)$, respectively, to denote the number of directed (outgoing) and undirected edges leaving the subset S.*

These notions of degree can be easily extended to partitions of a graph. Recall that a *partition* of set V is a collection of subsets $S_1, \ldots, S_k \subseteq V$ such that $S_1 \cup \cdots \cup S_k = V$ and $S_i \cap S_j = \varnothing$ for $i \neq j$.

Definition 2.11 (Maximal directed and total degree of a parition)**.** *Let $G = (V, E)$ be a mixed graph and $\mathcal{P} = \{S_1, \ldots, S_k\}$ a partition of its vertices V. The* maximal directed degree *and* total degree *of this partition is, respectively,*

$$\vec{d}(\mathcal{P}) := \max_{1 \leq i \leq k} \vec{d}(S_i), \tag{2.4}$$

$$d^*(\mathcal{P}) := \max_{1 \leq i \leq k} d(S_i) + \vec{d}(S_i). \tag{2.5}$$

Definition 2.12 (Number of edges of a partition)**.** *Let $G = (V, E)$ be a mixed graph and $\mathcal{P} = \{S_1, \ldots, S_k\}$ a partition of V. The* number of edges *of $\mathcal{P}$, denoted by $T(\mathcal{P})$, is the total number of edges (including directed and undirected) between different subsets S_i of $\mathcal{P}$. Intuitively, $T(\mathcal{P})$ characterizes the number of quantum communications between different subsets. Attention here we do not count the single-node-edges (output qubits).*

2.5 Tensors and tensor networks

For the purpose of simulating quantum circuits, it is useful to add extra structure to the corresponding graph by assigning a tensor to every vertex, resulting in a structure known as a tensor network. Intuitively, tensors are multilinear generalizations of vectors and matrices—they are multidimensional arrays whose entries are specified by several indices, each of which has a certain range.

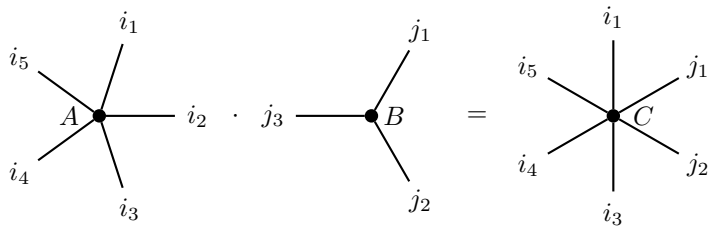

Figure 5: An illustration of tensor contraction (we contract index i_2 of A with index j_3 of B).

Definition 2.13 (Order-k tensor)**.** *An* order-k tensor A *is a k-dimensional array over $\mathbb{C}$, with the entries of A specified using k indices: $A_{i_1,\ldots,i_k}$.*

If there is no special request, we will usually take $i_1, \ldots, i_k \in \{0,1\}$ so that an order-k tensor has 2^k entries in total.

Definition 2.14 (Norm of a tensor)**.** *The norm of a tensor $A_{i_1,\ldots,i_k}$ is denoted by $|A|$:*

$$|A| := \sum_{i_1,\ldots,i_k \in \{0,1\}} |A_{i_1,\ldots,i_k}|^2. \tag{2.6}$$

If $|A| = 1$, we call A as a normalized tensor*.*

Definition 2.15 (Quantum encoding)**.** *It is convenient to represent a normalized order-k tensor A by a pure quantum state on k qubits:*

$$|A\rangle := \sum_{i_1,\ldots,i_k \in \{0,1\}} A_{i_1,\ldots,i_k} |i_1, \ldots, i_k\rangle. \tag{2.7}$$

We call $|A\rangle$ as the quantum encoding *of tensor A.*

Just like vectors and matrices can be combined using matrix multiplication, tensors can be combined using index contraction. If $A_{i_1,\ldots,i_k}$ and $B_{j_1,\ldots,j_l}$ are two tensors such that the ranges of indices i_1 and j_1 agree, we can contract these indices to obtain an order-$(k+l-2)$ tensor C whose entries are given by

$$C_{i_2,\ldots,i_k,j_2,\ldots,j_l} = \sum_i A_{i,i_2,\ldots,i_k} B_{i,i_2,\ldots,j_l}. \tag{2.8}$$

A similar expression can be used to contract any other pair of indices i_a and j_b of the two tensors, where $a \in [k]$ and $b \in [l]$. Note that such operation has an intuitive graphical representation (see Fig. 5).

Given a collection of tensors, one can combine them into a tensor network by identifying pairs of indices that must be contracted. Such a network is conveniently represented by a graph $G = (V, E)$ with vertices corresponding to tensors and edges corresponding to indices. To translate between graphs and tensors, we implicitly establish a bijection between the indices of $A(v)$ and the edges incident to v: every edge $uv \in E$ is identified with a pair of tensor indices—one from $A(u)$ and one from $A(v)$—in a way so that each index of $A(v)$ is associated to a different edge incident to v.

Definition 2.16 (Abstract tensor network)**.** Abstract tensor network *is a pair $(G, \mathcal{A})$ where $G = (V, E)$ is an undirected graph and $\mathcal{A} = \{A(v) : v \in V\}$ is a collection of tensors, one for every vertex of G, such that the order of $A(v)$ is equal to the degree $d(v)$ of v.*

Implicitly, an abstract tensor network also carries a function $a : V \times \mathbb{N} \to E$ such that $a(v, i) \equiv a_i(v)$ corresponds to the i-th edge incident with vertex $v \in V$. We will also write

$$a(v) := \big(a_1(v), \ldots, a_{d(v)}(v)\big) \subseteq E \tag{2.9}$$

to denote an ordered list of all edges incident with v. Furthermore, we will write $\{0,1\}^E$ to denote the set of all functions $E \to \{0,1\}$ that assign a label from $\{0,1\}$ to each element of E. For a given labelling

$s \in \{0,1\}^E$, we will write $s_e \in \{0,1\}$ to denote the label of $e \in E$ and, for any $E' \subseteq E$, we will write $s_{E'}$ to denote the substring of s that contains the labels of those edges that belong to the subset E'. In particular, for any $v \in V$, $s_{a(v)}$ denotes the substring of s that corresponds to $a(v)$. In other words, the string $s_{a(v)} \in \{0,1\}^{d(v)}$ describes what labels s assigns to those edges that are incident to $v \in V$.

Definition 2.17 (Value of an abstract tensor network)**.** *The* value *of an abstract tensor network $(G, \mathcal{A})$ is the squared absolute value of order-0 tensor obtained by contracting all edges of the network. It can be conveniently expressed using the notation introduced above as follows:*

$$T(G, \mathcal{A}) := \left| \sum_{s \in \{0,1\}^E} \prod_{v \in V} A(v)_{s_{a(v)}} \right|^2. \tag{2.10}$$

Remark: This definition is slightly different fomr [MS08] which has no squared absolute value. The reason is because our tensor network represents a pure state while theirs represents a mixed state and thus has twice as many indices.

The following somewhat more complicated definition corresponds to a scenario when the tensor network in question either has some free edges or is a piece of a larger tensor network.

Definition 2.18 (Subtensor)**.** *Let $(G, \mathcal{A})$ be an abstract tensor network where $G = (V, E)$. For any subset of nodes $S \subseteq V$, let $A(S)$ denote the order-$d(S)$* subtensor *obtained by contracting all nodes in S where $d(S)$ is the degree of S (i.e., the number of edges connecting S and $V \setminus S$). More formally, we first partition the set of all edges E into two disjoint subsets:*

$$E_1 := \{uv \in E : u \in S, v \in V \setminus S\}, \tag{2.11}$$
$$E_2 := \{uv \in E : u \in S, v \in S\}. \tag{2.12}$$

Then, for any $s_{E_1} \in \{0,1\}^{E_1}$, the s_{E_1}-entry of $A(S)$ is given by

$$A(S)_{s_{E_1}} := \sum_{s_{E_2} \in \{0,1\}^{E_2}} \prod_{v \in S} A(v)_{s_{a(v)}}, \tag{2.13}$$

where $s \in \{0,1\}^{E_1 \cup E_2}$ denotes the string obtained by combining the two disjoint substrings s_{E_1} and s_{E_2}.

The complexity of computing the value of a tensor network is known as *contraction complexity*. Here we quote its definition from [MS08]:

Definition 2.19 (Contraction complexity)**.** *Given an abstract tensor network $(G, \mathcal{A})$ which has m edges and an order of edges $\pi = \{\pi_1, \ldots, \pi_m\}$. The contracting process runs as following: For i^{th} turn, delete the edge π_i and contract two nodes connected by π_i into one node if these two nodes are different. The complexity of this order is defined by the maximal degree of nodes during contracting process. And the* contraction complexity *of $(G, \mathcal{A})$ defined by $cc(G)$ is the minimal complexity among all orders.*

Below we quote the main result from [MS08] without a proof.

Theorem 2.20 ([MS08])**.** *Given an abstract tensor network $(G, \mathcal{A})$ where $G = (V, E)$, we can classically compute the exact value of $A(V)$ in time* $\mathrm{poly}(|V| 2^{O(\mathrm{cc}(G))})$.

Here we slightly modify the definition from [MS08] of a tensor network representing a quantum circuit.

Definition 2.21 (Tensor network of a quantum circuit)**.** *Given a quantum circuit C, let $G = (V, E)$ be the graph associated to C via Definition 2.9. Then the* tensor network of quantum circuit *C, denoted by $(G, \mathcal{A})$, is a tensor network that satisfies the following additional constraints:*

1. *To each directed outgoing edge from i_k assign an index variable $e_{i_k} \in \{0,1\}$. To each undirected edge corresponding to the i^{th} two-qubit gate, assign an index variable $\alpha_i \in \{0, 15\}$.*
2. *To each node i_k that corresponds to a one-qubit gate $U \in \mathrm{U}(2)$, assign an order-two tensor $A(i_k)$ whose components are given by $A(i_k)_{e_{i_{k-1}}, e_{i_k}} := \langle e_{i_k} | U | e_{i_{k-1}} \rangle$.*

3. *For each pair of nodes (i_k, j_t) that corresponds to a two-qubit gate $U \in \mathrm{U}(4)$ (the l^{th} two-qubit gate), assume U can be decomposed as follows:*

$$U = \sum_{\alpha=0}^{15} c_\alpha V^\alpha \otimes W^\alpha, \tag{2.14}$$

where $V^\alpha, W^\alpha \in \mathrm{U}(2)$, $c_\alpha \geq 0$, and $\sum_\alpha c_\alpha^2 = 1$ (e.g., one can choose V^α, W^α as Pauli matrices and multiple some phase factor into V^α for guaranteeing $c_\alpha \geq 0$). Then define order-three tensors

$$A(i_k)_{e_{i_{k-1}}, e_{i_k}, \alpha_l} := \sqrt{c_\alpha} \langle e_{i_k} | V^{\alpha_l} | e_{i_{k-1}} \rangle, \tag{2.15}$$

$$A(j_t)_{e_{j_{t-1}}, e_{j_t}, \alpha_l} := \sqrt{c_\alpha} \langle e_{j_t} | W^{\alpha_l} | e_{j_{t-1}} \rangle. \tag{2.16}$$

Remark: Here we split two-qubit gates into two nodes so that we have more flexibility in terms of how we can partition the tensor network. However, our main approach does not make explicit use of the decomposition in Eq. (2.14), whereas it plays a major role in [BSS16]. In Section 5 we provide an alternative approach, based on ideas from [BSS16], that *does* make use of this decomposition.

If we contract the tensor network along with the direction of quantum circuit, the tensor we get will relate with the quantum states in a instant. If we use an n-bit string y to represent the indices of the output, it is easy to verify that:

$$A(V)_y = \langle y | C | 0^n \rangle. \tag{2.17}$$

where $A(S)_u$ represents the remainder tensor of $A(S)$ when given part of the indices as u. Here $A(V)$ is an n-order tensor. So when given y, it's a 0-order tensor which indicates a complex number.

So the expected value output by (C, f) is

$$\pi(C, f) = \sum_{y \in \{0,1\}^n} |A(V)_y|^2 f(y). \tag{2.18}$$

2.6 An illustration of the relevant parameters

For convenience, we illustrate some of the parameters which are frequently used in the statements of our results. Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of the nodes of G. Denote by T the total number of edges of $\mathcal{P}$ (see Definition 2.10). Assume $\mathcal{P}$ has maximal directed degree $\vec{d}$ and maximal total degree d^* (see Definition 2.11). Fig. 2 provides an example.

3 Main results

In this section, we derive our main results: Theorem 3.3, Theorem 3.9.

In Section 3.1, we deduce the theorem about simulating based on partition of tensor networks. The general idea is to cut the tensor network into small pieces and simulate each piece by a small quantum machine. Then, if the number of edges which connect different pieces is not too large, we can collect the results of all pieces and output the correct value after classical post-processing. Lemma 3.1 provides an approach to combine the small pieces by measuring the quantum encoding of tensors with Pauli matrix measurement. Then Lemma 3.2 tells how to get the quantum encoding of each pieces if the crossing edges connect them are only directed. The central technology used in Lemma 3.2 is inputting one qubit of the maximally entangled state to simulate the unknown state which comes from other pieces. By moving the undirected edge and its related two nodes into new pieces, we can transfer the crossing undirected edge into directed edges. This induces Theorem 3.3, which states that (C, f) has a $(2^{O(T)} O(\frac{k}{\epsilon^2}), d^*, \epsilon)$-simulator (See parameters from Section 2.6, following is the same so will be omitted). If recycling is permitted in simulating, we can save up the qubits which pay for each undirected edge to get Corollary 3.5, which claims that (C, f) has a $(2^{O(T)} O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$-simulator with recycling.

In Section 3.2, we consider the situation if f can be expressed as a sum of r product terms. It permits us to approximately tomography the tensor of small pieces because f is decomposed to be embedded into each piece instead of overall computing. Then the approach based on contracting complexity for combining

tensors can be used to replace the approach in Lemma 3.1. The Lemma 3.8 help bound the error during contracting process, by utilizing the property that any contracting tensor element is not too large because the tensor relates to some part of circuit. So finally we can conclude Theorem 3.9: (C, f) has a $(\mathrm{poly}(n + L, r)2^{O(cc(G^*))}O(\frac{1}{\epsilon^2}), d^*, \epsilon)$-simulator for small ϵ. Here $cc(G^*)$ denotes the contracting complexity of graph G^* which comes from contracting each piece into a single node, and C is a n-qubit circuit with L gates.

3.1 Simulation based on a partition of the tensor network

First, we introduce a lemma to tell how to combine the information of each piece for computing the value of the whole tensor network. The idea is remarkably simple, we only need to measure each qubit of quantum encoding of each piece in some Pauli matrix basis, and then multiply the results together.

Lemma 3.1. *Let $(G, \mathcal{A})$ be an abstract tensor network where $G = (V, E)$ and assume each $A(v)$ is a normalized tensor (see Definition 2.14). Given one copy of quantum encoding $|A(v)\rangle$ for each $v \in V$, there exists a quantum algorithm that outputs a real random variable ζ such that $\mathbb{E}[\zeta] = T(G, \mathcal{A})$ and $\zeta \in [-2^{|E|}, 2^{|E|}]$. The algorithm measures each qubit independently and uses $\mathrm{poly}(|V| + |E|)$ time of classical post-processing.*

Proof. We first describe an algorithm to produce ζ and then prove its correctness. Let us denote by I, X, Y, Z the Pauli matrices. Denote by $x(e), y(e)$ the two qubits which correspond to edge e in two nodes connected by e. First, for each edge $e \in E$, pick a uniformly random Pauli matrix σ_e. Then perform a measurement σ_e for two input qubits $x(e), y(e)$ respectively. Assume the output of Pauli measurement of each input qubit x is t_x. Finally, output

$$\zeta = \prod_{e \in E} 2t_{x(e)}t_{y(e)} \cdot (-1)^{\sigma_e = Y}. \tag{3.1}$$

It is easy to verify that $\zeta \in [-2^m, 2^m]$ because $t_x \in \{-1, 1\}$, so it only remains to prove that $\mathbb{E}[\zeta] = T(G, \mathcal{A})$.

Recall from Definition 2.17 that the value of the tensor network $(G, \mathcal{A})$ is given by:

$$T(G, \mathcal{A}) = \left| \sum_{s \in \{0,1\}^E} \prod_{v \in V} A(v)_{s_{a(v)}} \right|^2. \tag{3.2}$$

Since $A(v)_{s_{a(v)}} = \langle s_{a(v)}|A(v)\rangle$ where $|A(v)\rangle$ is the quantum encoding of $A(v)$, see Definition 2.15, we can write:

$$\prod_{v \in V} A(v)_{s_{a(v)}} = \prod_{v \in V} \langle s_{a(v)}|A(v)\rangle = \left(\bigotimes_{e \in E} \langle s_e, s_e|\right)\left(\bigotimes_{v \in V} |A(v)\rangle\right) =: \langle s, s|\mathcal{A}\rangle, \tag{3.3}$$

where $|s, s\rangle$ and $|\mathcal{A}\rangle$ are both $2|E|$-qubit states, with the qubits arranged appropriately so that each edge $uv \in E$ is aligned with the corresponding vertices $u, v \in V$. More specifically, $|s, s\rangle$ is a standard basis vector that repeats each bit of $s \in \{0, 1\}^E$ twice, with the bit corresponding to edge $uv \in E$ appearing both at vertex u and vertex v. The state $|\mathcal{A}\rangle$ is a tensor product of all $|A(v)\rangle$, so it conveniently encodes the whole tensor network in a single state.

Note that $\sum_{s \in \{0,1\}^E} |s, s\rangle = |\Phi\rangle^{\otimes |E|}$ where $|\Phi\rangle := |0, 0\rangle + |1, 1\rangle$ is the (unnormalized) maximally entangled state. If we let $\Phi := |\Phi\rangle\langle\Phi|$, we can re-express the value of the tensor network as follows:

$$T(G, \mathcal{A}) = \sum_{s,t \in \{0,1\}^E} \langle \mathcal{A}|s, s\rangle\langle t, t|\mathcal{A}\rangle = \langle \mathcal{A}|\left(\sum_{s,t \in \{0,1\}^E} |s, s\rangle\langle t, t|\right)|\mathcal{A}\rangle = \langle \mathcal{A}|\Phi^{\otimes |E|}|\mathcal{A}\rangle. \tag{3.4}$$

If $\{I, X, Y, Z\} \equiv \{\sigma_I, \sigma_X, \sigma_Y, \sigma_Z\}$ denote the *Pauli matrices* then

$$\Phi = \frac{1}{2}(I \otimes I + X \otimes X - Y \otimes Y + Z \otimes Z) = \frac{1}{2} \sum_{j \in \{I,X,Y,Z\}} (-1)^{j=Y} \sigma_j \otimes \sigma_j, \tag{3.5}$$

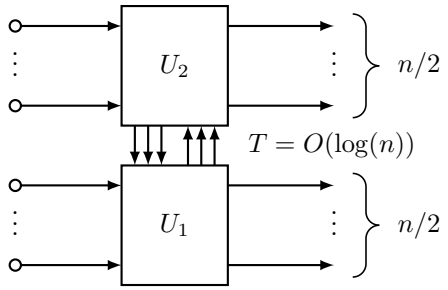

Figure 6: Here is an example of Lemma 3.2. We can use a $\frac{n}{2} + O(\log(n))$-qubit machine to simulate this circuit.

where $(-1)^{j=Y} = -1$ if $j = Y$ and 1 otherwise. Observe that

$$\left(\frac{1}{2}\sum_{j\in\{I,X,Y,Z\}}(-1)^{j=Y}\sigma_j\otimes\sigma_j\right)^{\otimes|E|} = \frac{1}{4^{|E|}}\sum_{p\in\{I,X,Y,Z\}^E}(-1)^{|p|_Y}\bigotimes_{e\in E}2(\sigma_{p_e}\otimes\sigma_{p_e}) \tag{3.6}$$

where $|p|_Y$ denotes the number of Y's in the string p. Note that $\bigotimes_{e\in E}(\sigma_{p_e}\otimes\sigma_{p_e}) = \bigotimes_{v\in V}\sigma_{p_{a(v)}}$ where $\sigma_{p_S} := \bigotimes_{e\in S}\sigma_{p_e}$ for any $S \subseteq E$ and $p \in \{I,X,Y,Z\}^S$. Putting everything together,

$$T(G,\mathcal{A}) = \frac{1}{4^{|E|}}\sum_{p\in\{I,X,Y,Z\}^E}(-1)^{|p|_Y}2^{|E|}\prod_{v\in V}\langle A(v)|\sigma_{p_{a(v)}}|A(v)\rangle. \tag{3.7}$$

Since $\langle A(v)|\sigma_{p_{a(v)}}|A(v)\rangle = \mathbb{E}[\prod_{e\in a(v)} t_e]$, we get the desired result. □

There are two types of communication between different quantum circuit pieces based on the type of crossing edges. The first one is directed edge, which transmits some qubits from one subset to another. The second one is undirected edge, which represents the crossing gates applied to two parts together. In some sense, they are interchangeable (for example, we can use SWAP gate and ancilla to represent qubit transmitting). But this changing may introduce additional cost instead of free. Here we consider the directed edge type preferentially.

The lemma following tells the way to get the quantum encoding (see Definition 2.15) of each piece if there are only directed edges between different pieces. The central obstacle to simulate the piece of quantum circuit is lacking knowledge of inputing qubits come from other subsets. We find a neat way to solve it, which only needs to input a pair of maximally entangled state. Then we apply Lemma 3.1 to combine the information.

Lemma 3.2. *Let (C,f) be a QC algorithm and $(G,\mathcal{A})$ be the tensor network of C. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of the nodes of G such that each two nodes corresponding to the same 2-qubit gate belong to the same subset. If $\mathcal{P}$ has maximal directed degree $\vec{d} := \vec{d}(\mathcal{P})$ (see Definition 2.11) and T edges (see Definition 2.10), then there is a $(2^{O(T)}O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$-simulator of (C,f). (see Fig. 6 as an example)*

Proof. For any subset S_i and its related tensor $A(S_i)$, denote by x_i the set of indices of incoming edges, by y_i the set of indices of outgoing edges to other subsets, and by u_i the set of indices of outgoing edges to the final outputs. Consider some inputs of $|0\rangle$ represented as nodes in S_i, the overall input-output transformation can be denoted as a unitary tranformation U. Then by definition of tensor,

$$A(S_i)_{x_i,y_i,u_i} := \langle y_i u_i|U|0x_i\rangle. \tag{3.8}$$

Denote by $|x_i|$ the number of incoming edges of S_i. It's easy to see that $\frac{A(S_i)}{2^{|x_i|/2}}$ is a normalized tensor because

$$\sum_{x_i,y_i,u_i}|A(S_i)_{x_i,y_i,u_i}|^2 = \sum_{x_i\in\{0,1\}^{|x_i|}}\sum_{y_i,u_i}|\langle y_i u_i|U|0x_i\rangle|^2 = \sum_{x_i\in\{0,1\}^{|x_i|}}1 = 2^{|x_i|}. \tag{3.9}$$

If instead we input one qubit of the maximally entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ to each incoming edge of U, we get the following pure state as the *quantum encoding* of tensor $\frac{A(S_i)}{2^{|x_i|/2}}$:

$$|A(S_i)\rangle := \frac{1}{2^{|x_i|/2}} \sum_{x_i} |x_i\rangle U|0x_i\rangle = \frac{1}{2^{|x_i|/2}} \sum_{x_i,y_i,u_i} A(S_i)_{x_i,y_i,u_i}|x_i\rangle|y_i u_i\rangle. \tag{3.10}$$

Our algorithm works as follows: first, implement the quantum encoding $|A(S_i)\rangle$ of each tensor of S_i by running a $(\vec{d}(S_i) + |x_i|)$-qubit circuit. Then measure the final output qubits u_i in the computational basis and transfer $|A(S_i)\rangle$ to $|A(S_i)\rangle_{u_i}$. Assume the probability of getting outcome u_i is $\Pr(u_i)$, so

$$|A(S_i)\rangle_{u_i} = \frac{1}{2^{|x_i|}\sqrt{\Pr(u_i)}} \sum_{x_i,y_i} A(S_i)_{x_i,y_i,u_i}|x_i y_i\rangle. \tag{3.11}$$

Now $|A(S_i)\rangle_{u_i}$ is the quantum encoding for normalized form of the remainder tensor $A(S_i)$ when given u_i. By applying Lemma 3.1 to the set of $|A(S_i)\rangle_{u_i}$ whose induced tensor network has k nodes and T edges. We get a random variable $\zeta \in [-2^T, 2^T]$ such that

$$E[\zeta] = \left| \sum_{x,y} \prod_i \frac{A(S_i)_{x_i,y_i,u_i}}{\sqrt{\Pr(u_i)}2^{|x_i|/2}} \right|^2. \tag{3.12}$$

where $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ represent the whole graph's indices related to incoming edges and outgoing edges to other subsets, respectively. Finally, we output a new random variable $\chi := 2^T \zeta \cdot f(u_1, u_2, ..., u_k)$.

Let us now verify that $\pi(C, f) = \mathbb{E}[\chi]$ (recall that $\sum_{i=1}^k |x_i| = T$):

$$\mathbb{E}[\chi] = \sum_{u_1,\ldots,u_k} 2^T \Pr(u_1) \cdots \Pr(u_k) f(u_1, \ldots, u_k) \left| \sum_{x,y} \prod_i \frac{A(S_i)_{x_i,y_i,u_i}}{\sqrt{\Pr(u_i)}2^{|x_i|/2}} \right|^2 \tag{3.13}$$

$$= \sum_{u_1,\ldots,u_k} f(u_1, \ldots, u_k) \left| \sum_{x,y} \prod_i A(S_i)_{x_i,y_i,u_i} \right|^2 \tag{3.14}$$

$$= \sum_{u_1,\ldots,u_k} f(u_1, \ldots, u_k) |A(V)_{u_1,\ldots,u_k}|^2 \tag{3.15}$$

$$= \pi(C, f). \tag{3.16}$$

Also, we have $\chi \in [-2^{2T}, 2^{2T}]$ because $\zeta \in [-2^T, 2^T]$. Hence, by Corollary 2.2, we can repeat this process for $2^{O(T)}/O(\epsilon^2)$ times to approximate $\pi(C, f)$ with high probability.

In each round, we need to select a setting of Pauli measurements uniformly at random for all crossing edges. Then call the oracle k times. In the i^{th} time, input a $(\vec{d}(S_i) + |x_i|)$-qubit quantum circuit of S_i, with an additional one-qubit gate putting on each qubit before measurement to make sure Pauli measurements are transformed to the computational basis measurement. Note that for each incoming edge of S_i, we input two qubits of $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. For the first qubit, we just put a U and measure in the 0/1 basis. Because

$$(U \otimes I)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = (I \otimes U^{\mathsf{T}})\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{3.17}$$

We can simulate the first qubit classically: to guess the result of 0/1 and then put U^{T} to the second qubit according to the result. Hence we only need one qubit input for each incoming edge which means that this is at most $\vec{d}$-qubit quantum circuit.

In total, we need to call the oracle $2^{O(T)}O(\frac{k}{\epsilon^2})$ times, each time run a $\vec{d}$-qubit quantum circuit with at most $O(L + \vec{d})$ gates. This results in a $(2^{O(T)}O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$-simulator for (C, f). □

To tackle the undirected edges, we replace them by directed edge based on introducing some small subsets of the graph, which gives us the following theorem.

Theorem 3.3. *Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of the nodes of G (without any further restrictions as in Lemma 3.2). If $\mathcal{P}$ has maximal total degree d^* (see Definition 2.11) and T edges, then there is a $(2^{O(T)}O(\frac{k}{\epsilon^2}), d^*, \epsilon)$-simulator of (C, f).*

Proof. Compared to Lemma 3.2, we now have some undirected edges between different subsets. For each two-qubit gate corresponding to nodes i_k, j_t which belong to different subsets, we add a new subset S' consisting of only i_k, j_t and remove them from the original subset. Then, there are only directed edges between different subsets again. Because this modification turns one undirected edge into four directed edges crossing the subsets, the total number of crossing edges still remains $O(T)$.

Each new subset that consists of only two nodes we can simulate by classical processing when call the quantum oracle. And $\vec{d}(S_i)$ now becomes $\vec{d}(S_i) + d(S_i)$ because of adding one directed outgoing edge when removing one undirected edge. Then, we can apply Lemma 3.2 which finishes the proof. □

Attention that for removing each undireted edge, related subset will add one outgoing edge and one incoming edge. Acutally, because these two edges has no correlation, we can recycle this qubit by measuring it in the outgoing edge and then input the same qubit into the corresponding incoming edge. The recycling approach is a standard approach to optimize the number of qubits needed for a circuit [PWD16] [PP00]. Here we introduce the definition of the recycle-simulator.

Definition 3.4 (Simulator with recycling)**.** *Let (C, f) be a QC algorithm (see Definition 2.5) where C is an n-qubit quantum circuit with L gates. Then a* (Q, m, ϵ)-simulator with recycling *of (C, f) is a classical* $\mathrm{poly}(Q, n, L)$*-time algorithm that has access to a quantum oracle for at most Q calls.*

For each call, the quantum oracle has initially $|0^m\rangle$ resources and run $O(L + m)$ steps. Each step, we can apply a one-qubit or two-qubit gate to some qubits, or measure some qubit in $0/1$ basis and recycle it as a new initial qubit $|0\rangle$. Finally, the quantum oracle would output the results of all measurement. The cost of a call would be the number of steps it needed.

Such simulator has accuracy ϵ for simulating (C, f) if its output $\theta \in \mathbb{R}$ satisfies:

$$\Pr[|\theta - \pi(C, f)| < \epsilon] > \frac{2}{3}. \tag{3.18}$$

Then, there is a stronger version of Theorem 3.3 if we permit recycle approach.

Corollary 3.5. *Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of the nodes of G. If $\mathcal{P}$ has maximal directed degree $\vec{d}$ and T edges, then (C, f) has a $(2^{O(T)}O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$-simulator with recycling.*

Proof. For each subset which adds an incoming edge and an outgoing edge because of removing some node from it, we can recycle the qubit as the input of the incoming edge after measuring the qubit of the outgoing edge. That means, $\vec{d}(S_i)$ would remain. So there is a $(2^{O(T)}O(\frac{k}{\epsilon^2}), \vec{d}, \epsilon)$-recycle-simulator of (C, f). □

Remark: Compare to Lemma 3.2, it implies that increasing number of crossing undirected edges would not increase the number of qubits we need, but exponentially increase the number of calls (increase the number of edges T of $\mathcal{P}$).

3.2 Improvement assuming a decomposition of the classical function

After getting Corollary 3.5, an immediate thought is to limit the scale of piece to $\log(n)$ then find some results of classical simulating of quantum circuit.

Corollary 3.6. *Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C where C is an n-qubit quantum circuit with L gates. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of the nodes of G. If $\mathcal{P}$ has maximal directed degree $\vec{d} = O(log(n))$ and T edges, there exists a $(2^{O(T)}O(\frac{1}{\epsilon^2}), 0, \epsilon)$-simulator of (C, f), i.e., a classical simulator.*

Proof. Because $\vec{d} = O(log(n))$, we can simulate the quantum oracle of the simulator classically within time $poly(L, n)$. So it's equivalent to removing quantum oracle. And then we can remove factor k from $2^{O(T)}O(\frac{k}{\epsilon^2})$ because here it only relates to the classical running time $\mathrm{poly}(2^{O(T)}O(1/\epsilon^2), n, L)$. □

An interesting comparison is to Markov and Shi [MS08] which related to the contracting complexity of the tensor network:

Theorem 3.7 ([MS08])**.** *Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C. Assume the contracting complexity of G is $\mathrm{cc}(G)$. By using post-selection and apply Theorem 2.20, C can be classically produced within $\mathrm{poly}(2^{O(\mathrm{cc}(G))}Ln)$ time where L is the number of gates in C, n is the number of qubits of C. Then, by Corollary 2.7, there is a $(2^{O(\mathrm{cc}(G))}O(\frac{1}{\epsilon^2}), 0, \epsilon)$-simulator of (C, f).*

Remark: Because if each node of G has constant degree, contrating complexity $\mathrm{cc}(G) = O(\mathrm{treewidth}(G))$, the result above is also often represented by treewidth instead of contracting complexity.

For classical simulation, our result shows some deficiency because contrating complexity is usually smaller than the number of edges of a graph. But with ansistance of small quantum computer, our results may show progress because the number of crossing edges decreases when the scale of subests grows.

In addition, if the classical function f of a QC model (C, f) satisfies some low-rank decomposition property, we can improve our results with exponential growth of contracting complexity instead of the total number of edges. The basic idea is to get the classical approximate description of each tensor of piece instead of the quantum encoding, then contract them together from some optimal contracting order [MS08]. Here we fisrt introduce a lemma about error analysis of approximate tensor contraction.

Lemma 3.8. *Let $(G, \mathcal{A})$ be an abstract tensor network where $G = (V, E)$. Denote $|V| = n$. Index all nodes from 1 to n. Assume that for any subset S of nodes of G, the tensor $A(S)$ obtained by contracting nodes in S into one node has all entry with absolute value at most one, i.e., $|A(S)_i| \leq 1$ for any $S \subseteq V$ and $i \in \{0,1\}^{d(S)}$. Let $\mathcal{B}$ be another tensor network that is entry-wise ϵ-close to $\mathcal{A}$, i.e., for all $v \in V$ and $i \in \{0,1\}^{d(v)}$,*

$$|A(v)_i - B(v)_i| \leq \epsilon. \tag{3.19}$$

If $\epsilon \leq \frac{1}{n}2^{-10d}$ where d denotes the maximal degree of nodes in G, then

$$|A(V) - B(V)| \leq n2^{4d}\epsilon. \tag{3.20}$$

Proof. We denote the bias tensor network $e(v)_i = B(v)_i - A(v)_i$, so $|e(v)_i| \leq \epsilon$. Then we have

$$|B(V) - A(V)| \tag{3.21}$$

$$= \left| \sum_{s\in\{0,1\}^E} \prod_{v\in V} \left(A(v)_{s_{a(v)}} + e(v)_{s_{a(v)}}\right) - \sum_{s\in\{0,1\}^E} \prod_{v\in V} A(v)_{s_{a(v)}} \right| \tag{3.22}$$

$$\tag{3.23}$$

Now, we could assign each node v with tensor $A(v)$ or $e(v)$. If we enumerate the set S which is assigned by $A(v)$, then

$$|B(V) - A(V)| \leq \left| \sum_{S\subseteq V, S\neq V} \sum_{s\in\{0,1\}^E} \left(\prod_{v\in S} A(v)_{s_{a(v)}} \cdot \prod_{v\in V-S} e(v)_{s_{a(v)}} \right) \right| \tag{3.24}$$

$$\tag{3.25}$$

Contract nodes in S as a whole, denote $E'(S) = E - \{(u, v) \in E | u \in S, v \in S\}$, then

$$|B(V) - A(V)| \leq \sum_{S\subseteq V, S\neq V} \sum_{s\in\{0,1\}^{E'(S)}} \left(|A(S)_{s_{a(S)}}| \cdot \left| \prod_{v\in V-S} e(v)_{s_{a(v)}} \right| \right) \tag{3.26}$$

$$\leq \sum_{S\subseteq V, S\neq V} \sum_{s\in\{0,1\}^{E'(S)}} \left| \prod_{v\in V-S} e(v)_{s_{a(v)}} \right| \qquad |A(S)_i| \leq 1 \tag{3.27}$$

$$\leq \sum_{S\subseteq V, S\neq \Phi} \sum_{s\in\{0,1\}^{E'(S)}} \epsilon^{|V-S|} \tag{3.28}$$

Because $|E'(S)| \leq d \cdot |V - S|$, where d is the maximal degree, then

$$|B(V) - A(V)| \leq \sum_{S \subseteq V, S \neq V} 2^{d|V-S|} \epsilon^{|V-S|} \leq \sum_{k=1}^{n} \binom{n}{k} (2^d \epsilon)^k \leq \sum_{k=1}^{n} \frac{(n2^d\epsilon)^k}{k!}. \tag{3.29}$$

Because $n2^d\epsilon \leq 1$, there is

$$|B(V) - A(V)| \leq \sum_{k=1}^{n} \frac{(n2^d\epsilon)^k}{2^{k-1}} \leq \sum_{k=1}^{n} \frac{(n2^d\epsilon)}{2^{k-1}} \leq 2(n2^d\epsilon) \leq n2^{d+1}\epsilon \leq n2^{4d}\epsilon. \tag{3.30}$$

□

Following we give the formal description of the theorem. The key step is to get the classical information of each tensor of piece based on the condition that f can be decomposed. This can be done by generating mutiple copy of quantum encoding and then measure them. The obstacle before is that we cannot implement post-selection for output results so that we cannot get multiple copy of the quantum encoding when given output results.

Theorem 3.9. *Let (C, f) be a QC algorithm and $(G, \mathcal{A})$ be the tensor network of C where $G = (V, E)$. Given a partition $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ of nodes V. Assume P has the maximal total degree d^*. Denote an induced graph G^* which induced from contracting each S_i to one node and removing the final output edges. Denote by $\mathrm{cc}(G^*)$ as the contracting complexity of G^* (see Definition 2.19).*

If S_i has l_i edges for final outputs, i.e., $l_1 + \cdots + l_k = n$, let $u_i \in \{0,1\}^{l_i}$ denote their indices. If the classical polynomial function f can be decomposed in the following form:

$$f(u_1, u_2, \ldots, u_k) = \sum_{i=1}^{r} f_1^i(u_1) f_2^i(u_2) \cdots f_k^i(u_k) \tag{3.31}$$

where $f_j^i : \{0,1\}^{l_i} \to [0,1]$ is also classical polynomial function for all $j \in [k]$ and $i \in [r]$, there is a $\left(poly(n+L, r) 2^{O(\mathrm{cc}(G^*))} O(\frac{1}{\epsilon^2}), d^*, \epsilon\right)$-simulator of (C, f) if $\epsilon \leq 2^{-O(\mathrm{cc}(G^*))} r$.

Proof. For simplification, we firstly compute $f_1^1(u_1) f_2^1(u_2) \cdots f_k^1(u_k)$. Rewrite it as following form (without ambiguity, we rewrite f^1 to f):

$$f(u_1, \ldots, u_k) = f_1(u_1) f_2(u_2) \ldots f_k(u_k). \tag{3.32}$$

And similarly, we remove undirected edges like in Theorem 3.3. Because new subsets are added, assume the number of party now is K (add S_i for $i > k$). It's natural to generalize $(u_1, \ldots, u_k)$ to $(u_1, \ldots, u_K)$ such that $\forall i > k, u_i = \Phi$. And also, we suppose $f_i(u_i) = 1$ for $i > k$. Hence

$$f(u_1, \ldots, u_k) = f_1(u_1) f_2(u_2) \ldots f_K(u_K). \tag{3.33}$$

Assume we get the quantum encoding $|A(S_i)\rangle_{u_i}$ for any $i \in [K]$ after running the quantum circuit for each part and measure the final output qubits to get u_i. Then from the quantum encoding, we can get the value of the tensor netowrk $(G', \mathcal{A}')$ they induces, which has T' edges and K nodes. From Eq. (3.7), there is

$$T(G', \mathcal{A}') \tag{3.34}$$

$$= \frac{1}{4^{|T'|}} \sum_{p \in \{I,X,Y,Z\}^{T'}} (-1)^{|p|_Y} 2^{|T'|} \prod_{j \in [K]} \langle A(S_j)_{u_j} | \sigma_{p_{a(S_j)}} | \mathcal{A}(S_j)_{u_j} \rangle. \tag{3.35}$$

$$= \sum_{p \in \{I,X,Y,Z\}^{T'}} \frac{1}{2^{|T'|}} \prod_{j \in [K]} i^{|p_{a(S_j)}|_Y} \langle \mathcal{A}(S_j)_{u_i} | \sigma_{p_{a(S_j)}} | \mathcal{A}(S_j)_{u_j} \rangle \tag{3.36}$$

$$\tag{3.37}$$

where $i = \sqrt{-1}$.

From Eq. (3.13), we know

$$\pi(C,f) \tag{3.38}$$

$$= \sum_u 2^{|T'|} f(u_1,\dots,u_K)\Pr(u_1)\dots\Pr(u_K)T(G',\mathcal{A}') \tag{3.39}$$

$$= \sum_u 2^{|T'|} f(u_1,\dots,u_K)\Pr(u_1)\dots\Pr(u_K)\left(\sum_{p\in\{I,X,Y,Z\}^{T'}} \frac{1}{2^{|T'|}}\prod_{j\in[K]} i^{|p_{a(S_j)}|_Y}\langle\mathcal{A}(S_j)_{u_i}|\sigma_{p_{a(S_j)}}|\mathcal{A}(S_j)_{u_j}\rangle\right) \tag{3.40}$$

$$= \sum_{p\in\{I,X,Y,Z\}^{T'}}\prod_{j\in[K]}\sum_{u_j}\Pr(u_j) i^{|p_{a(S_j)}|_Y}\langle\mathcal{A}(S_j)_{u_i}|\sigma_{p_{a(S_j)}}|\mathcal{A}(S_j)_{u_j}\rangle f(u_j). \tag{3.41}$$

So we denote a set of new tensor $\mathcal{B} = \{B(S_i)\}$ on the graph G' such that

$$B(S_j)_{p_{a(S_j)}} \tag{3.42}$$

$$= \sum_{u_j}\Pr(u_j) i^{|p_{a(S_j)}|_Y}\langle A(S_j)_{u_i}|\sigma_{p_{a(S_j)}}|A(S_j)_{u_j}\rangle f(u_j) \tag{3.43}$$

$$= \mathbb{E}_{u_j} i^{|p_{a(S_j)}|_Y}\langle A(S_j)_{u_i}|\sigma_{p_{a(S_j)}}|A(S_j)_{u_j}\rangle f(u_j). \tag{3.44}$$

where the indices of edge comes from $\{I,X,Y,Z\}$. Then, we rewrite $\pi(C,f)$,

$$\pi(C,f) = \sum_{p\in\{I,X,Y,Z\}^{T'}}\prod_{j\in[K]} B(S_j)_{p_{a(S_j)}} = \mathcal{B}(V'). \tag{3.45}$$

Hence we can compute $\pi(C,f)$ by contracting $(G',\mathcal{B})$.

Assume in G', the degree of S_j is at most d. For each element $B(S_j)_{p_{a(S_j)}}$, attention that it's the expectation of some complex number which has norm 1 and comes from the experiment that measures the quantum encoding of $A(S_j)$. Assume we do M experiments for each element's real part and image part respectively to get tensor set $\mathcal{B}'$. Then, for fix $B(S_j)_{p_{a(S_j)}}$, from the Lemma 2.1, with probability $1-4e^{-\frac{M\epsilon^2}{8}}$,

$$|\operatorname{Re}(B(S_j)_{p_{a(S_j)}} - B'(S_j)_{p_{a(S_j)}})| \le \frac{\epsilon}{2}, |\operatorname{Im}(B(S_j)_{p_{a(S_j)}} - B'(S_j)_{p_{a(S_j)}})| \le \frac{\epsilon}{2}. \tag{3.46}$$

So that

$$|B(S_j)_{p_{a(S_j)}} - B'(S_j)_{p_{a(S_j)}}| \le \epsilon. \tag{3.47}$$

Then, with $2M4^dK$ times experiments in total, with prob $(1-4e^{-\frac{M\epsilon^2}{8}})^{4^dk}$, for any $B(S_j)_{p_{a(S_j)}}$, there is

$$|B(S_j)_{p_{a(S_j)}} - B'(S_j)_{p_{a(S_j)}}| \le \epsilon.$$

Also we know, for contracting any subset S of $\mathcal{B}$, there is

$$B(S)_{p_{a(S)}} = \mathbb{E}_{u_S} i^{|p_{a(S)}|_Y}\langle A(S)_{u_S}|\sigma_{p_{a(S)}}|A(S)_{u_S}\rangle f(u_S). \tag{3.48}$$

Hence $|B(S)_{p_{a(S)}}| \le 1$.

Let $\epsilon \le 2^{-10d}\frac{1}{K}$, $M = 100\frac{Kdr}{\epsilon^2}$.

Now, we can apply Lemma 3.8, within $2^{O(\mathrm{cc}(G'))}K$ time, to output the value of $\mathcal{B}'(V)$ such that

$$|\mathcal{B}'(V) - \mathcal{B}(V)| = |\mathcal{B}'(V) - \pi(C,f)| \le K2^{4d}\epsilon. \tag{3.49}$$

Let $\mu = K2^{4d}\epsilon r$, then $\mu \le 2^{-6d}r$. We enumerate $1 \le i \le r$ to compute each decomposition function of f. Then with probability

$$(1-4e^{-\frac{M\epsilon^2}{8}})^{4^dKr} \ge 1 - 4^{d+1}Kre^{-\frac{M\epsilon^2}{8}} \ge 1 - 2^{2d+2}Kre^{-M\epsilon^2/8} \ge 1 - e^{-M\epsilon^2/8+2d+2+\ln K+\ln r} \ge \frac{2}{3}, \tag{3.50}$$

the final output has error $r \cdot K 2^{4d} \epsilon = \mu$.

And the times we need to call oracle is the number of experiments:

$$2M4^d Kr = 2^{O(d)} O(1/\mu^2) \operatorname{poly}(K, r). \tag{3.51}$$

d is the number of max degree in G', so will be not large than $\mathrm{cc}(G')$. And K is the number of parties, should be $O(n+L)$. Compare to G^* which not remove undirected edges, $\mathrm{cc}(G') \leq O(\mathrm{cc}(G^*))$, because we can firstly contract new subsets to old subsets and increase the degree of original subsets at most constant multiple. So the number of calls should be less than

$$\operatorname{poly}(n+L, r) 2^{O(\mathrm{cc}(G^*))} O\left(\frac{1}{\mu^2}\right). \tag{3.52}$$

So finally, there is a $(\operatorname{poly}(n+L,r)2^{O(\mathrm{cc}(G^*))}O(\frac{1}{\epsilon^2}), d^*, \mu)$-simulator of (C, f) if $\mu \leq 2^{-O(\mathrm{cc}(G^*))} r$. □

Similarly, we can use the approach of recycle to decrease the number of qubit we need for oracle.

Corollary 3.10. *Given the condition of Theorem 3.9. Assume P has maximal directed degree $\vec{d}$, there is a $(poly(n+L,r)2^{O(\mathrm{cc}(G^*))}O(\frac{1}{\epsilon^2}), \vec{d}, \epsilon)$-recycle-simulator of (C, f) if $\epsilon \leq 2^{-O(\mathrm{cc}(G^*))} r$.*

Then, we can limit the scale to $\log(n)$ to get the following classical simulation results.

Corollary 3.11. *Given the condition of Corollary 3.10. If $\vec{d} \leq O(\log(n))$, there is a classical simulator for (C, f) which has parameters $(poly(n+L,r)2^{O(\mathrm{cc}(G^*))}O(\frac{1}{\epsilon^2}), 0, \epsilon)$ if $\epsilon \leq 2^{-O(\mathrm{cc}(G^*))} r$.*

So if $r = \operatorname{poly}(L+n)$, this result partially implies Theorem 3.7. An open question is whether or not we can remove the limitation of low-rank decomposition of f. It seems that we need some power of post-selection to achieve this. But we have to mention that low-rank decomposition function has already covered many interesting classes people are interested. For example, compute the probability of outputing 1 of some qubits.

4 Applications

Following we give two possible applications of our results and approaches.

First one is about evaluating an abstract tensor network. We generalize the conclusion of Section 3 into more general tensor network instead of only quantum circuits. Corollary 4.1 provides a framework to evaluate the value of tensor network with free edges if we can obtain the quantum encoding of small pieces from some oracle, which coming from the generalization of Lemma 3.2 by replacing the quantum gate by general tensor. Corollary 4.2 is slight different from Corollary 4.1 by removing free edges of overall network, which can be regarded as a generalization of Theorem 3.9 where f is low-rank decomposed to be embedded into each piece.

Second application is about an n-qubit 2D circuit where $n = x \cdot y$ arranged in x rows and y columns. Assume this circuit has low connectivity between rows and relaxing constraints of applying gates to neighbor between columns. By partitioning each row into one party, it's natural to apply Corollary 3.5 and Corollary 3.10 (when f is low-rank decomposed) to the 2D circuit. Then we can get Corollary 4.3 and Corollary 4.4, respectively, which both claim y qubits is enough to simulate the whole one.

4.1 Evaluating an abstract tensor network

Evaluating abstract tensor networks is an important problem in several different fields, including statistical physics and machine learning. An interesting question is thus whether a small quantum device can help with this task. Based on our results, we provide two general approaches for computing the value of an abstract tensor network on a small device.

The first one comes from a generalization of Lemma 3.2, which replaces quantum gates by general tensors.

Corollary 4.1. *Given an abstract tensor network $(G, \mathcal{A})$ whose graph $G = (V, E)$ has n free edges indexed by u (recall that G has only undirected edges), and given a classical function $f : \{0,1\}^n \to [0,1]$, denote the value of $(G, \mathcal{A})$ as*

$$T(G, \mathcal{A}) := \sum_u |A(V)_u|^2 \cdot f(u). \tag{4.1}$$

Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of V. Denote by T the number of edges of $\mathcal{P}$, which excludes the free edges. Assume a d-qubit quantum machine can produce the quantum encoding $|A(S_i)\rangle$ for the tensor $\frac{A(S_i)}{|A(S_i)|}$ in time L.

Then in time $k2^{O(T)}O\left(\frac{\prod_i |A(S_i)|^2 L}{\epsilon^2}\right)$ and using a d-qubit and L-gate quantum oracle, we can output a random number ζ such that

$$\mathbb{E}[|\zeta - T(G, A)| < \epsilon] > \frac{2}{3}. \tag{4.2}$$

If G has no free edges, we can generalize the results from Theorem 3.9.

Corollary 4.2. *Given an abstract tensor network $(G, \mathcal{A})$ and a partition $\mathcal{P} = \{S_1, \ldots, S_k\}$ of its nodes, assume that, for each of the tensors $\frac{A(S_i)}{|A(S_i)|}$, a d-qubit quantum machine can produce its quantum encoding $|A(S_i)\rangle$ in time L. Denote by G^* the graph induced by $\mathcal{P}$. Then, with assistance of a d-qubit and L-gate quantum oracle, we can output a random number ζ in time $k2^{O(\mathrm{cc}(G^*))}O\left(\frac{\prod_i |A(S_i)|^2 L}{\epsilon^2}\right)$, such that*

$$\mathbb{E}[|\zeta - T(G, A)| < \epsilon] > \frac{2}{3}. \tag{4.3}$$

The approach of Arad and Landau [AL10] can be regarded as a way to get the quantum encoding of a general tensor by using a so-called bubbling process. Their approach may require a large number of ancillas, comparable to the number of nodes for each party at worst, and experience the difficulty of exponential growth related to the tensor norm. Theorems 3.3 and 3.9 can be regarded as an application of the general framework to a special case: the quantum circuit case. By utilizing the properties of quantum circuits themselves, we can decrease the number of qubits needed to get the quantum encoding, while keeping the norm of the system small. An interesting question is to find other applications of tensor networks to apply these two types of general simulation frameworks.

4.2 2D circuits with low connectivity

Low-dimensional quantum circuits have been studied for a long time. Jozsa [Joz06] has shown that a 1-D circuit with $\log(n)$ depth can be simulated classically. However, general approaches for simulating 2D circuits are not known. Here we apply our results to 2D circuits under the constraint that their connectivity is low along one of the dimensions.

Firstly, we apply our Corollary 3.5 to the 2D version.

Corollary 4.3. *Let (C, f) be a QC algorithm which has $n = x \cdot y$ qubits indexed by (i, j) where $i \in [x], j \in [y]$, i.e., arranged in x rows and y columns. Assume that for any two-qubit gate applied to (i_1, j_1), (i_2, j_2), we have $|i_1 - i_2| \leq 1$. Assume the number of gates which cross different rows is T. Then (C, f) has a $\left(2^{O(T)}O(\frac{x}{\epsilon^2}), y, \epsilon\right)$-simulator with recycling.*

For example, if $x = y$ and $T = \log(x)$, we can use a $\sqrt{n}$-qubit machine to simulate the n-qubit 2D circuit in polynomial time.

If f has a low-rank decomposition, we can apply Corollary 3.10 to this circuit and get following result.

Corollary 4.4. *Let (C, f) be a QC algorithm that satisfies the conditions in Corollary 4.3 where C has L gates. Assume for any $i \in [x-1]$, the number of gates which connect the i^{th} and the $(i+1)^{th}$ row is at most T'. Denote the indices of the l^{th} output qubit by u_l. Assume f can be decomposed as follows:*

$$f(u_1, \ldots, u_k) = \sum_{i=1}^{r} f_1^i(u_1) f_2^i(u_2) \cdots f_k^i(u_k). \tag{4.4}$$

Then (C, f) has a $\left(\mathrm{poly}(x + L, r)2^{O(T')}O(\frac{x}{\epsilon^2}), y, \epsilon\right)$-simulator with recycling if $\epsilon \leq 2^{-O(T')}r$.

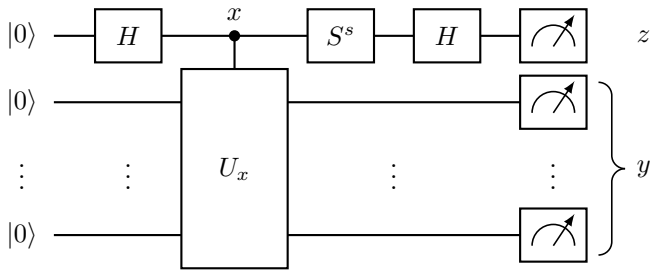

Figure 7: Procedure for simulating the real and imaginary parts of $\langle 0^n|U_0^\dagger|y\rangle\langle y|U_1|0^n\rangle$, where U_0 and U_1 are two n-qubit quantum circuits and $y \in \{0,1\}^n$ is a random string obtained by measuring the last n qubits in the computational basis. It uses the following single-qubit gates: $H := \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)/\sqrt{2}$ and $S := \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$. We denote this circuit by $C_s(U_0, U_1)$ where $s \in \{0,1\}$ (with $0 \equiv$ Re and $1 \equiv$ Im).

Proof. Note that after contracting each row to one node, the circuit becomes a 1-D circuit. The degree of each node is at most T', which implies the contraction complexity is at most $O(T')$. □

If $r = \text{poly}(x + L), x = y, T' = \log(x)$, we can simulate this 2D circuit on a $\sqrt{n}$-qubit quantum machine.

Finally, if instead of qubits we have qutrits and each two-qutrit gate is the special gate Ω defined in Appendix A, we can improve the simulation even further. Namely, instead of taking time $O(y)$ to prepare the quantum encoding of the tensor corresponding to each row of the network, we could instead apply all $O(y)$ two-qubit gates in parallel and prepare the encoding in constant quantum time.

These results show that a small quantum machine can provide help for 2D circuits with low connectivity along one dimension. An interesting open question is whether can we find more circuit classes which can be essentially simulated by small quantum machines. Furthermore, one may characterize the hierarchy of different circuit classes by the minimal quantum resources they need.

5 An alternative approach based on [BSS16]

In this section, we will introduce Theorem 5.3: an alternative proof for slightly different version of Corollary 3.5, based on cutting gates in the quantum circuit instead of thinking it as cutting edge in a tensor network. It provides some advantage in the perceptual intuition and removing the requirement of recycle, but lacking the ability to generalize like tensor network.

The idea can be regarded as a generalization of the approach in [BSS16], which is based on the decomposition of quantum gates. Compare to [BSS16], it improves in two aspects. Firstly transfer the payoff on qubits to the payoff on the number of entangled gates, which makes the limitation of costant-qubit party to arbitary large. It is an important progress which makes the partition more flexible. On the other aspect, we permit the small amount of qubit transformation between different party instead of just cutting gate. This makes the cutting solution involve not only qubit space but also evolution of time.

Lemma 5.1 is a lemma which was originally shown in [BSS16]. We slightly change the approach to make it more unify and easy to generalize. Lemma 5.2 tackles with crossing gates connect to different pieces: isolates each piece based on gate decomposition and computes each piece by Lemma 5.1. Then Theorem 5.3 generalizes Lemma 5.2 to cover directed edges, with replacing directed edges by SWAP gates and ancillas.

Lemma 5.1 ([BSS16])**.** *Given two n-qubit circuits that implement $U_0, U_1 \in \mathrm{U}(2^n)$, there exists an $(n+1)$-qubit quantum algorithm that outputs a pair of random variables (ζ, Y) such that for any n-bit string y,*

$$\mathbb{E}[\zeta|Y = y] \cdot \Pr(Y = y) \equiv \langle 0|U_0^+|y\rangle\langle y|U_1|0\rangle. \tag{5.1}$$

Proof. For $s \in \{0,1\}$, let $C_s(U_0, U_1)$ denote the circuit shown in Fig. 7. We pick $s \in \{0,1\}$ uniformly at random and execute $C_s(U_0, U_1)$. We denote the measurement outcomes obtained at the end of the circuit by $y \in \{0,1\}^n$ and $z \in \{0,1\}$, see Fig. 7. Let ζ be the following function of s and z:

$$\zeta(s, z) := 2i^s(-1)^{s+z}. \tag{5.2}$$

Hence the algorithm produces two random variables: ζ and $Y := y$.

Let us analyze the expectation of (ζ, Y). It is easy to verify that the quantum state produced by $C_s(U_0, U_1)$ before the final measurement is

$$|\phi_s\rangle = \frac{1}{2}\left(|0\rangle\left(U_0 + i^s U_1\right)|0\rangle + |1\rangle\left(U_0 - i^s U_1\right)|0\rangle\right). \tag{5.3}$$

Therefore,

$$\begin{aligned}
&\mathbb{E}(\zeta|Y=y)\cdot\Pr(Y=y) && (5.4)\\
&= \sum_{s,z\in\{0,1\}} \Pr(s,z,y)\zeta(s,z) && (5.5)\\
&= \frac{1}{2}\left(2|\langle 0y|\phi_0\rangle|^2 - 2|\langle 1y|\phi_0\rangle|^2 - 2i|\langle 0y|\phi_1\rangle|^2 + 2i|\langle 1y|\phi_1\rangle|^2\right) && (5.6)\\
&= \frac{1}{4}\left(\langle 0|(U_0^\dagger + U_1^\dagger)|y\rangle\langle y|(U_0+U_1)|0\rangle - \langle 0|(U_0^\dagger - U_1^\dagger)|y\rangle\langle y|(U_0-U_1)|0\rangle\right) && (5.7)\\
&- \frac{1}{4}i\left(\langle 0|(U_0^\dagger - iU_1^\dagger)|y\rangle\langle y|(U_0+iU_1)|0\rangle - \langle 0|(U_0^\dagger + iU_1^\dagger)|y\rangle\langle y|(U_0-iU_1)|0\rangle\right) && (5.8)\\
&= \mathrm{Re}(\langle 0|U_0^\dagger|y\rangle\langle y|U_1|0\rangle) + \mathrm{Im}(\langle 0|U_0^\dagger|y\rangle\langle y|U_1|0\rangle) && (5.9)\\
&= \langle 0|U_0^\dagger|y\rangle\langle y|U_1|0\rangle. && (5.10)
\end{aligned}$$

□

Then, we introduce the gate decomposition approach of multi-party. The idea is to fix decomposition parameters and apply Lemma 5.1 to each party, then combine them together. The reason it can improve the results of [BSS16] is because it measures the results of each party instead of enumerating all results of one party.

Lemma 5.2. *Let (C, f) be a QC algorithm and (G, A) be the tensor network of C where $G = (V, E)$. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of nodes V such that there are only undirected edges connecting different subsets. If $\mathcal{P}$ has degree $d := \vec{d}(\mathcal{P})$ (which in this case is just the maximum number of final outputs of some partition) and T edges, then there is a $(2^{O(T+k)}O(\frac{k}{\epsilon^2}), d+1, \epsilon)$-simulator of (C, f).*

Proof. Because each crossing two-qubits gate U_i is decomposed as following form (see Eq. (2.14)):

$$U_i = \sum_{\alpha_i=0}^{15} c_{\alpha_i} V_{\alpha_i} \otimes W_{\alpha_i}, \tag{5.11}$$

We can decompose the whole circuit C into linear combination of tensor of k parties:

$$C = \sum_{\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_T\}} c_\alpha V_\alpha^1 \otimes V_\alpha^2 \ldots \otimes V_\alpha^k \tag{5.12}$$

Each V_α^i is an unitary matrix which correspondings to S_i and applies to quantum system not larger than m. Because there is only undirected edge between different subsets, all T edges represent cutting of two-qubit gates.

And then we rewrite $\pi(C, f)$:

$$\begin{aligned}
&\pi(C,f)\\
&= \sum_{y\in\{0,1\}^{n+m}} \langle 0^{n+m}|C^\dagger|y\rangle\langle y|C|0^{n+m}\rangle f(y)\\
&= \sum_{y_1,\ldots,y_k} \langle 0^n|\sum_\alpha c_\alpha^* V_\alpha^{1\dagger}\otimes\ldots\otimes V_\alpha^{k\dagger}|y_1\rangle\langle y_1|\otimes\ldots\otimes|y_k\rangle\langle y_k|\sum_\beta c_\beta V_\beta^1\otimes\ldots\otimes V_\beta^k|0^n\rangle f(y_1,\ldots,y_k) \qquad (5.13)\\
&= \sum_{\alpha,\beta} c_\alpha^* c_\beta \sum_{y_1,..y_k} \langle 0|V_\alpha^{1\dagger}|y_1\rangle\langle y_1|V_\beta|0\rangle\cdot\ldots\cdot\langle 0|V_\alpha^{k\dagger}|y_k\rangle\langle y_k|V_\beta^k|0\rangle f(y_1,\ldots,y_k).
\end{aligned}$$

Recall that $c_{\alpha_i} \geq 0, \sum_{\alpha_i} c_{\alpha_i}^2 = 1$, so $c_\alpha = \prod_i c_{\alpha_i}$ also satisfy $c_\alpha \geq 0, \sum_\alpha c_\alpha^2 = 1$. Denote

$$S = \sum_\alpha c_\alpha. \tag{5.14}$$

Then

$$\pi(C,f) = \sum_{\alpha,\beta} \frac{c_\alpha}{S}\frac{c_\beta}{S} \sum_{y_1,\ldots,y_k} \langle 0|V_\alpha^{1\dagger}|y_1\rangle\langle y_1|V_\beta|0\rangle \cdot \ldots \cdot \langle 0|V_\alpha^{k\dagger}|y_k\rangle\langle y_k|V_\beta^k|0\rangle \cdot S^2 \cdot f(y_1,\ldots,y_k). \tag{5.15}$$

Our algorithm works as following: firstly randomly pick α, β with probability $\frac{c_\alpha}{S}, \frac{c_\beta}{S}$. Then by applying Lemma 5.1, we constrcut k circuits where i^{th} circuit given input V_α^i, V_β^i outputs random varible (ζ_i, y_i). Finally output the random variable σ as following:

$$\sigma = Re(\prod_{i=1}^k \zeta_i \cdot f(y_1,\ldots,y_k) \cdot S^2). \tag{5.16}$$

Now we prove that $E[\sigma] = \pi(C,f)$.

$$\begin{aligned}
&E[\sigma]\\
=&Re\left(\sum_{\alpha,\beta}\frac{c_\alpha}{S}\frac{c_\beta}{S}\sum_{\zeta_i,y_i}\left(\prod_{i=1}^k Pr(\zeta_i,y_i)\cdot\zeta_i\right)\cdot f(y_1,\ldots,y_k)\cdot S^2\right)\\
=&Re\left(\sum_{\alpha,\beta}\frac{c_\alpha}{S}\frac{c_\beta}{S}\sum_{y_i}\left(\prod_{i=1}^k\sum_{\zeta_i} Pr(\zeta_i,y_i)\cdot\zeta_i\right)\cdot f(y_1,\ldots,y_k)\cdot S^2\right)\\
=&Re\left(\sum_{\alpha,\beta}\frac{c_\alpha}{S}\frac{c_\beta}{S}\sum_{y_i}\left(\prod_{i=1}^k\langle 0|V_\alpha^{i\dagger}|y_i\rangle\langle y_i|V_\beta^i|0\rangle\right)\cdot f(y_1,\ldots,y_k)\cdot S^2\right)\\
=&Re(\pi(C,f))\\
=&\pi(C,f).
\end{aligned} \tag{5.17}$$

Also, it's easy to see that σ is bounded:

$$|\sigma| \leq 2^k S^2 \leq 2^{k+4T}. \tag{5.18}$$

where $\sum_{\alpha=1}^{16^T} c_\alpha^2 = 1$ induce $\sum_\alpha c_\alpha = S \leq 4^T$.

By Corollary 2.2, the number of experiments we need to achieve ϵ error is $2^{O(k+T)}O(1/\epsilon^2)$. Each experiment, we need call k times quantum oracle, to run a $d+1$ quantum circuit. Then, there is a $(2^{O(k+T)}O(\frac{k}{\epsilon^2}), d+1, \epsilon)$-simulator of (C,f). □

Now we apply Lemma 5.2 to prove a slightly weak version of Corollary 3.5. The slight difference is that we use the $d+1$-qubit non-recycle simulator to replace exactly d-qubit recycle simulator, with paying $2^{O(k+T)}$ cost in quantum calls instead of $2^{O(T)}$.

Theorem 5.3. *Let (C,f) be a QC algorithm and (G,A) be the tensor network of C where $G=(V,E)$. Let $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ be a partition of nodes V. If $\mathcal{P}$ has degree $d := \vec{d}(\mathcal{P})$ and T edges, then there is a $(2^{O(T+k)}O(\frac{k}{\epsilon^2}), d+1, \epsilon)$-simulator of (C,f).*

Proof. Our approach is to replace the directed edge (qubit transmitting) between subests by SWAP gate. For each directed edge connect from subset S_i to S_j, we add an ancilla qubit in S_j and then replace the edge by a SWAP gate. The new tensor network is equivalent to original one.

Finishing the process, the total number of crossing edges is still T, and each subset S_i now have $\vec{d}(S_i)$ qubits to run because of adding ancilla qubits. By lemma 5.2, there is a $(2^{O(T+k)}O(\frac{k}{\epsilon^2}), d+1, \epsilon)$-simulator of (C,f). □

Acknowledgements

MO acknowledges Leverhulme Trust Early Career Fellowship (ECF-2015-256) for financial support and MIT. TP acknowledges support from the Top Open program in Tsinghua University, China. Most of this work was done while MO and TP were visiting the Center for Theoretical Physics at MIT; we acknowledge CTP for its hospitality.

References

[AAEL07] Dorit Aharonov, Itai Arad, Elad Eban, and Zeph Landau. Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane. 2007. arXiv:quant-ph/0702008.

[AJL09] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the Jones polynomial. *Algorithmica*, 55(3):395–421, 2009. arXiv:quant-ph/0511096, doi:10.1007/s00453-008-9168-0.

[AL10] Itai Arad and Zeph Landau. Quantum computation and the evaluation of tensor networks. *SIAM Journal on Computing*, 39(7):3089–3121, 2010. arXiv:0805.0040, doi:10.1137/080739379.

[BB02] Jean-Luc Brylinski and Ranee Brylinski. Universal quantum gates. In Goong Chen and Ranee K. Brylinski, editors, *Mathematics of Quantum Computation*, Computational Mathematics, chapter 4. CRC Press, 2002. URL: https://books.google.com/books?id=evPKBQAAQBAJ&pg=PA101, arXiv:quant-ph/0108062.

[BDD+02] Michael J. Bremner, Christopher M. Dawson, Jennifer L. Dodd, Alexei Gilchrist, Aram W. Harrow, Duncan Mortimer, Michael A. Nielsen, and Tobias J. Osborne. Practical scheme for quantum computation with any two-qubit entangling gate. *Phys. Rev. Lett.*, 89(24):247902, Nov 2002. arXiv:quant-ph/0207072, doi:10.1103/PhysRevLett.89.247902.

[BSL+16] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A.G. Fowler, B. Campbell, Yu Chen, et al. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, 2016. arXiv:1511.03316, doi:10.1038/nature17658.

[BSS16] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Phys. Rev. X*, 6(2):021043, Jun 2016. arXiv:1506.01396, doi:10.1103/PhysRevX.6.021043.

[FKW02] H. Michael Freedman, Alexei Kitaev, and Zhenghan Wang. Simulation of topological field theories by quantum computers. *Communications in Mathematical Physics*, 227(3):587–603, 2002. arXiv:quant-ph/0001071, doi:10.1007/s002200200635.

[FLW02] H. Michael Freedman, Michael Larsen, and Zhenghan Wang. A modular functor which is universal for quantum computation. *Communications in Mathematical Physics*, 227(3):605–622, 2002. arXiv:quant-ph/0001108, doi:10.1007/s002200200645.

[GAL+15] Dardo Goyeneche, Daniel Alsina, José I. Latorre, Arnau Riera, and Karol Życzkowski. Absolutely maximally entangled states, combinatorial designs, and multiunitary matrices. *Phys. Rev. A*, 92(3):032316, Sep 2015. arXiv:1506.08857, doi:10.1103/PhysRevA.92.032316.

[GE07] David Gross and Jens Eisert. Novel schemes for measurement-based quantum computation. *Phys. Rev. Lett.*, 98(22):220503, May 2007. arXiv:quant-ph/0609149, doi:10.1103/PhysRevLett.98.220503.

[GESPG07] David Gross, Jens Eisert, Norbert Schuch, and David Pérez-García. Measurement-based quantum computation beyond the one-way model. *Phys. Rev. A*, 76(5):052315, Nov 2007. arXiv:0706.3401, doi:10.1103/PhysRevA.76.052315.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.2307/2282952.

[IBM] IBM Quantum Experience. URL: http://www.research.ibm.com/quantum/.

[Joz06] Richard Jozsa. On the simulation of quantum circuits. 2006. arXiv:quant-ph/0603163.

[MNM+16] Thomas Monz, Daniel Nigg, Esteban A. Martinez, Matthias F. Brandl, Philipp Schindler, Richard Rines, Shannon X. Wang, Isaac L. Chuang, and Rainer Blatt. Realization of a scalable Shor algorithm. *Science*, 351(6277):1068–1070, 2016. arXiv:1507.08852, doi:10.1126/science.aad9480.

[MS08] Igor L. Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008. arXiv:quant-ph/0511069, doi:10.1137/050644756.

[PGVWC07] David Pérez-García, Frank Verstraete, Michael M. Wolf, and J. Ignacio Cirac. Matrix product state representations. *Quantum Information & Computation*, 7(5&6):401–430, Jul 2007. URL: http://www.rintonpress.com/journals/qiconline.html#v7n56, arXiv:quant-ph/0608197.

[PP00] S. Parker and M. B. Plenio. Efficient factorization with a single pure qubit and log$N$ mixed qubits. *Phys. Rev. Lett.*, 85:3049–3052, Oct 2000. URL: http://link.aps.org/doi/10.1103/PhysRevLett.85.3049, doi:10.1103/PhysRevLett.85.3049.

[PWD16] A. Paler, R. Wille, and S. J. Devitt. Wire Recycling for Quantum Circuit Optimization. *ArXiv e-prints*, September 2016. arXiv:1609.00803.

[SDV06] Yao-Yun Shi, Lu-Ming Duan, and Guifré Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Phys. Rev. A*, 74(2):022320, Aug 2006. arXiv:quant-ph/0511070, doi:10.1103/PhysRevA.74.022320.

[Vid03] Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.*, 91(14):147902, Oct 2003. arXiv:quant-ph/0301063, doi:10.1103/PhysRevLett.91.147902.

[Vid04] Guifré Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett.*, 93(4):040502, Jul 2004. URL: quant-ph/0310089, doi:10.1103/PhysRevLett.93.040502.

[Vid08] Guifré Vidal. Class of quantum many-body states that can be efficiently simulated. *Phys. Rev. Lett.*, 101(11):110501, Sep 2008. arXiv:quant-ph/0610099, doi:10.1103/PhysRevLett.101.110501.

[VMC08] Frank Verstraete, Valentin Murg, and J. Ignacio Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008. arXiv:0907.2796, doi:10.1080/14789940801912366.

A Example of a universal gate set with no temporal order

In this appendix we describe a universal gate set, for a qutrit-based quantum computer, with a peculiar feature: none of the gates have designated input and output wires. Our construction is based on a certain 4-qutrit *absolutely maximally entangled* (AME) state [GAL+15].

The (unnormalized) form of the state in question is as follows:

$$|\Omega\rangle := \sum_{i,j=0}^{2} |i\rangle|j\rangle|i+j\rangle|i+2j\rangle, \tag{A.1}$$

where the arithmetic on indices is performed modulo 3. This means that $|\Omega\rangle$ is maximally entangled across any bipartition of the four qutrits into two groups of two. Equivalently, if any two qutrits are measured in the standard basis, the state of the other two can be uniquely recovered from the measurement outcomes. Indeed, if we identify the two variables by vectors in $\mathbb{Z}_3^2$ via $i \equiv (1,0)$ and $j \equiv (0,1)$, this is equivalent to saying that any pair of rows of

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \tag{A.2}$$

are linearly independent over $\mathbb{Z}_3$, which one can easily check.

One way of exploiting the properties of the state $|\Omega\rangle$ is by considering any of the six possible matrices obtained by turning any two of the kets in Eq. (A.1) into bras. For example, consider the following 9×9 matrix:

$$\Omega := \sum_{i,j=0}^{2} |i+j\rangle|i+2j\rangle\langle i|\langle j|. \tag{A.3}$$

It is not hard to convince oneself that this is a permutation matrix (indeed, from i and j one can uniquely determine $i+j$ and $i+2j$ and vice versa), hence we can treat Ω as a two-qutrit unitary gate.

As a quantum gate, Ω has the important property of being *entangling*, i.e., it can produce an entangled state out of two product states. This is most easily seen from the following example:

$$\Omega\left(\frac{|0\rangle + |1\rangle + |2\rangle}{\sqrt{3}} \otimes |0\rangle\right) = \frac{|00\rangle + |11\rangle + |22\rangle}{\sqrt{3}}. \tag{A.4}$$

From this observation and [BB02, BDD+02] we can conclude that Ω is universal.

Claim A.1. *The two-qutrit gate Ω together with all single-qutrit gates is universal for quantum computation.*

This means that without loss of generality we can consider quantum circuits based only on the gate set $\{\Omega\} \cup \mathrm{U}(3)$. This has interesting consequences for quantum circuit simulation. Since any two indices of the 4-tensor Ω can be considered as inputs and the other two as outputs, there is much more freedom in terms of the order in which we can contract a tensor network or execute a quantum circuit that we want to simulate. We provide an example to illustrate this in more detail.

Example A.2. *Assume the rectangle of the 2D grid circuit shown in Fig. 1 has height x and width y. Then the total number of tensors is $O(xy)$ and the total number of free edges is $O(x+y)$. Since each free edge is either an input or output (and the number of inputs and outputs has to be the same) the total amount of memory required to implement this circuit is $O(x+y)$, irrespectively of which edges we consider as inputs and hence irrespectively also of the order in which we implement the gates.*

Note that for the purpose of preparing a quantum encoding of the reduced tensor, which is obtained by contracting all edges, we can choose the inputs and outputs arbitrarily. However, it is clear that considering the left-ward pointing edges as inputs and the right-ward pointing ones as outputs, the number of time-steps necessary for implementing the circuit is $O(y)$. However, if we instead consider the upward pointing edges as inputs and the downward pointing ones as outputs, the number of time-steps is $O(x)$. Note that in this case we would be implementing $O(y)$ two-qutrit gates in parallel at each time-step, as opposed to $O(x)$.

The important point is that if $x \ll y$, it is much more advantageous to evaluate the tensor network from top to bottom rather than from left to right. Note that we have this freedom only thanks to the properties of the gate Ω. In particular, if we do not work in the tensor network framework but rather directly with the circuit, we cannot take advantage of this effect.