Simulating quantum circuits using tree tensor networks

Philipp Seitz[1], Ismael Medina[2], Esther Cruz[3], Qunsheng Huang[1], and Christian B. Mendl[1]

[1]Technical University of Munich, Department of Informatics, Boltzmannstraße 3, 85748 Garching, Germany
[2]University of Göttingen, Campus Institute Data Science
[3]Max-Planck-Institute of Quantum Optics, Hans-Kopfermann-Straße 1, 85748 Garching, Germany
Wednesday 22nd March, 2023

We develop and analyze a method for simulating quantum circuits on classical computers by representing quantum states as rooted tree tensor networks. Our algorithm first determines a suitable, fixed tree structure adapted to the expected entanglement generated by the quantum circuit. The gates are sequentially applied to the tree by absorbing single-qubit gates into leaf nodes, splitting two-qubit gates via singular value decomposition and threading the resulting virtual bond through the tree. We theoretically analyze the applicability of the method as well as its computational cost and memory requirements and identify advantageous scenarios in terms of required bond dimensions as compared to a matrix product state representation. The study is complemented by numerical experiments for different quantum circuit layouts up to 37 qubits.

1 Introduction

Tensor networks provide a well-established framework and method for analyzing and simulating strongly correlated quantum systems [1–3]. Recently, these methods have been adapted to circuit-based (digital) quantum computers, either by representing the statevector as a matrix product state (MPS) [4], or interpreting the overall quantum circuit as a tensor network. Such networks are contracted with the aid of heuristics to obtain a good contraction order and "Feynman-simulator"-type delayed contractions [5–10].

In this work, we aim to combine the advantages of both approaches by representing statevectors as tree tensor networks (TTNs). The properties and capabilities of such tree-type tensor networks have been studied in detail in the past. In particular, the "multi-scale entanglement renormalization ansatz" (MERA) [11, 12] can efficiently describe critical ground states of one-dimensional systems. The use of TTNs to simulate strongly correlated chemistry systems [13–17],

Philipp Seitz: philipp.seitz@tum.de
Ismael Medina: ismael.medina@cs.uni-goettingen.de
Esther Cruz: esther.cruz@mpq.mpg.de
Qunsheng Huang: keefe.huang@tum.de
Christian B. Mendl: christian.mendl@tum.de

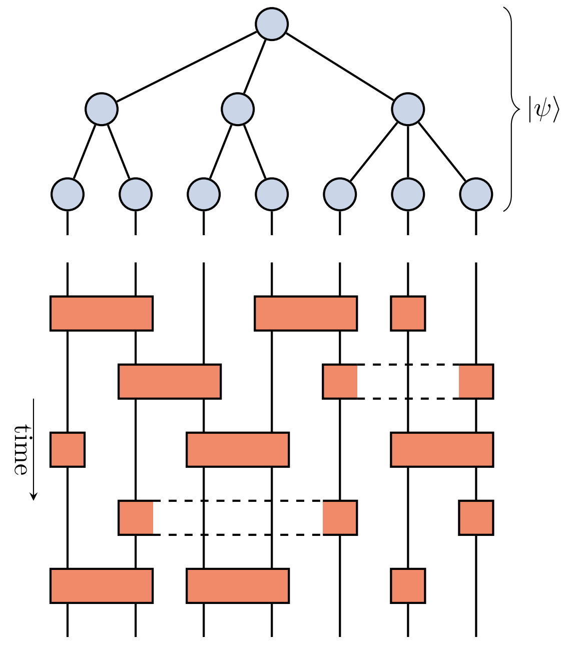


Figure 1: Principal algorithmic paradigm in this work: an initial N-qubit quantum state $|\psi\rangle$ represented as tree tensor network, to which the gates of a quantum circuit are applied while preserving the tree structure.

two-dimensional quantum systems [18], and tensor differential equations [19] has been investigated, but a dedicated study and analysis in the domain of digital quantum computers are still missing. A similar study on hybrid tensor networks (including classical and quantum tensors) is done in [20] which includes some aspects of hybrid TTNs. Ref. [21] studies the entanglement properties of Shor's algorithm on tree tensor networks specifically. Many useful properties of TTNs have been studied in the context of many-body physics [14]. Optimizing tree layouts based on the system characteristics is just one of these, which we aim to apply to circuit simulation.

In general, TTNs share important advantages with MPSs [22]: TTNs are efficiently contractable, incurring only polynomial cost on a classical computer for tensor nodes with bounded dimensions. As a consequence, one can extract many quantities of physical interest from a quantum state in a TTN representation, for example, two-point (static) correlation functions, and one can draw unbiased (Born) probability samples from them [23]. We note, however, that the *distance* between any pair of leaf nodes in a TTN, defined as the number of nodes traversed along the shortest path between the two chosen nodes, scales as $\mathcal{O}(\log(N))$, where N denotes the overall number of lattice sites or qubits in the circuit. This compares favorably to the MPS representation, where the distance between leaf nodes scales as $\mathcal{O}(N)$. Since connected correlation functions (i.e., covariances) typically decay exponentially with path length within a tensor network, TTNs can capture longer-range correlations as compared to MPS [13]. This property becomes relevant for quantum circuits with multi-qubit gates acting on "distant" qubits, or, in other words, in cases where one cannot devise a canonical linear ordering of qubits in which gates act solely in local neighborhoods. More concretely, we will identify scenarios where a tree representation provides a genuine advantage over an MPS in terms of the scaling of required internal bond dimensions, see Sect. 3.4 below. In terms of tensor networks, reduced bond dimensions directly translate to improved contraction and simulation efficiency.

The method studied in this work can be classified as a statevector-based quantum circuit simulator and is summarized in Fig. 1: a quantum state $|\psi\rangle$ (typically starting as a product state, e.g., $|0\rangle^{\otimes n}$) is represented as a TTN with an advantageous tree structure. The gates of the circuit are applied to $|\psi\rangle$ sequentially by absorbing them into the tree, as described in Sect. 2.2.

Definitions and conventions. We consider qubits throughout to simplify the exposition, but the methods presented here are generalizable to "qudits". We also assume that the circuit contains solely single- and two-qubit gates, with the remark that our algorithm can be extended to handle higher-order gates as well.

We say that a tree tensor network is of *canonical form* (i.e., orthonormalized) if each node in the tree obeys the condition in Fig. 2, i.e., contracting a node tensor with its complex conjugate along its downstream legs gives the identity map. Regarding the root node of the tree, one can formally attach a dummy upward leg with dimension 1 to enforce this requirement. With this property, a quantum state represented as a TTN is then normalized.

2 Algorithm

Our principal algorithm takes a quantum circuit description and a specification of the initial quantum state as inputs. The algorithm consists of two phases, as illustrated in Fig. 1:

1. Expressing the initial quantum state as a TTN by mapping logical qubits to leaf nodes and identifying an advantageous rooted tree graph (i.e., connectivity of nodes) based on the given circuit.

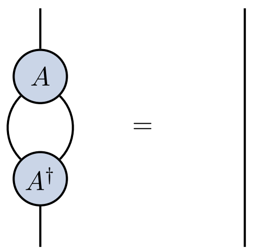


Figure 2: Orthonormalization property of a tree node; the tree tensor network is of the canonical form if all nodes in the tree have this property.

2. Sequentially applying quantum circuit gates to the tree while preserving its graph structure, interleaved with re-orthonormalization of the tree tensor network.

To ensure that the tree tensor network can be simulated on a classical computer, we require that each edge dimension is bounded by a chosen constant $D_{\max}$. Naturally, this restricts the set of quantum circuits that can be simulated by our approach. We will study how the $D_{\max}$ restriction is reflected in the feasible pattern of two-qubit gates in more detail in Sect. 3, and define the algorithm to determine a tree structure in the upcoming section.

2.1 Tree structure search

In the first phase, the initial quantum state is converted to a TTN. Each logical qubit is mapped to a leaf node, which we group into subtrees, generating the tree-like structure.

Finding a truly optimal tree structure for a given problem depends on the problem statement [18]. A similar technique in [24] uses TTNs in a variational way to find ground states of given Hamiltonians. They consider each node as a physical site, whereas in our case the physical sites live on the leaf nodes. Instead of direct optimization, we employ a heuristic to find a tree topology that performs well in a variety of cases. That being said, the topology optimizations discussed in [15, 24] and our approach are similar in philosophy. In the case of [24], they compute the mutual information and entanglement within different orbitals to arrive at a good tree architecture. In our case, the heuristic used to create the tree structure also looks at how the entanglement is distributed within the different physical parties, according to the gates to be applied. Thus our approach is an alternative to the mentioned ones for the case of circuit simulation.

The heuristic is based on the insight that the maximal required dimension of internal edges can be upper-bounded a priori based on the to-be-applied circuit gates. Single-qubit gates can be applied without increasing edge dimensions, hence, we focus solely on the effect of two-qubit gates.

Similar to the case in an MPS, the application of a two-qubit gate increases the edge dimension between the corresponding leaf nodes and each intermediate node along the path that connects them, which we refer to as *threading* the bond wire through the TTN. As visualized in Fig. 3, edge dimensions are increased by a factor of k, which can be determined by a singular value decomposition (SVD). For the CNOT gate, $k = 2$, while in general $k = 4$, as in the case of the fSIM gate [25].

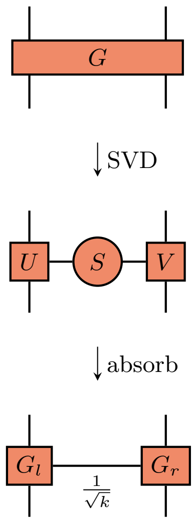


Figure 3: Splitting of a two-qubit gate via SVD. The diagonal matrix S contains the singular values. For the last step, one may absorb $\sqrt{S}$ both into U and V. k denotes the number of non-zero singular values, and the factor $1/\sqrt{k}$ serves as normalization when incorporating the connecting bond into the tree tensor network (see Fig. 4 below).

Thus, selecting subtrees of qubits that are highly entangled by the simulated circuit reduces the number of two-qubit gates spanning different subtrees, effectively increasing admissible circuit depth under the $D_{\max}$ restriction [26]. Additionally, threading connections through the tree incurs the overhead of renormalization of all nodes on the connecting path.

We identify such subtrees of qubits via the custom similarity function

$$s_{\text{qc}}(q_i, q_j) = |G(q_i) \cap G(q_j)| + \frac{1}{|G(q_i)| + |G(q_j)|} \quad (1)$$

for $i \neq j$ and $|G(q_i)| = \sharp(\text{2-qubit gates})$ on qubit q_i, where the first summand ranks qubit pairs with their entanglement and the second summand acts as a tie-breaker as it is < 1. A generic clustering algorithm uses s_{qc} to find similar-sized clusters over all qubits. For simplicity, subtrees are constructed bottom-up for each identified cluster, putting all qubit pairs with the same similarity value under a common node. Instead, one could recursively cluster all subtrees to a certain level and fall back to the bottom-up construction as a basis case. Similar ideas have been explored in [24, 27]. [27] construct adaptive-weighted binary TTNs bottom-up based on entanglement, which can be interpreted as a similarity function. One core difference is that our construction is not restricted to binary trees. For quantum chemistry, the *coordination number* of molecules can be used similarity function [24].

One insight from early prototypes is the advantage of similar-sized subtrees under the root node, compared to skewed structures. Favoring this pattern leads to similar subtree heights, and, consequently, ensures that the final tree structure is (almost) balanced. Thus, this is enforced in Algorithm 1, which shows the pseudocode for the first phase of our algorithm.

Algorithm 1: Tree structure search

```
1  Function find_tree_structure(qc, c):
       Data: qc = $Quantum\ Circuit$, c = $Number\ of\ clusters$
       Result: Tree structure
2      similarity ← similarity_matrix(qc.$gates$)  // $s_{\text{qc}}$
3      cluster_labels ← cluster(similarity, c)  // any clustering, e.g., SpectralClustering
4      cluster_roots ← $\emptyset$
5      foreach label in cluster_labels do
6          cluster_roots ← cluster_roots ∪ create_subtree(qc(label))
7      end
       /* $Node(children)$ creates a new subtree and builds the tree bottom up. */
8      return Node(cluster_roots)
9  Function create_subtree(qubits):
       Data: qubits = $Qubits$
       Result: Cluster
10     seen ← $\emptyset$
11     children ← $\emptyset$
12     pairs ← sort(Pair(qubits), $most\ similar$)
13     sim ← similarity(pairs[0][0], pairs[0][1])
       // similarity is based on Eq. (1)
14     foreach $q_0, q_1$ in pairs do
15         if $q_0 \notin$ seen then
16             seen ← seen ∪ $\{q_0\}$
17             children ← children ∪ $\{q_0\}$
18         if $q_1 \notin$ seen then
19             seen ← seen ∪ $\{q_1\}$
20             children ← children ∪ $\{q_1\}$
21         if sim > similarity($q_0$, $q_1$) then
22             children ← {Node(children)}
23             sim ← similarity($q_0$, $q_1$)
24     end
25     return Node(children)
```

2.2 Gate application procedure

Once the structure has been fixed, the tree tensor network has to represent the initial quantum state. Typically this is a computational basis state, e.g., all qubits starting from $|0\rangle$. For a computational basis state, one can set the dimensions of all internal edges to 1 and initialize the internal tensors with the single entry 1, while the two entries of a leaf tensor (of dimension 2×1) are the basis state entries of the corresponding qubit. In other words, the initial tree can be regarded as a scaffolding to which gates are applied.

Fig. 4 uses tensor diagram notation to illustrate incorporating (or "absorbing") a two-qubit gate into the tree, utilizing the decomposition step detailed in Fig. 3. After decomposing the gate, the connecting bond wire (red) is threaded through the tree structure. The items enclosed in dotted lines define the updated tensors. Note that the new non-leaf tensors (P', Q', R' in the figure) essentially result from an outer product with the identity matrix, which can be recorded symbolically at first. In other words, one may "lazy-update" these tensors by delaying the operation until it becomes necessary. The gate application procedure also preserves the orthonormality of non-leaf nodes – this point is illustrated in Fig. 5. Nevertheless, to return to canonical form, a re-orthonormalization sweep through the tree is necessary since the leaf nodes are no longer orthonormal.

Re-orthonormalization proceeds from the leaf nodes upwards through the tree, using QR-decompositions or SVDs, as shown in Fig. 6. By using "economical" SVDs (explained later), the edge dimensions can only decrease or remain the same.

A variant of our algorithm is an approximation of the true output quantum state by retaining only the leading $D_{\max}$ singular values during the SVD-orthonormalization procedure.

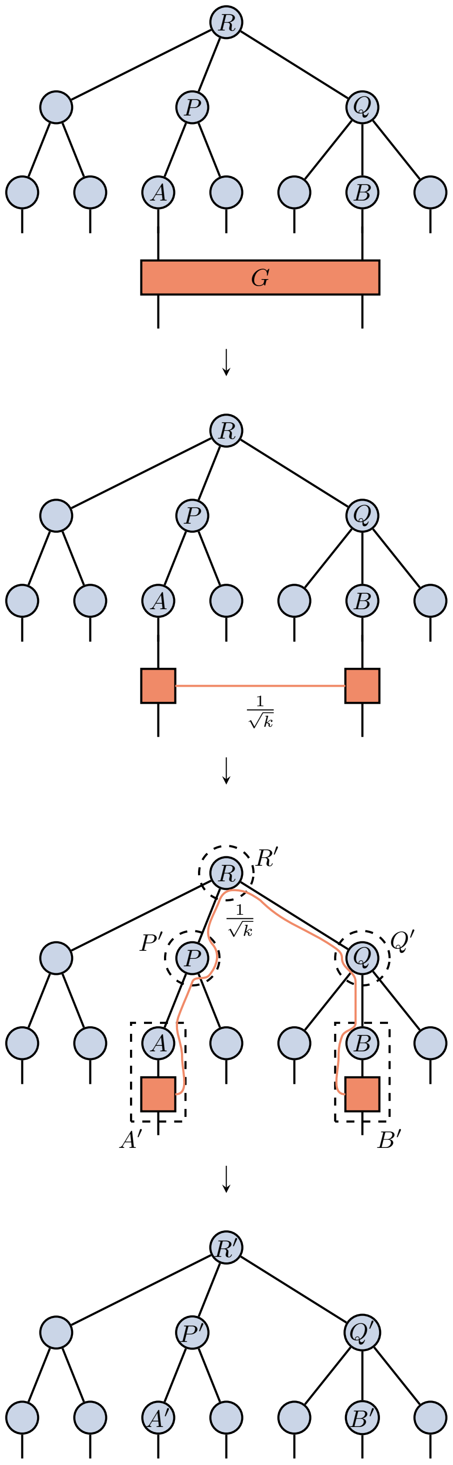


Figure 4: Applying a two-qubit gate to a quantum state represented as a tree tensor network, preserving the tree structure and orthonormalization of the non-leaf nodes. The dashed circles and rectangles define the updated tensors by the enclosed items.

3 Tree tensor network capability and algorithmic analysis

In this section, we analyze the algorithmic complexity and identify the required properties of the quantum circuit such that it can still be simulated efficiently using our approach. For specificity, our analysis refers to m-ary trees (see below), but note that the tree layout found by the tree structure search might result in more general architectures. These are mostly explored via numerical experiments in Sect. 4.

3.1 Overall architecture and assumptions

We impose the restriction that each edge dimension in the tree (i.e., dimension of any tensor leg) is bounded by a given number $D_{\max}$. To simplify the analysis, let us assume a perfect m-ary tree, i.e., each non-leaf node has m children, and that the tree is balanced. For example, a binary tree corresponds to $m = 2$. But note that the algorithm works for general rooted trees as well. $D_{\max}$ should be chosen such that storing and working with such tensors is still possible given the available computational resources, memory, and

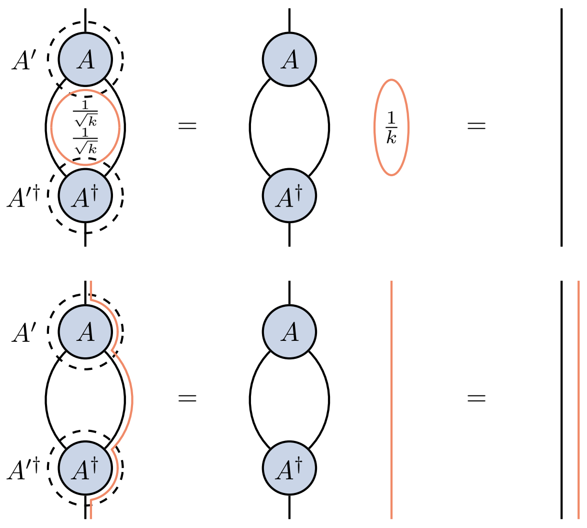


Figure 5: The canonical orthonormalization of a node tensor (see Fig. 2) is preserved when updating internal nodes of the tree during gate application (Fig. 4).

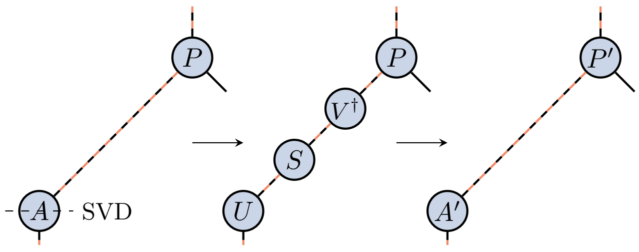


Figure 6: Orthonormalization of a leaf A in the tree via SVD: the A tensor is replaced by the isometry $A' = U$ from the SVD, and the diagonal matrix of singular values, as well as $V^\dagger$, are absorbed in the parent tensor P. The orthonormalization procedure is independent of gates as additional bonds can simply be bundled with the existing bonds indicated by the dashed line. For intermediate nodes, the procedure stays the same, but additional wires need to be bundled as well.

hardware architecture.

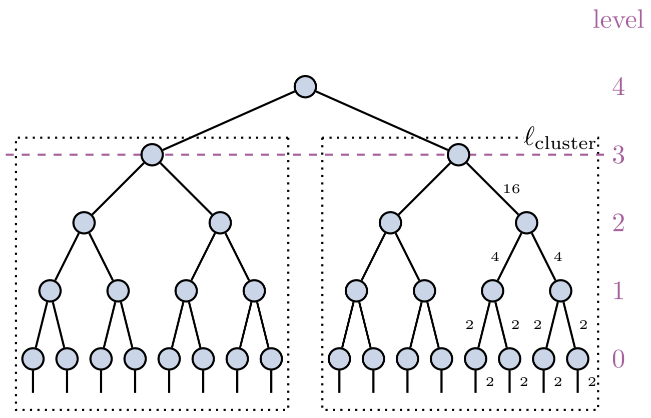


Figure 7: Maximally required edge dimensions up to level 3 for a binary tree, see Eq. (2). Assuming $D_{\max} = 16$, an arbitrary sequence of gates acting on the highlighted subtrees separately is feasible, while the number of two-qubit gates targeting one qubit from either subtree is restricted.

For the following, we first recall that an "economical" singular value decomposition of a matrix $A \in \mathbb{C}^{p\times q}$ is given by $A = USV^\dagger$, with isometries $U \in \mathbb{C}^{p\times k}$ and $V \in \mathbb{C}^{q\times k}$ where $k = \min(p, q)$, and a diagonal matrix S of singular values $\sigma_1 \geq \cdots \geq \sigma_k \geq 0$. In the context of Fig. 6, p is the product of the dimensions of the downward-pointing child connections of A. The term *child connection* is used to describe an edge between nodes from level ℓ to $\ell - 1$. Assuming that these are all equal to some integer D with $D \leq D_{\max}$, $p = D^m$. After the update, the connecting edge between A and P has dimension $k \leq p = D^m$. After an orthonormalization sweep through the tree starting from the leaf nodes, the dimensions of the child connections at level $\ell \geq 1$ are thus bounded by

$$D_\ell \leq 2^{(m^{\ell-1})}, \tag{2}$$

as illustrated in Fig. 7. We count levels starting from 0 at the leaf nodes. The base 2 in Eq. (2) stems from the dimension of a single qubit at the lowest level.

We define ℓ_{cluster} as the largest integer such that

$$2^{(m^{\ell_{\text{cluster}}-1})} \leq D_{\max}. \tag{3}$$

According to Eq. (2), the edge dimensions will never exceed $D_{\max}$ up to level ℓ_{cluster} in the tree. This implies that any gate sequence *within* a subtree of height ℓ_{cluster} is exact. In the following, we denote these subtrees as *clusters*. A cluster thus contains $m^{\ell_{\text{cluster}}}$ qubits (assuming uniform height). From another viewpoint, we can exactly represent any quantum state vector of the qubits in a cluster if it is unentangled with outside qubits.

Let us now consider the connecting edges above ℓ_{cluster} in the tree: we denote the bond dimension of a two-qubit gate G by k_G, see Fig. 3. We say that a gate *crosses* an edge e if the gate bond threaded through the tree traverses e; for example, in Fig. 4 the red curve crosses the edge between nodes P and R. The restriction imposed by $D_{\max}$ is thus certainly satisfied if

$$\prod_{G \text{ crossing } e} k_G \leq D_{\max} \tag{4}$$

for any edge e in the tree between nodes at or above level ℓ_{cluster}. In other words, the left term in Eq. (4) is an upper bound on the maximal entanglement acceptably generated by the gates crossing e. We used Eq. (4) as motivation for the similarity function Eq. (1). However, note that Eq. (4) does not consider possible simplifications in the contracted tensor structure, such as the (contrived) scenario when an arbitrary two-qubit gate G is immediately followed by $G^\dagger$, which does not increase entanglement at all. Thus, Eq. (4) is a "worst case" bound on the applicable gates.

The level of the root node, ℓ_{root}, denominates the overall height of the tree. Thus, it can represent a quantum state with up to $N = m^{\ell_{\text{root}}}$ qubits.

3.2 Computational cost and complexity analysis

3.2.1 Memory requirements

The number of nodes of the tree is less or equal to the node count of a perfect m-ary tree:

$$\sharp(\text{nodes}) \leq \sum_{\ell=0}^{\ell_{\text{root}}} m^{\ell_{\text{root}}-\ell} = \frac{m^{\ell_{\text{root}}+1}-1}{m-1}. \qquad (5)$$

As in Eq. (2), we denote the largest appearing bond dimension of the child connections at level ℓ by D_ℓ. Thus a tensor at level ℓ has at most $D_\ell^m D_{\ell+1}$ entries: m connections to its children and one connection to its parent. Consequently, an upper bound on the overall to-be-stored (complex) numbers for the tree is

$$\begin{aligned}\sharp(\text{entries})_{\text{tree}} &\leq m^{\ell_{\text{root}}} 2 D_1 \\ &+ \sum_{\ell=1}^{\ell_{\text{root}}-1} m^{\ell_{\text{root}}-\ell} D_\ell^m D_{\ell+1} + D_{\ell_{\text{root}}}^m. \end{aligned} \qquad (6)$$

By construction $D_0 = 2$ (see Fig. 7), for $1 \leq \ell \leq \ell_{\text{cluster}}$ the bound (2) holds, and $D_\ell \leq D_{\max}$ for $\ell > \ell_{\text{cluster}}$. Using Eq. (2) yields $D_\ell^m D_{\ell+1} \leq 4^{(m^\ell)}$ for $1 \leq \ell < \ell_{\text{cluster}}$. Inserting this bound into Eq. (6) leads to

$$\begin{aligned}\sharp(\text{entries})_{\text{tree}} &\leq 4\, m^{\ell_{\text{root}}} + \sum_{\ell=1}^{\ell_{\text{cluster}}-1} m^{\ell_{\text{root}}-\ell}\, 4^{(m^\ell)} \\ &+ \sum_{\ell=\ell_{\text{cluster}}}^{\ell_{\text{root}}-1} m^{\ell_{\text{root}}-\ell} D_{\max}^{m+1} + D_{\max}^m. \end{aligned} \qquad (7)$$

To arrive at a compact formula, we can use that $D_\ell \leq D_{\max}$, and thus

$$\sharp(\text{entries})_{\text{tree}} \leq \sharp(\text{nodes})\, D_{\max}^{m+1} \leq \frac{mN-1}{m-1} D_{\max}^{m+1}. \qquad (8)$$

Note that this bound is not reached because the root can only have up to $D_{\max}^m$ entries, and the leaf nodes have no children. In summary, $\sharp(\text{entries})_{\text{tree}}$ grows linearly with qubit count N (for fixed m and $D_{\max}$).

3.2.2 Gate application and orthonormalization

The orthonormalization sweep through the tree is the most computationally expensive step in the gate application procedure. In the worst case, the bond wire threaded through the tree visits all levels up to the root node. Thus $2\ell_{\text{root}}$ tensors along the path need to be re-orthonormalized. Recalling that the SVD decomposition of a $p \times q$ matrix costs $\mathcal{O}(\min(pq^2, p^2 q))$ floating-point operations, we arrive at the overall cost

$$\sharp(\text{FLOPS})_{\text{gate}} \leq \mathcal{O}\big(\ell_{\text{root}} \cdot D_{\max}^{m+2}\big) = \mathcal{O}\big(\log_m(N) \cdot D_{\max}^{m+2}\big) \qquad (9)$$

for an orthonormalization sweep after applying a gate.

3.3 Admissible gate patterns

Finally, we identify admissible gate patterns such that Eq. (4) certainly holds. To simplify the analysis, we do not take the option for truncation (omitting small singular values during orthonormalization) into account here, which could facilitate the application of additional gates while respecting the $D_{\max}$ restriction. Also, for simplicity we throughout set $k_G = 4$, the largest possible bond dimension of a two-qubit gate.

As described above, applying gates on qubits within the same cluster is unrestrained, so we only need to consider cases where the two qubits lie within different clusters.

Solving Eq. (4) for the number of gates crossing an edge e gives

$$\sharp(\text{gates crossing } e) \leq \log_4(D_{\max}) = \frac{1}{2}\log_2(D_{\max}). \qquad (10)$$

If the bond wire of a gate traverses a node above level ℓ_{cluster}, it multiplies the dimensions of two of its legs by the factor 4 (without truncation), i.e., a factor of 16 more entries is now required for this node. This gives an upper bound on the number of gates with bond wires crossing a given node A:

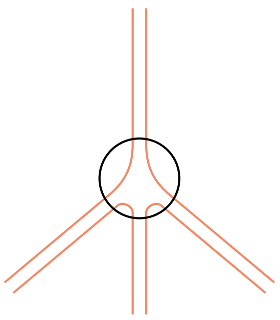
$$\sharp(\text{gates crossing } A) \leq \frac{1}{4}\log_2(D_{\max}^{m+1}) = \frac{m+1}{4}\log_2(D_{\max}). \qquad (11)$$

Fig. 8 shows admissible scenarios for $m = 3$ and $D_{\max} = 16$ and $D_{\max} = 64$, respectively.

This pattern can be applied recursively to construct a tree. Fig. 9 shows examples of the corresponding qubit connectivity and nested triangles defining the nodes on higher levels of the tree. For $m = 3$ and $D_{\max} = 16, 64$, one obtains $\ell_{\text{cluster}} = 2$, such that each cluster contains nine qubits. In the scenario of Fig. 9a, the inter-cluster bonds form a linear path visiting all clusters sequentially, whereas this is not the case in Fig. 9b.

3.4 Comparison with matrix product states

In this section, we argue that the tree representation can provide a genuine asymptotic advantage compared to an MPS: for certain circuits, the required virtual bond dimension of an MPS will diverge as the number of qubits $N \to \infty$, while the dimensions of a TTN remain bounded. To explain the argument, consider the connectivity pattern in Fig. 9b. First, note that an MPS imposes a linear ordering of the qubits and that a two-qubit gate acting on qubits i and k ($i < k$) increases the virtual bonds of all intermediate MPS tensors (sites j with $i < j < k$). Fig. 10 provides a simplified illustration of this point: the bond connecting 1 with 3 needs to be threaded through 2 in an MPS representation. For the connectivity pattern in Fig. 9b, and given an arbitrary ordering of the qubits, there exists a qubit playing the role of site 2 at each


(a) $m = 3,\ D_{\max} = 16$

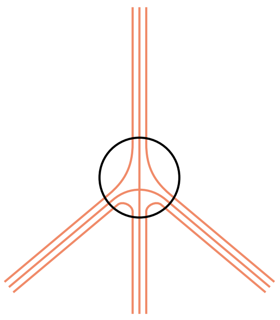

(b) $m = 3,\ D_{\max} = 64$

Figure 8: Examples of admissible threading of bond wires through a node with $m = 3$ children, such that the dimension of each edge remains bounded by $D_{\max}$. The subtrees can be connected pairwise by a gate, as well as upstream with other (more distant) qubits.

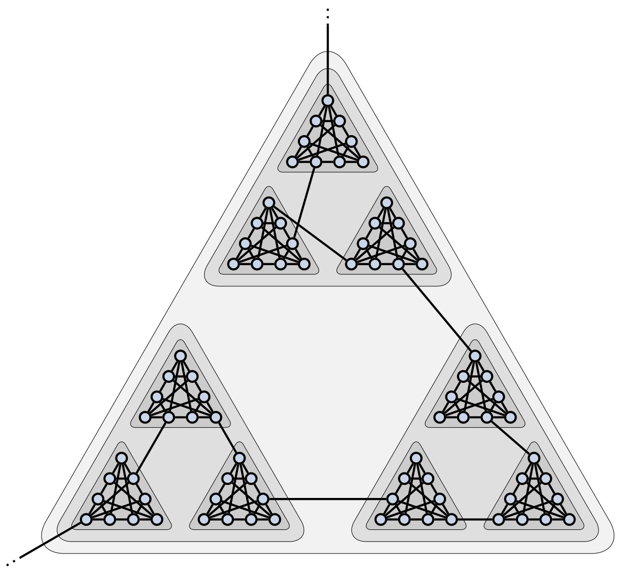
level within every triangle. Since this observation can be applied recursively to all levels, at least one of the virtual bond dimensions grows as 4^{ℓ}, with ℓ the number of levels. In contrast, using the TTN approach, a uniformly bounded bond dimension suffices.

Applied to the simulation this has the following effect. For the individual tree nodes, the cost of updating might be more expensive compared to the MPS. But this difference is offset by the bound dimensionality. The large cost of a full update is amortized by the savings of the common updates. More freely speaking, the necessity of a very expensive update is scarce in the TTN. In an MPS, the still expensive updates are common enough, to produce a disadvantage.

4 Numerical experiments

To complement the theoretical analysis in Sect. 3, we run and analyze numerical quantum circuit simulations based on the tree representation for the statevector, and compare them with a basic MPS-based simulator [1]. For our study, one-qubit gates are not considered as they do not impact the dimensionality, and the bond dimension of each gate is assumed to be $k_G = 4$ unless stated otherwise.

[1]Our implementation for the TTN and MPS simulators is available at https://github.com/Gistbatch/tree-tensor-network-simulator


(a) $m = 3,\ D_{\max} = 16$

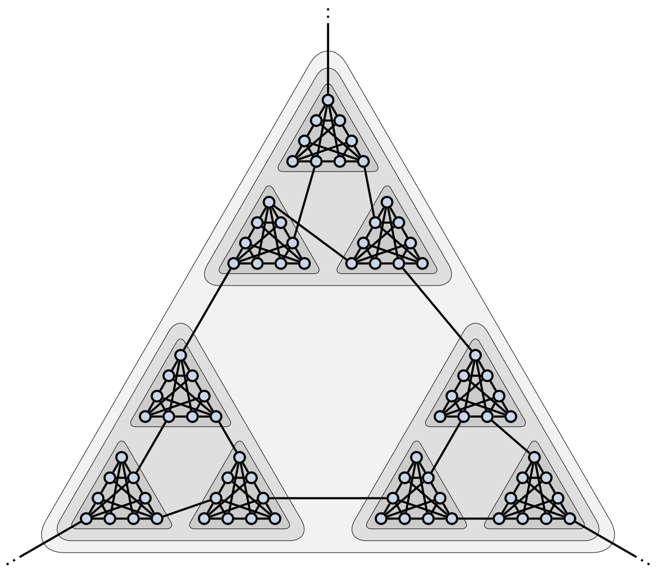

(b) $m = 3,\ D_{\max} = 64$

Figure 9: Top-down view of an admissible quantum gate pattern such that a perfect 3-ary tree representation of the output quantum state is feasible. The circles denote physical qubits, corresponding to leaf nodes of the tree, and the thick lines two-qubit gates (ordering in time not relevant). The nested triangles indicate a partitioning into subtrees, using nodes of the form shown in Fig. 8. An arbitrary number of quantum gates can act within the smallest triangles (clusters with 9 qubits), corresponding to subtrees at level $\ell_{\text{cluster}} = 2$.

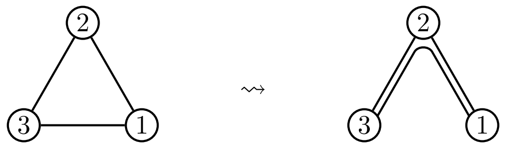


Figure 10: Pairwise connections between three qubits (or sites) require threading of one connection through an intermediate tensor in an MPS representation.

All experiments are performed on a CPU (AMD Ryzen 7 3700) with 32 GB of RAM. We explored both SVDs and QR decompositions for orthonormalization procedures, with comparable performance; the shown data uses SVDs throughout. To test our code, we simulate a locally interacting circuit on an $n \times n$ lattice (as in Fig. 11) with a depth of eight, based on Google Sycamore experiment [25]. This circuit will be referred to as a *lattice* circuit. We also design a *tree-like* circuit (as in Fig. 12) with a depth of six, such that the output state perfectly matches the tree layout, in order to showcase the potential of our simulator.

We compared the running times of the MPS and TTN simulators to validate our idea. Correctness is confirmed by contracting the networks to the full statevector and comparing them with a traditional statevector simulator.

One takeaway from this initial trial is the impact of the relation between the number of clusters and the number of leaves in a cluster. Too many clusters introduce overhead in intermediate nodes which have to be normalized each time a gate is applied. Only a few but bigger clusters require too much calculation on the individual normalization. For different purposes, these two aspects have to be fine-tuned to get the best result. Also, compared to applying gates, the structure search has no real impact on the performance.

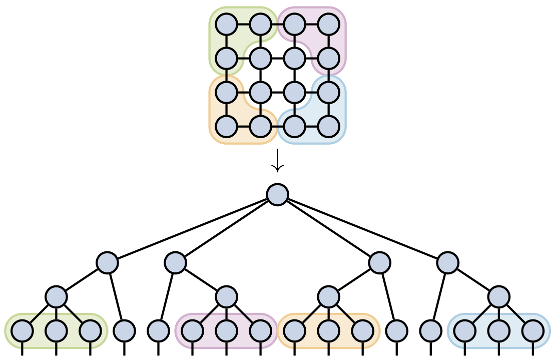


Figure 11: 4×4-qubit lattice with a nearest neighbor gate pattern, and corresponding tree architecture for representing the output quantum state.

4.1 Circuits

For our purposes, quantum circuits can be divided into two different categories, based on how well they can be clustered into a tree layout. Some circuits are not suitable due to their connectivity pattern, for example, the quantum Fourier transform with all-to-all connectivity or circuits with tight nearest-neighbor connectivity. This category is represented by the *lattice* circuit in the simulations. Fig. 11 shows the resulting tree for a 4×4 qubit lattice and gate pattern based on the Google Sycamore experiment [25] for randomized circuits. They specify a pattern of nearest neighbor gate activations alternating on each row and column, for an exact specification refer to [25]. All gates are chosen randomly from a set of predefined gates.

In some circuits, the output state almost perfectly matches the tree layout. The *tree-like* circuit showcases the potential of our simulator without depending on the recursive nature. A pattern is created from a highly entangled cluster with four qubits, which is repeated any number of times, similar to the example in Fig. 9. Gates crossing cluster boundaries are included, but kept under the $D_{\max}$ threshold by setting $k_G = 2$, only connecting to one central location. Fig. 12 shows an instance with four clusters and 17 qubits, derived from the given tree.

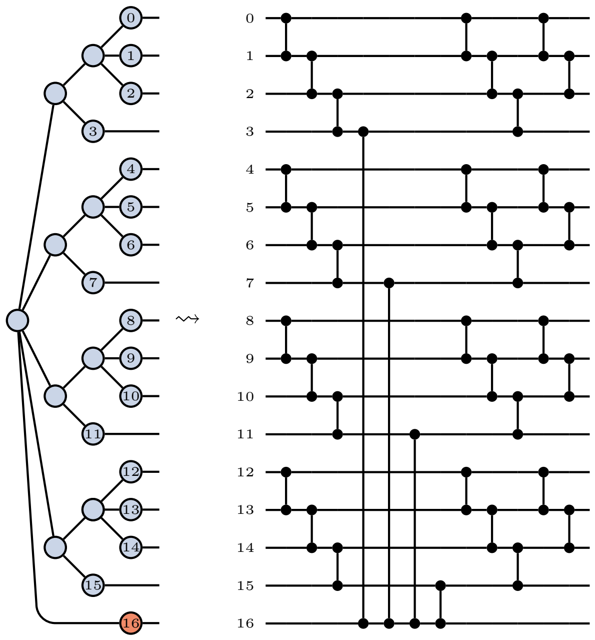


Figure 12: Well-structured circuit pattern resulting from a clusterable tree. The highlighted subtree contains all connections crossing boundaries.

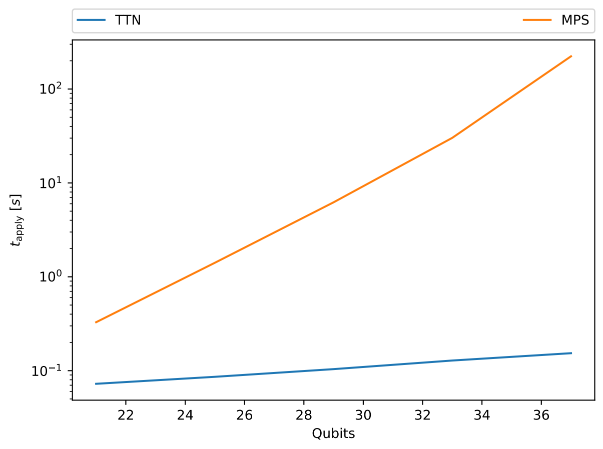


Figure 13: Wall-clock time of applying gates and re-orthonormalization, comparing a MPS with a TTN representation of the quantum state for the *tree-like* circuit.

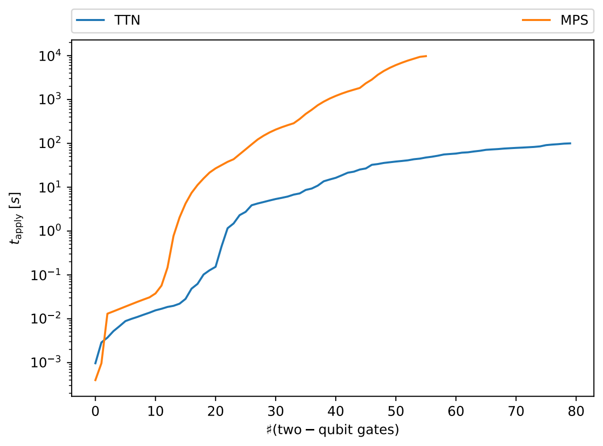


Figure 14: Accumulated wall-clock time of applying gates and re-orthonormalization, on the *lattice* circuit (for 25 qubits). For gate 57 the MPS fails to normalize.

4.2 Experiments and results

In this section, we discuss the experiments and results we achieve by running them. We also provide a rationale for why approximation techniques (based on truncating singular values) are not useful in this regime.

4.2.1 Wall-clock timing

The experiment is conducted for the *tree-like* circuit and scaled up to a maximum of 37 qubits which just fits into 256 GB of swap space. We simply measure the wall-clock time of simulating the complete circuit. The results from this experiment, shown in Fig. 13, indicate that the TTN outperforms the MPS. The memory requirements described in Sec. 3.2.1 can be fulfilled for certain tree structures. For some cluster sizes, the condition from Eq. (4) holds, which results in a decrease in internal bond dimensions. Timings are highly dependent on the real tensor structure. Our algorithm is tailored to provide a generally useful structure but this can be optimized. The implementation provided in Sec. 4 allows overwriting the structure by the user if necessary

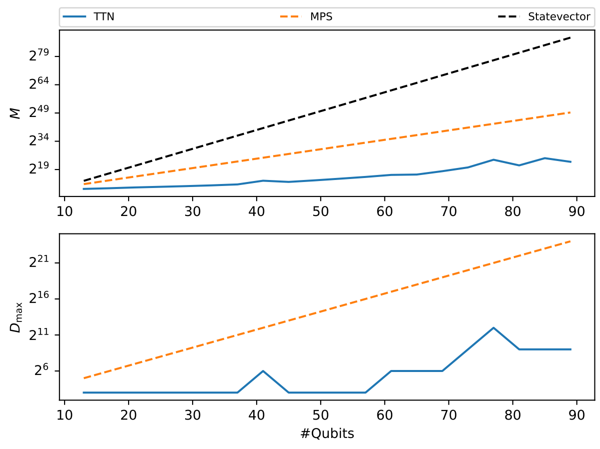
One unexpected outcome is the performance of TTNs on the *lattice* circuits. In a further trial on a 5×5 *lattice* the MPS fails to normalize some nodes. The Fortran BLAS implementation which we use for SVD and QR-decomposition does not converge due to the high internal dimensions. Fig. 14 shows the results up to the point of failure. In comparison, the TTN delivers a result in a reasonable time. A possible explanation for this could be the shorter path distance between some leaves, which means a fewer number of normalizations.

4.2.2 Scaling for a large number of qubits

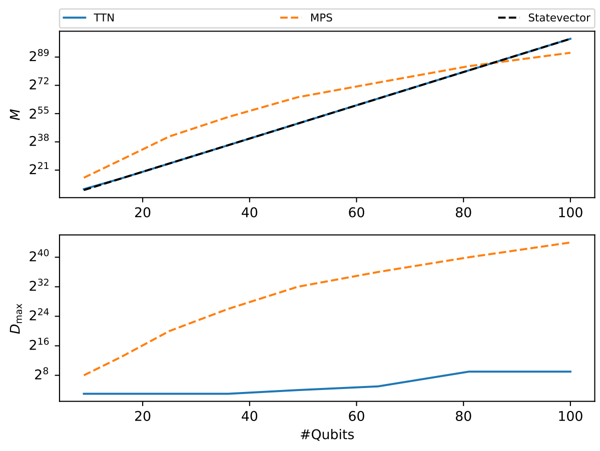
To circumvent numerical issues, we also *simulate* the gate application and orthonormalization procedure in an additional experiment. These “dry-runs” allow for calculating the internal bond dimensions without performing the actual tensor operations. This enables reasoning for qubit numbers above the classically possible threshold. For the comparison, we measure two aspects. Firstly, the time of the circuit calculation which includes finding the tree structure and applying the gates with a renormalization sweep afterward. Secondly, two metrics for bond dimensions are considered: the maximum internal bond dimension $D_{\max} = \max_{T\in\text{TN}} \max_i(\dim_i(T))$, and as a measure of the memory requirements the overall number of entries, i.e., the sum over all internal tensor sizes: $M = \sum_{T\in\text{TN}} \prod_i \dim_i(T)$. In this notation, TN refers to either the MPS or the TTN. We also used different initialization procedures for individual clusters not mentioned here. These variations represent restrictions imposed on the trees.

The dry-run experiment provides insight into the scalability of the TTN approach. On circuits with a large number of qubits, MPS and TTN scale similarly in regards to M. Fig. 15 shows the results for sizes up to 100 qubits. By construction, the circuit is shallow to conform with Eq. (4). Most advantages can only be achieved if the cluster boundaries are not crossed which will inevitably happen for deep circuits. In the *lattice* circuit, the number of connections already exceeds the threshold and the performance deteriorates.

Another aspect that arises from the results is the versatility of the approach. The creation and structure of the clusters can be tuned to fit different metrics. This can also be extended to include hardware specifications if necessary. Bad clustering, can harm the performance in certain cases. If leaves are distributed across clusters too unevenly, the biggest clus-

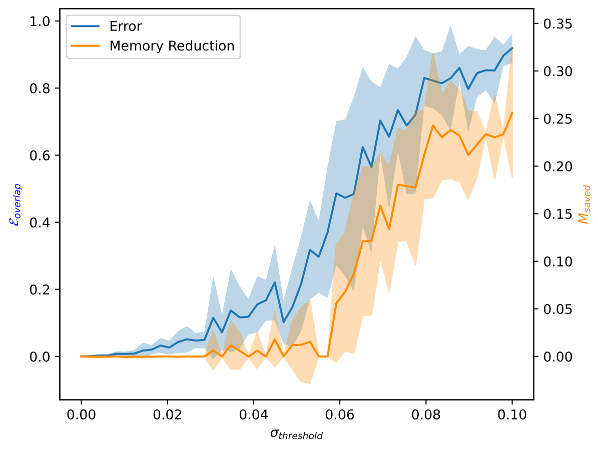


(a) *tree-like* circuit

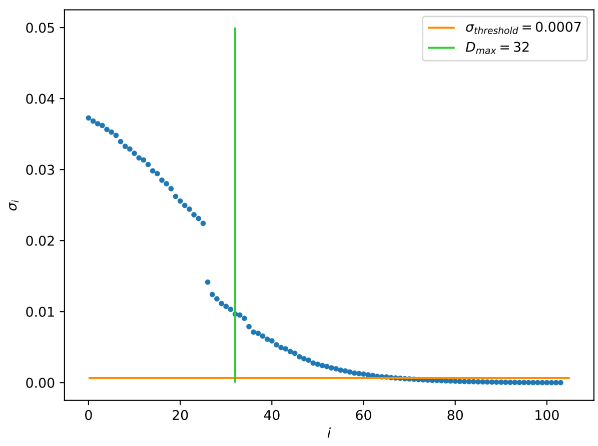


(b) *lattice* circuit

Figure 15: Results of the *dry-runs* with different settings for cluster generation compared to the MPS. The non-monotonicity for the *tree-like* circuit is caused by the heuristics in the tree creation. In some cases, the algorithm fails to find a good clustering, namely for *41* and *77* qubits.



(a) Randomized 16-qubit circuits



(b) Example for singular values

Figure 16: Errors and the resulting saved dimensionality for our proposed truncation scheme on randomized circuits. The relative saved memory is $\propto M_{\text{saved}} = 1 - \frac{M_{\text{truncated}}}{M_{\text{perfect}}}$. b) gives an example for an accumulated cut of off 0.1 which already induces an error of approximately 25%

ter dominates the performance cost, also evident in Fig. 15.

4.2.3 Effect of approximations

Similar to [4], we examine the effect of truncating singular values on error and memory savings at levels $> \ell_{\text{cluster}}$. In these cases, we discard the singular values $\sigma_{\text{truncated}}$, given $\sigma_1 \geq \cdots \geq \sigma_{\text{threshold}} > \sigma_{\text{truncated}} \geq 0$ after normalization. The fidelity of the state after truncation is quantified by the overlap error, which is equivalent to the trace distance when considering pure states and is far less memory intensive.

$$\mathcal{E}_{\text{overlap}} = 1 - |\langle \psi_{\text{tree}} | \psi_{\text{exact}} \rangle|^2 \tag{12}$$

We use the *lattice*-circuit with randomized nearest-neighbor interactions and run 10 trials for each tolerance. For small values of $\sigma_{\text{threshold}}$, the error is reasonable, but it barely results in any saved dimensionality. We see a significant increase in error given a singular value threshold of $7e-4$, which also corresponds to an accumulated cutoff of 0.1, see Fig. 16b. The resulting saved memory is not worth the tradeoff. Meaningful improvements can only be exchanged with a larger error rate. This is a side effect of the proposed cluster construction. In our truncation scheme, this can be explained by two factors:

1. The relative size of the singular values.
2. The number of necessary truncations.

The first aspect is mainly caused by the chosen cluster size. Smaller clusters perform better, but on smaller tensors truncating even a small number of singular values introduces significant errors. The second aspect arises from the normalization procedure which requires renormalization after each gate is applied. Once a given $D_{\max}$ is reached, each additional gate -would need truncation. Since we cannot avoid either of the two scenarios, we found the resulting error to be unacceptably large in this experiment. The only exceptions are circuits where cluster boundaries are no longer crossed after a fixed point in time.

Similarly, problems arise when controlling the bond dimension directly. Fixing $D_{\max}$ to a smaller bound, suffers the same pitfalls as before. Fig. 16b showcases the problem as one truncation can already produce a high error. Analogously, each following gate potentially introduces a similar error. It is open to see if other combinations of construction and truncation schemes might be more successful. For now, other approximation techniques remain superior in terms of the error-to-performance tradeoff.

5 Conclusions and outlook

In this work, we introduce a novel method to perform circuit simulation that exploits circuit structure using a TTN representation, which can be handled efficiently on a classical computer. A method to generate an advantageous tree structure for the initial quantum state which restricts bond dimension growth during circuit simulation is presented. Numerical simulations were run, demonstrating an advantage in overall simulation time (wall-clock time) between a naive MPS formulation and the presented method for specific circuits. Additionally, a specific circuit layout is presented in which MPSs would see exponential bond dimension growth, while states in the TTN representation would retain a uniformly bounded bond dimension.

A practical use case for the introduced method could be the now-ubiquitous Quantum Approximate Optimization Algorithm (QAOA), which is used to solve combinatorial optimization problems [28]. In this algorithm, the optimal solution to the problem is formulated as the ground state of a constructed Hamiltonian. The time evolution of such Hamiltonians often requires irregular connectivity, for which our TTN approach is suitable, especially as the layers of irregularly connected gates are repeated several times. One classical example is the Max-Cut problem, which is described in detail in [28], which, when mapped to a graph, exhibits the same connectivity as the original tree structure.

Another instance where TTNs might be useful is in entangled ancilla protocols as in [29]. Depending on the construction, multiple ancillae are entangled with a low number of gates and then only connected to their respective circuit. The problem remains intractable on classical computers for large circuit sizes, but the simulator could be used for confirming correctness on smaller problem instances.

TTNs have also shown to be a valuable technique for many-body physics [14]. Most approaches and algorithms studied are restricted to binary TTNs. However, we are not aware of any approaches that allow for an arbitrary *m-ary* or non-regular trees and exploration in a more generalized tree structure could yield interesting results. In quantum chemistry [15, 24] such TTNs have already been used and the importance of topology optimization is shown [24]. Our algorithm is designed to work with arbitrary similarity functions but is restricted to quantum circuits. A more general approach could incorporate entanglement or utilize problem-specific similarity functions. For example, [14] considers a fixed bond dimension and provides an algorithm to find an advantageous binary TTN. One could lift this restriction to binary TTNs in certain well-structured regions to allow for a more compact representation. In the extreme, optimization for each tensor might be possible which could allow generalizing to more complicated topologies.

From an algorithmic perspective, several avenues of future work still exist: the current algorithm assumes that two-qubit gates always introduce the maximum amount of entanglement, resulting in the restriction Eq. (4). The introduced method would benefit from a "look-ahead" estimation of entanglement generated by two-qubit gates; which is tractable if we reduce simulated circuits to more ubiquitous gate sets. Another promising generalization would be dynamically adapting the tree layout, i.e., creating or merging subtrees and branches when applying gates.

While not further explored in this work, an additional advantage is the possibility of interpretable and quantifiable compression by truncating the singular values of internal bonds. Namely, the singular values characterize the entanglement between subsystems and thus have a physical interpretation. On the other hand, the effects of truncating singular values within a general tensor network contraction are typically less predictable, and hence such contractions are usually avoided.

We hope that our algorithm for the tree structure search and theoretical analysis is applicable and insightful for other use cases of TTNs as well, e.g., condensed matter or quantum chemistry simulations.

Acknowledgments

This project is supported by the Federal Ministry for Economic Affairs and Climate Action on the basis of a decision by the German Bundestag through project QuaST, as well as by the Bavarian Ministry of Economic Affairs, Regional Development and Energy with funds from the Hightech Agenda Bayern.

We also thank the Munich Center for Quantum Science and Technology (MCQST) and the Munich Quantum Valley (MQV) initiative.

References

[1] F. Verstraete, V. Murg, and J. I. Cirac. "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems". Adv. Phys. **57**, 143–224 (2008).

[2] U. Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Ann. Phys. **326**, 96–192 (2011).

[3] Jacob C. Bridgeman and Christopher T. Chubb. "Hand-waving and interpretive dance: an introductory course on tensor networks". J. Phys. Math. Theor. **50**, 223001 (2017).

[4] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal. "What limits the simulation of quantum computers?". Phys. Rev. X **10**, 041038 (2020).

[5] Feng Pan, Pengfei Zhou, Sujie Li, and Pan Zhang. "Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations". Phys. Rev. Lett. **125**, 060503 (2020).

[6] Cupjin Huang, Fang Zhang, Michael Newman, Junjie Cai, Xun Gao, Zhengxiong Tian, Junyin Wu, Haihong Xu, Huanjun Yu, Bo Yuan, Mario Szegedy, Yaoyun Shi, and Jianxin Chen. "Classical simulation of quantum supremacy circuits" (2020). arXiv:2005.06787.

[7] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. "Simulating large quantum circuits on a small quantum computer". Phys. Rev. Lett. **125**, 150504 (2020).

[8] Johnnie Gray and Stefanos Kourtis. "Hyper-optimized tensor network contraction". Quantum **5**, 410 (2021).

[9] Feng Pan and Pan Zhang. "Simulating the sycamore quantum supremacy circuits" (2021). arXiv:2103.03074.

[10] Danylo Lykov, Roman Schutski, Alexey Galda, Valeri Vinokur, and Yuri Alexeev. "Tensor network quantum simulator with step-dependent parallelization". In 2022 IEEE International Conference on Quantum Computing and Engineering (QCE). Pages 582–593. (2022).

[11] G. Vidal. "Entanglement renormalization". Phys. Rev. Lett. **99**, 220405 (2007).

[12] G. Vidal. "Class of quantum many-body states that can be efficiently simulated". Phys. Rev. Lett. **101**, 110501 (2008).

[13] V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack. "Simulating strongly correlated quantum systems with tree tensor networks". Phys. Rev. B **82**, 205105 (2010).

[14] M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero. "Unconstrained tree tensor network: An adaptive gauge picture for enhanced performance". Phys. Rev. B **90**, 125154 (2014).

[15] V. Murg, F. Verstraete, R. Schneider, P. R. Nagy, and Ö. Legeza. "Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems". J. Chem. Theory Comput. **11**, 1027–1036 (2015).

[16] Klaas Gunst, Frank Verstraete, Sebastian Wouters, Örs Legeza, and Dimitri Van Neck. "T3NS: Three-legged tree tensor network states". J. Chem. Theory Comput. **14**, 2026–2033 (2018).

[17] Florian Schröder, David Turban, Andrew Musser, Nicholas Hine, and Alex Chin. "Tensor network simulation of multi-environmental open quantum dynamics via machine learning and entanglement renormalisation". Nat. Commun.**10** (2019).

[18] L. Tagliacozzo, G. Evenbly, and G. Vidal. "Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law". Phys. Rev. B **80**, 235127 (2009).

[19] Gianluca Ceruti, Christian Lubich, and Hanna Walach. "Time integration of tree tensor networks". SIAM J. Numer. Anal. **59**, 289–313 (2021).

[20] Xiao Yuan, Jinzhao Sun, Junyu Liu, Qi Zhao, and You Zhou. "Quantum simulation with hybrid tensor networks". Phys. Rev. Lett. **127**, 040501 (2021).

[21] Eugene Dumitrescu. "Tree tensor network approach to simulating shor's algorithm". Phys. Rev. A **96**, 062322 (2017).

[22] Shi-Ju Ran, Emanuele Tirrito, Cheng Peng, Xi Chen, Luca Tagliacozzo, Gang Su, and Maciej Lewenstein. "Tensor Network Contractions". Springer Cham. (2020).

[23] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. "Tree tensor networks for generative modeling". Phys. Rev. B **99**, 155131 (2019).

[24] Szilárd Szalay, Max Pfeffer, Valentin Murg, Gergely Barcza, Frank Verstraete, Reinhold Schneider, and Örs Legeza. "Tensor product methods and entanglement optimization for ab initio quantum chemistry". Int. J. Quantum Chem. **115**, 1342–1391 (2015).

[25] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. "Quantum supremacy using a programmable superconducting processor". Nature **574**, 505–510 (2019).

[26] Adam S. Jermyn. "Efficient tree decomposition of high-rank tensors". J. Comput. Phys. **377**, 142–154 (2019).

[27] Giovanni Ferrari, Giuseppe Magnifico, and Simone Montangero. "Adaptive-weighted tree tensor networks for disordered quantum many-body systems". Phys. Rev. B **105**, 214201 (2022).

[28] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm" (2014). arXiv:1411.4028.

[29] Benjamin F. Schiffer, Jordi Tura, and J. Ignacio Cirac. "Adiabatic spectroscopy and a variational quantum adiabatic algorithm". PRX Quantum **3**, 020347 (2022).