Arbitrary Tensor Network Algorithm: Theory, Methods and Applications

Feng Pan

Centre for Quantum Technologies, National University of Singapore

IPAM Tensor Network 2024 workshop
Contents

➢ Background & Motivation

➢ Theoretical Foundation and Methods of arbitrary TN algorithm

➢ Approximate arbitrary TN algorithm

➢ Exact arbitrary TN algorithm

➢ Conclusion
Background & Motivation
Background – Tensor Network Notations

Scalar

Vector

Matrix

Tensor

Diagram showing nodes A, B, C, D, E, F with connections labeled by indices i, j, k, l, m, n, o, p, q, r, s.
Background – Tensor Network Operations

(1) \( A \otimes B = C \)

\[ \sum_j A_{ij} B_j = C_i \]

(2) \( A \otimes B \otimes C = D \)

\[ \sum_j A_{ij} B_{jk} = C_{ik} \]

(3) \( A \rightarrow \)

\[ \sum_i A_{ii} = Tr(A) \]

(4) \( A \otimes B \otimes C \rightarrow \)

\[ \sum_{kl} A_{ijkl} B_{klm} = C_{ijm} \]
Motivation

Matrix Product State / Tensor Train

Tree Tensor Network / Hierarchical Tucker

PEPS

MERA

Figure credit: http://tensornetwork.org/
Motivation

Matrix Product State / Tensor Train

Tree Tensor Network / Hierarchical Tucker

PEPS

MERA

system without any structure?

Figure credit: http://tensornetwork.org/
Motivation

\[ \sum_s e^{-\beta \sum_i H_i(s_i)} \]

#P problem generally

➢ Mean-field approximation

➢ Tree approximation with Belief Propagation

➢ \[ \sum_s \prod_i T^i_{s_i} \] Tensor network contraction
Theoretical Foundation and Methods of arbitrary TN algorithm
Complexity lower bound

Theorem: The space complexity lower bound of an arbitrary tensor network contraction is exponential to the tree width of its line graph.

Tree decomposition and contraction order
Methods for finding good contraction orders

Finding a good contraction order is a hard combinatorial optimization problem, due to

complex target function + large solution space + sequence dependence

Here we introduce four ways to find contraction orders

- Greedy method
- Branch and Bound method
- Graph partition method
- Local update method
Greedy method:

Branch and Bound method:

Graph partition method:

Local update method:


Other techniques in the real numerical contraction

- Tensor slicing

\[ = \sum_k \]

- Computational efficiency
  Balance the memory read/write operations and floating-point operations
Approximate arbitrary TN algorithm
Approximate arbitrary TN algorithm

1. Exponentially large complexity
   Using matrix product state (MPS)
   Using the canonical form of MPS

2. How to do approximation?
   DMRG like low-rank approximation scheme
   \[ D_{\text{max}} \quad \chi_{\text{max}} \]

3. How to control the error?
   Greedily chosen from the current TN

4. Contraction order?
MPS calculus operations

(a) $i \approx j \approx i \approx j \approx i$

(b) $\approx \approx \approx \approx$

(c) $i \approx j \approx i \approx j \approx i$
Contract an Arbitrary TN with an example

Map calculation of physical properties to TN

\[
F(\beta) = -\frac{1}{\beta N} \ln \left( \sum_{\vec{s}} \prod_{(i,j)} e^{\beta J_{ij} s_i s_j} \right) \\
= -\frac{1}{\beta N} \ln \left( \sum_{\vec{b}} \bigotimes_i \mathcal{L}_{b_i}^{d_i} \otimes \mathcal{B}_{b_i b_j} \right)
\]
Some results on partition function calculation

16x16 2D lattice
RRG graphs
n=80, k=3

Small-world
n=70, c=4

SK model
n=20
Exact arbitrary TN algorithm
Exact arbitrary TN algorithm

There are some circumstances when problems require exact results or there is no intrinsic low-rank structure:

For example, Simulation of quantum circuits with quantum gate (fSim) whose decomposition spectrum is flat.

Hence, we will also need exact arbitrary TN algorithms.
Exact arbitrary TN algorithm - Challenges

- Exponentially large complexity
- How to do approximation?
- How to control the error?
- Contraction order
- For quantum circuit simulations: how to sample bitstrings
Sycamore chip and Random Circuit Sampling


200 seconds 1,000,000 bit-strings 0.2% XEB fidelity
Tensor Network and QCircuits

Initial states: Vectors

Single-qubit gates: Unitary matrices

Two-qubit gates: Unitary 4-way tensors

Another \#P problems without any low-rank structure to utilize

\[ \langle 0 | U_c | x \rangle = \sum \prod_{s \in S} U^{i_s} \hat{S}_i \]
Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google’s choice)
2. Tensor network contraction algorithm
Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrödinger-Feynman algorithm (Google’s choice)

2. Tensor network contraction algorithm
Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrödinger-Feynman algorithm (Google’s choice)

2. Tensor network contraction algorithm
Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google’s choice)
2. Tensor network contraction algorithm

<table>
<thead>
<tr>
<th>qubits, n</th>
<th>cycles, m</th>
<th>total #paths</th>
<th>fidelity</th>
<th>run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>12</td>
<td>$4^{17}\times2^4$</td>
<td>1.4%</td>
<td>2 hours</td>
</tr>
<tr>
<td>53</td>
<td>14</td>
<td>$4^{21}\times2^3$</td>
<td>0.9%</td>
<td>2 weeks</td>
</tr>
<tr>
<td>53</td>
<td>16</td>
<td>$4^{25}\times2^3$</td>
<td>0.6%</td>
<td>4 years</td>
</tr>
<tr>
<td>53</td>
<td>18</td>
<td>$4^{28}\times2^3$</td>
<td>0.4%</td>
<td>175 years</td>
</tr>
<tr>
<td>53</td>
<td>20</td>
<td>$4^{31}\times2^4$</td>
<td>0.2%</td>
<td>10,000 years</td>
</tr>
</tbody>
</table>

TABLE XI. Approximate qsimh run times using one million CPU cores extrapolated from the average simulation run time for 1000 simulation paths on one CPU core.
Fidelity and Linear cross entropy benchmark (XEB)

• DM of the output state \( \rho = \mathcal{F}\ket{\psi_U}\bra{\psi_U} + (1 - \mathcal{F}) I/2^n \)

• Traditional definition of fidelity \( \mathcal{F} = \bra{\psi_U} \rho \ket{\psi_U} \)

• It is impossible to calculate the fidelity for such experiment, use XEB fidelity instead

\[
\mathcal{F}_{\text{XEB}} = \frac{2^n}{L} \sum_{i=1}^{L} p_U(x_i) - 1
\]

• XEB can be spoofed: samples with large probabilities
Big-batch method
Results and the spoofing of XEB

<table>
<thead>
<tr>
<th></th>
<th># bitstrings</th>
<th>Time complexity</th>
<th>Space complexity</th>
<th>Computational time</th>
<th>Computational hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google [1]</td>
<td>$10^6$</td>
<td>—</td>
<td>—</td>
<td>10,000 years</td>
<td>Summit supercomputer</td>
</tr>
<tr>
<td>Cotengra [12]</td>
<td>$1$</td>
<td>$3.10 \times 10^{22}$</td>
<td>$2^{27}$</td>
<td>3,088 years</td>
<td>One NVIDIA Quadro P2000</td>
</tr>
<tr>
<td>Alibaba [18]</td>
<td>$64$</td>
<td>$6.66 \times 10^{18}$</td>
<td>$2^{29}$</td>
<td>267 days</td>
<td>One V100 GPU</td>
</tr>
<tr>
<td>Ours</td>
<td>$2097152$</td>
<td>$4.51 \times 10^{18}$</td>
<td>$2^{30}$</td>
<td>149 Days</td>
<td>One A100 GPU</td>
</tr>
</tbody>
</table>

TABLE II. Comparison of computational cost among different methods on Sycamore circuit with 53 qubits and 20 cycles.

5 days in 60 GPUs

2,097,152 bitstring samples with 0% XEB fidelity

1,000,000 bitstring samples with 73.9% XEB fidelity

Full amplitude simulation

We can also use big-batch method to do full amplitude simulation, by enumerating configurations of closed qubits.

50 qubits, 14 cycles, EFGH sequence using 100 GPUs in 10 days
Feedbacks of this work

Can be defended using a slightly more complicated benchmark such as adding a preprocessing step to detect correlated samples.

---

Another axe swung at the Sycamore

So there's an interesting new paper on the arXiv by Feng Pen and Pan Zhang, entitled "Simulating the Sycamore supremacy circuits." It's about a new tensor contraction strategy for classically simulating Google's 53-qubit quantum supremacy experiment from Fall 2019. Using their approach, and using just 60 GPUs running for a few days, the authors say they managed to generate a million correlated 53-bit strings—meaning, strings that all agree on a specific subset of 20 or so bits—that achieve a high linear cross-entropy score.

Alas, I haven't had time this weekend to write a "proper" blog post about this, but several people have by now emailed to ask my opinion, so I thought I'd share the brief response I sent to a journalist.

This does look like a significant advance on simulating Sycamore-like random quantum circuits! Since it's based on tensor networks, you don't need the literally largest supercomputer on the planet filling up tens of petabytes of hard disk space with amplitudes, as in the brute-force strategy proposed by IBM. Pan and Zhang's strategy seems most similar to the strategy previously proposed by Alibaba, with the key difference being that the new approach generates millions of correlated samples rather than just one.

I guess my main thoughts for now are:

1. Once you knew about this particular attack, you could evade it and get back to where we were before by switching to a more sophisticated verification test—namely, one where you not only computed a Linear XEB score for the observed samples, you also made sure that the samples didn't share too many bits in common. (Strangely, though, the paper never mentions this point.)

2. The other response, of course, would just be to redo random circuit sampling with a slightly bigger quantum computer, like the ~70-qubit devices that Google, IBM, and others are now building!

Anyway, very happy for thoughts from anyone who knows more.
Sparse-state tensor network simulation

Observation of quantum supremacy experiments:

➢ The number of bitstrings obtained by experiments will not be exponentially large.
➢ These bitstrings will compose a sparse-state of the full Hilbert space.

Thus, calculating multiple bitstring amplitudes becomes the tensor network contraction below
Observation of quantum supremacy experiments:

➢ The number of bitstrings obtained by experiments will not be exponentially large.
➢ These bitstrings will compose a sparse-state of the full Hilbert space.

Thus, calculating multiple bitstring amplitudes becomes the tensor network contraction below
Sparse-state contraction

The core of sparse-state contraction is explicitly listing the dimension of open qubits and determine which entries should be calculated.

Arbitrary bit-string means we can circumvent the correlated bitstring problems, and there is no way to tell from our samples from the quantum samples.
Results
## Results

<table>
<thead>
<tr>
<th>Data</th>
<th>Original</th>
<th>Branch merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_c$ head one sub-task</td>
<td>$2.3816 \times 10^{13}$</td>
<td>$6.967 \times 10^{13}$</td>
</tr>
<tr>
<td>$T_c$ tail one sub-task</td>
<td>$2.9425 \times 10^{13}$</td>
<td>$8.796 \times 10^{13}$</td>
</tr>
<tr>
<td>Overall $T_c$ ($2^{16}$ sub-tasks)</td>
<td>$3.489 \times 10^{18}$</td>
<td>$1.033 \times 10^{19}$</td>
</tr>
<tr>
<td>Space complexity</td>
<td>$2^{30}$</td>
<td></td>
</tr>
</tbody>
</table>
# Results

<table>
<thead>
<tr>
<th>Data</th>
<th>Original</th>
<th>Branch merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_e ) head one sub-task</td>
<td>( 2.3816 \times 10^{13} )</td>
<td>( 6.967 \times 10^{13} )</td>
</tr>
<tr>
<td>( T_e ) tail one sub-task</td>
<td>( 2.9425 \times 10^{13} )</td>
<td>( 8.796 \times 10^{13} )</td>
</tr>
<tr>
<td>Overall ( T_e ) (( 2^{16} ) sub-tasks)</td>
<td>( 3.489 \times 10^{18} )</td>
<td>( 1.033 \times 10^{19} )</td>
</tr>
<tr>
<td>Space complexity</td>
<td>( 2^{30} )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># bitstrings</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google [1]</td>
<td>( 10^6 )</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Cotengra [12]</td>
<td>1</td>
<td>( 3.10 \times 10^{22} )</td>
<td>( 2^{27} )</td>
</tr>
<tr>
<td>Alibaba [18]</td>
<td>64</td>
<td>( 6.66 \times 10^{18} )</td>
<td>( 2^{29} )</td>
</tr>
<tr>
<td>Ours</td>
<td>\textbf{2097152}</td>
<td>\textbf{4.51 \times 10^{18}}</td>
<td>( 2^{30} )</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Data</th>
<th>Original</th>
<th>Branch merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_c$ head one sub-task</td>
<td>$2.3816 \times 10^{13}$</td>
<td>$6.967 \times 10^{13}$</td>
</tr>
<tr>
<td>$T_c$ tail one sub-task</td>
<td>$2.9425 \times 10^{13}$</td>
<td>$8.796 \times 10^{13}$</td>
</tr>
<tr>
<td>Overall $T_c$ ($2^{16}$ sub-tasks)</td>
<td>$3.489 \times 10^{18}$</td>
<td>$1.033 \times 10^{19}$</td>
</tr>
<tr>
<td>Space complexity</td>
<td></td>
<td>$2^{30}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># bitstrings</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google [1]</td>
<td>$10^6$</td>
<td>—</td>
</tr>
<tr>
<td>Cotengra [12]</td>
<td>1</td>
<td>$3.10 \times 10^{22}$</td>
</tr>
<tr>
<td>Alibaba [18]</td>
<td>64</td>
<td>$6.66 \times 10^{18}$</td>
</tr>
<tr>
<td>Ours</td>
<td>$2097152$</td>
<td>$4.51 \times 10^{18}$</td>
</tr>
</tbody>
</table>

15 hours in 512 GPUs $\rightarrow$ dozens of seconds in exaflops supercomputer

Quantum supremacy on Sycamore53 does not hold!
## Results

<table>
<thead>
<tr>
<th>Data</th>
<th>Original</th>
<th>Branch merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_c ) head one sub-task</td>
<td>(2.3816 \times 10^{13})</td>
<td>(6.967 \times 10^{13})</td>
</tr>
<tr>
<td>( T_c ) tail one sub-task</td>
<td>(2.9425 \times 10^{13})</td>
<td>(8.796 \times 10^{13})</td>
</tr>
<tr>
<td>Overall ( T_c ) (2(^{16}) sub-tasks)</td>
<td>(3.489 \times 10^{18})</td>
<td>(1.033 \times 10^{19})</td>
</tr>
<tr>
<td>Space complexity</td>
<td>(2^{30})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th># bitstrings</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google [1]</td>
<td>(10^6)</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Cotengra [12]</td>
<td>1</td>
<td>(3.10 \times 10^{22})</td>
<td>(2^{27})</td>
</tr>
<tr>
<td>Alibaba [18]</td>
<td>64</td>
<td>(6.66 \times 10^{18})</td>
<td>(2^{29})</td>
</tr>
<tr>
<td>Ours</td>
<td>(2097152)</td>
<td>(4.51 \times 10^{18})</td>
<td>(2^{30})</td>
</tr>
</tbody>
</table>
Results
Results
Results

![Graph 1: Probability vs Np]

- Probability (Pr(Np)) on a logarithmic scale.
- Np range from 0 to 10.

![Graph 2: Squared norm vs Estimated Fidelity]

- Squared norm on a logarithmic scale.
- Estimated fidelity range from $2^{-11}$ to $2^{-8}$.

The graphs illustrate the relationship between Np and Pr(Np), as well as the squared norm against the estimated fidelity.
Results

Advantage on Power Consumption?

1. improved slicing scheme
2. post-selection

Zhao, Zhong, Pan, et al in preparation
Conclusion
Conclusion

• Arbitrary tensor network algorithms are very efficient tools for solving specific \#P problems, either with low-rank structure or not.

• Their improvements are highly related to algorithmic and hardware developments (the hardware requirements are highly similar to large language models).

• There are many applications:
  • Classical simulation/validation of quantum computational tasks
  • Calculation of physical properties defined on complex systems
  • Exploring solution space of combinatorial optimization problems (tropical algebra)
Thanks for your attention!
\[
E^* = - \lim_{\beta \to \infty} \frac{1}{\beta} \sum_s e^{-\beta E(s)}
\]

\[
= - \lim_{\beta \to \infty} \frac{1}{\beta} \sum_s \prod_{i \neq j} e^{\beta J_{ij}s_is_j} \prod_i e^{h_is_i}
\]

\[
\lim_{\beta \to \infty} \frac{1}{\beta} \ln(e^{\beta x} + e^{\beta y}) = x \oplus y, \quad \frac{1}{\beta} \ln(e^{\beta x} \cdot e^{\beta y}) = x \odot y.
\]
