Tensor network based quantum simulation with Yao.jl

(@ Lausanne)

Jin-Guo Liu (GiggleLiu)

HKUST(GZ) - FUNH - Advanced Materials Thrust

2024-12-02

Outline

Outline

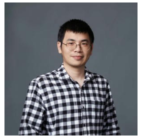
Yao.jl - a Julia package for quantum simulation

2min

Yao.jl (幺 - means unitary)

- One of the first quantum simulators dedicated to **differentiable quantum simulation** (Luo et al., 2020).
  - Simulation of variational quantum algorithms, e.g. quantum machine learning (Mitarai et al., 2018), variational quantum eigensolver (Tilly et al., 2022), quantum circuit Born machine (Liu & Wang, 2018) et al.
  - Quantum control, e.g. design control pulses.

Xiu-Zhe Luo,



Jin-Guo Liu, Lei Wang and Pan Zhang @ 2018

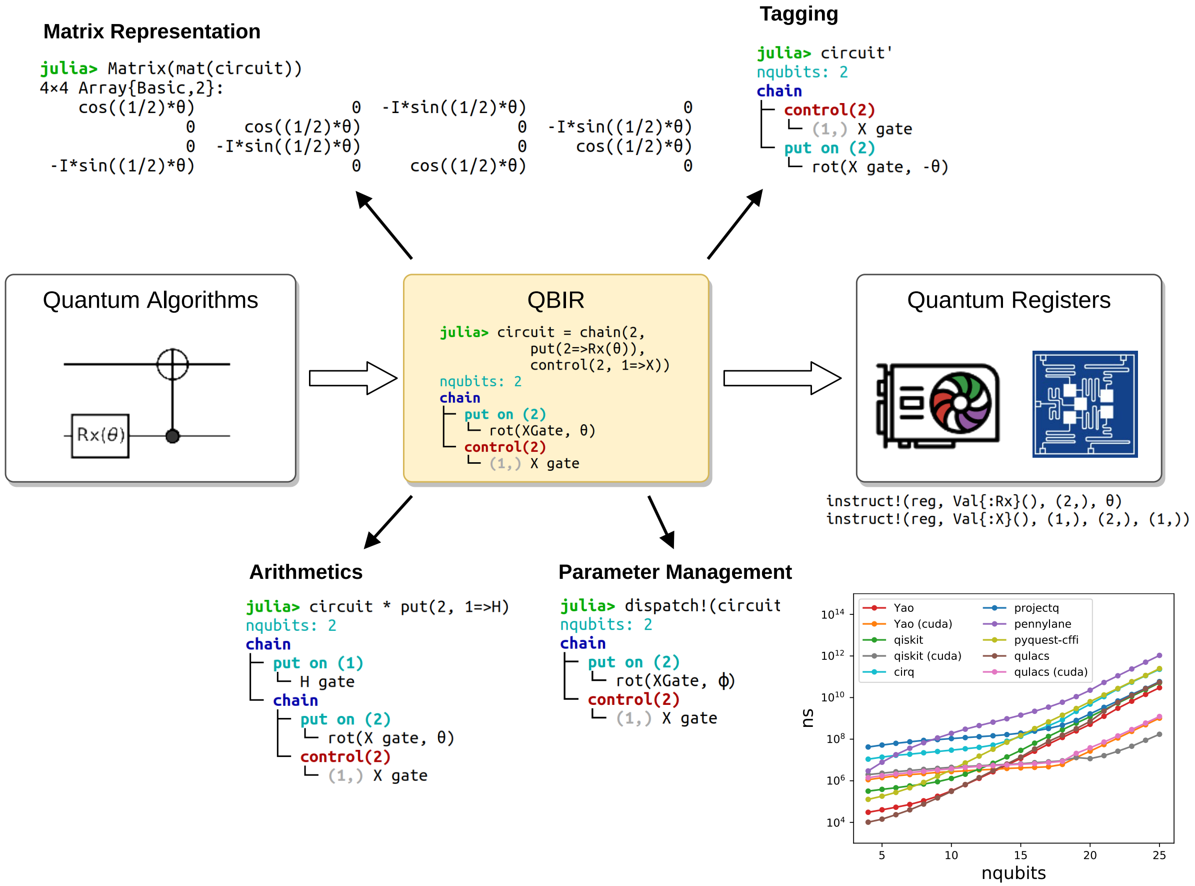
Yao.jl features in v0.6

4min

Features in v0.6

- Differentiable quantum circuit
- Matrix representation
- Operator arithmetics

- State-of-the-art performance
- GPU backend



v0.6-v0.9, the updates

1. Bloqade.jl @ QuEraComputing

6min

Bloqade.jl is a package for the quantum computation and quantum simulation based on the neutral-atom architecture.

- Extended qubit to **qudit** simulation.
- Allows simulation in a **subspace** of the Hilbert space.

2. Classical benchmarking quantum circuits & Quantum error correction

- Tensor network backend.
- Basic noise channel and density matrix simulation.

3. Community packages include:

- FLOYao.jl: A fermionic linear optics simulator backend for Yao.jl (Jan Lukas Bosse et al)
- QAOA.jl: This package implements the Quantum Approximate Optimization Algorithm and the Mean-Field Approximate Optimization Algorithm.

Outline

Yao @ v0.9 - What's new?

Fast prototyping with Yao.jl

Tensor network based quantum simulation

Discussion: Tensor network contraction order optimization

Finding the ground state of a Rydberg PXP chain.

8min

The Hamiltonian of a Rydberg PXP chain is given by

$$H = \sum_{i=1}^{n} P_{i-1} X_i P_{i+1}$$

where $P_i = |0\rangle_i \langle 0|_i$ is a projector to state $|0\rangle_i$, and X is the Pauli-X operator. Periodic boundary condition is applied, i.e. $0 = n$.

One line for solving the ground state

12min

```
julia> using Yao, KrylovKit

julia> @time eigsolve(mat(sum([kron(20, mod1(i-1, 20)=>ConstGate.P0, i=>X,
mod1(i+1, 20)=>ConstGate.P0) for i in 1:20])), 1, :SR; ishermitian=true);
  5.259707 seconds (74.84 k allocations: 5.315 GiB, 18.48% gc time, 0.57%
compilation time)
```

- `KrylovKit.eigsolve(m, 1, :SR; ishermitian=true)` finds the lowest 1 eigenvalue and eigenvector of a Hermitian matrix m. `KrylovKit` is also the time evolution backend for Yao.
- `mat(op)` converts an operator to a sparse matrix.
- `kron(n, pairs...)` raises an operator to a larger Hilbert space,
- $P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$

Outline

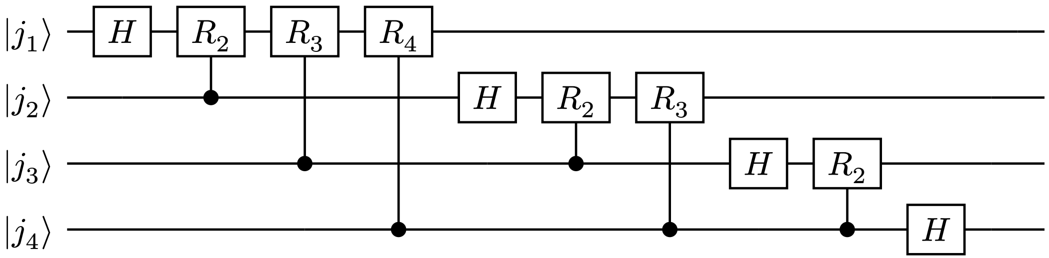
A minimum example

14min

- A product state $|j_1\rangle \otimes |j_2\rangle \otimes \ldots \otimes |j_n\rangle$ as input,
- Goes through a shallow quantum circuit, here we use a quantum Fourier transform (QFT) circuit
- Q: What is the expectation value of a given observable, e.g. a product of Pauli operators $P_1 \otimes P_2 \otimes \ldots \otimes P_n$, where $P_i \in \{I, X, Y, Z\}$.

Step 1. Create the quantum Fourier transform (QFT) circuit.

16min



$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is a Hadamard gate, $\mathrm{CR}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{\pi}{2^{k-1}}} \end{pmatrix}$ is a controlled phase gate.

One line for creating a QFT circuit

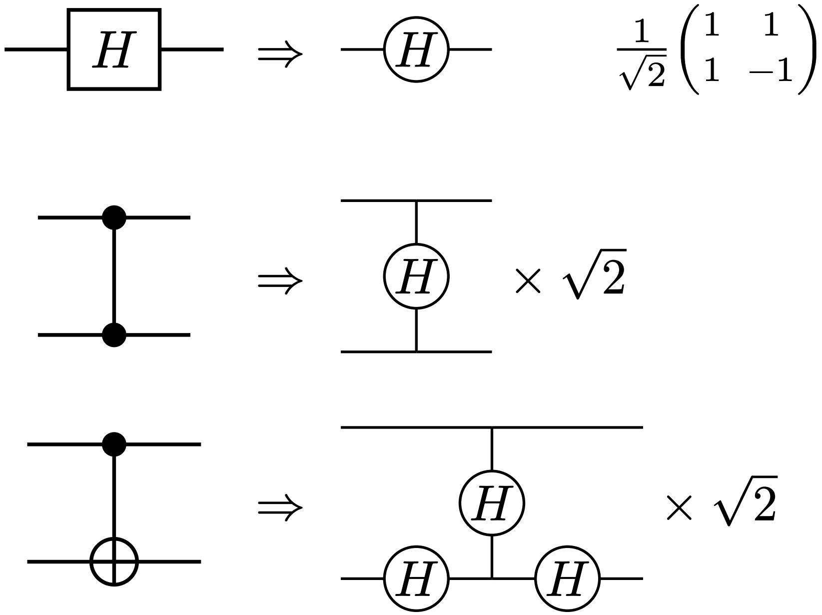
18min

```
julia> qft = chain(4, chain(4, i==j ? put(i=>H) : control(4, i, j=>shift(2π/(2^(j-i+1)))) for j in i:4) for i = 1:4)
```

- `chain(n, gates...)` creates a n-qubit circuit by concatenating the gates.
- `put(n, loc=>op)` raises an operator to an n-qubit Hilbert space.
- `control(n, ctrl_locs, target_loc=>op)` creates a controlled operator in an n-qubit Hilbert space.
- `shift(`θ`)`$= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$ is a phase shift gate.

Step 2. Convert a quantum circuit to a tensor network

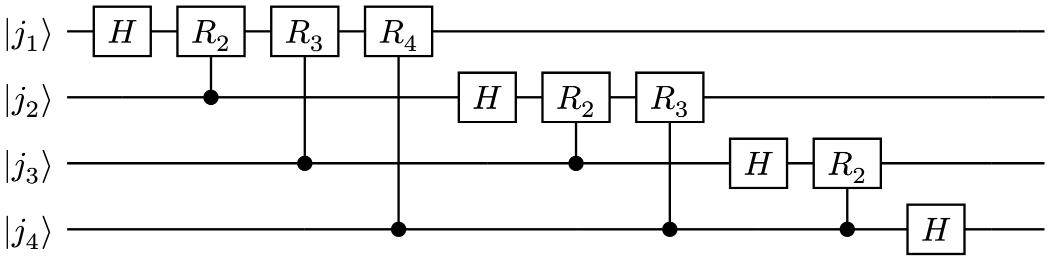
19min

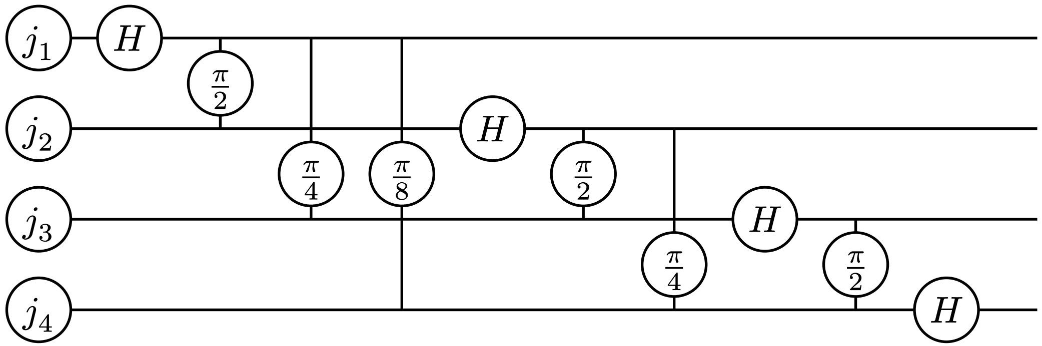


Note: $\perp$ is a hyperedge (or delta tensor).

QFT tensor network

20min

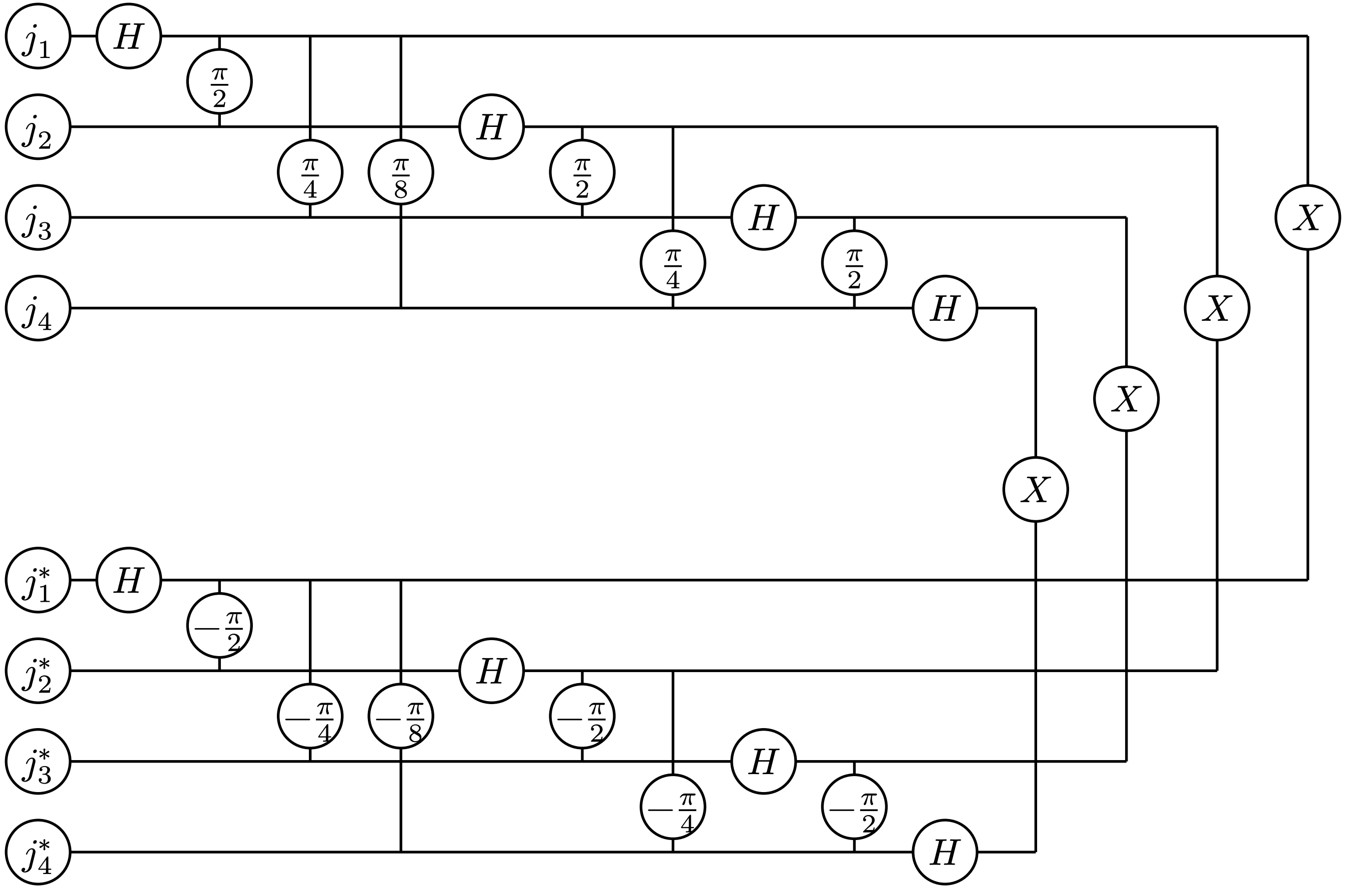




where $\theta = \begin{pmatrix} 1 & 1 \\ 1 & e^{i\theta} \end{pmatrix}$

The tensor network for the expectation value

21min



One line for creating a tensor network

23min

```
julia> qft_net = yao2einsum(chain(qft, chain(4, [put(4, i=>X) for i in 1:4]),
qft'), initial_state = Dict([i=>zero_state(1) for i=1:4]), final_state =
Dict([i=>zero_state(1) for i=1:4]), optimizer = TreeSA(nslices=2))
TensorNetwork
Time complexity: 2^9.108524456778169
Space complexity: 2^2.0
Read-write complexity: 2^10.199672344836365
```

- `yao2einsum(circuit; initial_state, final_state, optimizer)` maps a quantum circuit to a tensor network. Initial and final states are specified by a dictionary.
- `circuit'` is the adjoint of `circuit`.
- `TreeSA(; nslices)` is a heuristic contraction order optimizer with `nslices` slices.

Step 3: One line to contract a tensor network

24min

```
julia> contract(qft_net) # calculate <reg|qft' observable qft|reg>
0-dimensional Array{ComplexF64, 0}:
0.9999999999999993 + 0.0im
```

- `contract(tensor_network)`, use the `OMEinsum.jl` to contract the tensor network.
- Time complexity is the number of multiplications. Space complexity is the number of elements in the largest tensor. Read-write complexity is the number of reads and writes.

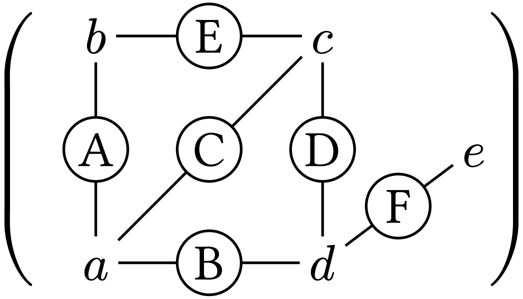
Outline

Tensor network contraction is a sum of products

25min

Tensor network contraction ↔ sum of products of tensor elements

contract



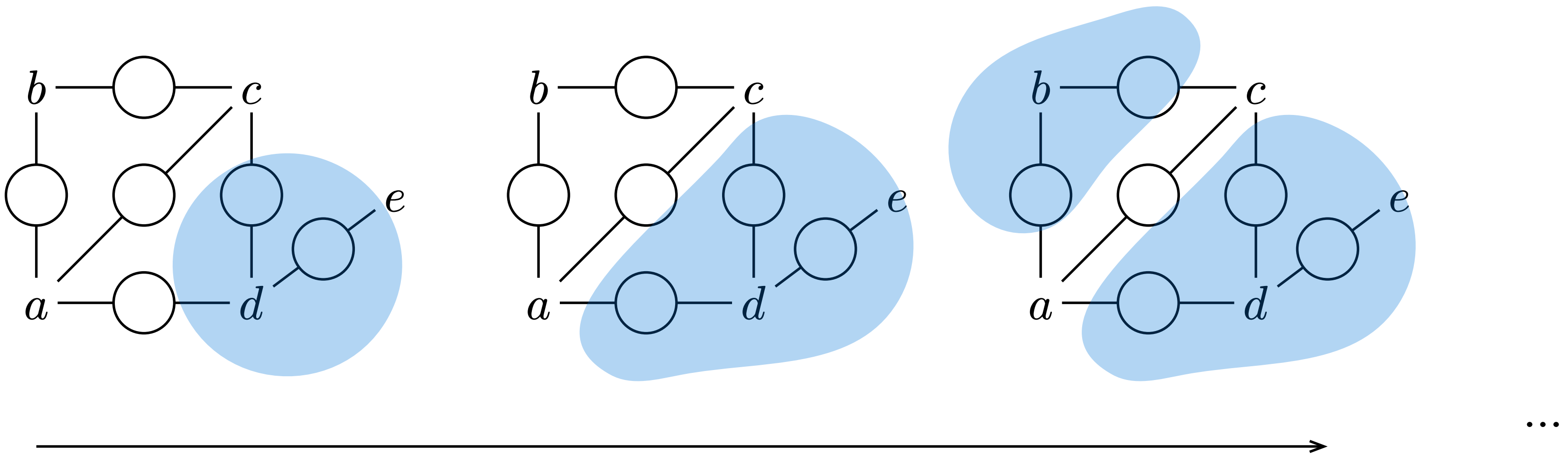
$$= \sum_{abcde} A_{ab} B_{ad} C_{ac} D_{cd} E_{bc} F_{de}$$

- Multiplication is commutative,
- Addition and multiplication are distributive.

$$\text{Note: In this talk, } \text{tensor network} = \text{einsum}$$
$$= \text{sum-product network}$$

Tensor network contraction order

26min



- Contraction is performed in pair-wise manner.
- The pair-wise contraction order determines the complexity (time, space, read-write).

The hardness of finding optimal contraction order

27min

NP-complete

Theorem (Markov & Shi, 2008): Let C be a quantum circuit (tensor network) with T gates (tensors) and whose underlying circuit graph is G_C. Then C can be simulated deterministically in time $T^{O(1)} \exp[O(\mathrm{tw}(G_C))]$.

Tree width (measures how similar a graph is to a tree, the smaller the more tree-like):

- Tree graphs and line graphs: 1
- $L \times L$ grid graph: $O(L)$
- n-vertex 3-regular graph: $\approx \frac{n}{6}$

Heuristic search for optimal contraction order

28min

`OMEinsum.jl` (GSoC 2019, 2024)

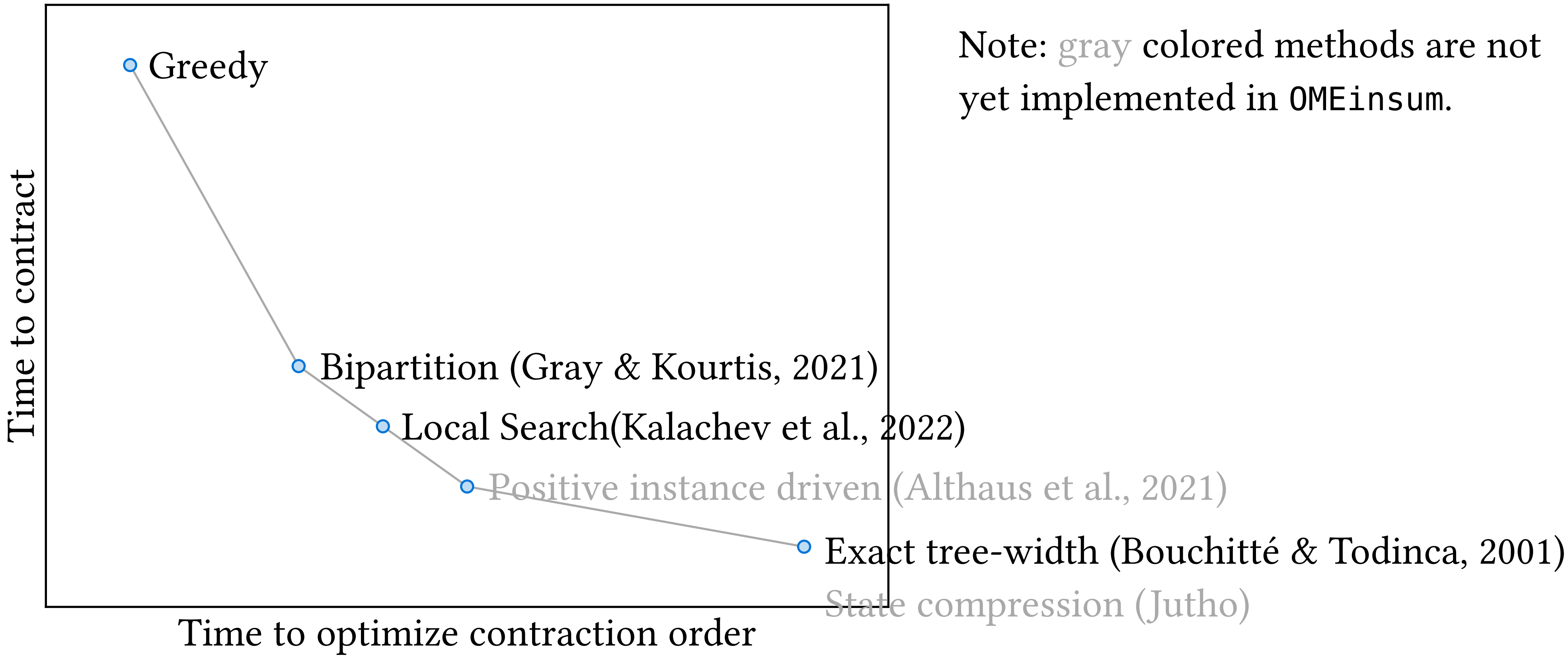
Can handle $> 10^4$ tensors!

- `GreedyMethod`: fast but not optimal
- `ExactTreewidth`: optimal but exponential time (Bouchitté & Todinca, 2001)
- `TreeSA`: heuristic local search, close to optimal, **slicing** supported (Kalachev et al., 2022)
- `KaHyParBipartite` and `SABipartite`: min-cut based bipartition, better heuristic for extremely large tensor networks (Gray & Kourtis, 2021)

Check the blog post for more details: https://arrogantgao.github.io/blogs/contractionorder/

Heuristic search for optimal contraction order

30min



Note: gray colored methods are not yet implemented in `OMEinsum`.

Pros and cons

31min

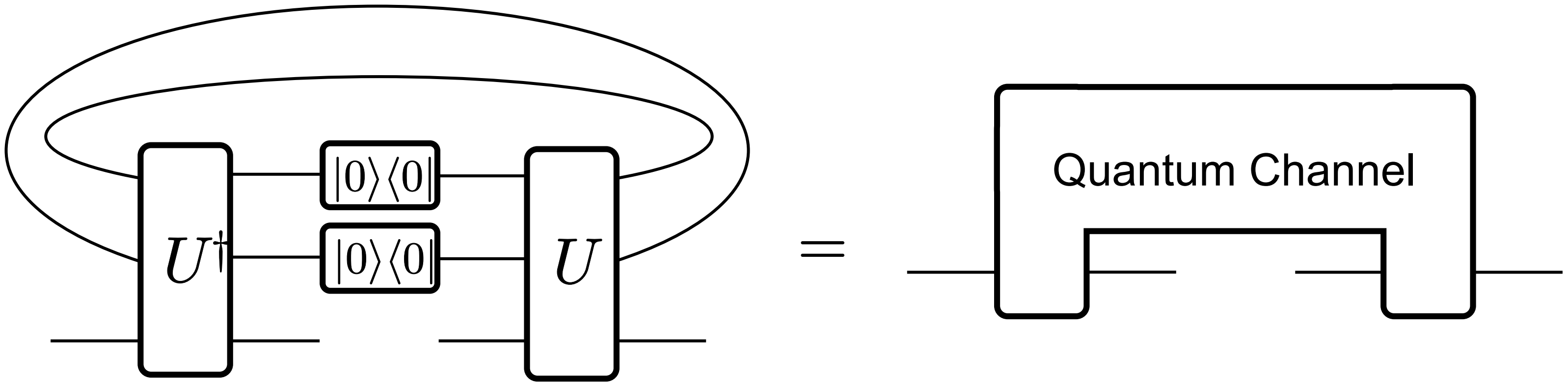
- Suited for **shallow** quantum circuit simulation, e.g. solving the sampling problem of the sycamore quantum circuits (53 qubits) (Pan et al., 2022)
- Can handle common tasks, such as **sampling** and **obtaining expectation values**.
- Can easily generalize the noisy quantum systems (Gao et al., 2024).

- For general circuits, the simulation is still exponentially hard.

Example application: Quantum error correction

32min

Using tensor network as the simulation backend for studying **coherent errors**(Ni et al., 2024).



TensorQEC

Summary

33min

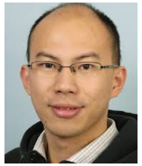
Yao.jl: a utility for quantum onliners.

- Yao paper: (Luo et al., 2020)
- GitHub repo: Yao.jl

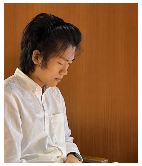
1000 - # of stars = 65!

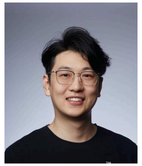
Collaborators


Xiu-Zhe Luo


Lei Wang


Pan Zhang


Xuan-Zhao Gao (TreeWidthSolver.jl)


Zhong-Yi Ni (TensorQEC.jl)

Bibliography

[1] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, "Yao.Jl: Extensible, Efficient Framework for Quantum Algorithm Design," *Quantum*, vol. 4, p. 341, Oct. 2020, doi: 10.22331/q-2020-10-11-341.

[2] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 32309, 2018.

[3] J. Tilly *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[4] J.-G. Liu and L. Wang, "Differentiable learning of quantum circuit born machines," *Physical Review A*, vol. 98, no. 6, p. 62324, 2018.

[5] I. L. Markov and Y. Shi, "Simulating Quantum Computation by Contracting Tensor Networks," *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, Jan. 2008, doi: 10.1137/050644756.

[6] V. Bouchitté and I. Todinca, "Treewidth and Minimum Fill-in: Grouping the Minimal Separators," *SIAM Journal on Computing*, vol. 31, no. 1, pp. 212–232, 2001, doi: 10.1137/S0097539799359683.

[7] G. Kalachev, P. Panteleev, and M.-H. Yung, "Multi-Tensor Contraction for XEB Verification of Quantum Circuits." [Online]. Available: https://arxiv.org/abs/2108.05665

[8] J. Gray and S. Kourtis, "Hyper-optimized tensor network contraction," *Quantum*, vol. 5, p. 410, Mar. 2021, doi: 10.22331/q-2021-03-15-410.

[9] E. Althaus, D. Schnurbusch, J. Wüschner, and S. Ziegler, "On Tamaki's Algorithm to Compute Treewidths," p. 18, 2021.

[10] F. Pan, K. Chen, and P. Zhang, "Solving the sampling problem of the sycamore quantum circuits," *Physical Review Letters*, vol. 129, no. 9, p. 90502, 2022.

[11] X. Gao, M. Kalinowski, C.-N. Chou, M. D. Lukin, B. Barak, and S. Choi, "Limitations of linear cross-entropy as a measure for quantum advantage," *PRX Quantum*, vol. 5, no. 1, p. 10334, 2024.

[12] Z.-Y. Ni, Y.-S. Zhao, and J.-G. Liu, "Universal quantum computing with a single arbitrary gate." [Online]. Available: https://arxiv.org/abs/2409.20025