Tensor network algorithm for solving quantum physics on high-performance computing clusters

Student: Rocco Barač

Mentor: Simone Montangero

Università degli Studi di Padova

Funded by the European Union

AQTIVATE

The *AQTIVATE* project receives funding from the European Union's HORIZON MSCA Doctoral Networks programme, under Grant Agreement No. 101072344.

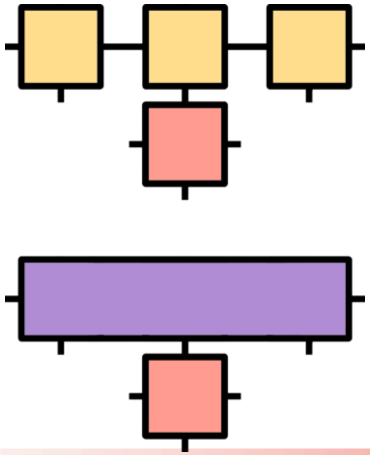
- Starting point: Quantum TEA library (Current main developers: Alice Pagano, Marco Ballarin, Nora Reinić, ...)

- Tensor networks are a classical way of simulating systems at low entanglement – useful for quantum hardware simulations (superconducting qubits, trapped ions, Rydberg atoms, etc.)

- Master thesis serves as an introduction for my PhD project (Project 8: Quantum computing and tensor networks for (2+1)D and (3+1)D QED)

Introduction

- Quantum many-body physics
- Many systems do not have analytical solutions
- Exponential Hilbert space growth problem – storing a state vector of 60 qubits already requires $2^{60} \sim 10^9$ GB of memory
- Numerical solutions using tensor networks (TNs) – time evolution and statics (such as ground state search)

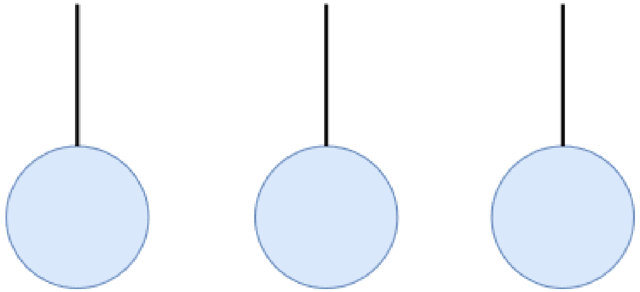


Tensor networks example
Image source:
https://en.wikipedia.org/wiki/Tensor_network

Tensor networks basics

- Mathematical framework for working with tensors
- Used in quantum many-body physics as a class of variational wave functions
- Easier to work with due to their intuitive diagrammatic notation
- Bond dimension: parameter controling the specific expressivity of a tensor network (TN)

$$|\psi_{MF}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$$



$$T^{s_1 s_2 s_3 s_4 s_5 s_6} = \sum_{\{\alpha\}} A^{s_1}_{\alpha_1} A^{s_2}_{\alpha_1\alpha_2} A^{s_3}_{\alpha_2\alpha_3} A^{s_4}_{\alpha_3\alpha_4} A^{s_5}_{\alpha_4\alpha_5} A^{s_6}_{\alpha_5}$$

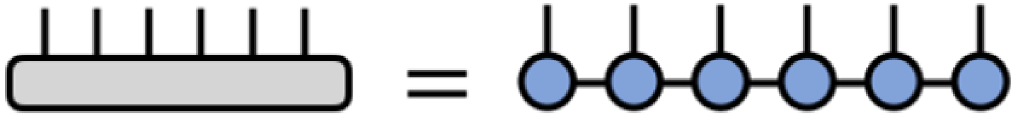
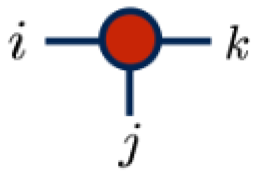


Image source: https://tensornetwork.org/mps/

Tensor network notation basics

- Low order tensors:

vector	v_j	
matrix	M_{ij}	
3-index tensor	T_{ijk}	

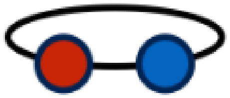
Source of images: https://tensornetwork.org/diagrams/

Tensor network notation basics

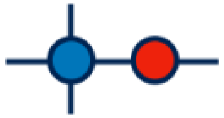
- Contractions examples:

$$= \sum_j M_{ij} v_j$$

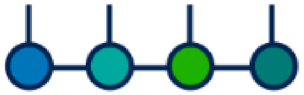
$$= A_{ij} B_{jk} = AB$$



$$= A_{ij} B_{ji} = \mathrm{Tr}[AB]$$



$$= \sum_k T_{ijkl} V_{km}$$

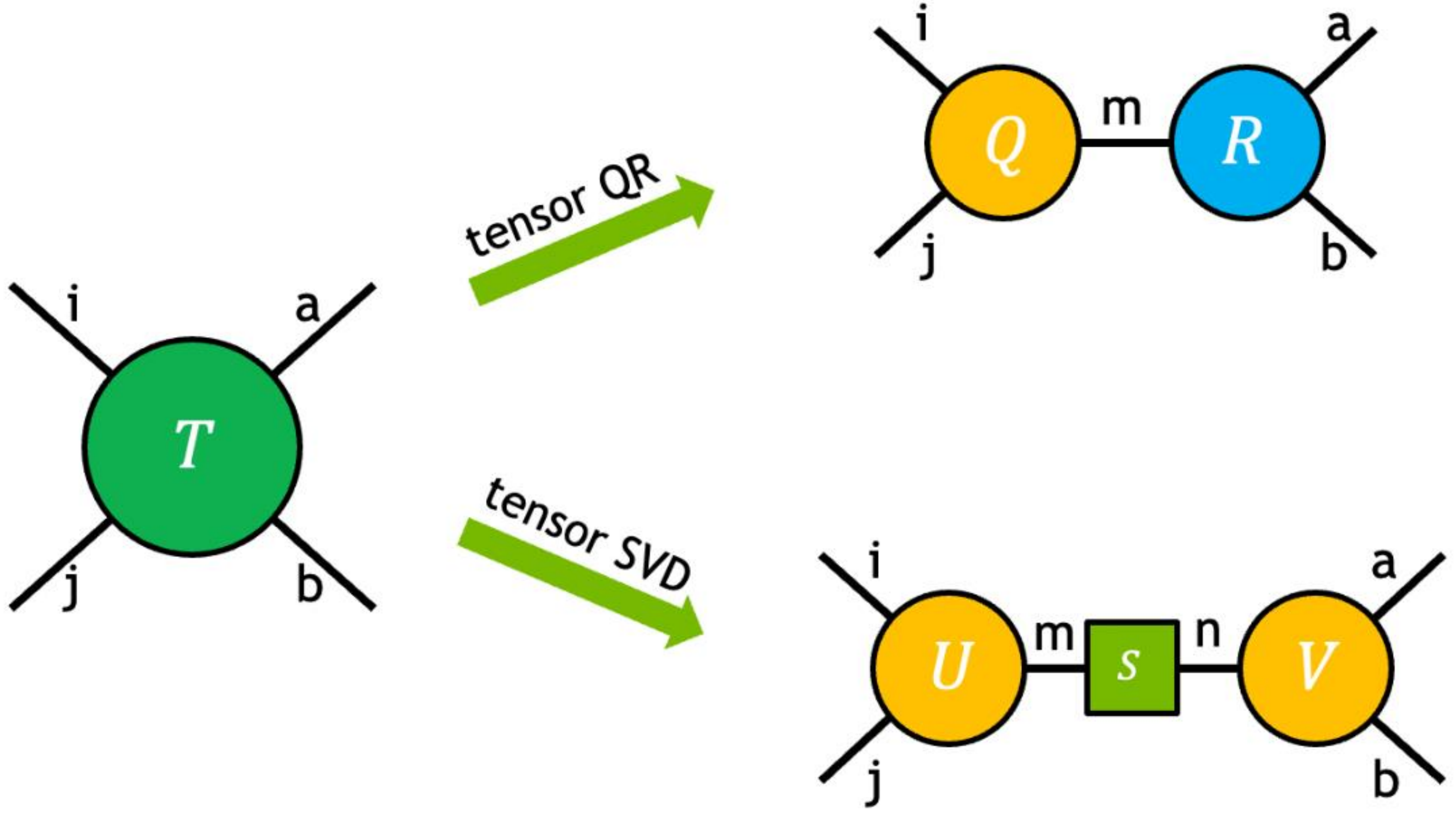


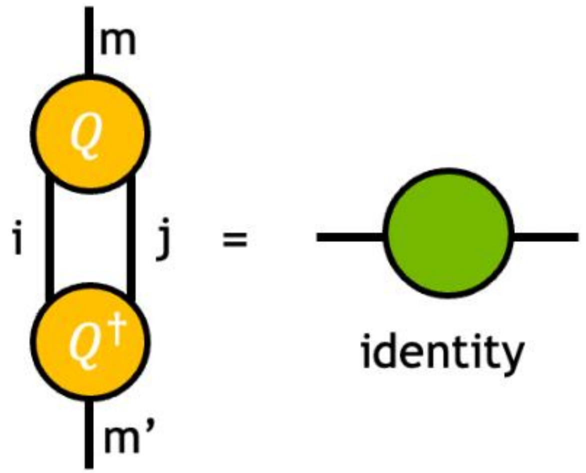
$$= \sum_{\alpha_1,\alpha_2,\alpha_3} A^{s_1}_{\alpha_1} B^{s_2}_{\alpha_1\alpha_2} C^{s_3}_{\alpha_2\alpha_3} D^{s_4}_{\alpha_3}$$

Source of images: https://tensornetwork.org/diagrams/

Tensor network notation basics

- Tensor decompositions:





Source: https://docs.nvidia.com/cuda/cuquantum/23.06.0/cutensornet/overview.html

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

$$\lambda_3 \ll 1$$

$$
\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix}
\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}
\begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}
$$

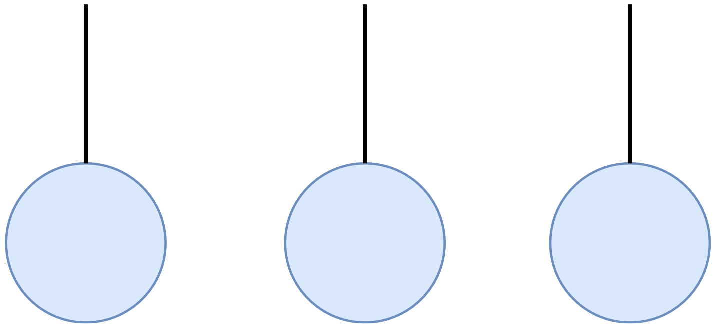
(2x2)x(2x2)x(3x2) => (2x2)

AQTIVATE

Matrix product state (MPS)

- factorization of a tensor with N indices into a chain-like product of three-index tensors

$$|\psi_{MF}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$$



$$T^{s_1 s_2 s_3 s_4 s_5 s_6} = \sum_{\{\alpha\}} A^{s_1}_{\alpha_1} A^{s_2}_{\alpha_1\alpha_2} A^{s_3}_{\alpha_2\alpha_3} A^{s_4}_{\alpha_3\alpha_4} A^{s_5}_{\alpha_4\alpha_5} A^{s_6}_{\alpha_5}$$

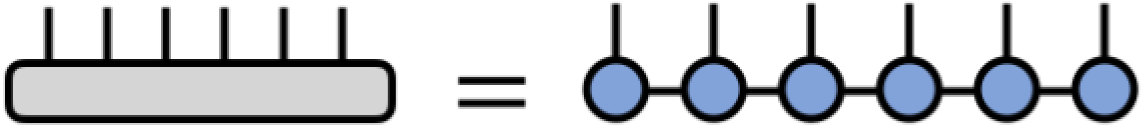


Image source: https://tensornetwork.org/mps/

Isometry center

- Isometrizing is done through a series of tensor decompositions and contractions
- Isometry centers have useful properties for simplifying many algorithms
- Isometrized TN = there is only one non-unitary tensor

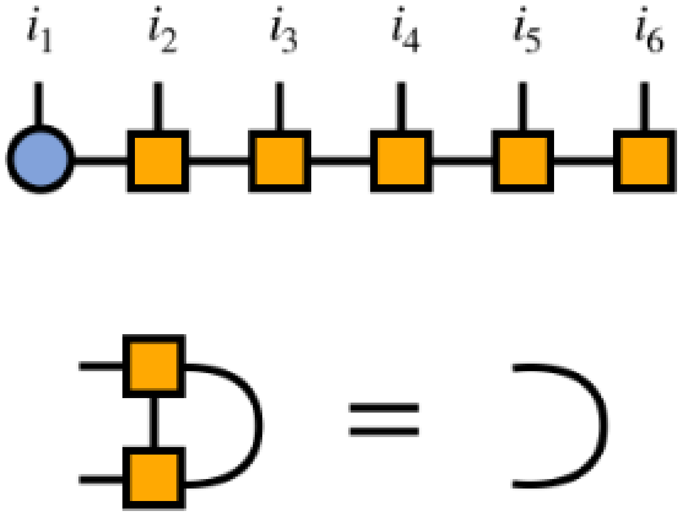
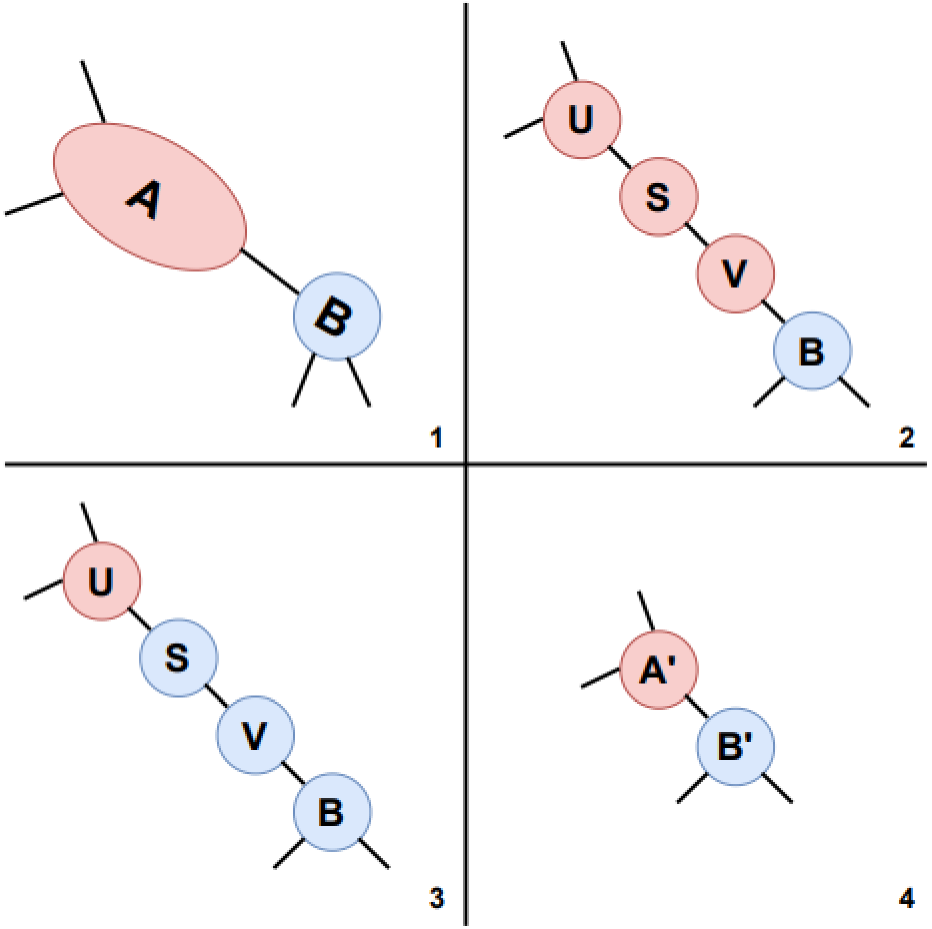
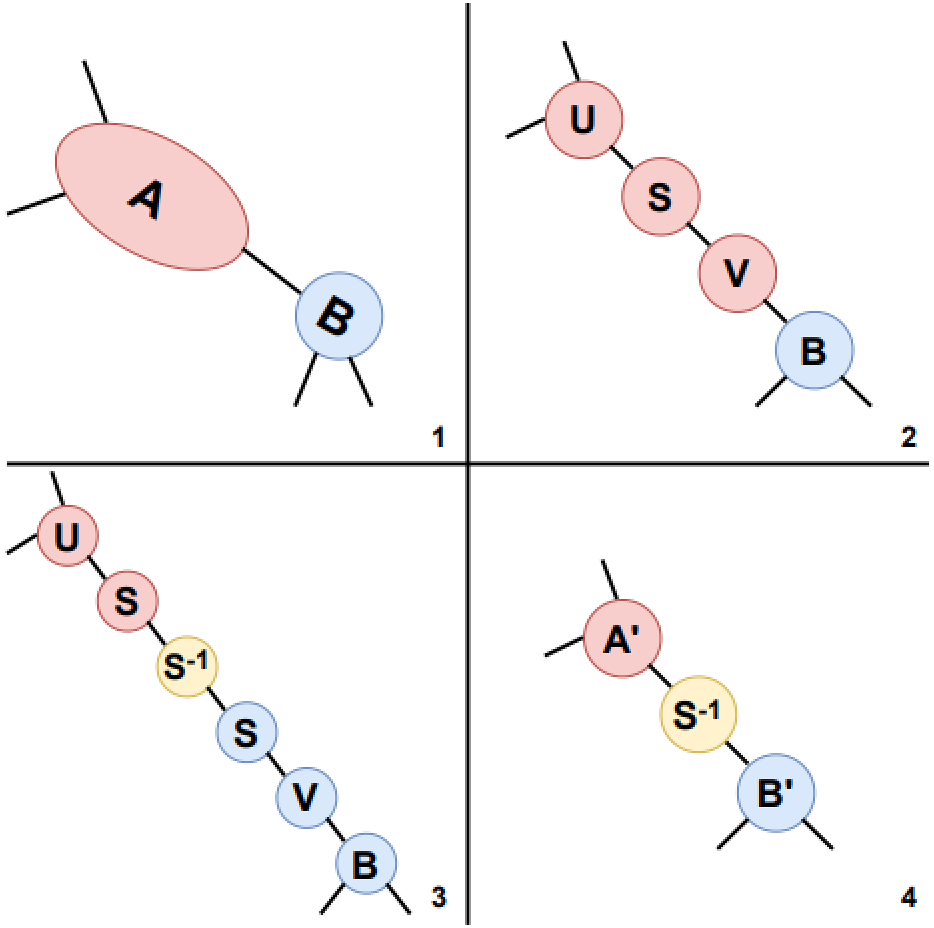


Image Source: https://tensornetwork.org/mps/algorithms/dmrg/



Moving the isometry center from A to B

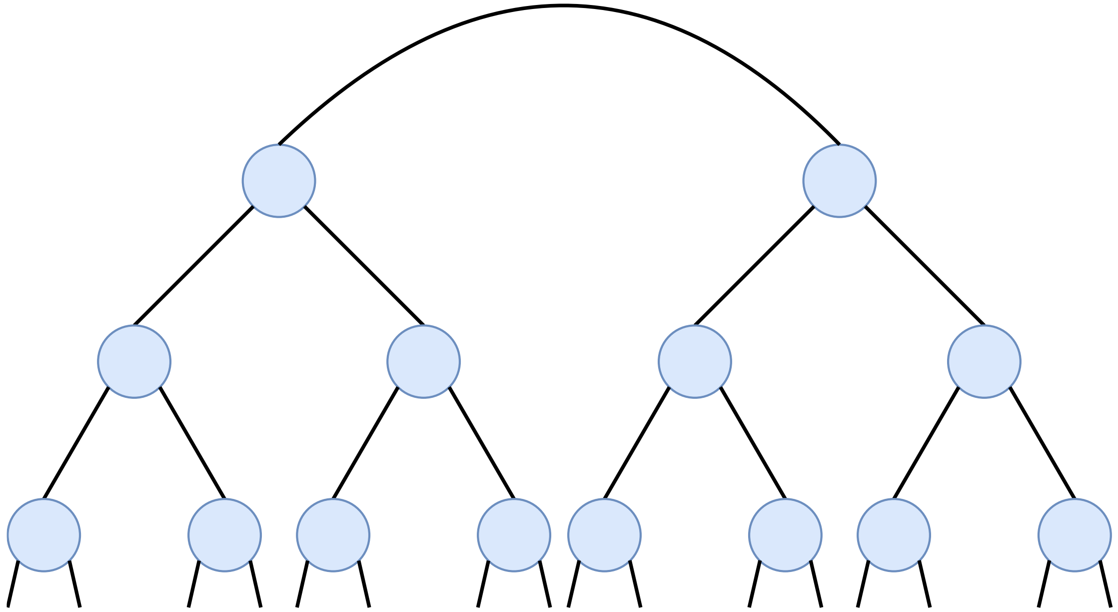


Splitting the isometry center from A into B

Tree Tensor Networks (TTNs)

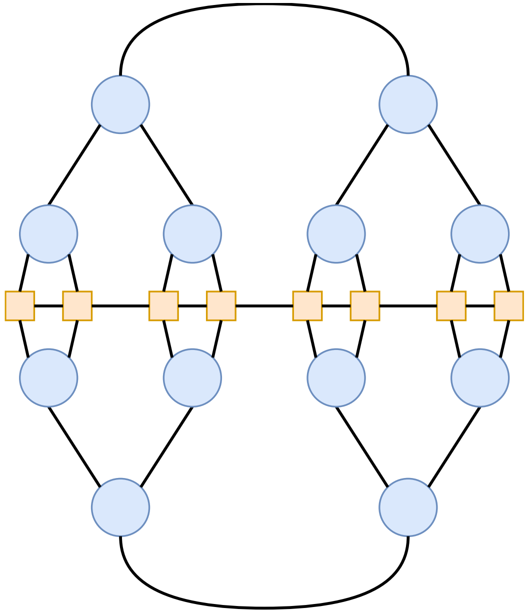
- Algorithms developed for this thesis are for manipulating TTNs

$$|\psi\rangle$$



3 layer TTN

$$\langle\psi|\, H \,|\psi\rangle$$



Hamiltonian expectation value calculation via TTN

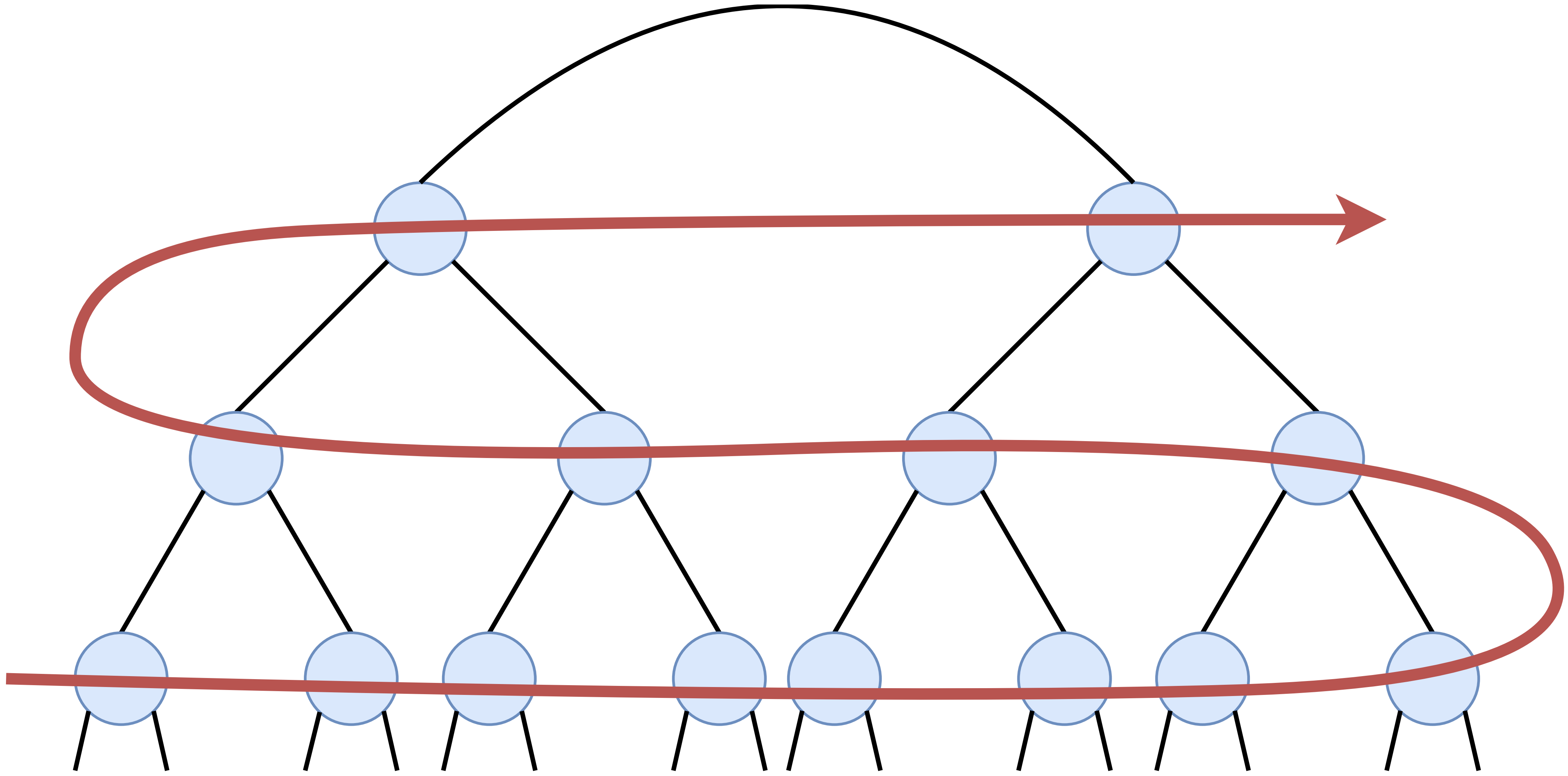
1D Quantum Ising model

- Theoretical model to study phase transitions
- A lattice is modeled as a chain of spins with an external magnetic field
- Hamiltonian has terms which correspond to neighboring spin interactions and effects of the external transverse magnetic field

$$H = -J \sum_i \sigma_i^z \sigma_{i+1}^z - g \sum_i \sigma_i^x$$

Serial algorithm TTN groundstate search (starting point for parallel)

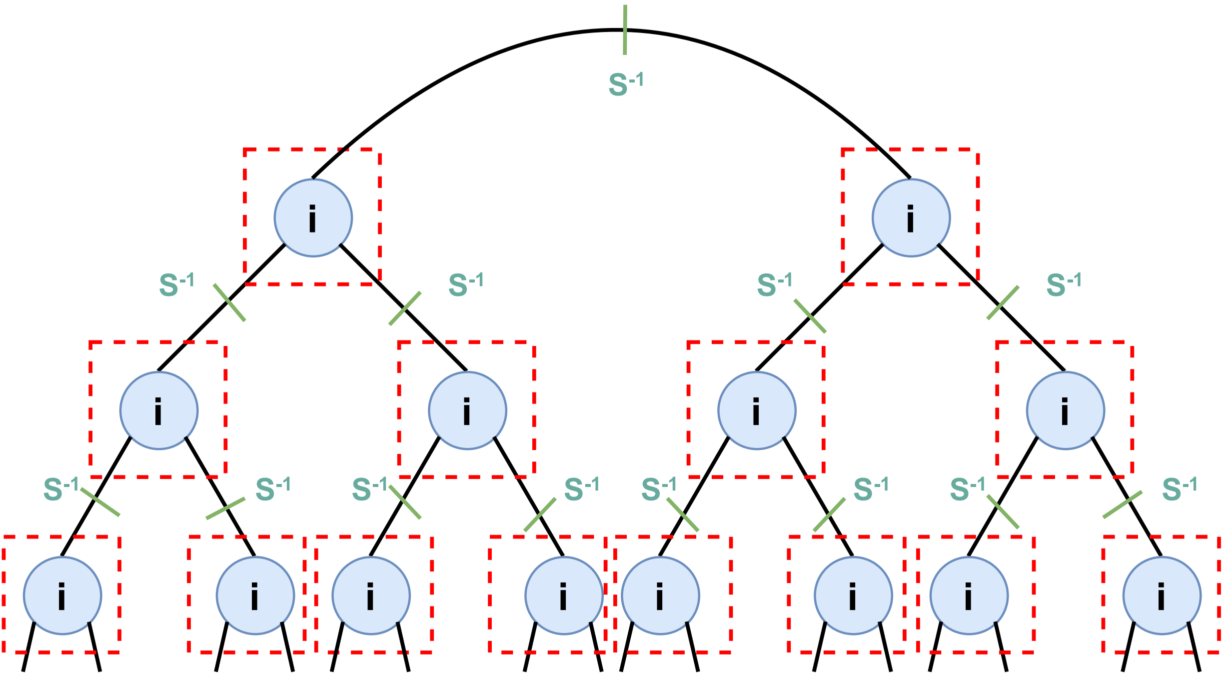
- Start with a random state TTN
- The operator used here is the quantum Ising model Hamiltonian
- Perform a series of single tensor updates (using Lanczos) by doing a sweep through the TTN (i.e. minimize tensors one by one while keeping the rest of the TTN constant)
- Iterate the previous step until converging to a result



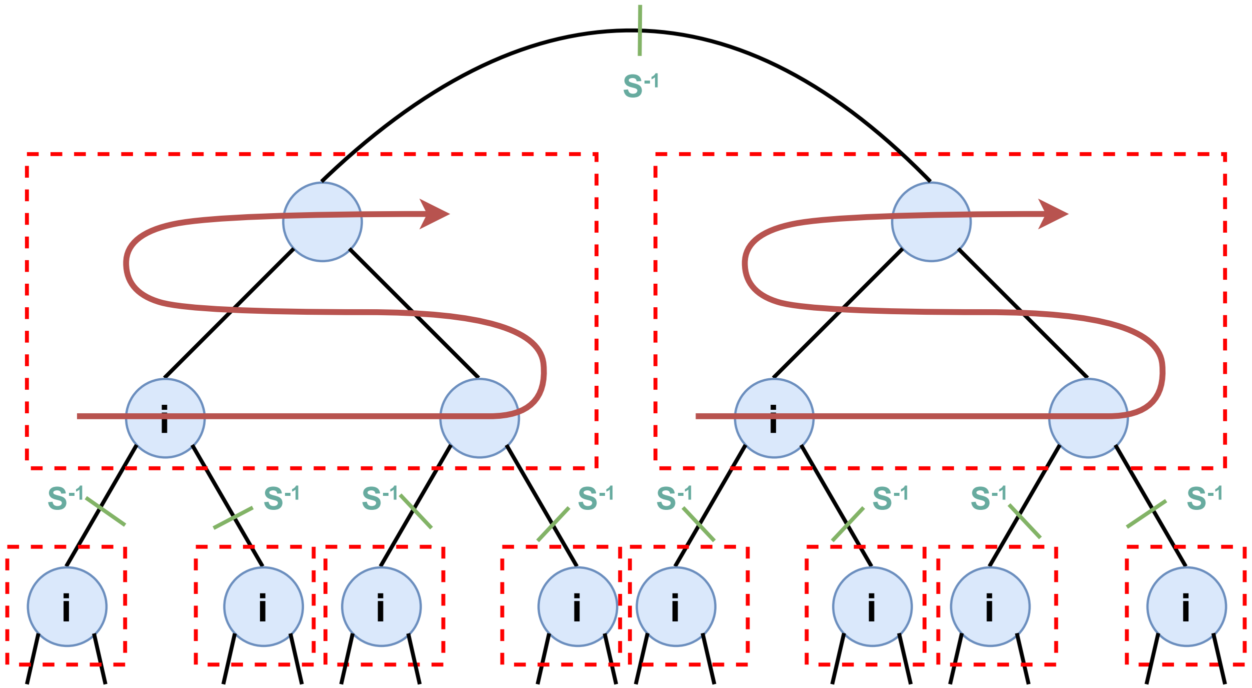
TTN optimization sweep, the isometry center is moved to the tensor currently being optimized

Parallel algorithm TTN groundstate search

- Start with a random state TTN
- The operator used is the quantum Ising model Hamiltonian
- Split the isometry center into all of the TTN tensors
- Perform simultaneous single tensor updates (using Lanczos) on all tensors on different threads.
- Update the TTN with the information coming in from all cores
- Iterate the previous 2 steps until converging to a result



Algorithm for the thesis was developed for a situation where each tensor is assigned its own thread.



Future algorithms should be able to combine both approaches because real-world computers have a limited number of available threads.

AQTIVATE

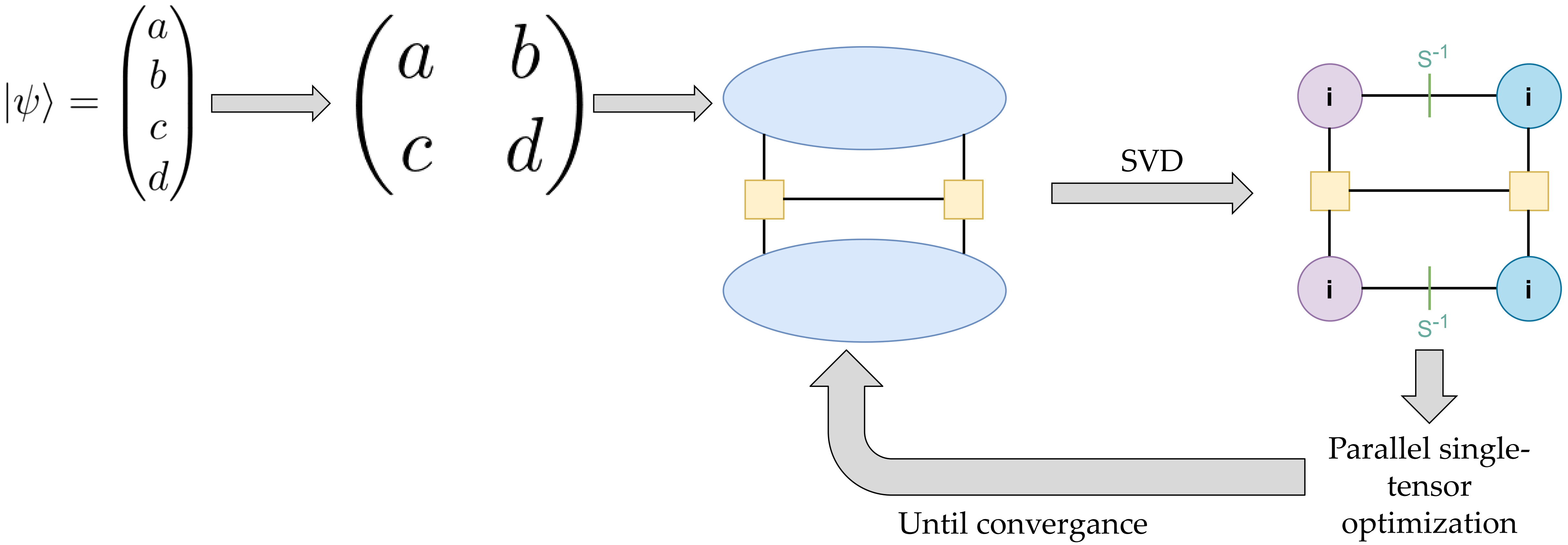
2-qubit example

- Simplest example
- 1D quantum Ising model Hamiltonian:

$$H = -J\sigma_x \otimes \sigma_x - g_1\sigma_z \otimes I_2 - g_2 I_2 \otimes \sigma_z$$

- This Hamiltonian is easily diagonalized => gives the exact ground state energy
- Also easily solvable using a serial algorithm

2-qubit example



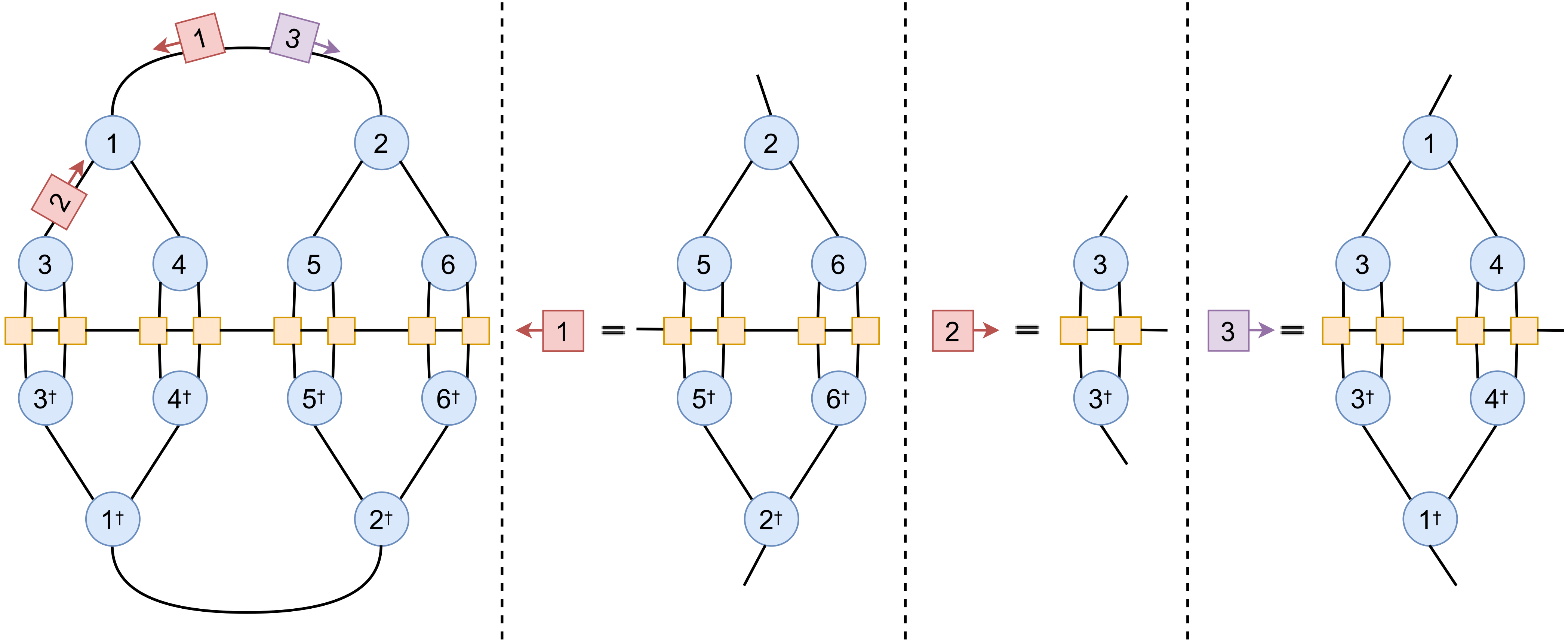
Conclusion

- Parallel algorithms have potential advantages over serial algorithms in solving many problems using tensor networks
- Further development of parallel algorithms should let us simulate quantum systems with tensor networks faster than ever before

Thank you!

Effective operators

- Contractions of (large)parts of a tensor network
- Usually done after isometrizing

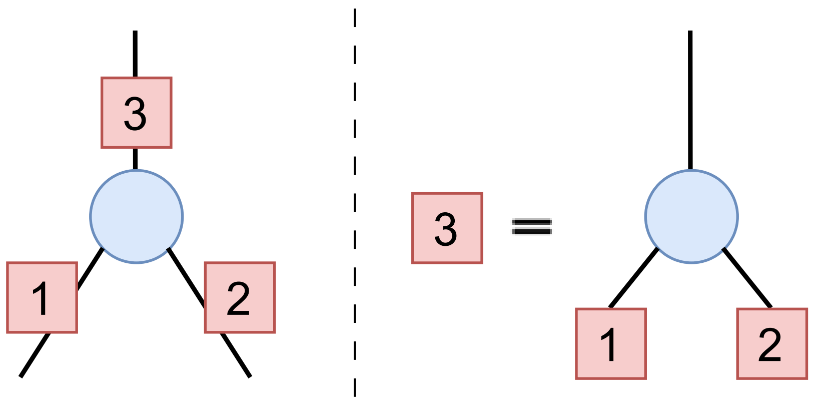


Lanczos algorithm

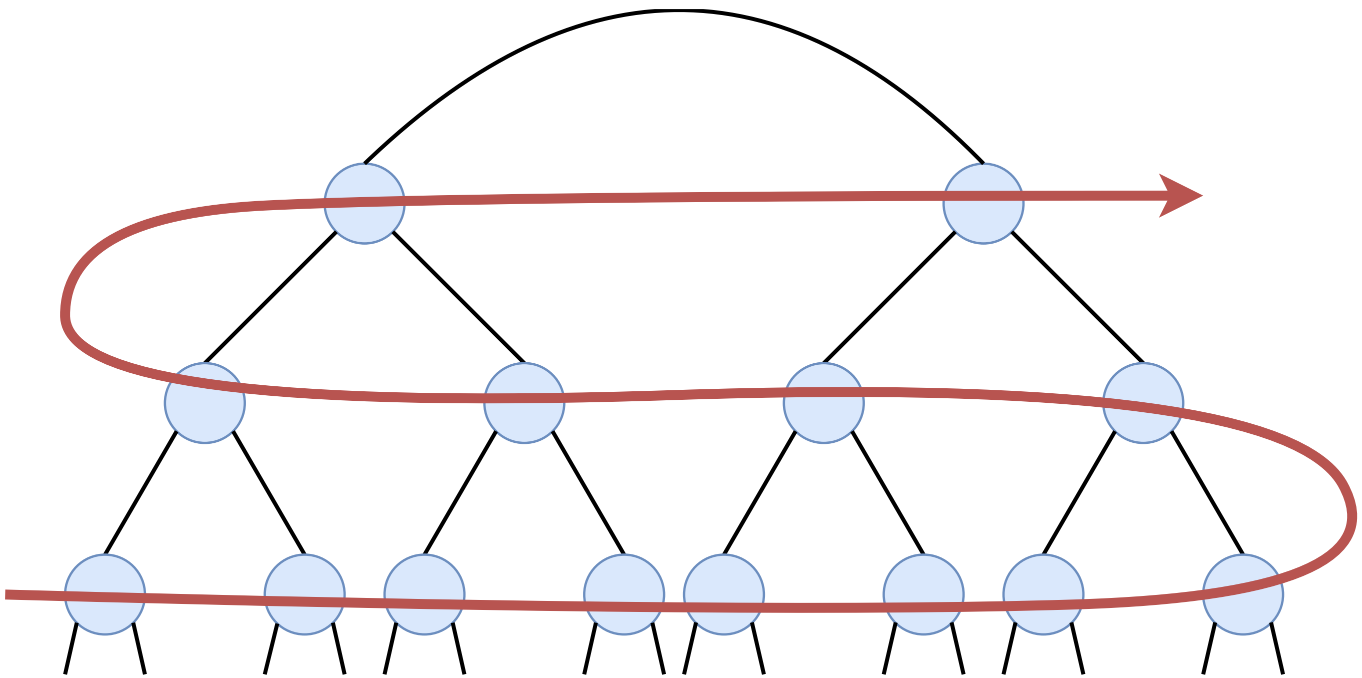
- Iterative algorithm
- It can be applied to the eigenproblem
- Applying an operator to a vector many times makes it converge to the operator's eigenvector
- The algorithm can be efficiently used to find some of the most extreme eigenvalues (useful in QM problems like ground state search)
- It can be applied to higher rank tensors by reshaping them to vectors

Serial algorithm TTN groundstate search (starting point for parallel)

- Start with a random state TTN
- The operator used here is the quantum Ising model Hamiltonian
- Isometrize towards the top left tensor
- Build effective operators (from bottom to top)
- Perform a series of single tensor updates (using Lanczos) by doing a sweep through the TTN
- Iterate the previous step until converging to a result



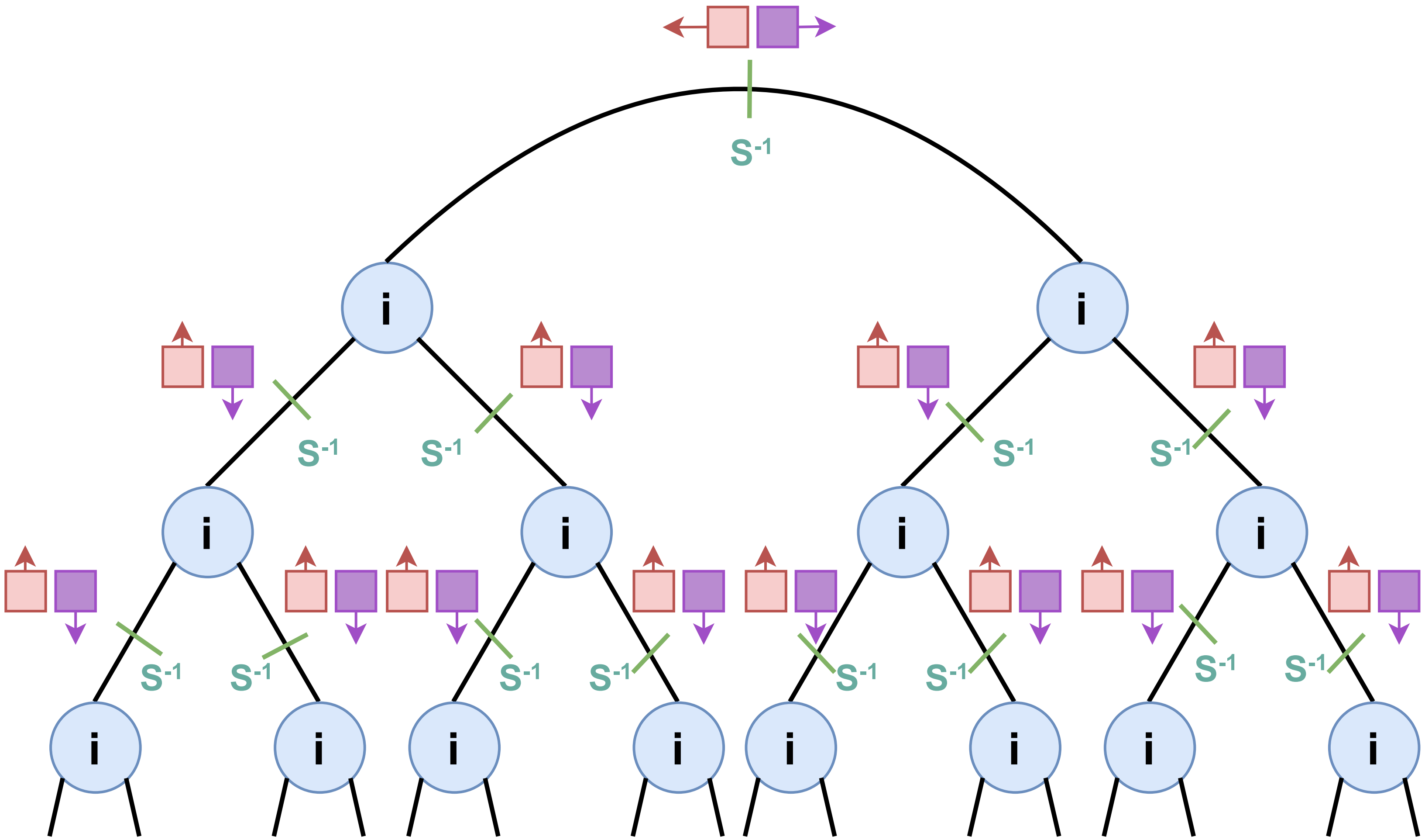
Building the effective operators in the TTN by using the effective operators from the layer below



TTN optimization sweep

Parallel algorithm TTN groundstate search

- Start with a random state TTN
- The operator used is the quantum Ising model Hamiltonian
- Isometrize towards the top left tensor
- Split the isometry center into all of the TTN tensors
- Build effective operators (from bottom to top in one direction and in the other direction while updating)
- Perform simultaneous single tensor updates (using Lanczos) on all tensors on different threads/cores
- Update the effective operators
- Iterate the previous 2 steps until converging to a result



TTN with the isometry centers split to each tensor, ready for parallel single tensor updates.