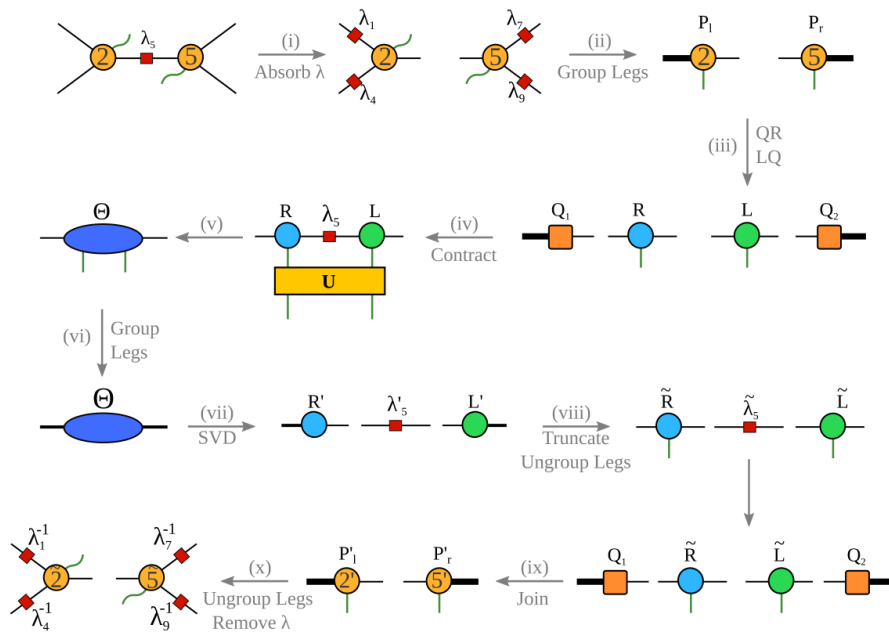


## BP-SU

The BP-SU algorithm consists of the following steps:

- ① `qbp(T_list, e_list) => m_list`
- ② `BP_gauging(T_list, e_dict, m_list) => T_list, w_dict`
- ③ `apply_2local_gate(T_list, e_list, w_list, U_list)`  
`=> T_list, w_dict`
- ④ `merge_SU_weights(T_list, w_list) => T_list`

# The Simple - Update procedure

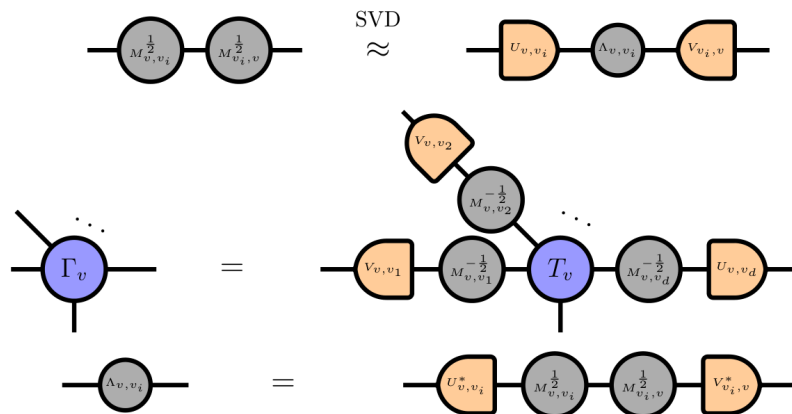


apply-2local-gate ( $T_i, T_j, \omega_i, \omega_j, \omega, op, \max D, eps$ )

# BP-Gauging

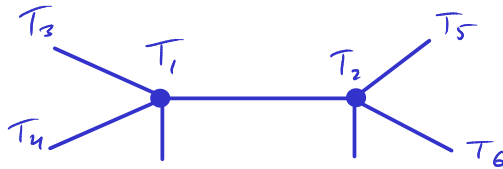
"Gauging tensor networks with belief propagation", Joseph Tindall and Matt Fishman,

SciPost Phys. 15, 222 (2023)

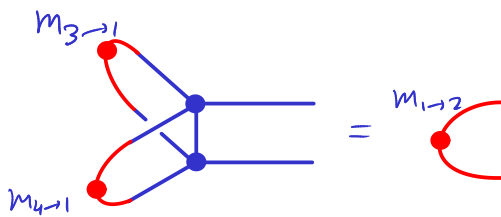


○

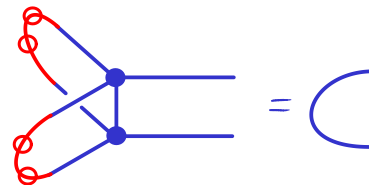
## Explanation



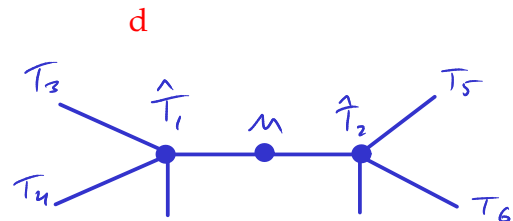
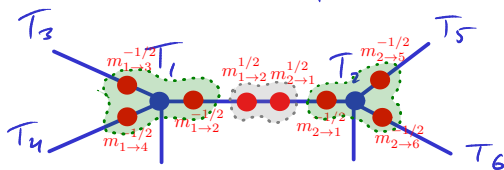
The BP fixed points assure:



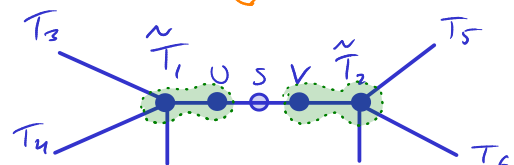
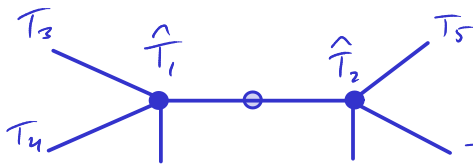
We want to take it to the Vidal Gauge:



We achieve this by:



⇓ SVD



The BP-Gauge fixing alg<sup>1</sup> should go like that:

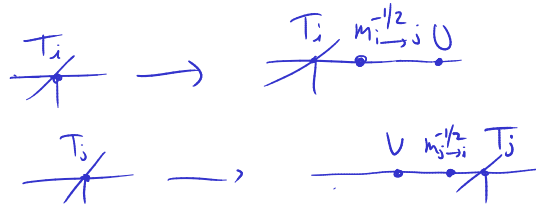
(\*) Go over all edges  $e = (i, j)$

1. Calc  $M_{i \rightarrow j}^{\pm 1/2}$ ,  $m_{j \rightarrow i}^{\pm 1/2}$

2. Calc  $M_{ij} = m_{i \rightarrow j}^{1/2} \cdot m_{j \rightarrow i}^{1/2}$

3. SVD:  $M_{ij} = U \cdot S \cdot V$

4. Absorb

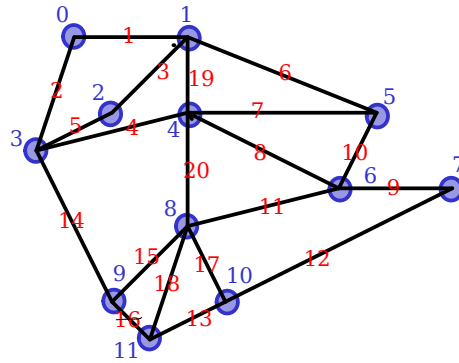


The diagram illustrates the absorption of a gauge field into a fermion line. It consists of two parts. The top part shows a fermion line with a vertex labeled  $T_i$  and a gauge field line labeled  $m_{i \rightarrow j}^{-1/2} U$ . The bottom part shows a fermion line with a vertex labeled  $T_j$  and a gauge field line labeled  $U m_{j \rightarrow i}^{-1/2}$ .

## Example on a random PEPS

The example-BP-SU.py file contains an example of using the BPSU library for running SU on a random PEPS (on some random graph), and using BP to move to the Vidal gauge.

The structure of the PEPS is as follows:



Note that blue labels denote the vertex number and red labels denote the edge labels