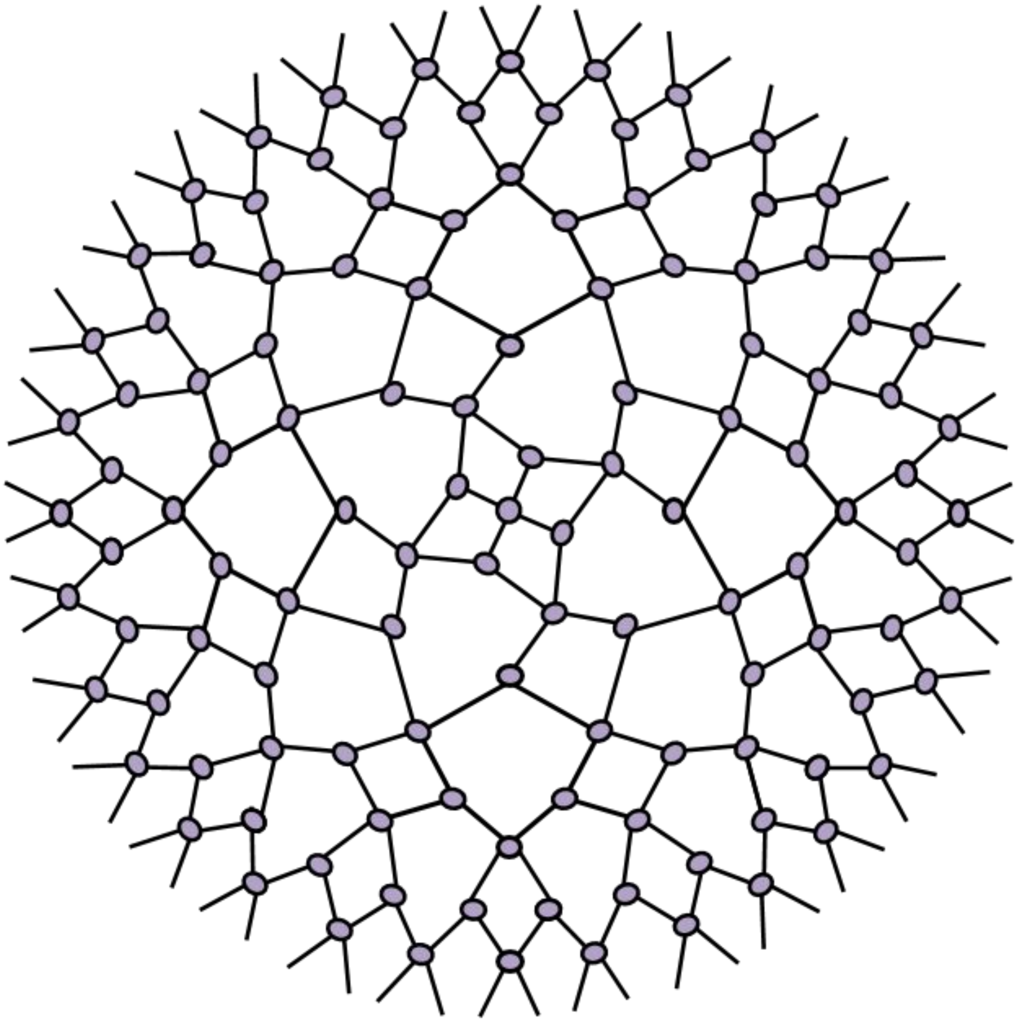


Tensor Contraction

JinGuo Liu

IOP T03 Journal Club

Table of contents

Flop

A flop is a floating point add, subtract, multiply, or divide

Algorithm 1.1.6 (Dot Product Matrix Multiplication) If $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{r \times n}$, and $C \in \mathbb{R}^{m \times n}$ are given, then this algorithm overwrites C with $C + AB$.

for $i = 1{:}m$
    for $j = 1{:}n$
        $C(i,j) = C(i,j) + A(i,:) \cdot B(:,j)$
    end
end

Operation	Dimension	Flops	Big O notation
$\alpha = x^T y$	$x, y \in \mathbb{R}^n$	$2n$	O(N)
$y = y + ax$	$a \in \mathbb{R},\ x, y \in \mathbb{R}^n$	$2n$	O(N)
$y = y + Ax$	$A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m$	$2mn$	$O(N^2)$
$A = A + yx^T$	$A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m$	$2mn$	$O(N^2)$
$C = C + AB$	$A \in \mathbb{R}^{m\times r},\ B \in \mathbb{R}^{r\times n},\ C \in \mathbb{R}^{m\times n}$	$2mnr$	$O(N^3)$

Table 1.1.2. *Important flop counts*

Einstein Summation

Quiz count flop for the following einsum notations

```
"i,i->i"        # element wise product
"i,i->"         # inner product
"i,j->ij"       # outer product

"ii->"          # trace
"ii->i"         # diag
"ij->ji"        # transpose
"ij->i"         # summation over columns
"ij,jk->ik"     # matmul
"ij,ik,il->jkl"   # star contraction

"ijk,jkl->il"   # tensordot
```

for $i = 1{:}m$
  for $j = 1{:}n$
    $C(i,j) = C(i,j) + A(i,:)\cdot B(:,j)$
  end
end

$$y_{ik} = A_{ij}B^j_k = \sum_j A_{ij}B_{jk}$$

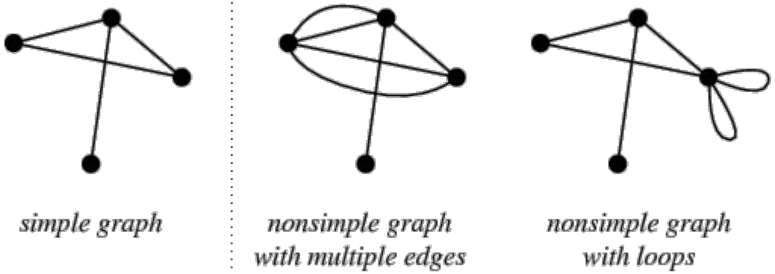
numpy notation: y = einsum("ij,jk->ik", A, B) # run over all indices, accumulate products to y.

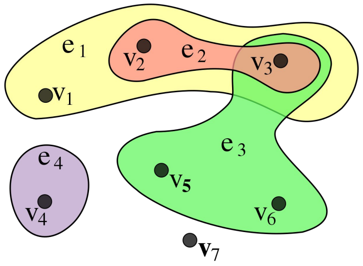
An Einsum is even more general than a Hypergraph

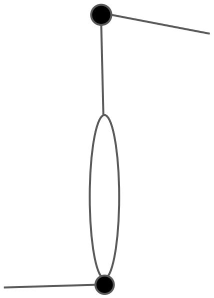
Simple Graph

Multigraph

Hypergraph







Most tensor network algorithms falls
into this category

star-contraction

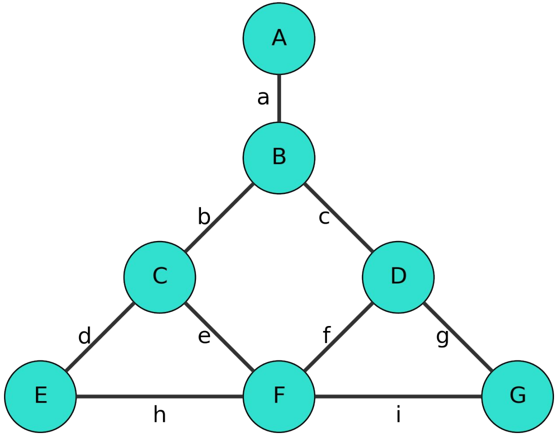
Duplicate indices
within a tensor

Table of contents

What about this one?

$O(N^9)$

less if we contract step by step

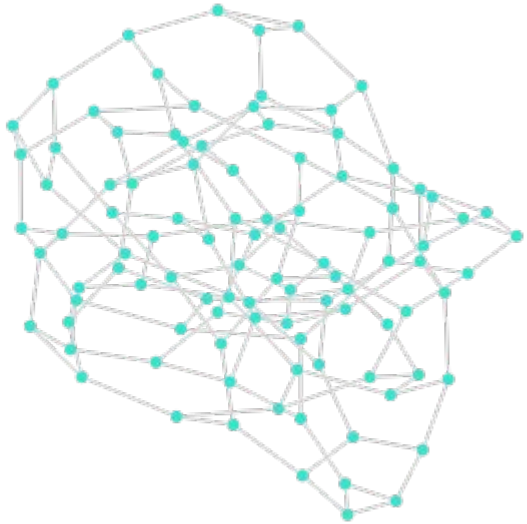



```
using TensorOperations

dims = [1, 3, 3, 3, 2, 4, 2]
A, B, C, D, E, F, G = map(nd->randn(fill(20, nd)...), dims)

@tensor begin
    Y1[b, c] := A[a] * B[a,b,c]           # N^3
    Y2[c,d,e] := Y1[b,c] * C[b,d,e]         # N^4
    Y3[d,e,f,g] := Y2[c,d,e] * D[c,f,g]       # N^5
    Y4[d,g,h,i] := Y3[d,e,f,g] * F[e,f,h,i]       # N^6
    Y5[g,i] := Y4[d,g,h,i] * E[d,h]       # N^4
    Y5[g,i] * G[g,i]        # N^2
end
```

y = einsum(“a,abc,bde,cfg,dh,efhi,gi->” A, B, C, D, E, F, G)

Quiz Find the optimal contraction order

Contracting a general structured tensor network efficiently



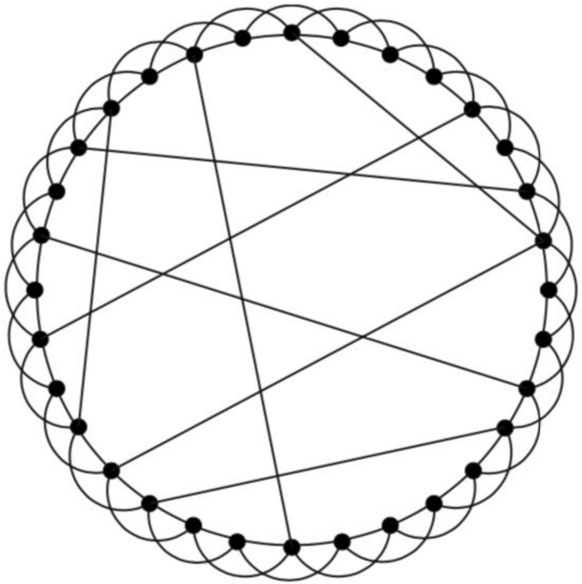
Applications

1. Obtaining the self energy of a randomly connected Ising model.
2. Simulating a Quantum Circuit
3. As a duality of a probabilistic graph model

Collective dynamics of 'small-world'networks

DJ **Watts**, SH **Strogatz** - nature, 1998 - nature.com

Networks of coupled dynamical systems have been used to model biological oscillators 1, 2, 3, 4, Josephson junction arrays 5, 6, excitable media 7, neural networks 8, 9, 10, spatial games 11, genetic control networks 12 and many other self-organizing systems. Ordinarily …



1. Watts-Strogatz model

Features

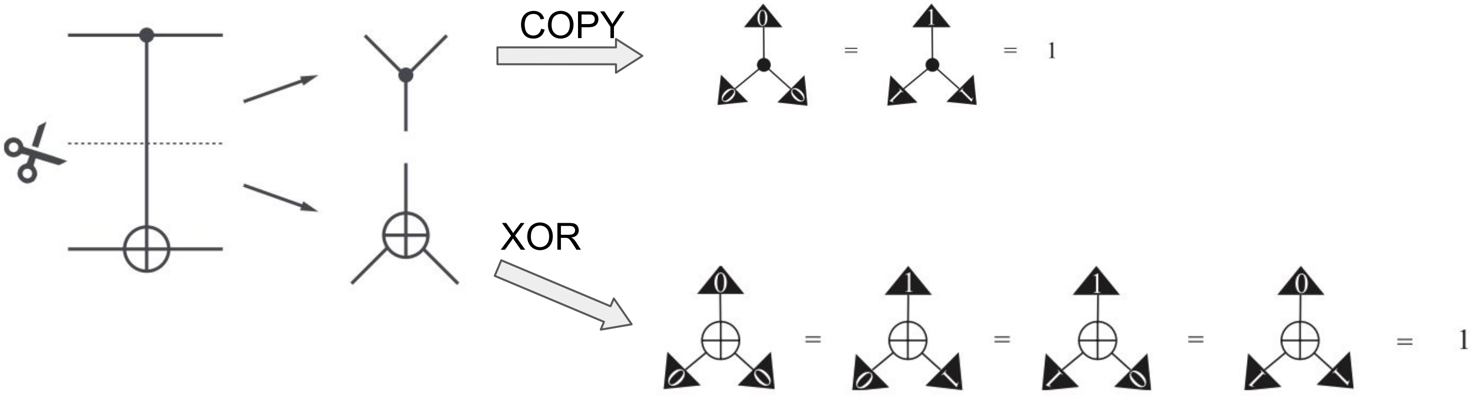
- short average path lengths
- high clustering

Applications

- power grid
- neural network of C. elegans
- networks of movie actors

Dorogovtsev, Sergey N., Alexander V. Goltsev, and José FF Mendes. "Critical phenomena in complex networks." *Reviews of Modern Physics* 80.4 (2008): 1275.

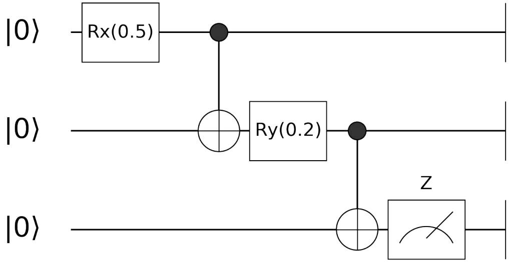
2. Quantum Circuits



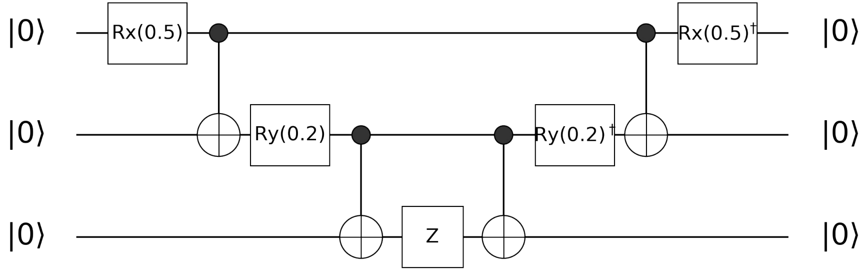
Markov, I. L., & Shi, Y. (n.d.). Simulating quantum computation by contracting tensor networks, 1–21.

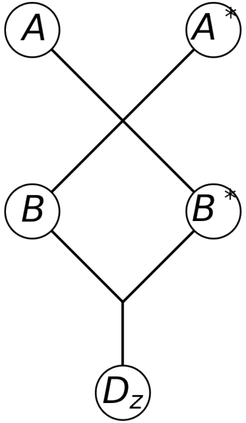
Biamonte, Jacob, and Ville Bergholm. "Tensor networks in a nutshell." *arXiv preprint arXiv:1708.00006* (2017).

Example: $\langle Z \rangle = \langle \psi | Z | \psi \rangle$

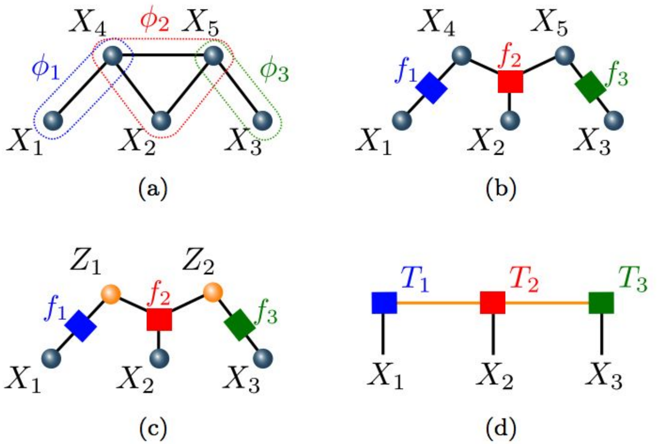



```
A := Rx(0.5)[:,1]
B := Ry(0.2)
Dz := diag(Z)
einsum("i,i,ij,ij,j->", A,conj(A),B,conj(B),Dz)
```





3. Probability Graph and Tensor Networks



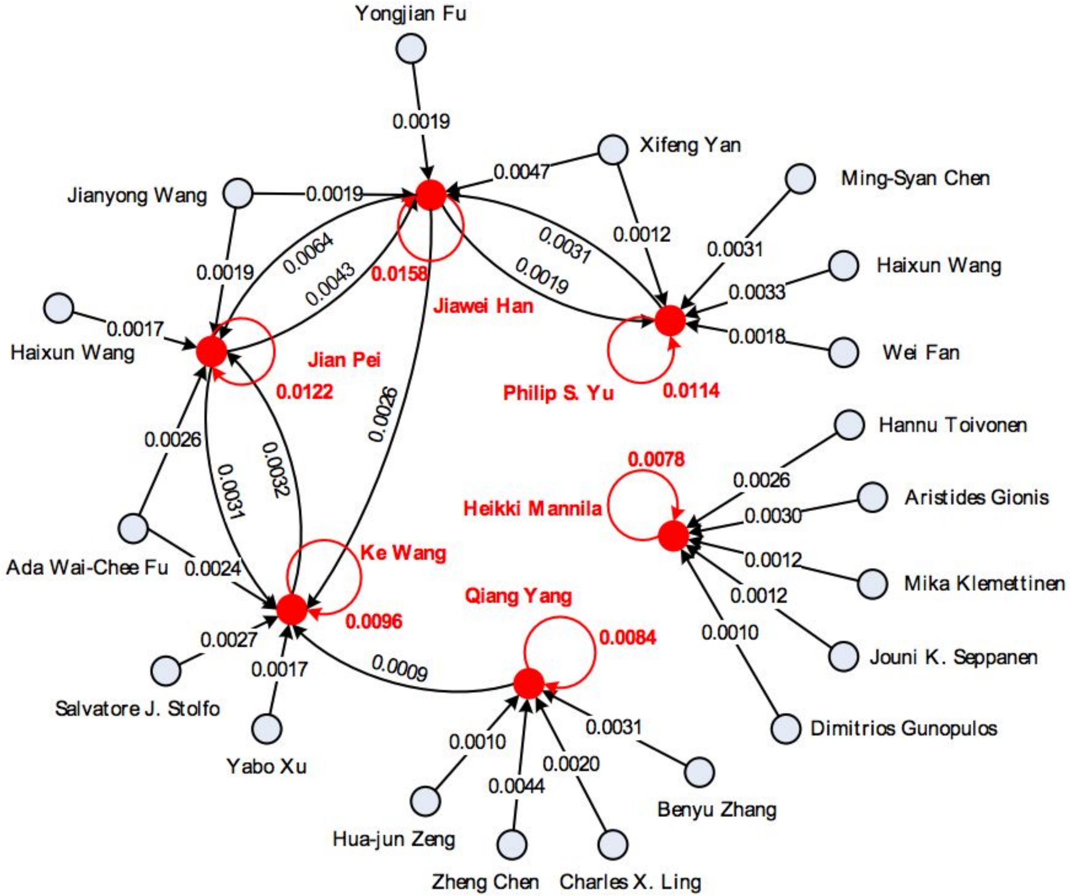
$$p(X_1, X_2, X_3, X_4, X_5) = \phi_1(X_1, X_4)\phi_2(X_2, X_4, X_5)\phi_3(X_3, X_5)$$
$$\text{or } f_1(X_1, X_4) f_2(X_2, X_4, X_5) f_3(X_3, X_5)$$

Marginalize X_4 and X_5

$$p(X_1, X_2, X_3) = \sum_{X_4, X_5} f_1(X_1, X_4) f_2(X_2, X_4, X_5) f_3(X_3, X_5)$$
$$= \sum_{X_4, X_5} T_1^{X_1, X_4} T_2^{X_2, X_4, X_5} T_3^{X_3, X_5}$$

- Glasser, Ivan, Nicola Pancotti, and J. Ignacio Cirac. "Supervised learning with generalized tensor networks." *arXiv preprint arXiv:1806.05964* (2018).
- Robeva, Elina, and Anna Seigal. "Duality of graphical models and tensor networks." *arXiv preprint arXiv:1710.01437* (2017).
- Gao, Xun, Zhengyu Zhang, and Luming Duan. "An efficient quantum algorithm for generative machine learning." *arXiv preprint arXiv:1711.02038* (2017).

Social community analysis



Yang, Zi, et al. "Social community analysis via a factor graph model." *IEEE Intelligent Systems* 3 (2010): 58-65.

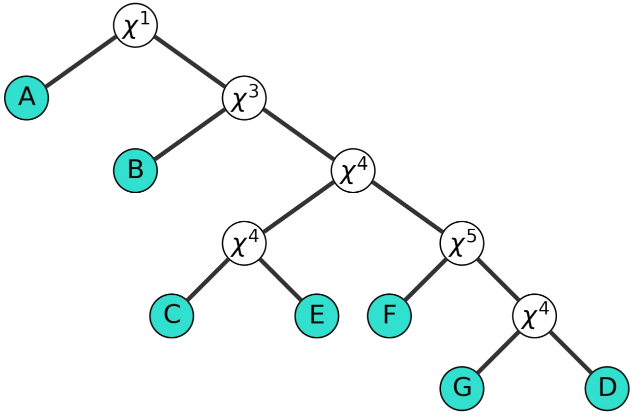
Table of contents

A domain-specific algorithm: NCon

```
julia> @optimalcontractiontree A[a] * B[a,b,c] * C[b,d,e] * D[c,f,g] * E[d,h] *
F[e,f,h,i] * G[g,i]
((1, (2, ((3, 5), (6, (7, 4))))), 1*χ^5 + 3*χ^4 + 1*χ^3 + 0*χ^2 + 1*χ + 0)
```

contraction tree

cost



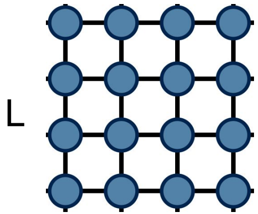
Pfeifer, Robert NC, Jutho Haegeman, and Frank Verstraete. "Faster identification of optimal contraction sequences for tensor networks." *Physical Review E* 90.3 (2014): 033315.

https://github.com/Jutho/TensorOperations.jl

- ✓ Multiple edges
- × Selfloops
- × Hyperedges

Treewidth based approach

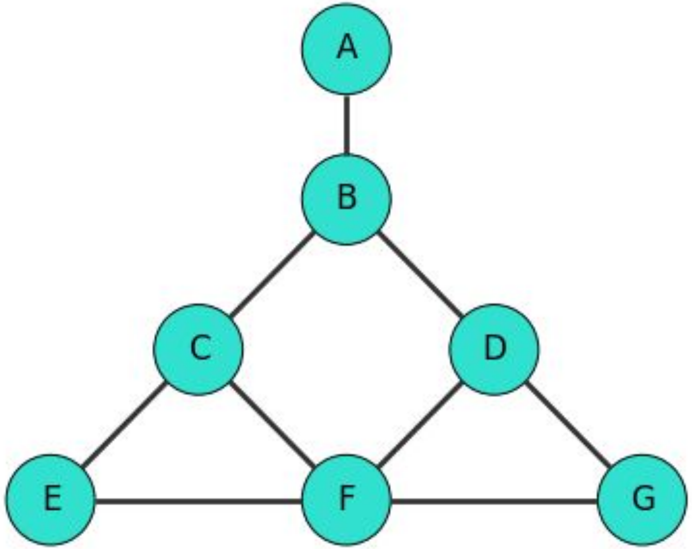
Definition 4.1. The *contraction of an edge* e removes e and replaces its end vertices (or vertex) with a single vertex. A *contraction ordering* π is an ordering of all the edges of G, $\pi(1), \pi(2), \ldots, \pi(|E(G)|)$. The complexity of π is the maximum degree of a merged vertex during the contraction process. The *contraction complexity* of G, denoted by $\mathrm{cc}(G)$, is the minimum complexity of a contraction ordering.



Quiz what is cc of this tensor network?

Markov, I. L., & Shi, Y. (n.d.). Simulating quantum computation by contracting tensor networks, 1–21.

Proposition 4.2. *For any graph $G = (V, E)$, $\mathrm{cc}(G) = \mathrm{tw}(G^*)$. Furthermore, given a tree decomposition of G^* of width d, there is a deterministic algorithm that outputs a contraction ordering π with $\mathrm{cc}(\pi) \leq d$ in polynomial time.*



Line graph G*

$$V(G^*) \stackrel{\text{def}}{=} E(G)$$

$$E(G^*) \stackrel{\text{def}}{=} \{\{e_1, e_2\} \subseteq E(G) : e_1 \neq e_2, \exists v \in V(G)$$

Contraction ordering of G is equivalent to **elimination ordering** of G*

Definition (Treewidth (tw)). *A* tree decomposition *of a graph G is a tree T with a function f mapping nodes in T to* bags *(sets) of vertices from G, such that the following conditions hold:*

1. *All vertices are represented:* $\bigcup_{t \in V(T)} f(t) = V(G)$.

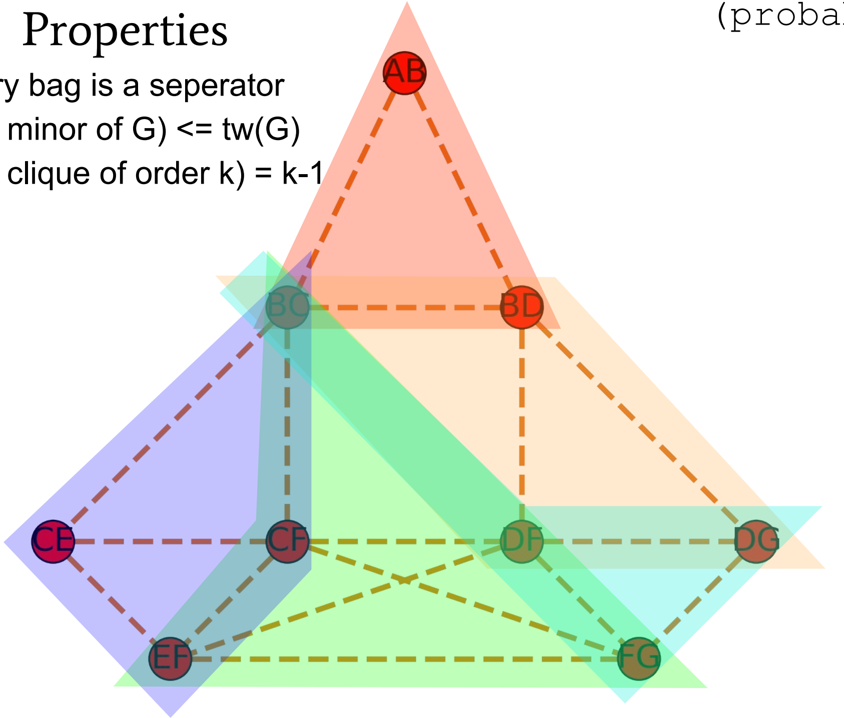
2. *All edges are represented:* $\forall (u, v) \in E(G), \exists t \in V(T)$ *s.t.* $u, v \in f(t)$.

3. *Graph vertices induce a (connected) subtree of T: if $w \in f(r) \cap f(s)$ for $r, s \in V(T), w \in V(G)$, then $w \in f(t)$ for all t on the path from r to s in T.*

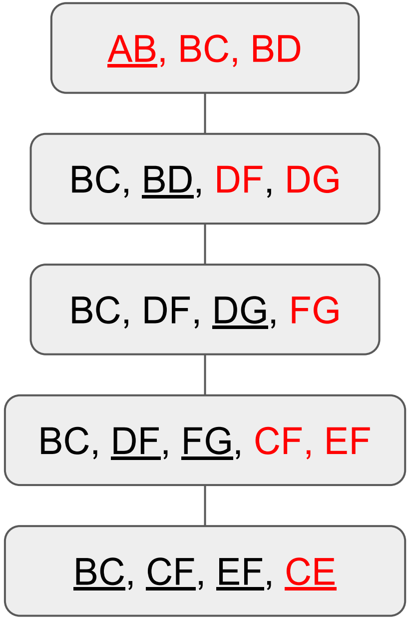
The width *of a tree decomposition is* $\max_{t \in V(T)} |f(t)| - 1$, *and the* treewidth *of a graph G, denoted $tw(G)$, is the minimum width over all valid tree decompositions of G.*

Properties

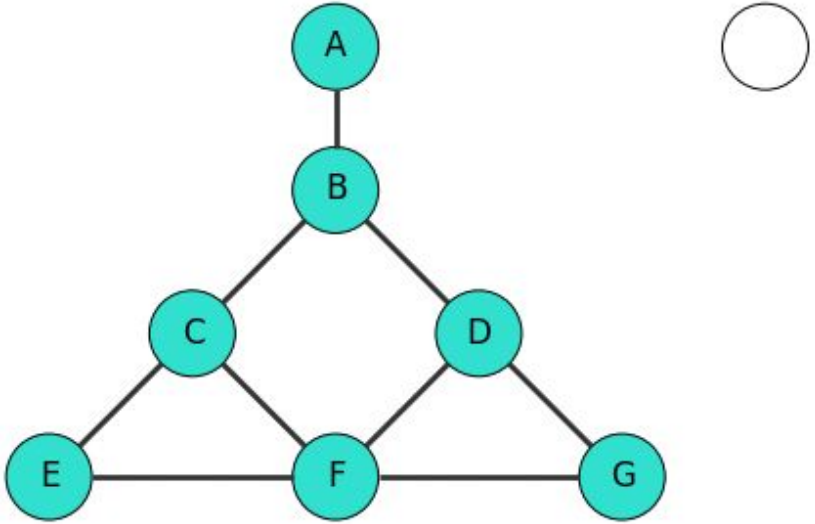
- Every bag is a seperator
- tw(a minor of G) <= tw(G)
- tw(a clique of order k) = k-1



(probably) the simplest tree decomposition



Elimination Order: AB-2, BD-3, DG-3, (DF-4, FG-3), (BC-3, CF-2, EF-1,CE-0)

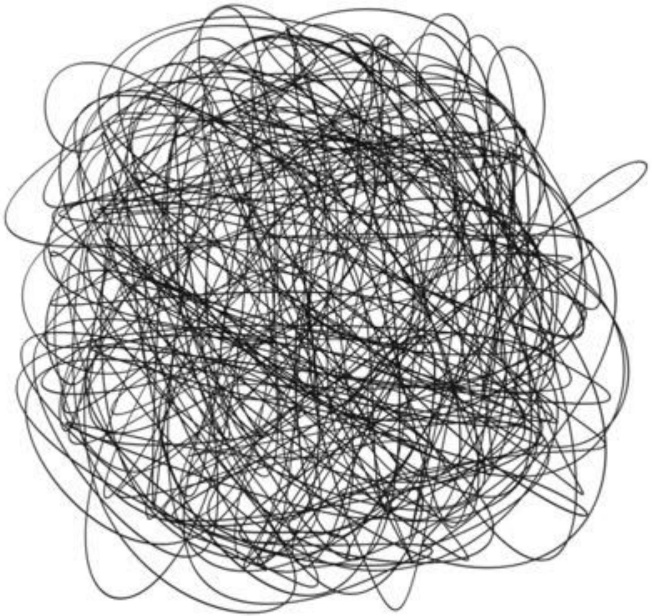


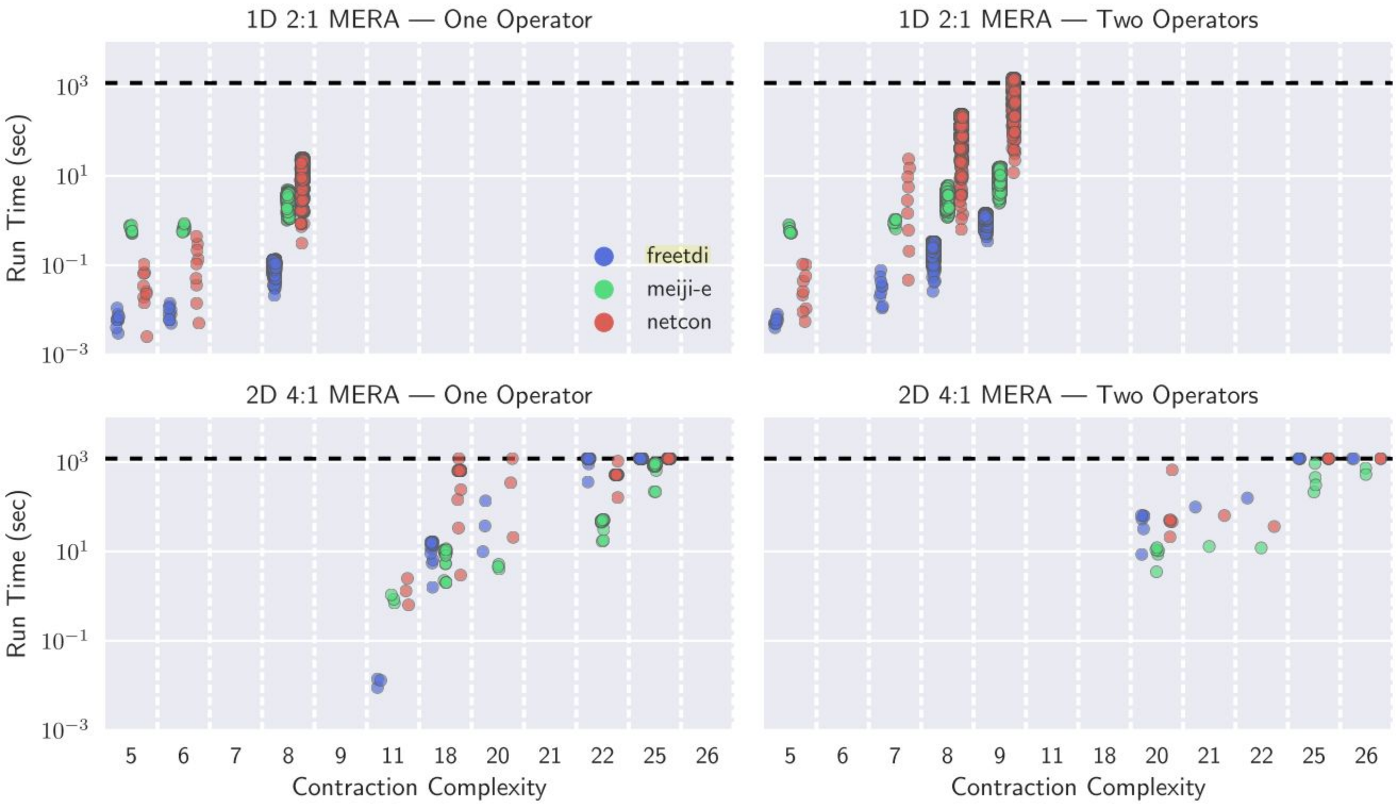
✓ Multiple edges
✓ Selfloops
✓ Hyperedges

Elimination Order: AB-2, BD-3, DG-3, (DF-4, FG-3), (BC-3, CF-2, EF-1,CE-0)
Contraction Order: AB-2, BD-3, DG-3, (DF-4, FG-3), (BC-3, CF-2, EF-1,CE-0)

Note: A selfloop counts one edge

Finding the optimal treewidth

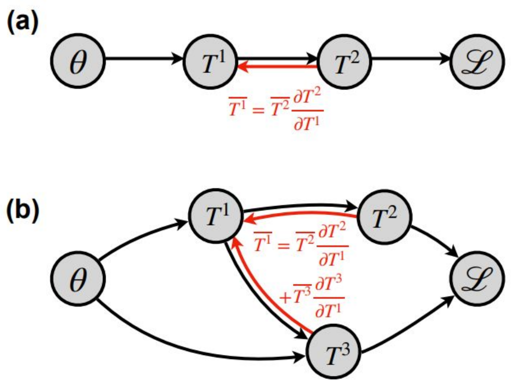




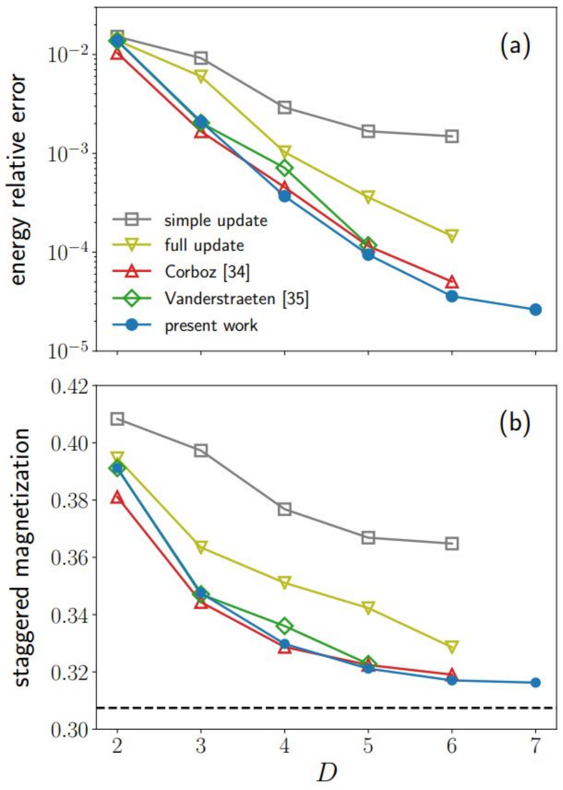
Dumitrescu, E. F., Fisher, A. L., Goodrich, T. D., Travis, S., Sullivan, B. D., & Wright, A. L. (n.d.). Benchmarking treewidth as a practical component of tensor-network–based quantum simulation, 1–20.

Table of contents

Automatic differentiation



$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial x_i}$$



Liao, H., Liu, J., Wang, L., & Xiang, T. (n.d.). Differentiable Programming Tensor Networks, 1–10.

Autodiff $\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial x_i}$ for einsum

```
"i,i->i"       # element wise product
"i,i->"        # inner product
"i,j->ij"      # outer product

"ii->"         # trace
"ii->i"        # diag
"ij->ji"       # transpose
"ij->i"        # summation over columns
"ij,jk->ik"    # matmul
"ij,ik,il->jkl"  # star contraction

"ijk,jkl->il"  # tensordot
```

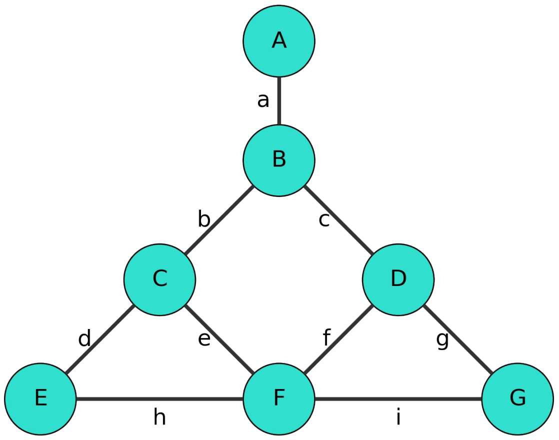
```
"i,i->i"
",i->i"
"ij,j->i"

"->ii"
"i->ii"
"ji->ij"
"i->ij"
"ik,jk->ij"
"jkl,ik,il->ij"

"il,jkl->ijk"
```

BP Rules

1. Exchange indices of x_i and indices of y in the code
2. Feed dL/dy as new input
3. Supply output shape if necessary
4. Output is dL/dx

Proof: https://giggleliu.github.io/2019/04/02/einsumbp.html

What about this one?



y = einsum(“a,abc,bde,cfg,dh,efhi,gi->” A, B, C, D, E, F, G)

Quiz Get dL/dC from dL/dy

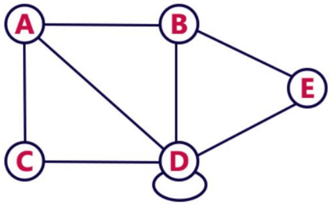
Reading Materials

1. Cichocki, A. (2013). Tensor Networks for Big Data Analytics and Large-Scale Optimization Problems, 1–36.
2. Biamonte, J., & Bergholm, V. (n.d.). Quantum Tensor Networks in a Nutshell, 1–34.
3. Stoudenmire, E. M., & Schwab, D. J. (2016). Supervised Learning with Quantum-Inspired Tensor Networks. arXiv Preprint, 11. Retrieved from http://arxiv.org/abs/1605.05775
4. Han, Z.-Y., Wang, J., Fan, H., Wang, L., & Zhang, P. (2017). Unsupervised Generative Modeling Using Matrix Product States, 1–11. Retrieved from http://arxiv.org/abs/1709.01662
5. Markov, I. L., & Shi, Y. (n.d.). Simulating quantum computation by contracting tensor networks, 1–21.
6. Dumitrescu, E. F., Fisher, A. L., Goodrich, T. D., Travis, S., Sullivan, B. D., & Wright, A. L. (n.d.). Benchmarking treewidth as a practical component of tensor-network–based quantum simulation, 1–20.
7. Tamaki, H. (n.d.). Positive-instance driven dynamic programming for treewidth ⋆, 1–21.
8. Liao, H., Liu, J., Wang, L., & Xiang, T. (n.d.). Differentiable Programming Tensor Networks, 1–10.
9. Gao, X., Zhang, Z., & Duan, L. (2017). An efficient quantum algorithm for generative machine learning.

Thanks

The data structure of a graph

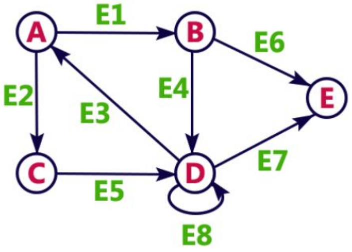
Adjacency Matrix



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

× Multiple edges
✓ Selfloops
× Hyperedges

Incidence Matrix



	E1	E2	E3	E4	E5	E6	E7	E8
A	1	1	-1	0	0	0	0	0
B	-1	0	0	1	0	1	0	0
C	0	-1	0	0	1	0	0	0
D	0	0	1	-1	-1	0	1	1
E	0	0	0	0	0	-1	-1	0

✓ Multiple edges
✓ Selfloops
✓ Hyperedges