From Treewidth to Tensor Network Contraction Order

Xuanzhao Gao

The Hong Kong University of Science and Technology

Acknowledgement

Instructor: Jin-Guo Liu
HKUST(GZ)

GSoC 2024

The Julia Language

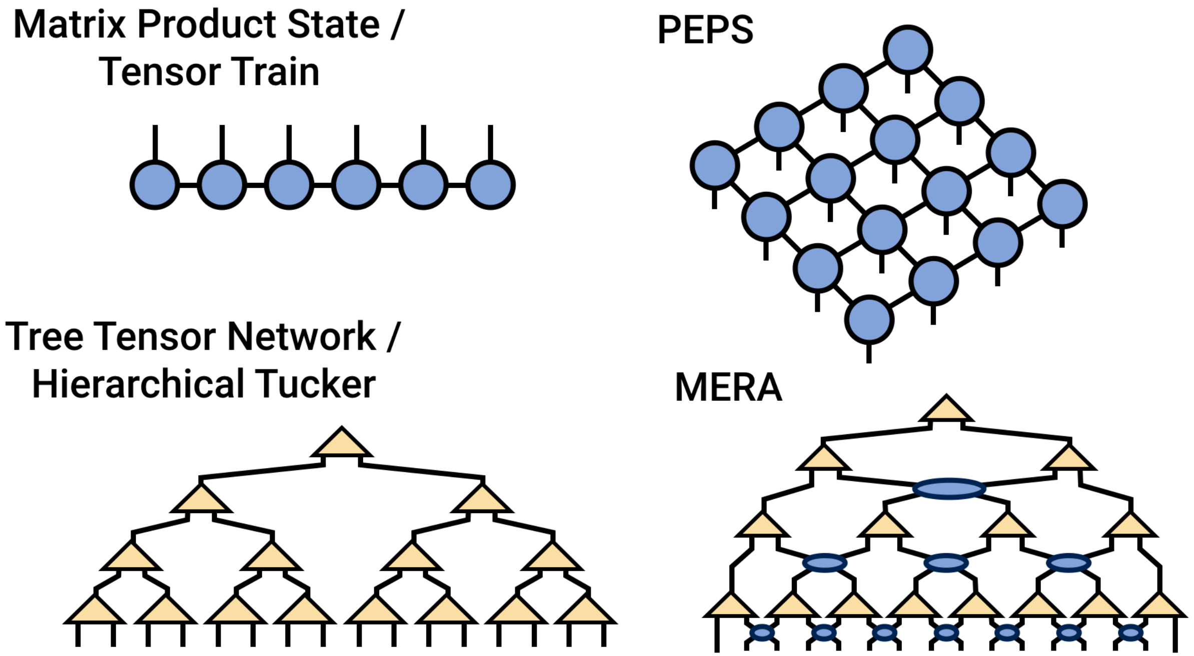
CONTENTS

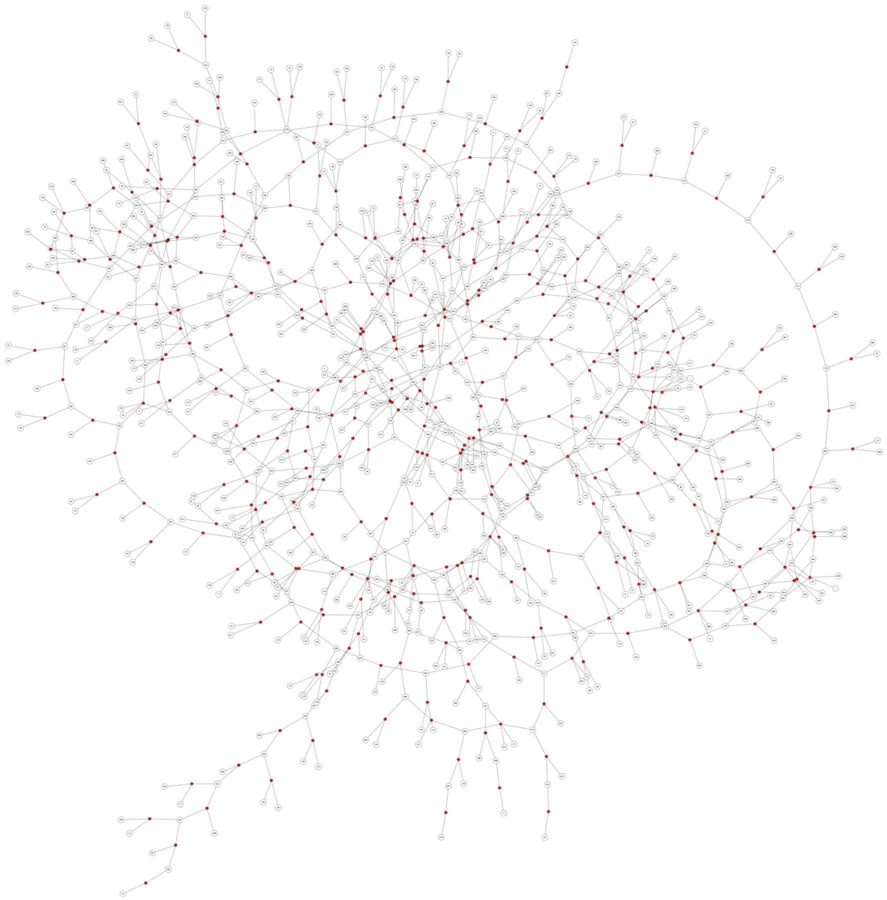
- Tensor network and its contraction order
- Treewidth and tree decomposition
- From tree width to contraction order
- Tensor network with open indices

Tensor Network and Its Contraction Order

What are tensor networks?

- A set of tensors connected as a network.

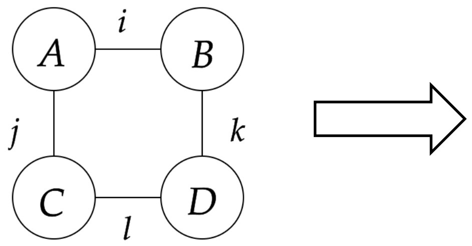




What are tensor networks?

- One can use the Einstein summation formula to represent the tensor network as high dimensional arrays' multiplication.

$$R_{i,j,k,\ldots} = \sum_{a,b,c,\ldots} A_{a,\ldots} B_{b,\ldots} \ldots$$

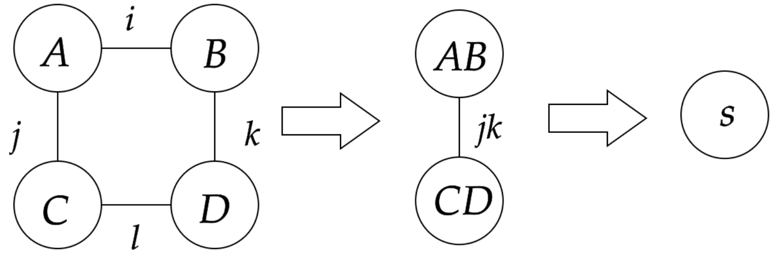


$$s = \sum_{i,j,k,l} A_{ij} B_{ik} C_{jl} D_{lk},$$

```
julia> using OMEinsum

julia> einsum = ein"ij, ik, jl, lk -> "
ij, ik, jl, lk ->
```

What are contraction orders?

- Different ways can be used to contraction a tensor network, we can choose an order for the indices to be eliminated.
- A naïve way is to directly loop over all indices, $O(D^4)$ operations needed.
- Another way is shown below, $O(2D^3 + D^2)$ operations needed.




```
julia> nested_ein = ein"(ij, ik), (jl, lk) -> "
jk, jk ->
├─ ij, ik -> jk
│  ├─ ij
│  └─ ik
└─ jl, lk -> jk
   ├─ jl
   └─ lk
```

Why contraction orders so important?

- For each contraction order, define time and space complexity
  - **Time complexity**: the number of Floating Point operations required to calculate the result
  - **Space complexity**: the largest size of the intermediate tensors

```
# here we take D = 16
julia> size_dict = uniformsize(einsum, 2^4)

julia> contraction_complexity(einsum, size_dict)
Time complexity: 2^16.0
Space complexity: 2^0.0
Read-write complexity: 2^10.001408194392809

julia> contraction_complexity(nested_ein, size_dict)
Time complexity: 2^13.044394119358454
Space complexity: 2^8.0
Read-write complexity: 2^11.000704269011246
```

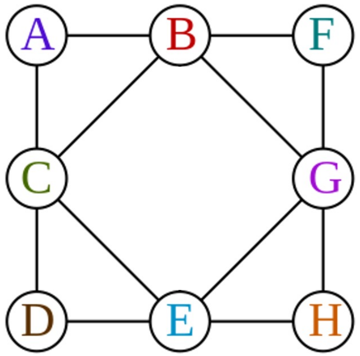
How to get contraction order?

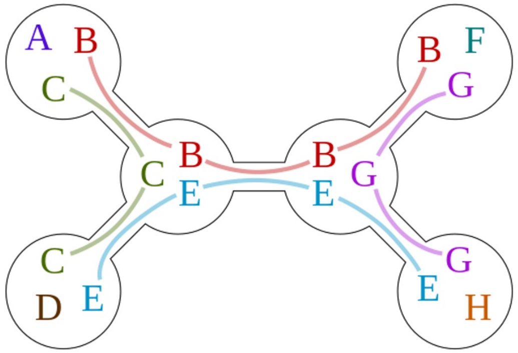
Optimizer	Description	Available in
Exhaustive Search	Slow, exact	TensorOperations.jl
Greedy Algorithm	Fast, heuristic	OMEinsumContractionOrders.jl, Cotengra
Binary Partition	Fast, heuristic	OMEinsumContractionOrders.jl, Cotengra
Local Search	Fast, heuristic	OMEinsumContractionOrders.jl
Exact Treewidth	Slow, exact	OMEinsumContractionOrders.jl

Treewidth and Tree Decomposition

Tree decomposition of a graph

- The tree decomposition of a graph is a tree whose nodes are subsets of the vertices of the graph, and the following conditions are satisfied:
  - Each vertex of the graph is in at least one node of the tree.
  - For each edge of the graph, there is a node of the tree containing both vertices of the edge.
  - Bags containing the same vertices must be connected in the tree

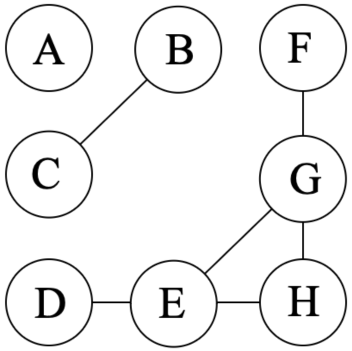


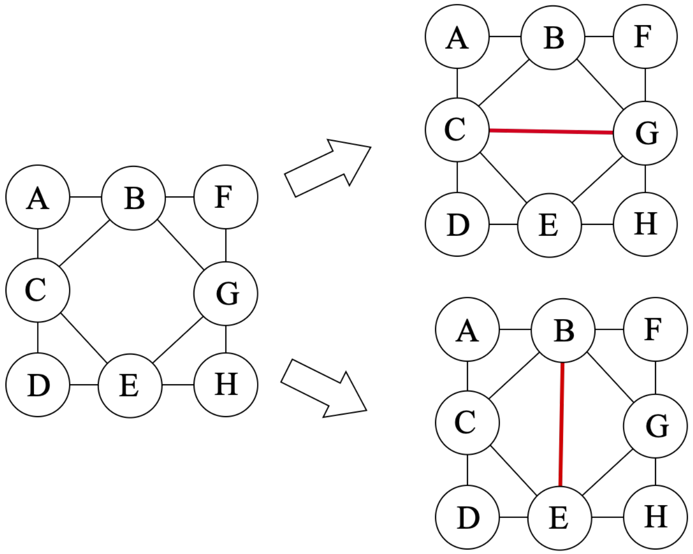


The nodes of the tree are called tree bags, and the tree width of a graph is the minimum width of all possible tree decompositions.

Potential maximal clique

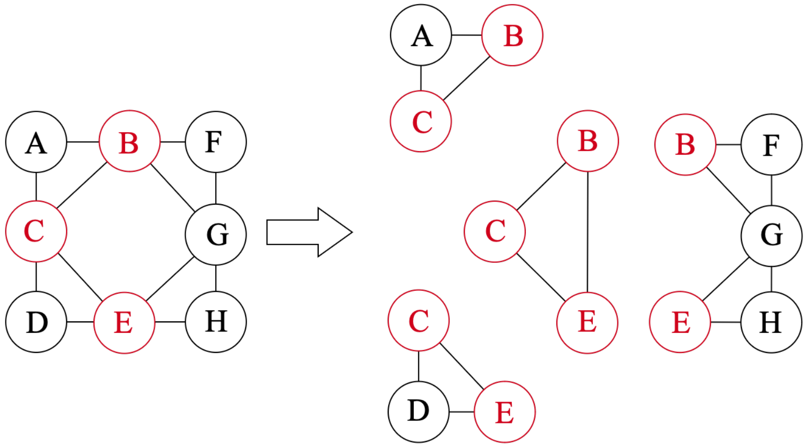
- Minimal separator: Let G be a graph. A set of vertices S⊆V(G) is a minimal separator of G if it is a minimal a, b-separator for some pair a,b∈V(G).
- Potential maximal clique: A set of vertices Ω⊆V(G) is a potential maximal clique of a graph G if there is a minimal triangulation H of G such that Ω is a maximal clique of H. A set of vertices is a maximal clique if it is a clique and no strict superset of it is a clique.





The Bouchitté–Todinca algorithm

- The Bouchitté–Todinca (BT) algorithm is an exact method based on dynamic programming.
- It first list all possible tree bags and the intersection of these tree bags, then search for a set of tree bags that reconstruct the graph and with minimum width.
- For more details, please refer to the original article or my blog.



Bouchitté, Vincent, and Ioan Todinca. "Treewidth and Minimum Fill-in: Grouping the Minimal Separators." SIAM Journal on Computing 31, no. 1 (January 2001): 212–32.

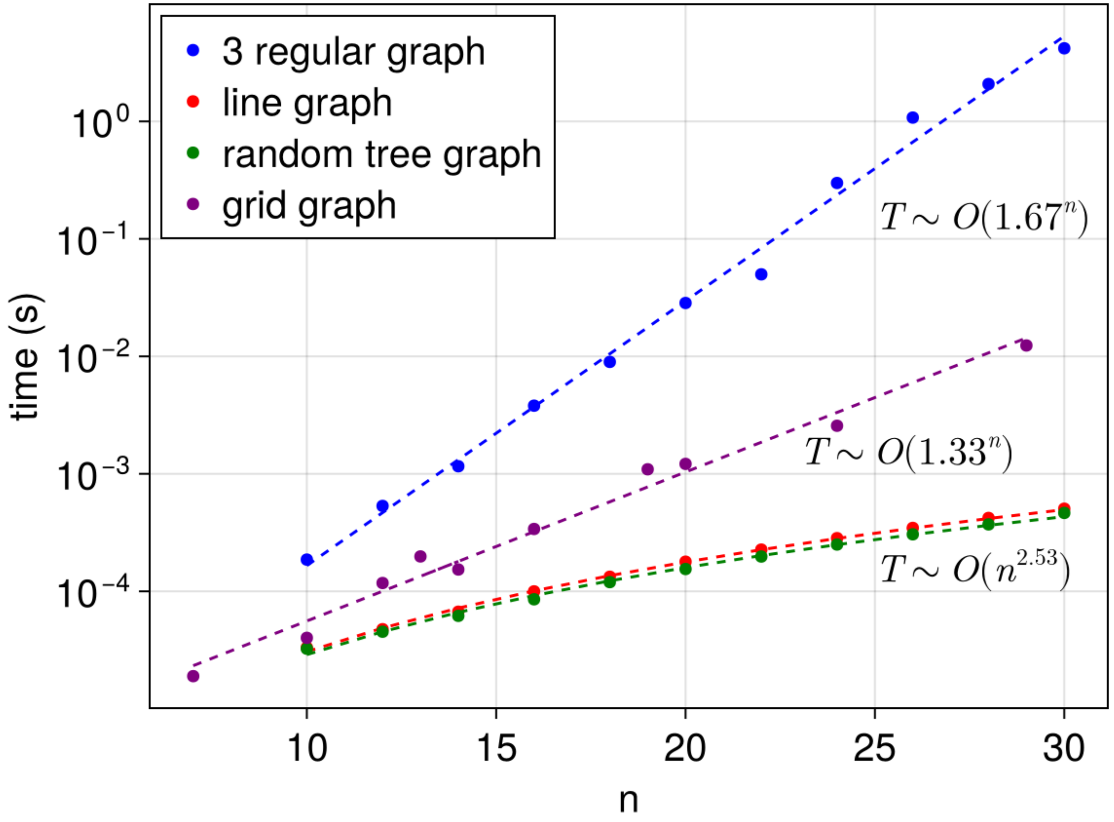
TreeWidthSolver.jl

- TreeWidthSolver.jl is a pure Julia implementation of the BT algorithm.

```
julia> using TreeWidthSolver, Graphs

julia> g = smallgraph(:petersen)
{10, 15} undirected simple Int64 graph

# calculate the exact treewidth of the graph
julia> exact_treewidth(g)
4.0

# show more information
julia> exact_treewidth(g, verbose = true)
[ Info: computing all minimal separators
[ Info: allminseps: 10, 15
[ Info: all minimal separators computed, total: 15
[ Info: computing all potential maximal cliques
[ Info: vertices: 9, Δ: 15, Π: 0
[ Info: vertices: 8, Δ: 14, Π: 9
[ Info: vertices: 7, Δ: 13, Π: 16
[ Info: vertices: 6, Δ: 9, Π: 24
[ Info: vertices: 5, Δ: 6, Π: 35
[ Info: vertices: 4, Δ: 5, Π: 36
[ Info: vertices: 3, Δ: 2, Π: 43
[ Info: vertices: 2, Δ: 1, Π: 44
[ Info: vertices: 1, Δ: 1, Π: 44
[ Info: computing all potential maximal cliques done, total: 45
[ Info: computing the exact treewidth using the Bouchitté-Todinca algorithm
[ Info: precomputation phase
[ Info: precomputation phase completed, total: 135
[ Info: computing the exact treewidth done, treewidth: 4.0
4.0
```

```
# construct the tree decomposition
julia> decomposition_tree(g)
tree width: 4.0
tree decomposition:
Set([5, 6, 7, 3, 1])
├─ Set([7, 2, 3, 1])
├─ Set([5, 4, 6, 7, 3])
│  └─ Set([4, 6, 7, 9])
└─ Set([5, 6, 7, 10, 3])
   └─ Set([6, 10, 8, 3])

# similar for the elimination order
julia> elimination_order(g)
6-element Vector{Vector{Int64}}:
 [1, 3, 7, 6, 5]
 [10]
 [8]
 [4]
 [9]
 [2]

# one can also assign labels to the vertices
julia> elimination_order(g, labels = ['a':'j'...])
6-element Vector{Vector{Char}}:
 ['a', 'c', 'g', 'f', 'e']
 ['j']
 ['h']
 ['d']
 ['i']
 ['b']
```

TreeWidthSolver.jl



Tree Decomposition → Contraction Order

The Theorem

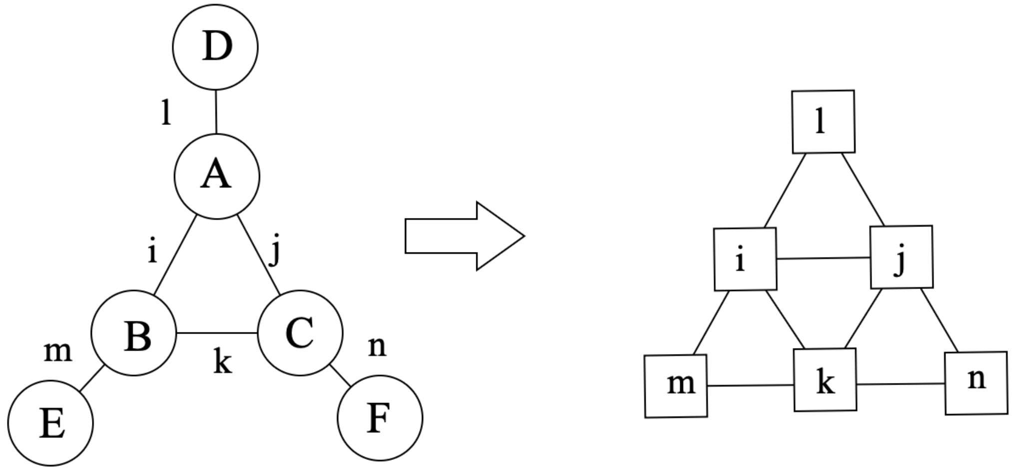
Theorem 1.1. *Let C be a quantum circuit with T gates and whose underlying circuit graph is G_C. Then C can be simulated deterministically in time* $T^{O(1)}\exp[O(\mathrm{tw}(G_C))]$.

In our language, the time complexity of contracting a tensor network is upper bounded by the tree width of its corresponding line graph.

Markov, Igor L., and Yaoyun Shi. “Simulating Quantum Computation by Contracting Tensor Networks.” SIAM Journal on Computing 38, no. 3 (January 2008): 963–81

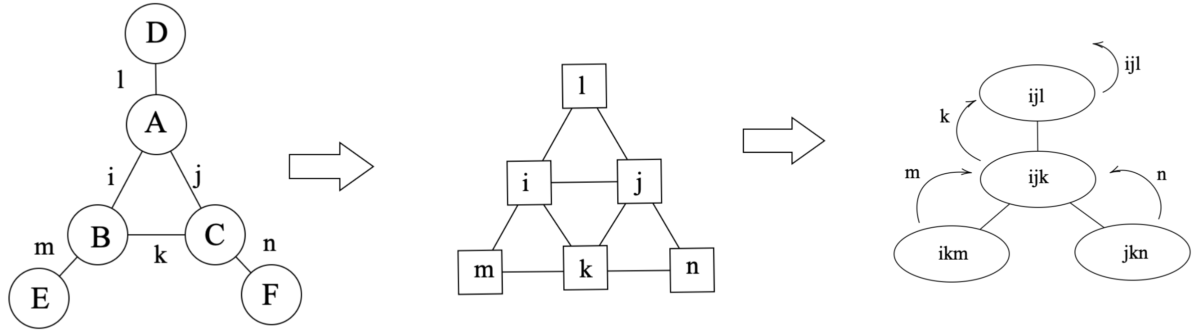
Line graph

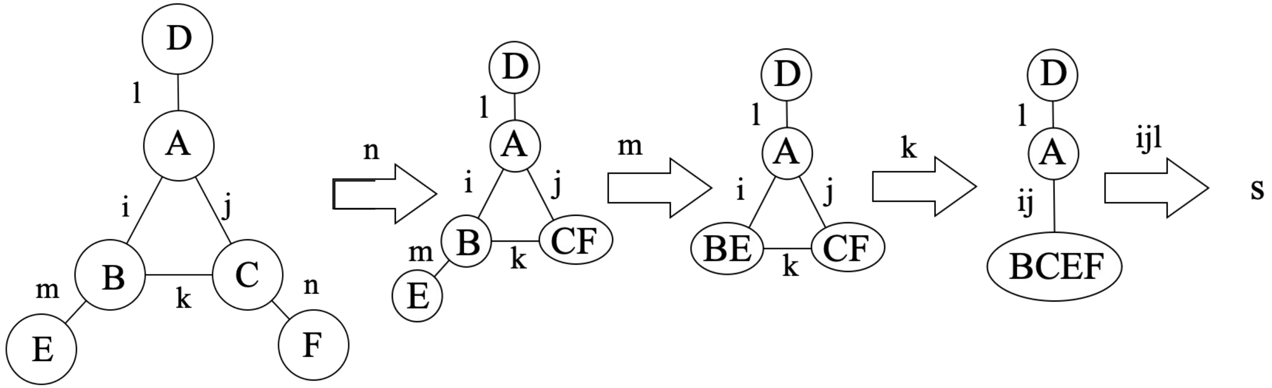
- Given a graph G, its line graph L(G) is a graph such that:
  - Each vertex of L(G) represents an edge of G;
  - Two vertices of L(G) are adjacent if and only if their corresponding edges share a common endpoint ("are incident") in G.



Vertex elimination order contraction order

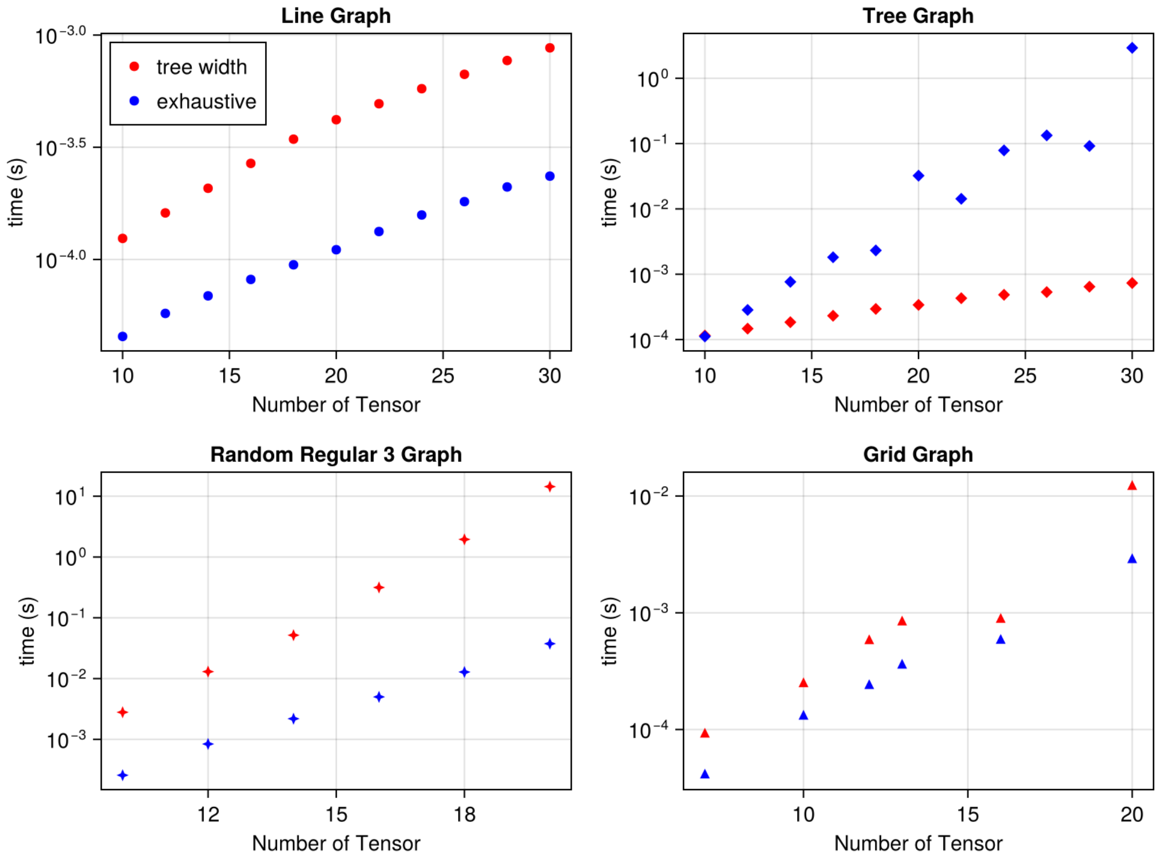
- Vertex elimination order, can be obtained from the tree decomposition
- Vertex elimination order can be converted to be a contraction order





OMEinsumContractionOrders.jl

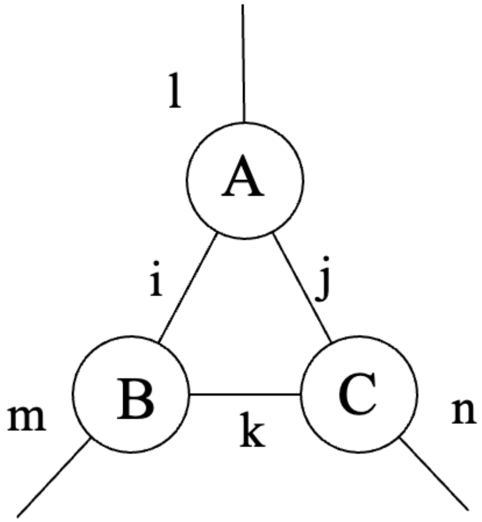
```
julia> using OMEinsum, OMEinsumContractionOrders

# define the contraction using Einstein summation
julia> code = ein"ijl, ikm, jkn, l, m, n -> "
ijl, ikm, jkn, l, m, n ->

ulia> optimizer = ExactTreewidth()
ExactTreewidth{GreedyMethod{Float64, Float64}}(GreedyMethod{Float64, Float64}(0.0,

# set the size of the indices
julia> size_dict = uniformsize(code, 2)
Dict{Char, Int64} with 6 entries:
  'n' => 2
  'j' => 2
  'i' => 2
  'l' => 2
  'k' => 2
  'm' => 2

julia> optcode = optimize_code(code, size_dict, optimizer)
n, n ->
├─ jk, jkn -> n
│  ├─ ij, ik -> jk
│  │  ├─ ijl, l -> ij
│  │  │  ├─ ijl
│  │  │  └─ l
│  │  └─ ikm, m -> ik
│  │     ├─ ikm
│  │     └─ m
│  └─ jkn
└─ n
```



Tensor Networks with Open Indices

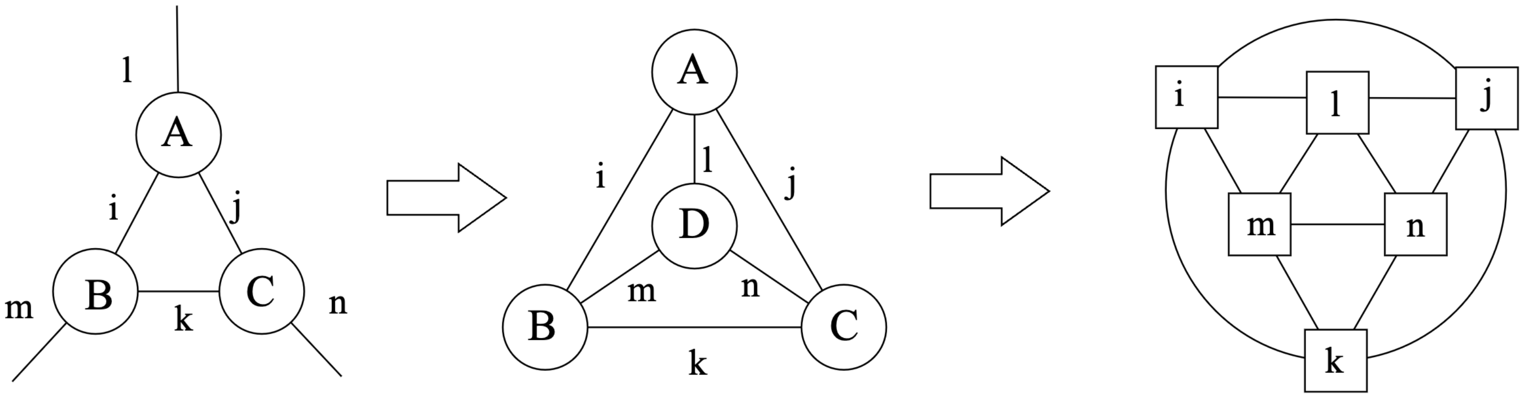
Tensor Network with Open Indices

- How to optimize the contraction order of a tensor network with open indices?
- It is hard to use graph-based methods directly.



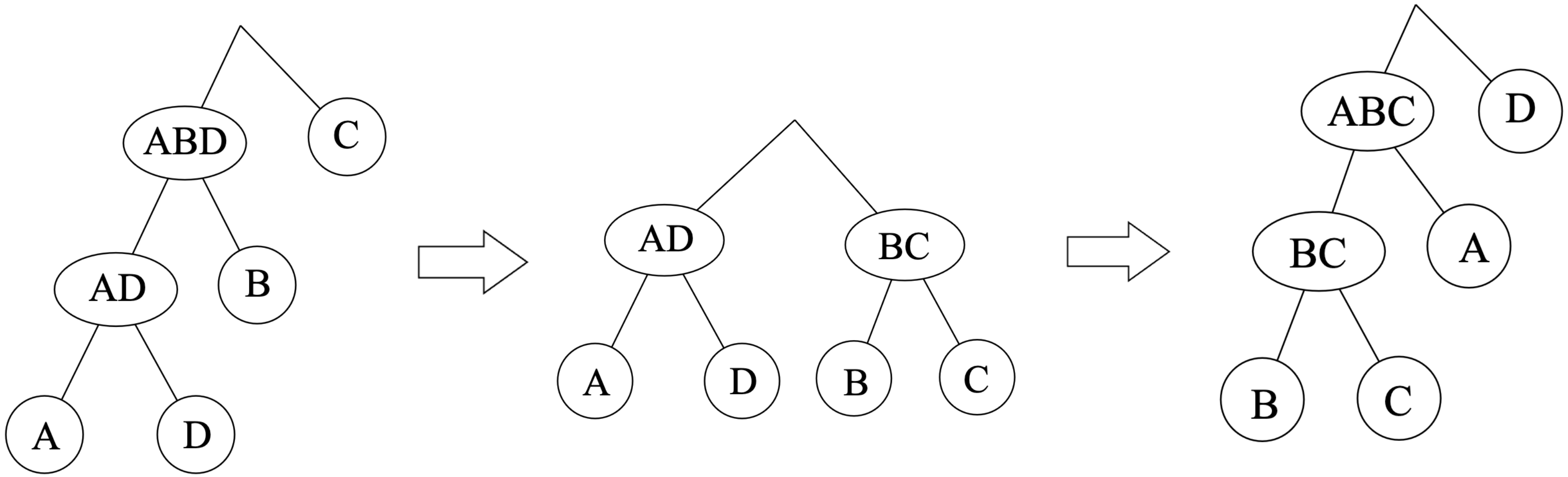
Tensor Network with Open Indices

- A simple method can be applied, we add a tensor D and connect all open indices to D.



Tensor Network with Open Indices

- This leads to a contraction order with D, then we *rotate* the contraction tree so that D is at the top of the tree without changing the complexity



SUMMARY

- Implementation of an exact treewidth solver in pure Julia
- Improving the contraction order optimizer with the solver

arrogantgao.github.io/blogs/treewidth

TreeWidthSolver.jl

OMEinsumContractionOrders.jl

Thanks!