Tree Decompositions in Julia

Richard Samuelson

January 2025

Tree Decompositions

Let $G := (G_V, G_E)$ be a connected simple graph. A *tree decomposition* of G is a pair (T, f), where $T := (T_V, T_E)$ is a tree and

$$f : T_V \to 2^{G_V}$$

is a function mapping the vertices of T to subsets of vertices of G, called *bags*.

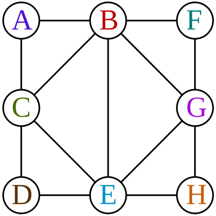
Tree Decompositions

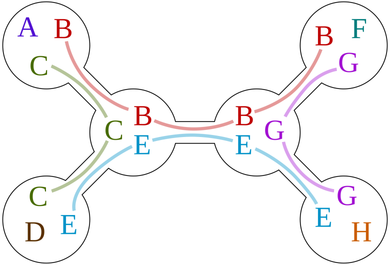
Coverage

For all edges $\{u, v\} \in G_E$, there exists a vertex $i \in T_V$ such that $\{u, v\} \in f(i)$.

Coherence

For all $v \in G_V$, the preimage $f^{-1}\{v\} \subseteq T_V$ induces a connected subtree of T.


A
B
F
C
G
D
E
H


A B C
B C E
B E G
B F G
C D E
E G H

Width

Tree decompositions are used by many graph algorithms. Also,

1. constraint satisfaction [1]
2. probabilistic inference [2]
3. tensor network contraction [3]
4. matrix factorization [4]
5. convex optimization [4]

The running time of these algorithms is parametrized by the *width* of the decomposition: one minus the size of its largest bag.

Tree Decompositions in Julia

Some Julia libraries construct tree decompositions.

library	application	active
COSMO.jl	convex	✓
Clarabel.jl	convex	✓
Chordal.jl	convex	
TreeWidthSolver.jl	tensors	✓
QXGraphDecompositions.jl	tensors	

The functions in COSMO.jl and Clarabel.jl are internal, and function in TreeWidthSolver.jl has an exponential running time.

JunctionTrees.jl

```
julia> using StructuredDecompositions.JunctionTrees

julia> graph = [
           0 1 1 0 0 0 0 0
           1 0 1 0 0 1 0 0
           1 1 0 1 1 0 0 0
           0 0 1 0 1 0 0 0
           0 0 1 1 0 0 1 1
           0 1 0 0 0 0 1 0
           0 0 0 0 1 1 0 1
           0 0 0 0 1 0 1 0
       ];

julia> label, tree = junctiontree(graph);

julia> tree
6-element JunctionTree:
[6, 7, 8]
├─ [1, 6, 7]
├─ [4, 6, 8]
│  └─ [3, 4, 6]
│     └─ [2, 3, 6]
└─ [5, 7, 8]
```

JunctionTrees.jl

A graph elimination algorithm. The options are

type	name	complexity
`MCS`	maximum cardinality search	$O(m + n)$
`RCM`	reverse Cuthill-Mckee	$O(m\Delta)$
`AMD`	approximate minimum degree	$O(mn)$
`SymAMD`	column approximate minimum degree	$O(mn)$
`MMD`	multiple minimum degree	$O(mn^2)$
`NodeND`	nested dissection	
`FlowCutter`	FlowCutter	
`Spectral`	spectral ordering	
`BT`	Bouchitte-Todinca	$O(2.6183^n)$

for a graph with m edges, n vertices, and maximum degree Δ.

Benchmarks

Both COSMO.jl and Clarabel.jl call the function `ldl` exported by QDLDL.jl.

library	name	edges	time	memory
StructuredDecompositions	mycielskian4	23	2.449 µs	6.70 KiB
QDLDL	mycielskian4	23	1.029 µs	4.70 KiB
StructuredDecompositions	can_292	1124	47.500 µs	146.36 KiB
QDLDL	can_292	1124	28.958 µs	146.08 KiB
StructuredDecompositions	wing	121544	18.089 ms	28.59 MiB
QDLDL	wing	121544	97.048 ms	177.01 MiB
StructuredDecompositions	333SP	11108633	1.214 s	1.64 GiB
QDLDL	333SP	11108633	2.728 s	3.89 GiB

Graphs were sourced from the SuiteSparse Matrix Collection.

Bibliography

[1] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[2] Daphane Koller. *Probabilistic Graphical Models: Principles and Techniques*. 2009.

[3] Igor L Markov and Yaoyun Shi. "Simulating Quantum Computation by Contracting Tensor Networks". In: *SIAM Journal on Computing* 38.3 (2008), pp. 963–981.

[4] Lieven Vandenberghe, Martin S Andersen, et al. "Chordal Graphs and Semidefinite Optimization". In: *Foundations and Trends in Optimization* 1.4 (2015), pp. 241–433.