A SAT Approach to Branchwidth

Neha Lodha Sebastian Ordyniak Stefan Szeider
[neha,ordyniak,sz]@ac.tuwien.ac.at
Algorithms and Complexity Group, TU Wien, Vienna, Austria

In memory of Helmut Veith

Branch Decomposition

Applications of Branch Decompositions:

- Introduced by Robertson and Seymour
- Similar to tree decomposition

Branch Decomposition

Applications of Branch Decompositions:

- Introduced by Robertson and Seymour
- Similar to tree decomposition
- Used for decomposing combinatorial objects
  - Hypergraphs
  - Matroids
  - Integer-valued symmetric submodular functions
  - Propositional CNF formulas

Branch Decomposition

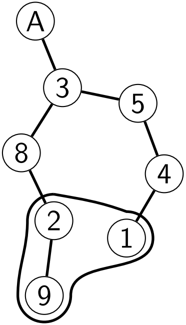
Applications of Branch Decompositions:

- Introduced by Robertson and Seymour
- Similar to tree decomposition
- Used for decomposing combinatorial objects
  - Hypergraphs
  - Matroids
  - Integer-valued symmetric submodular functions
  - Propositional CNF formulas
- Problems solved efficiently using dynamic programming on branch decomposition
  - Traveling salesman problem
  - #P-complete problem of propositional model counting
  - Generation of resolution refutations for unsatisfiable CNF formulas

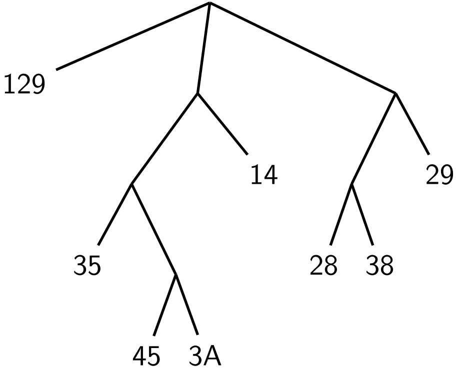
Branch Decomposition

Definition by Example:

Hypergraph G



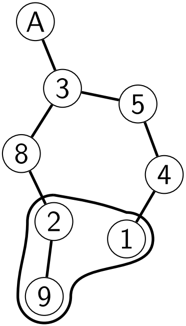
Branch Decomposition of G



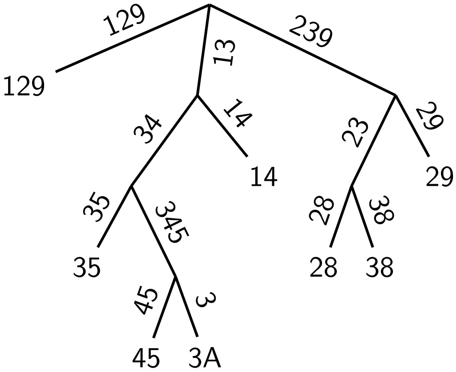
Branch Decomposition

Definition by Example:

Hypergraph G



Branch Decomposition of G

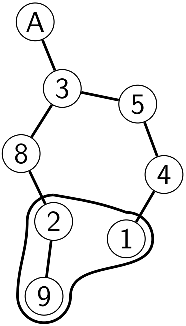


$Width = 3$

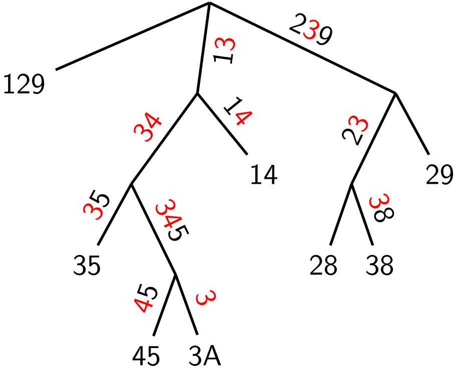
Branch Decomposition

Definition by Example:

Hypergraph G



Branch Decomposition of G



$Width = 3$

Branchwidth

$$bw = \min_{\{\text{all decompositions}\}} \{\text{width of decomposition}\}$$

Branchwidth

$$bw = \min_{\{\text{all decompositions}\}} \{\text{width of decomposition}\}$$

- *NP*-hard problems with bounded width in time and space polynomial in input size

Branchwidth

$$bw = \min_{\{\text{all decompositions}\}} \{\text{width of decomposition}\}$$

- *NP*-hard problems with bounded width in time and space polynomial in input size
- But (usually) exponential in the width parameters

Branchwidth

$$bw = \min_{\{\text{all decompositions}\}} \{\text{width of decomposition}\}$$

- *NP*-hard problems with bounded width in time and space polynomial in input size
- But (usually) exponential in the width parameters

Note: $bw \leq tw - 1 \leq \frac{3}{2} bw$

Why small widths?

Why small widths?

The following was noted by Kask et al. [2011]

> *... since inference is exponential in the tree-width, a small reduction in tree-width (say by even by 1 or 2) can amount to one or two orders of magnitude reduction in inference time.*

Why small widths?

The following was noted by Kask et al. [2011]

> *... since inference is exponential in the tree-width, a small reduction in tree-width (say by even by 1 or 2) can amount to one or two orders of magnitude reduction in inference time. Thus, huge computational gains are also possible by simply finding improved variable orderings.*

Task: Finding smallest width efficiently

Why small widths?

The following was noted by Kask et al. [2011]

> *... since inference is exponential in the tree-width, a small reduction in tree-width (say by even by 1 or 2) can amount to one or two orders of magnitude reduction in inference time. Thus, huge computational gains are also possible by simply finding improved variable orderings.*

Task: Finding smallest width efficiently

Note: Finding optimal width is *NP*-hard

Existing Algorithms for Branch Decompositions

- Heuristics : splitting graph iteratively along a small cut

Existing Algorithms for Branch Decompositions

- Heuristics : splitting graph iteratively along a small cut
- Exact :
  - Interger linear program (ILP) developed by Ulusal [2008]

Existing Algorithms for Branch Decompositions

- Heuristics : splitting graph iteratively along a small cut
- Exact :
  - Interger linear program (ILP) developed by Ulusal [2008]
  - An exponential-time tangle based algorithm (TANGLES), suggested by Robertson and Seymour [1991] and implemented by Hicks [2005]

Related work

Treewidth (tw)

- Samer and Veith [2009] introduced the first SAT encoding for treewidth.
- Berg and Järvisalo [2014] improved upon it and used techniques like incremental and MAX-SAT approach

Related work

Treewidth (tw)

- Samer and Veith [2009] introduced the first SAT encoding for treewidth.
- Berg and Järvisalo [2014] improved upon it and used techniques like incremental and MAX-SAT approach

Clique width (cw)

- Heule and Szeider [2015] introduced the first SAT encoding for clique width (This is the first exact algorithm for clique width)

Our Contribution

- Efficient SAT-encoding based on new partition based characterization of branch decomposition
- Using local improvement techniques to avail use of SAT for large instances

First Attempt

Encoding Branch Decomposition Tree

- Encoded the Branch Decomposition Tree to CNF formula

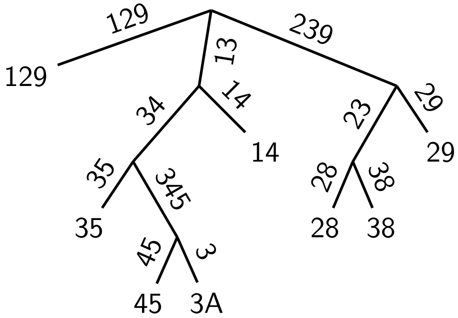
First Attempt

Encoding Branch Decomposition Tree

- Encoded the Branch Decomposition Tree to CNF formula
- This method could not solve even up to 30 edges

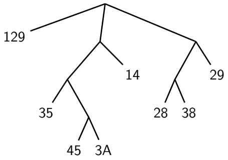
New Approach

Partition Based Characterization



New Approach

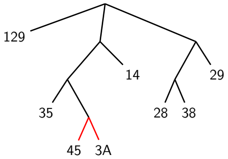
Partition Based Characterization



$$\Big\{\{129\}, \{35\}, \{45\}, \{3A\}, \{14\}, \{28\}, \{38\}, \{29\}\Big\}$$

New Approach

Partition Based Characterization

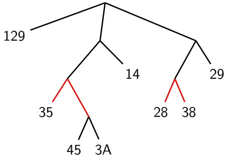


$$\Big\{\{129\},\{35\},\{45,3A\},\{14\},\{28\},\{38\},\{29\}\Big\}$$

$$\Big\{\{129\},\{35\},\{45\},\{3A\},\{14\},\{28\},\{38\},\{29\}\Big\}$$

New Approach

Partition Based Characterization



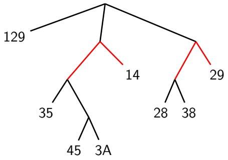
$$\Big\{\{129\},\{35,45,3A\},\{14\},\{28,38\},\{29\}\Big\}$$

$$\Big\{\{129\},\{35\},\{45,3A\},\{14\},\{28\},\{38\},\{29\}\Big\}$$

$$\Big\{\{129\},\{35\},\{45\},\{3A\},\{14\},\{28\},\{38\},\{29\}\Big\}$$

New Approach

Partition Based Characterization



$$\Big\{\{129\}, \{35, 45, 3A, 14\}, \{28, 38, 29\}\Big\}$$

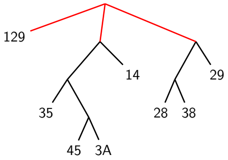
$$\Big\{\{129\}, \{35, 45, 3A\}, \{14\}, \{28, 38\}, \{29\}\Big\}$$

$$\Big\{\{129\}, \{35\}, \{45, 3A\}, \{14\}, \{28\}, \{38\}, \{29\}\Big\}$$

$$\Big\{\{129\}, \{35\}, \{45\}, \{3A\}, \{14\}, \{28\}, \{38\}, \{29\}\Big\}$$

New Approach

Partition Based Characterization



$$\Big\{\{129, 35, 45, 3A, 14, 28, 38, 29\}\Big\}$$

$$\Big\{\{129\}, \{35, 45, 3A, 14\}, \{28, 38, 29\}\Big\}$$

$$\Big\{\{129\}, \{35, 45, 3A\}, \{14\}, \{28, 38\}, \{29\}\Big\}$$

$$\Big\{\{129\}, \{35\}, \{45, 3A\}, \{14\}, \{28\}, \{38\}, \{29\}\Big\}$$

$$\Big\{\{129\}, \{35\}, \{45\}, \{3A\}, \{14\}, \{28\}, \{38\}, \{29\}\Big\}$$

SAT Encoding

Tools for encoding partition based characterization

- Equivalence classes $s(e, f, i)$
  To encode refinement as underlying tree is implicitly represented by refinement of partitions

SAT Encoding

Tools for encoding partition based characterization

- Equivalence classes $s(e, f, i)$
- Cut along the edges $c(e, u, i)$
  Set for every vertex that is cut vertex

SAT Encoding

Tools for encoding partition based characterization

- Equivalence classes $s(e, f, i)$
- Cut along the edges $c(e, u, i)$
- Cardinality constraints
  - To bound the number of cut vertices
  - Sequential unary counter

SAT Encoding

Tools for encoding partition based characterization

- Equivalence classes $s(e, f, i)$
- Cut along the edges $c(e, u, i)$
- Cardinality constraints

Output

For a (hyper) graph G and an integer k we produce formula $F(G, k)$ which is satisfiable iff G has a branch decomposition of width k.

Experimental setup

- SAT solver: Glucose 4.0
- System: 4-core Intel Xeon CPU E5649, 2.35GHz, 72GB RAM, Ubuntu 14.04
- Memory limit: 8GB
- Timeout: 6hours
- Instances: Famous named graphs

First Results

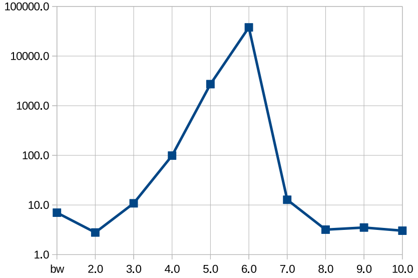
Graph	$\lvert V \rvert$	$\lvert E \rvert$	w
Watsin	50	75	6
Kittell	23	63	6
Holt	27	54	9
Shrikhande	16	48	8
Errera	17	45	6
Brinkmann	21	42	8
Clebsch	16	40	8
Folkman	20	40	6
Paley13	13	39	7
Poussin	15	39	6
Robertson	19	38	8

Limits

- We have observed a limit of 70 edges
  - Timeout
  - The size of encoding (some cases exceeded 1GB)
  - Same limit is observed for *tw* and *cw*

Limits

- We have observed a limit of 70 edges
  - Timeout
  - The size of encoding (some cases exceeded 1GB)
  - Same limit is observed for *tw* and *cw*
- But we can provide upper bounds for graphs with up to 150 edges in reasonable time



Branchwidth = 7

What do we do now?

What do we do now?

Extend SAT to large instances?

What do we do now?

Extend SAT to large instances?

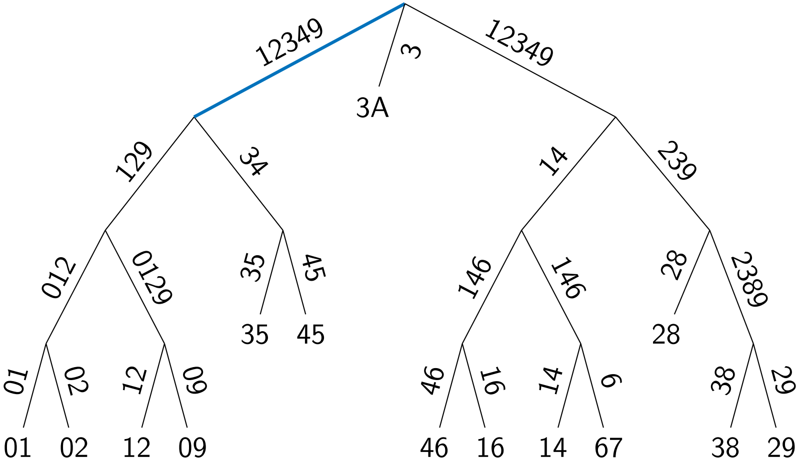
Use it locally!

How?

1. Generate heuristic decomposition (we used heuristics provided by Hicks)
2. Pick local branch decomposition around large cut (using specialized DSF procedure)
3. Use SAT to improve local branch decomposition and plug it back in

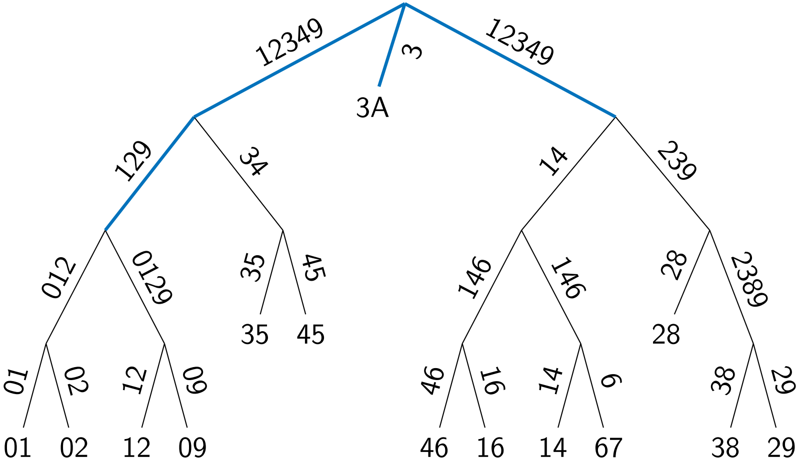
Repeat till no more improvement possible or timeout

By Example

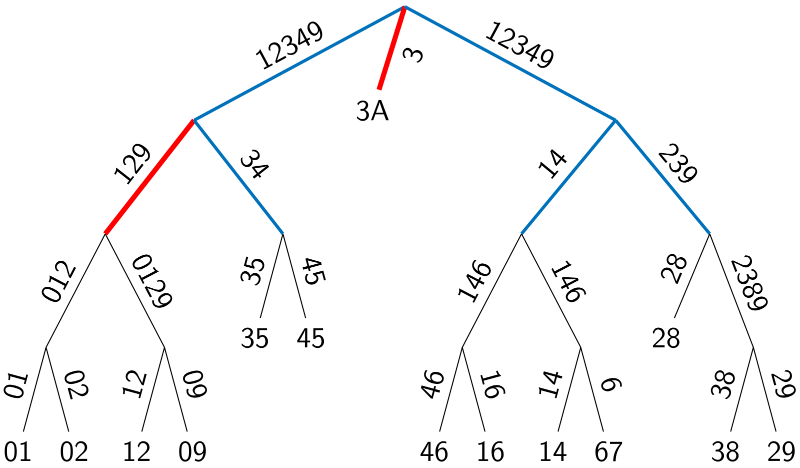


12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

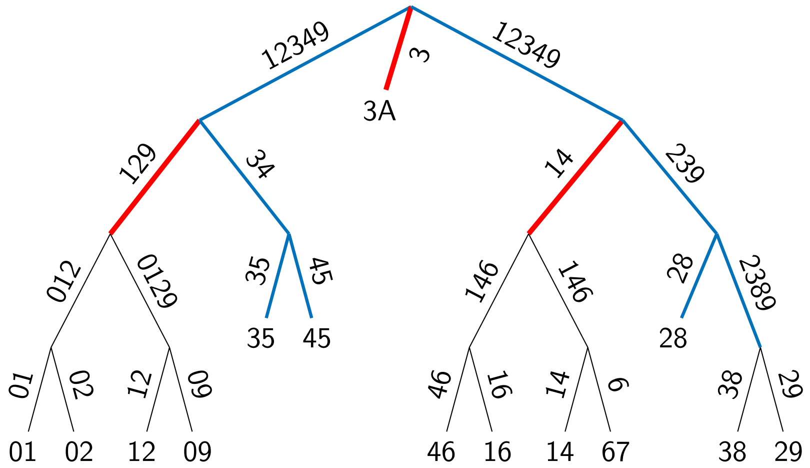
By Example


12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

By Example

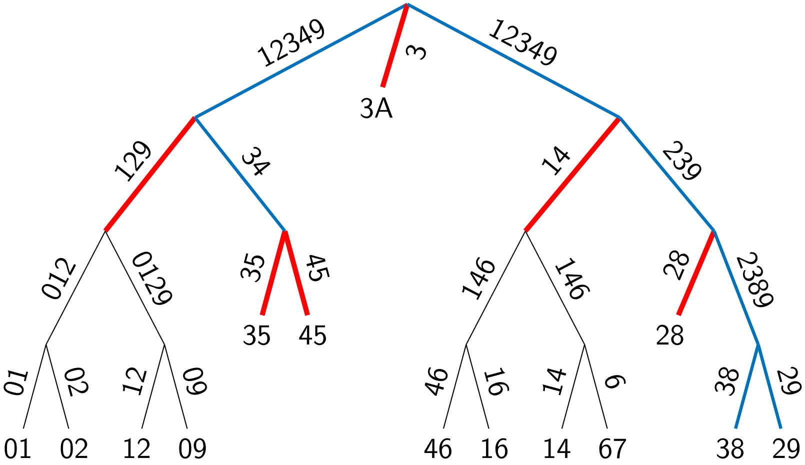

12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

By Example



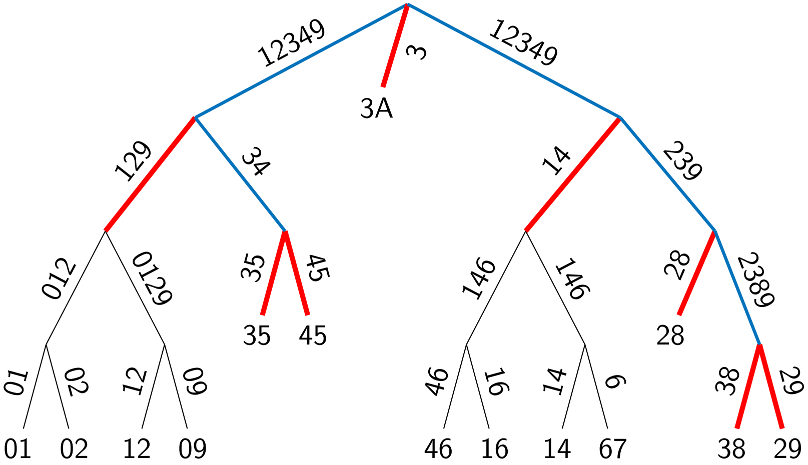
12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

By Example



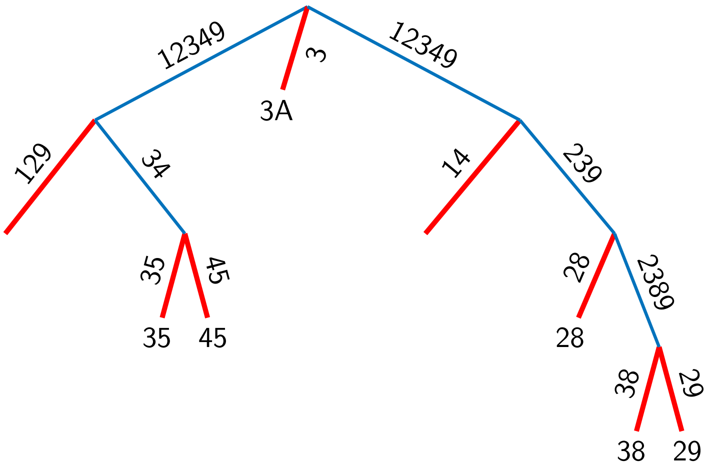
12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

By Example

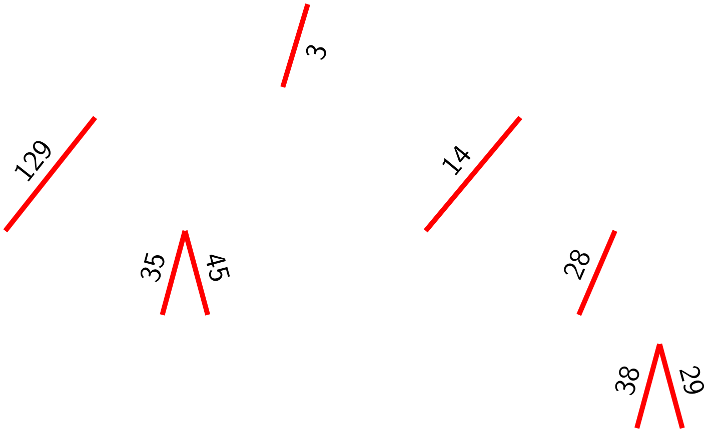

12349
3
12349
3A
129
34
14
239
012
0129
35
45
146
146
28
2389
35
45
28
01
02
12
09
46
16
14
6
38
29
01
02
12
09
46
16
14
67
38
29

By Example

By Example



By Example



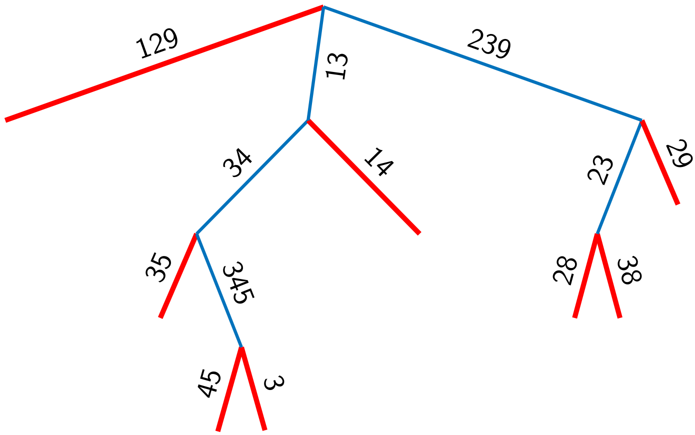
Hypergraph consisting of all boundary edges

By Example

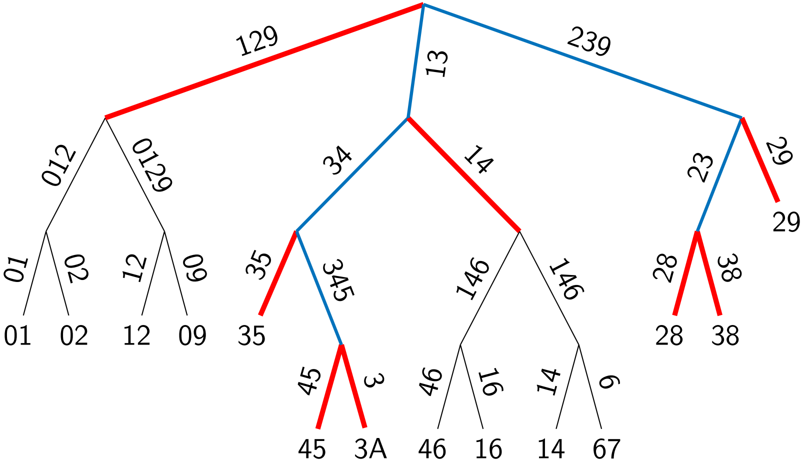
Input:
The hypergraph (H) consisting every boundary **edge**

Theorem

Improved branch decomposition of hypergraph (H) can be plugged in to the original decomposition.



129
13
239
34
14
23
29
35
345
28
38
45
3


129
13
239
012
0129
34
14
23
29
01
02
12
09
35
345
146
146
28
38
29
01
02
12
09
35
28
38
45
3
46
16
14
6
45
3A
46
16
14
67

Experimental setup

- SAT solver: Glucose 4.0
- System: 4-core Intel Xeon CPU E5649, 2.35GHz, 72GB RAM, Ubuntu 14.04
- Memory limit: 8GB
- Timeout: 6hours
- Timeout per SAT call: 600sec
- Size of local branch decomposition: 80edges
- Instances: TreewidthLIB

Results

Graph	$\lvert V\rvert$	$\lvert E\rvert$	iw	w	diff
inithx.i.2-pp	363	8897	55	45	10
fpsol2.i.2-pp	333	7910	48	39	9
graph13	458	1877	141	134	7
bn_31-pp	1148	3317	40	36	4
bn_4	100	574	42	38	4
celar08-pp-003	76	421	20	16	4
fpsol2.i.2	451	8691	53	49	4
graph05-wpp	94	397	28	24	4
graph04-pp	179	678	52	48	4
u724.tsp	724	2117	29	26	3
water-wpp	22	96	11	8	3
celar05-pp	80	426	18	15	3
mulsol.i.2-pp	116	2468	62	59	3

How are we compared to others?

Exact Encodings:

- TANGLES[1]
  - Exponential time and space algorithm ($\mathcal{O}(m^k)$)
  - Limit of branchwidth 8
  - Cannot deal with large graphs

[1] These results are based on older hardware and software, thus the comparison can not be concrete

How are we compared to others?

Exact Encodings:

- TANGLES[1]
  - Exponential time and space algorithm ($\mathcal{O}(m^k)$)
  - Limit of branchwidth 8
  - Cannot deal with large graphs
  - Cannot be used for local improvements

[1] These results are based on older hardware and software, thus the comparison can not be concrete

How are we compared to others?

Exact Encodings:

- TANGLES[1]
  - Exponential time and space algorithm ($\mathcal{O}(m^k)$)
  - Limit of branchwidth 8
  - Cannot deal with large graphs
  - Cannot be used for local improvements
- ILP[1]
  - Limit of 13 edges for hypergraphs

[1] These results are based on older hardware and software, thus the comparison can not be concrete

The SAT-based local improvement

- Scales to large instance with several thousands edges
- High branchwidth upper bounds

Future Work

1. Extending the encoding to obtain specialized decomposition to aid local improvement
2. Encoding various other parameters such as boolean width, rank width (similar decomposition scheme)
3. Extending the branch decomposition approach to apply in field of knowledge compilation
4. Extending current approach with incremental and MAXSAT solving

Summary

- SAT encoding for branchwidth based on new partition based characterization

Summary

- SAT encoding for branchwidth based on new partition based characterization
- SAT-based local improvements for branch decompositions
  - Provides the means for scaling the SAT-approach to much larger instances
  - New application field of SAT solvers

Summary

- SAT encoding for branchwidth based on new partition based characterization
- SAT-based local improvements for branch decompositions
  - Provides the means for scaling the SAT-approach to much larger instances
  - New application field of SAT solvers

Thank you.

References

[1] Berg, J. and Järvisalo, M. (2014). SAT-Based Approaches to Treewidth Computation: An Evaluation. In ICTAI 2014.

[2] Heule, M. and Szeider, S. (2015). A SAT Approach to Clique-Width. ACM Trans. Comput. Log. *16*.

[3] Hicks, I. V. (2005). Graphs, branchwidth, and tangles! Oh my! Networks *45*.

[4] Kask, K., Gelfand, A., Otten, L. and Dechter, R. (2011). Pushing the Power of Stochastic Greedy Ordering Schemes for Inference in Graphical Models. In AAAI 2011.

[5] Robertson, N. and Seymour, P. D. (1991). Graph minors X. Obstructions to tree-decomposition. J. Combin. Theory Ser. B *52*.

[6] Samer, M. and Veith, H. (2009). Encoding Treewidth into SAT. In SAT 2009.

[7] Ulusal, E. (2008). Integer Programming Models for the Branchwidth Problem. PhD thesis, Texas A&M University.