Treewidth: Preprocessing and Kernelization

Hans L. Bodlaender

Joint work with
Arie Koster, Frank van den Eijkhof,
Bart Jansen, Stefan Kratsch,
Vincent Kreuzen
This talk

- Survey of work on *preprocessing* and *kernelization* for treewidth
  1. Treewidth
  2. Reduction rules
  3. Safe separators
  4. Kernelization
  5. Conclusions
1

TREewidth
Treewidth

- Graph parameter telling how treelike a graph is
- Introduced by Robertson & Seymour (198*) in work on graph minors
- Several equivalent notions
- Many applications
  - Intractable problems often easier when restricted to graphs of bounded treewidth, e.g.:
  - Courcelle’s theorem: Each problem formulatable in Monadic Second Order Logic can be solved in linear time on graphs of bounded treewidth
  - Probabilistic networks: linear time algorithm for inference problem on graphs of bounded treewidth
A **tree decomposition** of a graph $G$ is a tree in which each node corresponds to a set of vertices from $G$ (a *bag*) with:

- Every vertex of $G$ occurs in some bag.
- For every edge of $G$ there is a bag containing both endpoints.
- All bags containing the same vertex form a connected subtree.

The **treewidth** of a tree-decomposition is the size of the largest bag minus one.
Lemmas

- If $W$ is a clique, then in any tree decomposition, there is a bag that contains all vertices in $W$.
- If $G'$ is obtained from $G$ by contracting edges, then the treewidth of $G'$ is at most the treewidth of $G$. 
Algorithms on tree decompositions

- Usual form:
  1. Find a tree decomposition of small width
  2. Run a dynamic programming algorithm on the tree decomposition

- Time of step 2 is (usually) exponential (or worse) in the width of the tree decomposition

- So, need for algorithms to determine treewidth and find tree decompositions of small width
Complexity of treewidth

- NP-complete
- B, 1997: For each fixed $k$, there is a linear time algorithm to test if the treewidth of a given graph is at most $k$, and if so, find a corresponding tree decomposition
- $O(n) = \Theta(n)$ time... (Röhrig, 1998; even if $k=4$)
- ... and many many more results ...
- Practical algorithms...
  - Heuristics
  - Fast(er) exact algorithms
  - Preprocessing
    - Transform your input to a smaller equivalent input
Two types of preprocessing

- **Reduction rules** (*Simplification*)
  - Rules that change G into a smaller `equivalent` graph
  - Maintains a lower bound variable for treewidth _low_

- **Safe separators** (*Divide and Conquer*)
  - Splits the graph into two or more smaller parts with help of a separator that is made to a clique
REDUCTION RULES
Safe rules that
- Make $G$ smaller
- Maintain optimality...
- Use for preprocessing graphs when computing treewidth

Reduction

Input Graph $G$

Preprocessing rules

Reduced Graph $H$

Compute Treewidth for $G$

Compute Treewidth for $H$

Tree decomposition for $G$

Tree decomposition for $H$

Undo preprocessing
Reduction rules

- Help of a variable $low$
- $low$ gives lower bound on treewidth of original input graph

- Safe rule: if $(G, low)$ is changed to $(G', low')$ then
  - $\max (\text{treewidth}(G), \text{low}) = \max (\text{treewidth}(G', \text{low'}))$

- Algorithm:
  - while we see a safe rule to apply (that simplifies instance) do
    - Apply it
Reduction rules for Treewidth

- Arnborg, Proskurowski, 1986: Rules that recognize graphs of treewidth 1, 2, 3
- Bodlaender, Koster, vd Eijkhof, 2005: Preprocessing heuristics for treewidth
- vd Eijkhof, Bodlaender, Koster, 2007: Generalization and weighted variants
- Sanders, 1996: Rules for treewidth 4
- Hein, Koster, 2011: Experimental evaluation of Sanders rules
- Bodlaender, Jansen, Kratsch, 2011: “New” rules and kernelization
- Kreuzen, 2011: Experimental evaluation of new rules
Example: Series Rule
(from Arnborg, Proskurowski, 1986)

- **Series Rule**: remove a vertex of degree 2 and connect its neighbors
- Safe if treewidth is at least 2 (low > 1)
Example

Reduce

Solve

Undo reductions
A vertex is **simplicial** if its neighbors form a clique.

A vertex $v$ is **almost simplicial** if $v$ has a neighbor $w$ such that all neighbors except $w$ form a clique.
Let \( v \) be a simplicial vertex in \( G \)

Remove \( v \)

Set \( \text{low} := \max(\text{low}, \text{degree}(v)) \)

Simplicial Rule is safe

Special cases: all vertices of degree 0 and 1
Almost Simplicial Rule (2005 version)

- Let $v$ be a almost simplicial vertex in $G$ and $\text{low} \geq \text{degree}(v)$
- Remove $v$
- Turn neighbors into clique

Almost Simplicial Rule is safe
Proof of safeness of Almost Simplicial Rule

- Suppose $G'$ is obtained from $G$
- $G'$ is obtained from $G$ by contracting, so $\text{treewidth}(G')$ is at most $\text{treewidth}(G)$
- If we have tree decomposition of $G'$: $N(v)$ is a clique, so there is a bag containing $N(v)$

\[
\begin{align*}
N(v) & \rightarrow v \\
N(v) & \rightarrow N(v)
\end{align*}
\]

- Add a new bag as shown
- New bag has size $\deg(v) + 1 \leq \text{low} + 1$
Buddy Rule

- Let $v, w$ be a buddy in $G$ and $low \geq 3$
- Remove $v, w$.
- Turn neighbors into clique

Safe; generalizations exist
(Extended) Cube Rule

- Let $\text{low} \geq 3$ and cube structure (see picture)
- Replace subgraph as shown
- Safe. Not often in practice but helps to increase lower bound

Original Graph $\rightarrow$ Reduced Graph
Lower bound rules

- Rules that increase *low*
  - E.g., if no rule applies and \( \text{low} = 3 \), then set \( \text{low} := 4 \)
  - Or run a lower bound heuristic
## Results for probabilistic networks

<table>
<thead>
<tr>
<th>instance</th>
<th>original</th>
<th>preprocessed</th>
<th></th>
<th>instance</th>
<th>original</th>
<th>preprocessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm</td>
<td>37 65</td>
<td>0 0 4</td>
<td></td>
<td>oesoca+</td>
<td>67 208</td>
<td>14 75 9</td>
</tr>
<tr>
<td>barley</td>
<td>48 126</td>
<td>26 78 4</td>
<td></td>
<td>oesoca</td>
<td>39 67</td>
<td>0 0 3</td>
</tr>
<tr>
<td>boblo</td>
<td>221 328</td>
<td>0 0 3</td>
<td></td>
<td>oesoca42</td>
<td>42 72</td>
<td>0 0 3</td>
</tr>
<tr>
<td>diabetes</td>
<td>413 819</td>
<td>116 276 4</td>
<td></td>
<td>oow-bas</td>
<td>27 54</td>
<td>0 0 4</td>
</tr>
<tr>
<td>link</td>
<td>724 1738</td>
<td>308 1158 4</td>
<td></td>
<td>oow-solo</td>
<td>40 87</td>
<td>27 63 4</td>
</tr>
<tr>
<td>mildew</td>
<td>35 80</td>
<td>0 0 4</td>
<td></td>
<td>oow-trad</td>
<td>33 72</td>
<td>23 54 4</td>
</tr>
<tr>
<td>munin1</td>
<td>189 366</td>
<td>66 188 4</td>
<td></td>
<td>pignet2</td>
<td>3032 7264</td>
<td>1002 3730 4</td>
</tr>
<tr>
<td>munin2</td>
<td>1003 1662</td>
<td>165 451 4</td>
<td></td>
<td>pigs</td>
<td>441 806</td>
<td>48 137 4</td>
</tr>
<tr>
<td>munin3</td>
<td>1044 1745</td>
<td>96 313 4</td>
<td></td>
<td>ship-ship</td>
<td>50 114</td>
<td>24 65 4</td>
</tr>
<tr>
<td>munin4</td>
<td>1041 1843</td>
<td>215 642 4</td>
<td></td>
<td>vsd</td>
<td>38 62</td>
<td>0 0 4</td>
</tr>
<tr>
<td>munin-kgo</td>
<td>1066 1730</td>
<td>0 0 5</td>
<td></td>
<td>water</td>
<td>32 123</td>
<td>22 96 5</td>
</tr>
<tr>
<td>wilson</td>
<td>21 27</td>
<td>0 0 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Some cases could be solved with preprocessing to optimality
- Often substantial reductions obtained
- Time needed for preprocessing was small (never more than a few seconds)
Safe separators

A set of vertices S is a **safe separator** if and only if

The treewidth of G equals the maximum over the treewidth of all graphs obtained by

- Taking a connected component W of G - S
- Take the graph, induced by W ∪ S
- Make S into a clique in that graph
Use of safe separators

- **Split** the graph with safe separators (making the separator to a clique)
  - Until no safe separators can be found
- Solve each component separately
- Treewidth is **maximum** of treewidths of components
- Also possible to construct tree decomposition of input graph

- Some sufficient conditions for separators to be safe...
What separators are safe?

- Clique separators (several authors)
- Minimal almost clique separators (B, Koster, 2006)
  - S is a minimal separator: it has no separator as a proper subset
  - S is almost a clique: there is a vertex \( v \) with \( S - \{ v \} \) is a clique
- All minimal separators of size 1 and 2 (corollary)
- All minimal separators of size 3 that split off more than one vertex (BK)
If one component is contracted to the red vertex, the separator turns into a clique.

So, the treewidth of the components is at most the treewidth of $G$.

We can combine tree decompositions of the components to one of $G$ (example on next slide).

Treewidth($G$) equals maximum of treewidth components.
## Safe separators:

### Results for probabilistic networks

| instance       | |V| |E| clique | almost-clique | size 3 | # graphs | # cliques | # To Do | low |
|----------------|-----------|------------|-----------|-----------|-----------|--------|---------|----------|--------|-----|
| barley-pp      | 26        | 78         | 0         | 7         | 0         | 8      | 7       | 1        | 5      |
| diabetes-pp    | 116       | 276        | 0         | 85        | 0         | 86     | 84      | 2        | 4      |
| link-pp        | 308       | 1158       | 0         | 0         | 0         | 1      | 0       | 1        | 4      |
| munin1-pp      | 66        | 188        | 0         | 2         | 0         | 3      | 2       | 1        | 4      |
| munin2-pp      | 165       | 451        | 6         | 13        | 4         | 24     | 12      | 12       | 4      |
| munin3-pp      | 96        | 313        | 2         | 2         | 2         | 7      | 4       | 3        | 4      |
| munin4-pp      | 215       | 642        | 3         | 4         | 0         | 8      | 2       | 6        | 4      |
| oesoca+-pp     | 14        | 75         | 0         | 0         | 0         | 1      | 0       | 1        | 9      |
| oow-trad-pp    | 23        | 54         | 0         | 0         | 1         | 2      | 1       | 1        | 4      |
| oow-solo-pp    | 27        | 63         | 0         | 0         | 1         | 2      | 0       | 2        | 4      |
| pathfinder-pp  | 12        | 43         | 0         | 5         | 0         | 6      | 6       | 0        | 6      |
| pignet2-pp     | 1002      | 3730       | 0         | 0         | 0         | 1      | 0       | 1        | 4      |
| pigs-pp        | 48        | 137        | 0         | 1         | 0         | 2      | 1       | 1        | 5      |
| ship-ship-pp   | 24        | 65         | 0         | 0         | 0         | 1      | 0       | 1        | 4      |
| water-pp       | 22        | 96         | 0         | 1         | 0         | 2      | 1       | 1        | 6      |
4

KERNELIZATION
Bounds on reduced instance sizes?

- Preprocessing + a proof on quality (size of resulting instances)
- What we cannot expect: always decrease input size

- **Lemma.** If $P \neq NP$, then for any NP-complete problem $Q$ there is no polynomial time algorithm that always transforms an input of $Q$ to a smaller equivalent input
  - **Proof:** Otherwise, repeat till $O(1)$ size input, and solve.

- **Kernelization:** bounds as a function of a parameter of the input...
A parameterized problem is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet $\Sigma$.

A parameterized problem $Q$ is said to have a **kernel** of size $f(k)$ if there is an algorithm $A$, mapping inputs of $Q$ to inputs of $Q$ with

- A uses time polynomial in $|x| + k$ on input $(x,k)$
- For all inputs $(x,k)$: $Q((x,k))$ if and only if $Q(A(x,k))$
- If $A$ maps $(x,k)$ to $(x', k')$ then
  - $|x'| \leq f(k)$
  - $k' \leq f(k)$

From fixed parameter tractability (Downey, Fellows)

Intuition: bound $f(k)$ with polynomial time preprocessing
Negative result

B, Downey, Fellows, Hermelin (2008)
Treewidth has no kernel of polynomial size unless the *and-distillation conjecture does not hold.*

Treewidth
- Given: Graph $G=(V,E)$, integer $k$
- Parameter: $k$
- Question: Is the treewidth of $G$ at most $k$

Next: other parameters
Kernels for treewidth with other parameters

- Treewidth with Given Vertex Cover
  - Given: Graph $G = (V, E)$, integer $k$, vertex cover $W$ of $G$
  - Parameter: $|W| = l$
  - Question: Is the treewidth of $G$ at most $k$?

- If we do not have a vertex cover: use a 2-approximation algorithm to find one (affects constant factor)

- B, Jansen, Kratsch, 2011: A kernel with $O(k^3)$ vertices for Treewidth with Given Vertex Cover
First two rules

- **Simplicial Vertex Rule variant:**
  - If \( v \) is a simplicial vertex of degree at most \( k \), then remove \( v \)
  - If \( v \) is a simplicial vertex of degree more than \( k \), then say no
    - the treewidth of \( G \) is at most \( k+1 \)

- **Trivial Decision Rule:**
  - If \( |W| \leq k \), then say yes
    - A simple construction of treewidth at most \( |W| \) exists
Common Neighbors Rule

- Taken from “linear time algorithm” B, 1997
- Safe rule

**Common Neighbors Rule**

- If \( x \) and \( y \) are in \( W \), and \( x \) and \( y \) have at least \( k+1 \) common neighbors, then add the edge \( \{x,y\} \)
Kernelization algorithm and bound

- The algorithm:
  - while a rule can be applied do apply it

- Theorem: The reduced graph has $|W|^{2k} = O(k^3)$ vertices.
  - Proof:
    Each vertex $v$ in $V' - W$ is not simplicial, so has two nonadjacent neighbors, which must be in $W$. Assign $v$ to such a pair. Each nonadjacent pair of vertices in $W$ has at most $k$ vertices assigned to it, otherwise the common neighbor rule applies.
Feedback vertex set parameter and almost simplicial vertices

- **Theorem** (B, Jansen, Kratsch, 2011): Treewidth with a Given Feedback Vertex Set has a kernel with $O(k^4)$ vertices.

- Feedback vertex set: a set of vertices $W$ such that $G[V-W]$ is a forest.

- A leaf in the forest that is not almost simplicial has two non-adjacent neighbors in $W$
  - Motivates to remove almost simplicial vertices

\[ = \text{almost simplicial} \]
New Almost Simplicial Vertex Rule

- If \( v \) is an almost simplicial vertex, then
  - If the degree of \( v \) is at most \( k \) then make the neighbors of \( v \) into a clique and remove \( v \)
    - Same as old rule
  - If the degree of \( v \) is at least \( k+2 \) then say no
    - \( G \) has a clique of size \( k+2 \) so treewidth is larger than \( k \)
  - If the degree of \( v \) is exactly \( k+1 \) then
    1. If for each pair of neighbors \( x, y \) of \( w \) we have that \( \{x, y\} \in E \) or there is a path from \( x \) to \( y \) that avoids \( N[v] \), then say no
    2. Otherwise, make the neighbors of \( v \) into a clique and remove \( v \)

Rule that removes all (regardless of degree) simplicial vertices

Test can be carried out in polynomial time
Experimental evaluation

- New almost simplicial vertex rule can be changed into preprocessing step
  - Instead of saying no, increase the lower bound by one
  - Eventually, all almost simplicial vertices are removed
- Also, common neighbors rule and generalization (disjoint paths, implemented with flow / Ford-Fulkerson) were implemented
- Work fall 2011, Kreuzen, (B, Kratsch)
Experiments and first results

- Experiment 1: use the New Almost Simplicial Vertex Rule on 570 graphs from “our” usual test set (all graphs except a few very large ones)
  - On 116 instances, additional reductions with the new rule could be applied
  - Additional reductions ranged from 1 vertex to all vertices of graph
  - Running time was somewhat larger, up to 1.7 seconds for 645 vertex graph initx.1.2, but still “good”

- Experiment 2: experiment 1 + Common Neighbors and Disjoint Paths Rule
  - Sometimes slow (worst case $O(n^7)$) but still practical
  - Further reductions obtained
  - Adding edges to graph actually is good!
### Some results of experiment 1

| graph     | $|V|$ | $|E|$ | Old $|V'|$ | Old $|E'|$ | Old low | New $|V'|$ | New $|E'|$ | New low |
|-----------|-----|-----|--------|--------|--------|--------|--------|--------|--------|
| barley    | 48  | 126 | 26     | 78     | 4      | 25     | 76     | 5      |
| celar04   | 340 | 1009| 114    | 524    | 6      | 105    | 476    | 8      |
| miles500  | 128 | 1170| 103    | 1068   | 8      | 79     | 827    | 22     |
| zeroin.i.2| 211 | 3541| 157    | 3541   | 4      | 57     | 1097   | 31     |
Ongoing work

- Better implementation for experiment 2 (mixing rules to speedup)
- Random graphs
- More test data

General conclusion: the new rules help in practice to obtain further reductions!
CONCLUSIONS
Conclusions

- This talk:
  - Preprocessing heuristics for treewidth: reductions and separators
  - Kernels for treewidth
    - “Vertex cover” result only used existing rules + a counting argument
    - “Feedback vertex set” result uses generalization of almost simplicial vertex rule to bound number of leaves in forest + new rules + counting argument

- Interaction between (algorithmic) graph theory and experimental work

- Ongoing work: kernels for pathwidth (B, Jansen, Kratsch)