Contracting chordal graphs

Petr A. Golovach  
Durham University

Marcin Kamiński  
Université Libre de Bruxelles

Daniël Paulusma  
Durham University

MFCS 2011, Warszawa
This talk is about containment relations in chordal graphs.

What are containment relations? What are chordal graphs?
This talk is about containment relations in chordal graphs.

What are containment relations? What are chordal graphs?
This talk is about containment relations in chordal graphs.

What are containment relations? What are chordal graphs?
CONTAINMENT RELATIONS
Containment relations

A containment relation is a poset on the set of graphs.

\[ H \leq G, \text{ where } \leq \text{ is being a}
\begin{itemize}
  \item subgraph,
  \item induced subgraph,
  \item minor,
  \item ...
\end{itemize}
Containment relations

A *containment relation* is a poset on the set of graphs.

\[ H \leq G, \text{ where } \leq \text{ is being a } \]

- subgraph,
- induced subgraph,
- minor,
- ...
Containment relations

A containment relation is a poset on the set of graphs.

\( H \preceq G \), where \( \preceq \) is being a

- subgraph,
- induced subgraph,
- minor,
- ...

Containment relations

A *containment relation* is a poset on the set of graphs. 

\[ H \leq G, \text{ where } \leq \text{ is being a} \]

- subgraph,
- induced subgraph,
- minor,
- ...

Containment relations

A *containment relation* is a poset on the set of graphs.

\[ H \leq G, \text{ where } \leq \text{ is being a} \]

- subgraph,
- induced subgraph,
- minor,
- ...
Containment relations

A containment relation is a poset on the set of graphs.

$H \leq G$, where $\leq$ is being a

- subgraph,
- induced subgraph,
- minor,
- ...

Basic operations

\[ G \backslash v = \text{vertex deletion} \]
\[ G \backslash e = \text{edge deletion} \]
\[ G / e = \text{edge contraction} \]
## Containment relations

<table>
<thead>
<tr>
<th>$G \setminus v$</th>
<th>$G \setminus e$</th>
<th>$G/ e$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>-</td>
<td>subgraph</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>-</td>
<td>induced subgraph</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>minor</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>induced minor</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>+</td>
<td>contraction</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>-</td>
<td>“spanning” subgraph</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>+</td>
<td>“spanning” minor</td>
</tr>
</tbody>
</table>
Algorithmic questions

We are interested in deciding whether $H \leq G$ for some containment relation $\leq$. 
1. The problem of deciding if a graph has a Hamiltonian cycle is NP-complete. [Karp 1972]

2. There is a $\mathcal{O}(|G|^{|H|})$ algorithm to decide whether $H$ is an (induced) subgraph of $G$.

Can we do better?
(Induced) subgraphs

1. The problem of deciding if a graph has a Hamiltonian cycle is NP-complete. [Karp 1972]

2. There is a $\mathcal{O}(|G|^{|H|})$ algorithm to decide whether $H$ is an (induced) subgraph of $G$.

Can we do better?
(Induced) subgraphs

1. The problem of deciding if a graph has a Hamiltonian cycle is NP-complete. [Karp 1972]

2. There is a $O(|G|^{|H|})$ algorithm to decide whether $H$ is an (induced) subgraph of $G$.

Can we do better?
(Induced) subgraphs

3. $\mathcal{O}(|G|^{|H|/3}) = \mathcal{O}(|G|^{.792|H|})$ [Nešetřil and Poljak 1985]

Much better?

4. Both problems are $\mathcal{W}[1]$-complete. Most likely no FPT algorithm $\Rightarrow |H|$ must stay in the exponent.
3. $O(|G|^{\alpha|H|/3}) = O(|G|^{0.792|H|})$ [Nešetřil and Poljak 1985]
  Much better?

4. Both problems are $\mathcal{W}[1]$-complete. Most likely no FPT algorithm $\Rightarrow |H|$ must stay in the exponent.
3. $O(|G|^{|H|/3}) = O(|G|^{0.792|H|})$ [Nešetřil and Poljak 1985]
   Much better?

4. Both problems are $\mathcal{W}[1]$-complete. Most likely no FPT algorithm $\Rightarrow |H|$ must stay in the exponent.
(Induced) subgraphs

3. $O(|G|^{|H|/3}) = O(|G|^{.792}|H|)$ [Nešetřil and Poljak 1985]
   Much better?

4. Both problems are $W[1]$-complete. Most likely no FPT algorithm $\Rightarrow |H|$ must stay in the exponent.
Complexity theory for parametrized problems.

Class FPT
Input: graph $G$ and parameter $k$
Running time: $f(k) \cdot |G|^{O(1)}$
Vertex Cover is FPT.

Class W[1]
Independent Set is W[1]-complete.

Conjecture. FPT $\neq$ W[1]
Complexity theory for parametrized problems.

Class FPT

Input: graph $G$ and parameter $k$
Running time: $f(k) \cdot |G|^{O(1)}$

Vertex Cover is FPT.

Class W[1]

Independent Set is W[1]-complete.

Conjecture. $\text{FPT} \neq \text{W[1]}$
Complexity theory for parametrized problems.

Class FPT

Input: graph $G$ and parameter $k$

Running time: $f(k) \cdot |G|^{O(1)}$

Vertex Cover is FPT.

Class $W[1]$

Independent Set is $W[1]$-complete.

Conjecture. $FPT \neq W[1]$
Complexity theory for parametrized problems.

Class FPT

Input: graph $G$ and parameter $k$

Running time: $f(k) \cdot |G|^{O(1)}$

Vertex Cover is FPT.

Class W[1]

Independent Set is W[1]-complete.

Conjecture. $\text{FPT} \neq \text{W[1]}$
Complexity theory for parametrized problems.

Class FPT

Input: graph $G$ and parameter $k$

Running time: $f(k) \cdot |G|^{O(1)}$

Vertex Cover is FPT.

Class W[1]

Independent Set is W[1]-complete.

Conjecture. $FPT \neq W[1]$
Complexity theory for parametrized problems.

Class FPT

Input: graph $G$ and parameter $k$

Running time: $f(k) \cdot |G|^{O(1)}$

Vertex Cover is FPT.

Class $W[1]$

Independent Set is $W[1]$-complete.

Conjecture. $\text{FPT} \neq W[1]$
Complexity theory for parametrized problems.

Class FPT
Input: graph $G$ and parameter $k$
Running time: $f(k) \cdot |G|^{O(1)}$
Vertex Cover is FPT.

Class W[1]
Independent Set is W[1]-complete.

Conjecture. FPT $\neq$ W[1]
Complexity theory for parametrized problems.

Class FPT
Input: graph $G$ and parameter $k$
Running time: $f(k) \cdot |G|^{O(1)}$
Vertex Cover is FPT.

Class W[1]
Independent Set is W[1]-complete.

Conjecture. FPT ≠ W[1]
Minors

N. Robertson, P. D. Seymour, Graph Minors. I. Excluding a Forest, JCTB, 35, 39-61, 1983
N. Robertson, P. D. Seymour, Graph Minors. II. Algorithmic Aspects of Tree-Width, J. of Alg., 7, 309-322, 1986
N. Robertson, P. D. Seymour, Graph Minors. III. Planar Tree-Width, JCTB, 36, 49-64, 1984
N. Robertson, P. D. Seymour, Graph Minors. IV. Tree-Width and Well-Quasi-Ordering, JCTB, 48, 227-254, 1990
N. Robertson, P. D. Seymour, Graph Minors. V. Excluding a Planar Graph, JCTB, 41, 92-114, 1986
N. Robertson, P. D. Seymour, Graph Minors. VI. Disjoint Paths across a Disc, JCTB, 41, 115-138, 1986
N. Robertson, P. D. Seymour, Graph Minors. VII. Disjoint Paths on a Surface, JCTB, 45, 212-254, 1988
N. Robertson, P. D. Seymour, Graph Minors. VIII. A Kuratowski Theorem for General Surfaces, JCTB, 48, 255-288, 1990
N. Robertson, P. D. Seymour, Graph Minors. IX. Disjoint Crossed Paths, JCTB, 49, 40-77, 1990
N. Robertson, P. D. Seymour, Graph Minors. X. Obstructions to Tree- Decomposition, JCTB, 52, 153-190, 1991
N. Robertson, P. D. Seymour, Graph Minors. XI. Circuits on a Surface, JCTB, 60, 72-106, 1994
N. Robertson, P. D. Seymour, Graph Minors. XII. Distance on a Surface, JCTB, 64, 240-272, 1995
N. Robertson, P. D. Seymour, Graph Minors. XIII. The Disjoint Paths Problem, JCTB, 63, 65-110, 1995
N. Robertson, P. D. Seymour, Graph Minors. XIV. Extending an Embedding, JCTB, 65, 23-50, 1995
N. Robertson, P. D. Seymour, Graph Minors. XV. Giant Steps, JCTB, 68, 112-148, 1996
N. Robertson, P. D. Seymour, Graph Minors. XVI. Excluding a Non-planar Graph, JCTB, 89, 43-76, 2003
N. Robertson, P. D. Seymour, Graph Minors. XVII. Taming a Vortex, JCTB, 77, 162-210, 1999
N. Robertson, P. D. Seymour, Graph Minors. XVIII. Tree-decompositions and well-quasi-ordering, JCTB, 89, 77-108, 2003
N. Robertson, P. D. Seymour, Graph Minors. XIX. Well-quasi-ordering on a Surface, JCTB, 90, 325-385, 2004
N. Robertson, P. D. Seymour, Graph Minors. XX. Wagner’s Conjecture, JCTB, 92, 325-357, 2004
N. Robertson, P. D. Seymour, Graph Minors. XXI. Graphs with Unique Linkages, JCTB, 99, 583-616, 2009
N. Robertson, P. D. Seymour, Graph Minors. XXII. Irrelevant Vertices in Linkage Problems, JCTB, TO APPEAR
N. Robertson, P. D. Seymour, Graph Minors. XXIII. Nash-Williams’ Immersion Conjecture, JCTB, 100, 181-205, 2010
Minors

N. Robertson, P. D. Seymour, *Graph Minors. III. Planar Tree-Width*, JCTB, 36, 49-64, 1984
N. Robertson, P. D. Seymour, *Graph Minors. V. Excluding a Planar Graph*, JCTB, 41, 92-114, 1986
N. Robertson, P. D. Seymour, *Graph Minors. VI. Disjoint Paths across a Disc*, JCTB, 41, 115-138, 1986
N. Robertson, P. D. Seymour, *Graph Minors. VII. Disjoint Paths on a Surface*, JCTB, 45, 212-254, 1988
N. Robertson, P. D. Seymour, *Graph Minors. IX. Disjoint Crossed Paths*, JCTB, 49, 40-77, 1990
N. Robertson, P. D. Seymour, *Graph Minors. X. Obstructions to Tree- Decomposition*, JCTB, 52, 153-190, 1991
N. Robertson, P. D. Seymour, *Graph Minors. XI. Circuits on a Surface*, JCTB, 60, 72-106, 1994
N. Robertson, P. D. Seymour, *Graph Minors. XII. Distance on a Surface*, JCTB, 64, 240-172, 1995
N. Robertson, P. D. Seymour, *Graph Minors. XIII. The Disjoint Paths Problem*, JCTB, 63, 65-110, 1995
N. Robertson, P. D. Seymour, *Graph Minors. XVI. Excluding a Non-planar Graph*, JCTB, 89, 43-76, 2003
N. Robertson, P. D. Seymour, *Graph Minors. XVII. Taming a Vortex*, JCTB, 77, 162-210, 1999
N. Robertson, P. D. Seymour, *Graph Minors. XIX. Well-quasi-ordering on a Surface*, JCTB, 90, 325-385, 2004
N. Robertson, P. D. Seymour, *Graph Minors. XX. Wagner's Conjecture*, JCTB, 92, 325-357, 2004
N. Robertson, P. D. Seymour, *Graph Minors. XXI. Graphs with Unique Linkages*, JCTB, 99, 583-616, 2009
N. Robertson, P. D. Seymour, *Graph Minors. XXII. Irrelevant Vertices in Linkage Problems*, JCTB, to appear
N. Robertson, P. D. Seymour, *Graph Minors. XXIII. Nash-Williams' Immersion Conjecture*, JCTB, 100, 181-205, 2010
Minors

Theorem (Robertson and Seymour)

Testing if a fixed $H$ is a minor of the input graph $G$ can be done in time $f(|H|)|G|^3$.

Minor testing is FPT.
Minors

**Theorem (Robertson and Seymour)**

*Testing if a fixed $H$ is a minor of the input graph $G$ can be done in time $f(|H|)|G|^3$.*

Minor testing is FPT.
Minors

Theorem (Matoušek and Thomas 1982)

Given two input graphs $G$ and $H$, to decide whether $H$ is a minor of $G$ is NP-complete even with one of the following restrictions:

- $H$ and $G$ are trees of bounded diameter,
- $H$ and $G$ are trees all of whose vertices but one have degree $\leq 5$. 
Minors

Theorem (Matoušek and Thomas 1982)

Given two input graphs $G$ and $H$, to decide whether $H$ is a minor of $G$ is NP-complete even with one of the following restrictions:

- $H$ and $G$ are trees of bounded diameter,
- $H$ and $G$ are trees all of whose vertices but one have degree $\leq 5$. 
Minors

Theorem (Matoušek and Thomas 1982)

Given two input graphs $G$ and $H$, to decide whether $H$ is a minor of $G$ is \textsc{NP}-complete even with one of the following restrictions:

- $H$ and $G$ are trees of bounded diameter,
- $H$ and $G$ are trees all of whose vertices but one have degree $\leq 5$. 
Induced minors

A tree $T_1$ is a minor a tree $T_2$ $\iff$ $T_1$ is an induced minor of $T_2$ $\iff$ $T_1$ is a contraction of $T_2$.

The theorem of Matoušek and Thomas applies to induced minors and contractions.
Induced minors

A tree $T_1$ is a minor a tree $T_2$

$\iff$

$T_1$ is an induced minor of $T_2$

$\iff$

$T_1$ is a contraction of $T_2$.

The theorem of Matoušek and Thomas applies to induced minors and contractions.
Induced minors

A tree $T_1$ is a minor a tree $T_2$ \iff $T_1$ is an induced minor of $T_2$ \iff $T_1$ is a contraction of $T_2$.

The theorem of Matoušek and Thomas applies to induced minors and contractions.
Induced minors

A tree $T_1$ is a minor a tree $T_2$  
$\iff$  
$T_1$ is an induced minor of $T_2$  
$\iff$  
$T_1$ is a contraction of $T_2$.

The theorem of Matoušek and Thomas applies to induced minors and contractions.
Induced minors

A tree $T_1$ is a minor a tree $T_2$ $\iff$ $T_1$ is an induced minor of $T_2$ $\iff$ $T_1$ is a contraction of $T_2$.

The theorem of Matoušek and Thomas applies to induced minors and contractions.
Downsides of induced minors

- There are non-recursive classes of graphs that are closed under induced minors. [Matoušek, Nešetřil, Thomas 1988]
- There exists a graph $H$ such that testing if $H$ is an induced minor of the input graph is NP-complete. $|H| = 68$. [Fellows, Kratochvíl, Middendorf, Pfeiffer 1995]
Downsides of induced minors

- There are non-recursive classes of graphs that are closed under induced minors. [Matoušek, Nešetřil, Thomas 1988]
- There exists a graph $H$ such that testing if $H$ is an induced minor of the input graph is NP-complete. $|H| = 68$. [Fellows, Kratochvıl, Middendorf, Pfeiffer 1995]
Downsides of induced minors

- There are non-recursive classes of graphs that are closed under induced minors. [Matoušek, Nešetřil, Thomas 1988]
- There exists a graph $H$ such that testing if $H$ is an induced minor of the input graph is NP-complete. $|H| = 68$. [Fellows, Kratochvíl, Middendorf, Pfeiffer 1995]
Testing for induced minors

Theorem (Fellows, Kratochvíl, Middendorf, Pfeiffer 1995)

Testing if $H$ is an induced minor of a planar graph $G$ is in FPT.
Theorem (Brouwer and Veldman 1987)

Let $H$ be a connected triangle-free graph. If $H$ is a star, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

NP-complete even for $H = P_4$ or $H = C_4$. 

Theorem (Brouwer and Veldman 1987)

Let $H$ be a connected triangle-free graph. If $H$ is a star, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

NP-complete even for $H = P_4$ or $H = C_4$. 

Contractions
Theorem (Brouwer and Veldman 1987)

Let $H$ be a connected triangle-free graph. If $H$ is a star, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

NP-complete even for $H = P_4$ or $H = C_4$. 

Contractions
Theorem (Brouwer and Veldman 1987)

Let $H$ be a connected triangle-free graph. If $H$ is a star, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is $\text{NP}$-complete.

$\text{NP}$-complete even for $H = P_4$ or $H = C_4$. 
Theorem (Brouwer and Veldman 1987)

Let $H$ be a connected triangle-free graph. If $H$ is a star, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is \textsc{NP}-complete.

\textsc{NP}-complete even for $H = P_4$ or $H = C_4$. 
Small patterns

Theorem (Levin, Paulusma, Woeginger 2003)
Let $H$ be a connected graph on at most 5 vertices. If $H$ has a dominating vertex, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

There is a graph $H$ (on 69 vertices) with a dominating vertex such that testing whether the input graph can be contracted to $H$ is NP-complete.
Small patterns

Theorem (Levin, Paulusma, Woeginger 2003)

Let $H$ be a connected graph on at most 5 vertices. If $H$ has a dominating vertex, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

There is a graph $H$ (on 69 vertices) with a dominating vertex such that testing whether the input graph can be contracted to $H$ is NP-complete.
Theorem (Levin, Paulusma, Woeginger 2003)

Let $H$ be a connected graph on at most 5 vertices. If $H$ has a dominating vertex, then testing whether the input graph can be contracted to $H$ is solvable in polynomial time; otherwise, it is NP-complete.

There is a graph $H$ (on 69 vertices) with a dominating vertex such that testing whether the input graph can be contracted to $H$ is NP-complete.
Theorem (with Paulusma and Thilikos 2010)

There exists an algorithm testing in time \(|G|^{O(|H|)}\) whether \(H\) is a contraction of a planar input graph \(G\).

Theorem (with Thilikos 2010)

There exists an algorithm testing in FPT time whether \(H\) is a contraction of a planar input graph \(G\).

Generalizes to other surfaces.
Theorem (with Paulusma and Thilikos 2010)
There exists an algorithm testing in time $|G|^\mathcal{O}(|H|)$ whether $H$ is a contraction of a planar input graph $G$.

Theorem (with Thilikos 2010)
There exists an algorithm testing in FPT time whether $H$ is a contraction of a planar input graph $G$.

Generalizes to other surfaces.
Contractions in planar graphs

Theorem (with Paulusma and Thilikos 2010)
There exists an algorithm testing in time $|G|^{O(|H|)}$ whether $H$ is a contraction of a planar input graph $G$.

Theorem (with Thilikos 2010)
There exists an algorithm testing in FPT time whether $H$ is a contraction of a planar input graph $G$.

Generalizes to other surfaces.
CHORDAL GRAPHS
A graph $G$ is **chordal** if it has no induced cycle of length at least 4. That is, every cycle of length at least 4 in $G$ has a chord.

Chordal graphs = intersection graphs of subtrees of a tree

Chordal graphs are closed under contractions and vertex deletions.
A graph $G$ is chordal if it has no induced cycle of length at least 4. That is, every cycle of length at least 4 in $G$ has a chord.

Chordal graphs = intersection graphs of subtrees of a tree

Chordal graphs are closed under contractions and vertex deletions.
A graph $G$ is **chordal** if it has no induced cycle of length at least 4. That is, every cycle of length at least 4 in $G$ has a chord.

chordal graphs = intersection graphs of subtrees of a tree

Chordal graphs are closed under contractions and vertex deletions.
Chordal graphs

A graph $G$ is **chordal** if it has no induced cycle of length at least 4. That is, every cycle of length at least 4 in $G$ has a chord.

chordal graphs = intersection graphs of subtrees of a tree

Chordal graphs are closed under contractions and vertex deletions.
Split graphs

A graph $G$ is **split** if its vertex set can be partitioned into a clique and an independent set.

Split graphs = intersection graphs of substars of a star

Split graphs are chordal.

Split graphs are closed under contractions and vertex deletions.
A graph $G$ is **split** if its vertex set can be partitioned into a clique and an independent set.

**split graphs = intersection graphs of substars of a star**

Split graphs are chordal.

Split graphs are closed under contractions and vertex deletions.
Split graphs

A graph $G$ is split if its vertex set can be partitioned into a clique and an independent set.

split graphs = intersection graphs of substars of a star

Split graphs are chordal.

Split graphs are closed under contractions and vertex deletions.
A graph $G$ is **split** if its vertex set can be partitioned into a clique and an independent set.

split graphs = intersection graphs of substars of a star

Split graphs are chordal.

Split graphs are closed under contractions and vertex deletions.
A **tree decomposition** of a graph $G$ is a pair $(X, T)$ such that $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,
2. for every $uv \in E(G)$, there exits $X_i$ such that $u, v \in X_i$, and
3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
A tree decomposition of a graph $G$ is a pair $(X, T)$ such that

1. $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree,
2. $V(T) = X$, and
3. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and
4. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 

Tree decomposition
A **tree decomposition** of a graph $G$ is a pair $(X, T)$ such that $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,
2. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and
3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
A tree decomposition of a graph $G$ is a pair $(X, T)$ such that $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,
2. for every $uv \in E(G)$, there exits $X_i$ such that $u, v \in X_i$, and
3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 

Tree decomposition
Tree decomposition

A tree decomposition of a graph $G$ is a pair $(X, T)$ such that $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,
2. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and
3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
A tree decomposition of a graph $G$ is a pair $(X, T)$ such that

$X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree,

$V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,

2. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and

3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
A tree decomposition of a graph $G$ is a pair $(X, T)$ such that

$X = \{X_1, \ldots, X_k\}, \ X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,

2. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and

3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
A **tree decomposition** of a graph $G$ is a pair $(X, T)$ such that $X = \{X_1, \ldots, X_k\}$, $X_i \subseteq V(G)$ for all $k = 1, \ldots, k$, $T$ is a tree, $V(T) = X$, and

1. $X_1 \cup \ldots \cup X_k = V(G)$,
2. for every $uv \in E(G)$, there exists $X_i$ such that $u, v \in X_i$, and
3. for every $v \in V(G)$, all nodes $X_i$ containing $v$ form a subtree of $T$. 
The width of a tree decomposition \((X, T)\) is the largest of \(|X_i|\) minus 1, for all \(i = 1, \ldots, k\).

The treewidth of a graph \(G\) is the minimum width over all tree decompositions of \(G\).

Many algorithmic problems are easy in classes of bounded treewidth. Dynamic programming.
Treewidth

The width of a tree decomposition \((X, T)\) is the largest of \(|X_i|\) minus 1, for all \(i = 1, \ldots, k\).

The treewidth of a graph \(G\) is the minimum width over all tree decompositions of \(G\).

Many algorithmic problems are easy in classes of bounded treewidth. Dynamic programming.
The **width** of a tree decomposition \((X, T)\) is the largest of \(|X_i|\) minus 1, for all \(i = 1, \ldots, k\).

The **treewidth** of a graph \(G\) is the minimum width over all tree decompositions of \(G\).

Many algorithmic problems are *easy* in classes of bounded treewidth. Dynamic programming.
The width of a tree decomposition \((X, T)\) is the largest of \(|X_i|\) minus 1, for all \(i = 1, \ldots, k\).

The treewidth of a graph \(G\) is the minimum width over all tree decompositions of \(G\).

Many algorithmic problems are easy in classes of bounded treewidth. Dynamic programming.
Tree decompositions of chordal graphs

Lemma (Gavril 1974)

A graph is chordal if and only if it has a tree decomposition into maximal cliques of $G$. 
CONTAINMENT RELATIONS
IN
CHORDAL GRAPHS
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.

2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.

3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Lemma
Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Subgraphs

Lemma

Deciding whether a fixed graph $H$ is a subgraph of the input chordal graph $G$ is FPT (linear-time).

Proof. Win-win!

1. Find the tree decomposition of $G$.
2. If the width of the decomposition is $< |H|$, then the graph is of treewidth $< |H|$. Apply dynamic programming.
3. Otherwise, there is a clique of size $\geq |H|$ in $G$. Hence, $H$ is a subgraph of $G$. 
Monotone containment relations

The same argument works for other containment relations that allow edge deletion and are *easy* on graphs of bounded treewidth.

Works for: subgraphs, minors, topological minors, immersions, ...
Monotone containment relations

The same argument works for other containment relations that allow edge deletion and are easy on graphs of bounded treewidth.

Works for: subgraphs, minors, topological minors, immersions, ...
Monotone containment relations

The same argument works for other containment relations that allow edge deletion and are *easy* on graphs of bounded treewidth.

Works for: subgraphs, minors, topological minors, immersions, ...
Monotone containment relations

The same argument works for other containment relations that allow edge deletion and are \textit{easy} on graphs of bounded treewidth.

Works for: subgraphs, minors, topological minors, immersions, ...
The same argument works for other containment relations that allow edge deletion and are *easy* on graphs of bounded treewidth.

Works for: subgraphs, minors, topological minors, immersions, ...
Theorem (with Golovach, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^f(|H|))$ whether a split graph $H$ is a contraction/induced minor of a chordal input graph $G$.

Remember: split graphs are chordal.

Theorem (with Belmonte, Golovach, Heggernes, van ‘t Hof, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^{|H|^2})$ whether a graph $H$ is a contraction/induced minor of a chordal input graph $G$. 
Theorem (with Golovach, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^f(|H|))$ whether a split graph $H$ is a contraction/induced minor of a chordal input graph $G$.

Remember: split graphs are chordal.

Theorem (with Belmonte, Golovach, Heggernes, van ’t Hof, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^{|H|^2})$ whether a graph $H$ is a contraction/induced minor of a chordal input graph $G$. 
Theorem (with Golovach, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^{|H|})$ whether a split graph $H$ is a contraction/induced minor of a chordal input graph $G$.

Remember: split graphs are chordal.

Theorem (with Belmonte, Golovach, Heggernes, van ’t Hof, Paulusma 2011)

There exists an algorithm that decides in time $O(|G|^{|H|^2})$ whether a graph $H$ is a contraction/induced minor of a chordal input graph $G$. 
Proof.

- Dynamic programming over the tree decomposition of $G$.
- Reduction to an auxiliary problem: restricted $k$-Disjoint-Paths. Each path can only use prescribed vertices.
Proof.

- Dynamic programming over the tree decomposition of $G$.
- Reduction to an auxiliary problem: restricted $k$-Disjoint-Paths. Each path can only use prescribed vertices.
Proof.

- Dynamic programming over the tree decomposition of $G$.
- Reduction to an auxiliary problem: restricted $k$-Disjoint-Paths. Each path can only use prescribed vertices.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $\mathcal{O}(|G|^{|H|})$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

Theorem

Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is W[1]-complete.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $O(|G|^{|H|})$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

**Theorem**

Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is W[1]-complete.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $O(|G|^{|H|})$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

Theorem
Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is $W[1]$-complete.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $O(|G|^{|H|})$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

Theorem
Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is W[1]-complete.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $O(|G||^H|)$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

Theorem
Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is $\text{W}[1]$-complete.
Contractions in chordal graphs

Can we improve the exponent?

The complexity is $\mathcal{O}(|G|^{|H|})$ for interval graphs. (Interval graphs are intersection graphs of subpaths of a path, and are chordal.)

FPT?

**Theorem**

*Testing whether $H$ is a contraction/induced minor/induced subgraph of a split input graph $G$ is W[1]-complete.*
Open problems

1. Is testing whether a graph $H$ is a contraction of an interval graph $W[1]$-hard?

2. Is testing whether a graph $H$ is a contraction of a proper interval graph FPT?
Open problems

1. Is testing whether a graph $H$ is a contraction of an interval graph W[1]-hard?

2. Is testing whether a graph $H$ is a contraction of a proper interval graph FPT?
Open problems

1. Is testing whether a graph $H$ is a contraction of an interval graph $W[1]$-hard?

2. Is testing whether a graph $H$ is a contraction of a proper interval graph FPT?
THANK YOU!