TECHNISCHE
UNIVERSITÄT
DRESDEN

DATABASE THEORY

Lecture 6: Tree-like Conjunctive Queries

Markus Krötzsch

TU Dresden, 12 May 2016

Overview

See course homepage [⇒ link] for more information and materials

Review

Conjunctive queries (CQs) are simpler than FO-queries:

- NP combined and query complexity (instead of $\mathrm{PS{\small PACE}}$)
- data complexity remains in AC^0

CQs become even simpler if they are tree-shaped:

- GYO algorithm defines acyclic hypergraphs
- acyclic hypergraphs have join trees
- join trees can be evaluated in P with Yannakakis' Algorithm

This time:

- Find more general conditions that make CQs tractable
  $\leadsto$ "tree-like" queries that that are not really trees
- Play some games

Is Yannakakis' Algorithm Optimal?

We saw that tree queries can be evaluated in polynomial time, but we know that there are much simpler complexity classes:

$$\mathrm{NC}^0 \subset \mathrm{AC}^0 \subset \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1 \subseteq \ldots \subseteq \mathrm{NC} \subseteq \mathrm{P}$$

Is Yannakakis' Algorithm Optimal?

We saw that tree queries can be evaluated in polynomial time, but we know that there are much simpler complexity classes:

$$\mathrm{NC}^0 \subset \mathrm{AC}^0 \subset \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1 \subseteq \ldots \subseteq \mathrm{NC} \subseteq \mathrm{P}$$

Indeed, tighter bounds have been shown:

Theorem (Gottlob, Leone, Scarcello: J. ACM 2001)

Answering tree BCQs is complete for LOGCFL.

LOGCFL: the class of problems $\textsc{LogSpace}$-reducible to the word problem of a context-free language:

$$\mathrm{NC}^0 \subset \mathrm{AC}^0 \subset \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{LOGCFL} \subseteq \mathrm{AC}^1 \subseteq \ldots \subseteq \mathrm{NC} \subseteq \mathrm{P}$$

$\leadsto$ highly parallelisable

Generalising Tree Queries

In practice, many queries are tree queries,
but even more queries are “almost” tree queries, but not quite . . .

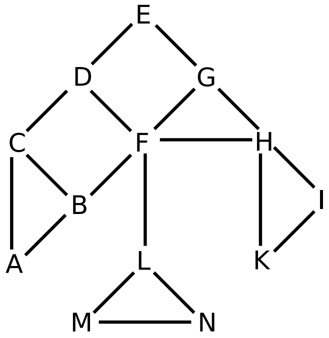
How can we formalise this idea?

Several attempts to define “tree-like” queries:

- Treewidth: a way to measure tree-likeness of graphs
- Query width: towards tree-like query graphs
- Hypertree width: adoption of treewidth to hypergraphs

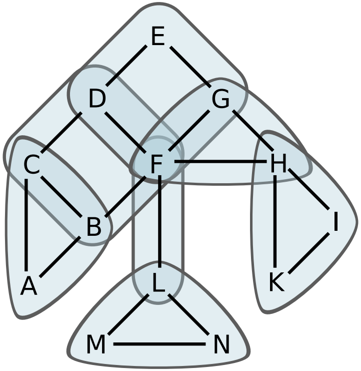
How to recognise trees . . .

. . . from quite a long way away:



How to recognise trees . . .

. . . from quite a long way away:



Tree Decompositions

Idea: if we can group the edges of a graph into bigger pieces, these pieces might form a tree structure

Definition

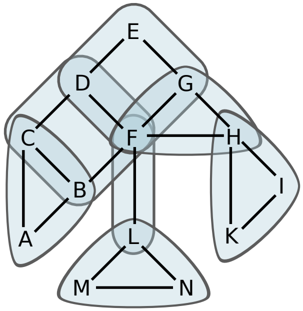
Consider a graph $G = \langle V, E\rangle$. A tree decomposition of G is a tree structure T where each node of T is a subset of V, such that:

- The union of all nodes of T is V.
- For each edge $(v_1 \to v_2) \in E$, there is a node N in T such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of T that contain v form a subtree of T; equivalently: if two nodes contain v, then all nodes on the path between them also contain v (connectedness condition).

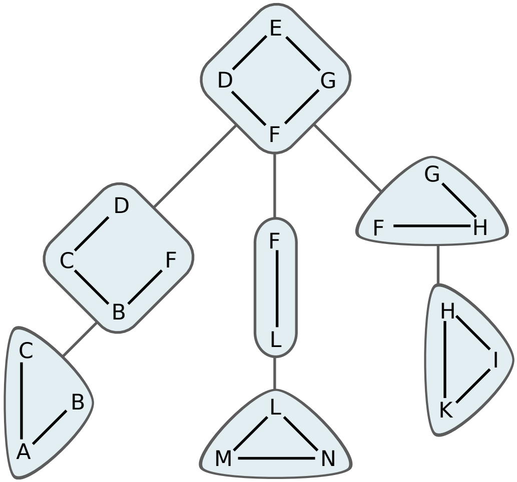
Nodes of a tree decomposition are often called bags

(not related to the common use of “bag” as a synonym for “multiset”)

Tree Decompositions: Example



Tree Decompositions: Example



Treewidth

The treewidth of a graph defines how “tree-like” it is:

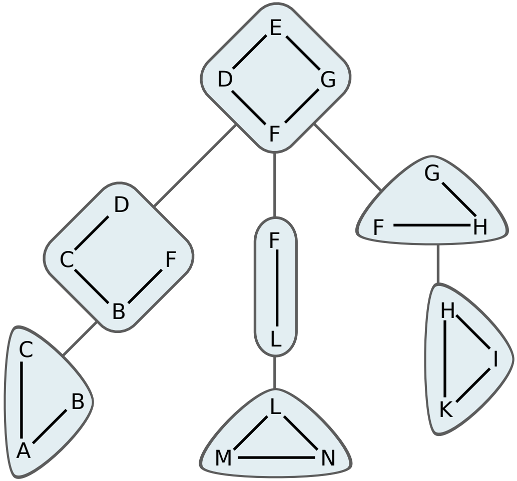
Definition

The width of a tree decomposition is the size of its largest bag minus one.
The treewidth of a graph G, denoted $\mathrm{tw}(G)$, is the smallest width of any of its tree decompositions.

Simple observations:

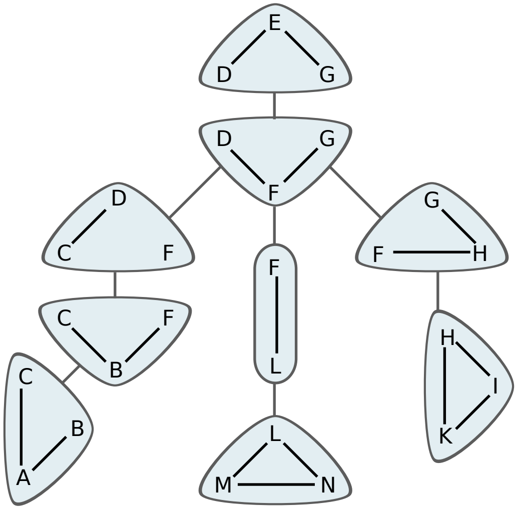
- If G is a tree, then we can decompose it into bags that contain only one edge $\leadsto$ trees have treewidth 1
- Every graph has at least one tree decomposition where all vertices are in one bag $\leadsto$ max. treewidth = number of vertices

Treewidth: Example



$\rightsquigarrow$ tree decomposition of width 3

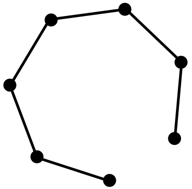
Treewidth: Example

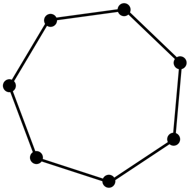


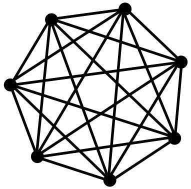
$\rightsquigarrow$ tree decomposition of width 2 = treewidth of the example graph

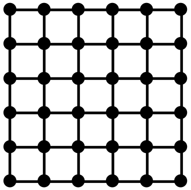
More Examples

What is the treewidth of the following graphs?









Treewidth and Conjunctive Queries

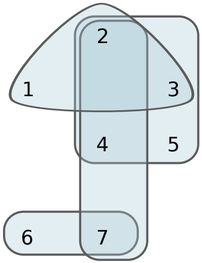
Treewidth is based on graphs, not hypergraphs

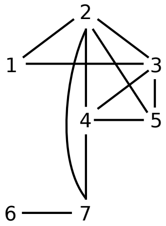
Treewidth and Conjunctive Queries

Treewidth is based on graphs, not hypergraphs

$\rightsquigarrow$ treewidth of CQ = treewidth of primal graph of query hypergraph

Query graph and corresponding primal graph:





$\rightsquigarrow$ Treewidth 3

Observation: acyclic hypergraphs can have unbounded treewidth!

Exploiting Treewidth in CQ Answering

Queries of low treewidth can be answered efficiently:

Theorem (Dechter/Chekuri+Rajamaran ’97/Kolaitis+Vardi ’98/Gottlob & al. ’98)

Answering BCQs of treewidth k is possible in time $O(n^k \log n)$, and thus in polynomial time if k is fixed.
The problem is also complete for LOGCFL.

Checking for low treewidths can also be done efficiently:

Theorem (Bodlaender ’96)

Given a graph G and a fixed number k, one can check in linear time if tw$(G) \leq k$, and the corresponding tree decomposition can also be found in linear time.

Warning: neither CQ answering nor tree decomposition might be practically feasible if k is big

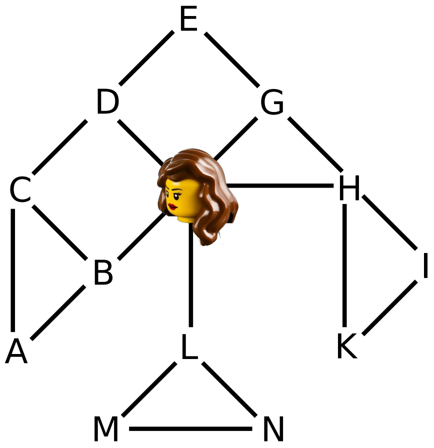
Treewidth via Games

Seymour and Thomas [1993] gave
an alternative characterisation of treewidth:

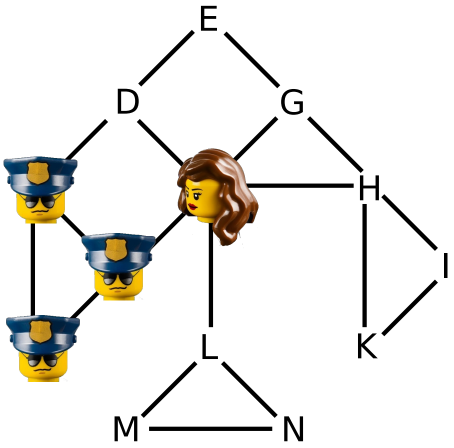
The Cops-and-Robber Game

- The game is played on a graph G
- There are k cops and one robber, each located at one vertex
- In each turn:
  - the cops can fly to an arbitrary vertex in the graph
  - the robber can run along the edges of the graph, as far as she likes, as long a she does not pass through any vertex that was occupied by a cop before or after the turn
    (the robber can run to a place where a cop was before the turn, but not pass through such a place)
- The goal of the cops is to catch the robber;
  the goal of the robber is never to be caught

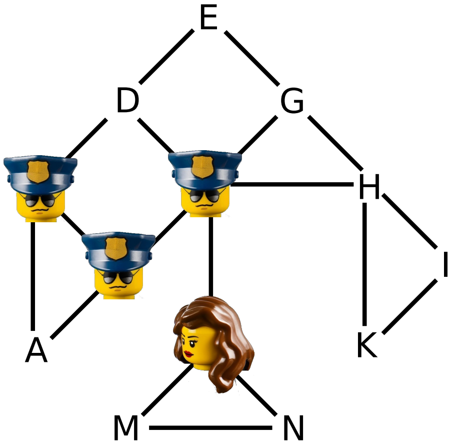
Cops and Robbers: Example



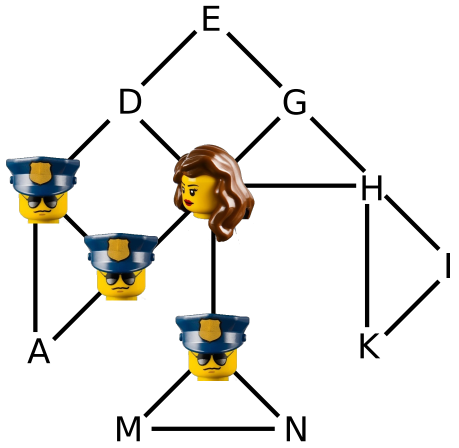
Cops and Robbers: Example



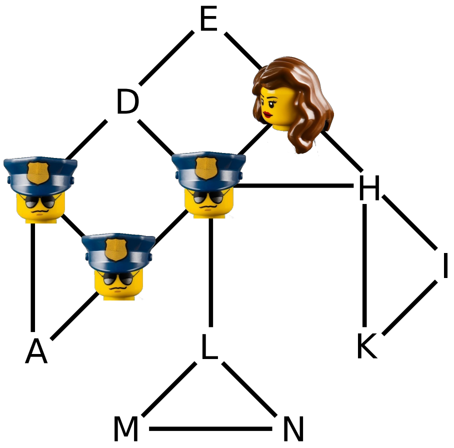
Cops and Robbers: Example



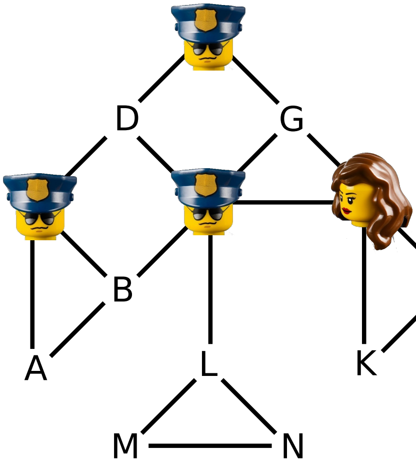
Cops and Robbers: Example



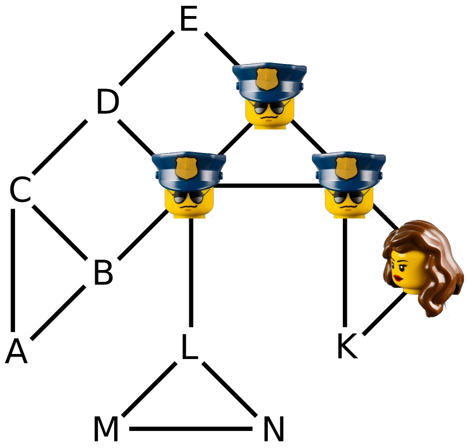
Cops and Robbers: Example



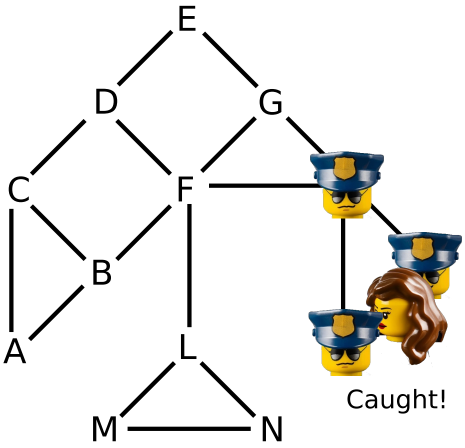
Cops and Robbers: Example



Cops and Robbers: Example



Cops and Robbers: Example



Cops & Robbers and Treewidth

Theorem (Seymour and Thomas)

A graph G is of treewidth $\leq k-1$ if and only if k cops have a winning strategy in the cops & robber game on G.

Intuition: the cops together can block even the widest branch and still move in on the robber

Treewidth via Logic

Kolaitis and Vardi [1998] gave a logical characterisation of treewidth

Bounded treewidth CQs correspond to certain FO-queries:

- We allow FO-queries with $\exists$ and $\wedge$ as only operators
- But operators can be nested in arbitrary ways (unlike in CQs)
- Theorem: A query can be expressed with a CQ of treewidth k if and only if it can be expressed in this logic using a query with at most $k + 1$ distinct variables

Intuition: variables can be reused by binding them in more than one $\exists$
$\leadsto$ Apply a kind of "inverted prenex-normal-form transformation"
$\leadsto$ Variables that occur in the same atom or in a "tightly connected" atom must use different names
$\leadsto$ minimum number of variables $\Leftrightarrow$ treewidth (+1)

Treewidth: Pros and Cons

Advantages:

- Bounded treewidth is easy to check
- Bounded treewidth CQs are easy to answer

Disadvantages:

- Even families of acyclic graphs may have unbounded treewidth
- Loss of information when using primal graph
  (cliques might be single hyperedges – linear! –
  or complex query patterns – exponential!)

$\leadsto$ Are there better ways to capture “tree-like” queries?

Query Width

Idea of Chekuri and Rajamaran [1997]:

- Create tree structure similar to tree decomposition
- But consider bags of query atoms instead of bags of variables
- Two connectedness conditions:
  1. Bags that refer to a certain variable must be connected
  2. Bags that refer to a certain query atom must be connected

Query width: least number of atoms needed in bags of a query decomposition

Query Width

Idea of Chekuri and Rajamaran [1997]:

- Create tree structure similar to tree decomposition
- But consider bags of query atoms instead of bags of variables
- Two connectedness conditions:
  (1) Bags that refer to a certain variable must be connected
  (2) Bags that refer to a certain query atom must be connected

Query width: least number of atoms needed in bags of a query decomposition

Theorem

Given a query decomposition for a BCQ, the query answering problem can be decided in time polynomial in the query width.

Problems with Query Width

Theorem (Gottlob et al. 1999)

Deciding if a query has query width at most k is NP-complete.

In particular, it is also hard to find a query decomposition

$\leadsto$ Query answering complexity drops from NP to P …
…but we need to solve another NP-hard problem first!

Generalised Hypertree Width

Gottlob, Leone, and Scarcello had another idea on defining tree-like hypergraphs:

Intuition:

- Combine key ideas of tree decomposition and query decomposition
- Start by looking at a tree decomposition
- But define the width based on query atoms:
  How many atoms do we need to cover all variables in a bag?

$\leadsto$ Generalised hypertree width
$\leadsto$ A technical condition is needed to get a simpler-to-check notion

Hypertree Width

Definition

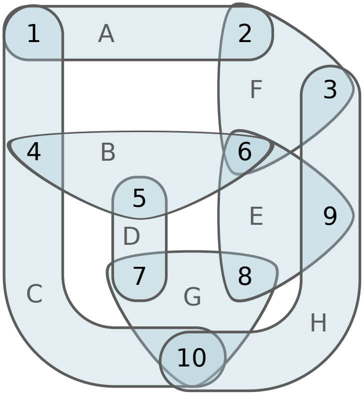
Consider a hypergraph $G = \langle V, E \rangle$. A hypertree decomposition of G is a tree structure T where each node n of T associated with a bag of variables $B_n \subseteq V$ and with a set of edges $G_n \subseteq E$, such that:

- T with B_n yields a tree decomposition of the primal graph of G.
- For each node n of T:
  (1) the vertices used in the edges G_n are a superset of B_n,
  (2) if a vertex v occurs in an edge of G_n and this vertex also occurs in B_m for some node m below n in T, then $v \in B_n$.

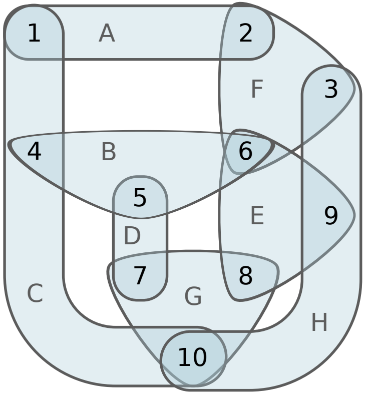
The width to T is the largest number of edges in a set G_n.
The hypertree width of G, hw(G), is the least width of its hypertree decompositions.

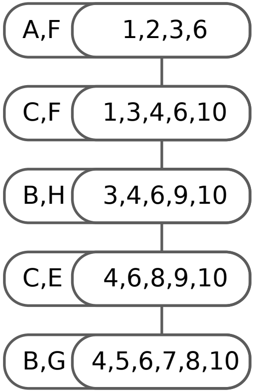
((2) is the "special condition": without it we get the generalised hypertree width)

Hypertree Width: Example

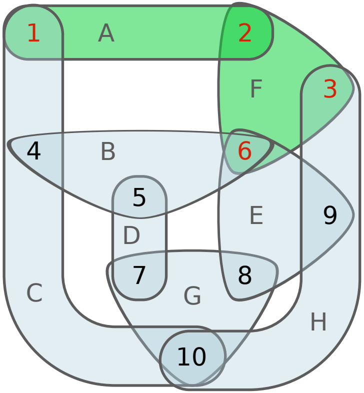


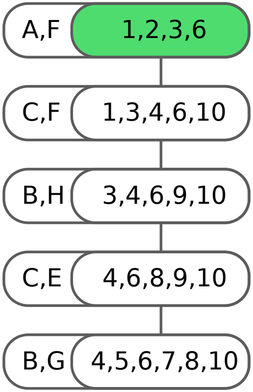
Hypertree Width: Example



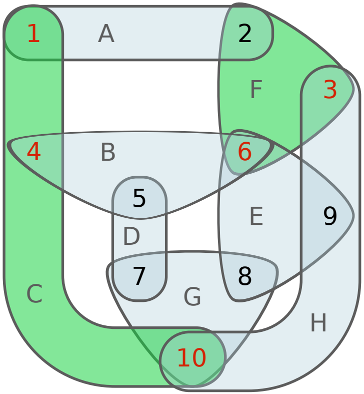


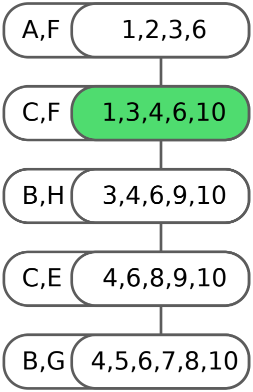
Hypertree Width: Example



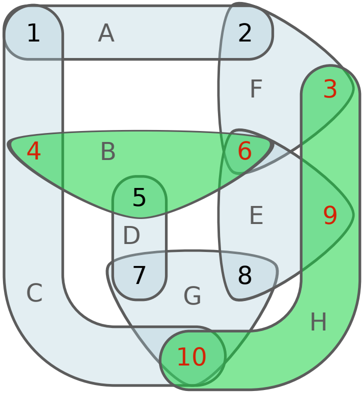


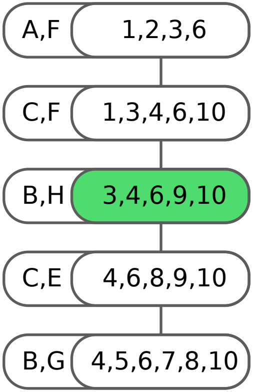
Hypertree Width: Example



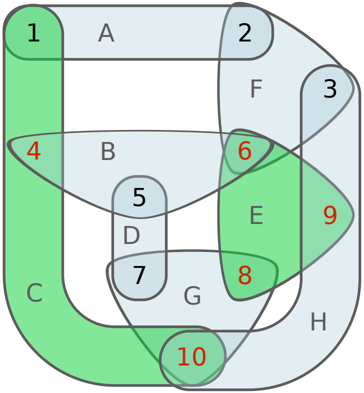


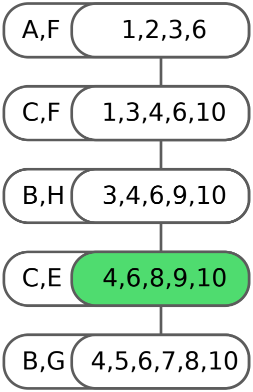
Hypertree Width: Example



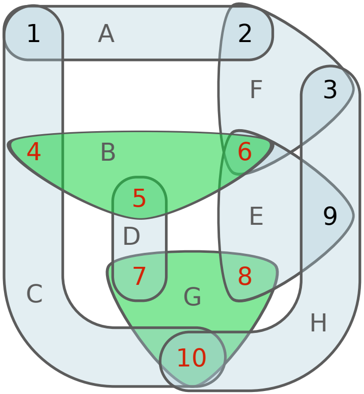


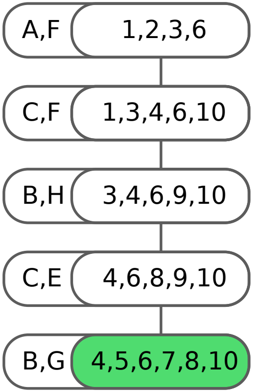
Hypertree Width: Example



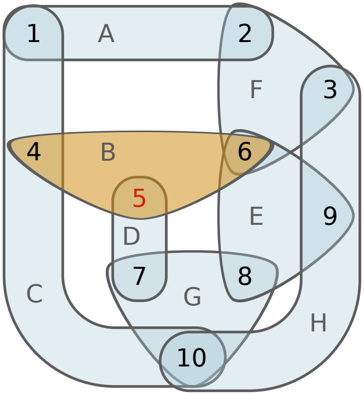


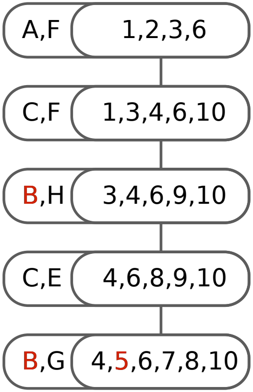
Hypertree Width: Example





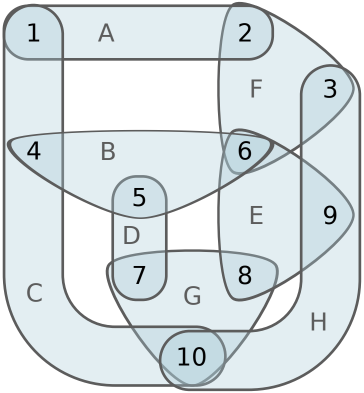
Hypertree Width: Example



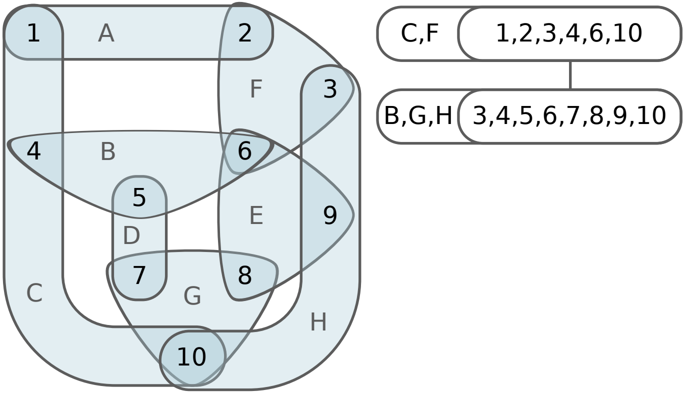


Special condition violated $\rightsquigarrow$ no hypertree decomposition
$\rightsquigarrow$ But generalised hypertree decomposition of width 2

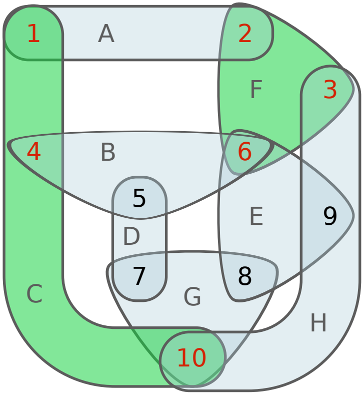
Hypertree Width: Example

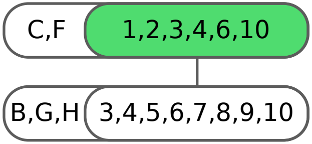


Hypertree Width: Example

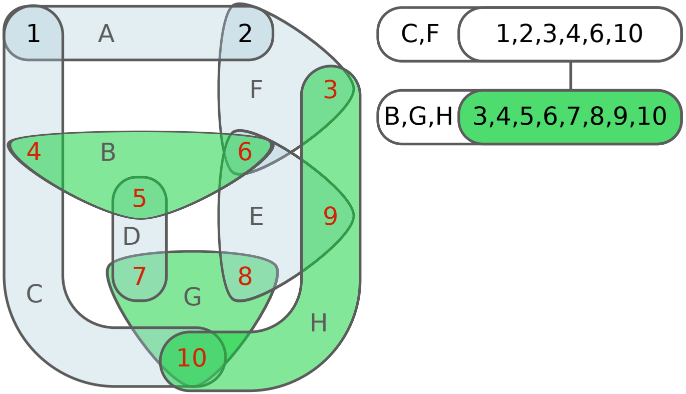


Hypertree Width: Example





Hypertree Width: Example



Special condition satisfied $\rightsquigarrow$ hypertree decomposition of width 3

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding “query width < k?” is NP-complete

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width < k?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width < k?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete
- Deciding "hypertree width < k?" is polynomial (LOGCFL)

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width < k?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete
- Deciding "hypertree width < k?" is polynomial (LOGCFL)
- Hypertree decompositions can be computed in polynomial time if k is fixed

Hypertree Width: Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width < k?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete
- Deciding "hypertree width < k?" is polynomial (LOGCFL)
- Hypertree decompositions can be computed in polynomial time if k is fixed

Theorem

For a BCQ of (generalised) hypertree width k, query answering can be decided in polynomial time, and is complete for LOGCFL.

. . . but the degree of the polynomial time bound is greater than k

Hypertree Width via Games

There is also a game characterisation of (generalised) hypertree width.

The Marshals-and-Robber Game

- The game is played on a hypergraph
- There are k marshals, each controlling one hyperedge, and one robber located at a vertex
- Otherwise similar to cops-and-robber game
- Special condition: Marshals must shrink the space that is left for the robber in every turn!

Hypertree width $\leq k$ if and only if k marshals have a winning strategy
$\leadsto$ hypergraph is acyclic iff 1 marshal has a winning strategy

Hypertree Width via Logic

There is also a logical characterisation of hypertree width.

Loosely k-Guarded Logic

- Fragment of FO with $\exists$ and $\wedge$
- Special form for all $\exists$ subexpressions:

$$\exists x_1, \ldots, x_n.(G_1 \wedge \ldots \wedge G_k \wedge \varphi)$$

  where G_i are atoms ("guards") and every variable that is free in φ occurs in one such atom G_i.

A query has hypertree width $\leq k$ if and only if it can be expressed as a loosely k-guarded formula
$\leadsto$ tree queries correspond to loosely 1-guarded formulae
("loosely 1-guarded" logic is better known as guarded logic and widely studied)

Summary and Outlook

Besides tree queries, there are other important classes of CQs that can be answered in polynomial time:

- Bounded treewidth queries
- Bounded hypertree width queries

General idea: decompose the query in a tree structure

Other possible characterisations via games and logic

Next topics:

- What else is there besides query answering? $\leadsto$ optimisation
- Measure expressivity rather than just complexity