CSE599d: Advanced Query Processing

Lecture 13: Tree Decomposition

Dan Suciu

University of Washington

Announcements

- HW3 due: Friday Feb. 27

Tree Decomposition

Tree Decomposition

$G = (V, E)$ undirected graph

Definition A tree decomposition is a pair (T, χ) where T = a tree and $\chi : \texttt{Nodes}(T) \rightarrow 2^V$ maps each node u in T to a subset of V (where $\chi(u)$ is called a bag), such that:

- Every edge (x, y) in G is contained in some bag $\chi(u)$.
- For every node x, the bags containing x are connected in T.

Tree Width

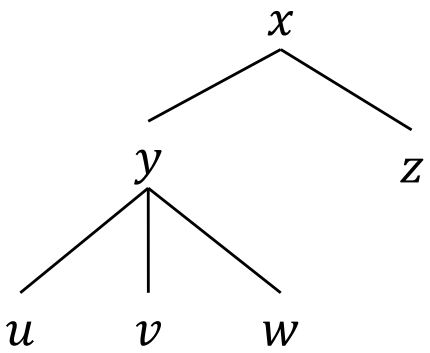
The width of $TD = (T, \chi)$ is the size of the largest bag, minus one:

$$\boxed{\text{TW}(TD) = \max_u |\chi(u)| - 1}$$

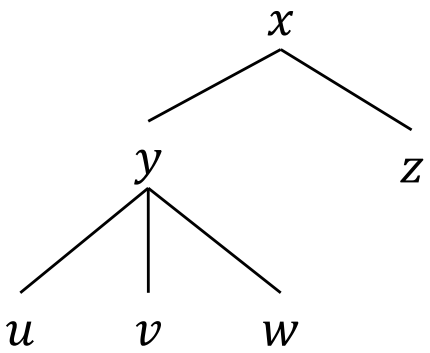
The tree-width of a graph is smallest width of any tree decomposition:

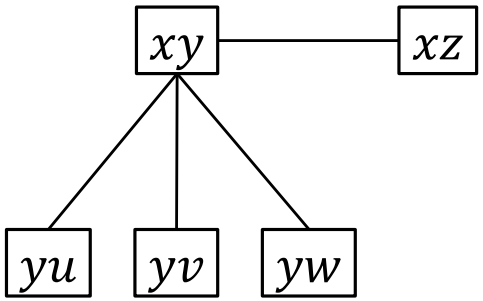
$$\boxed{\text{TW}(G) = \min_{TD} (\text{TW}(TD))}$$

Examples

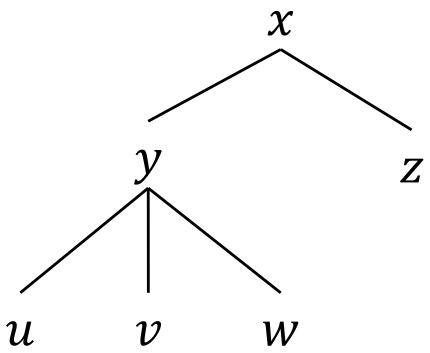


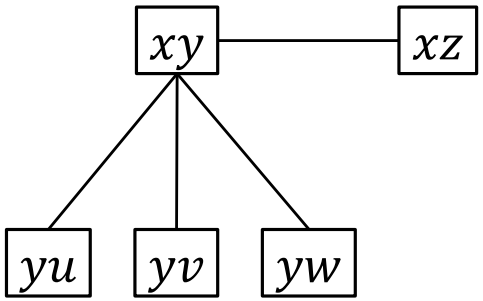
Examples





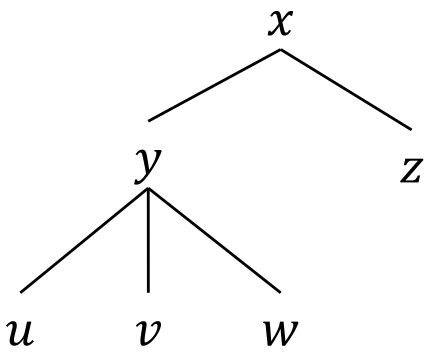
Examples

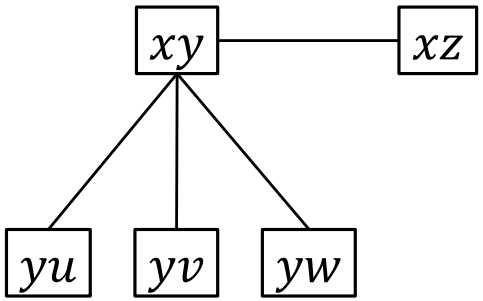




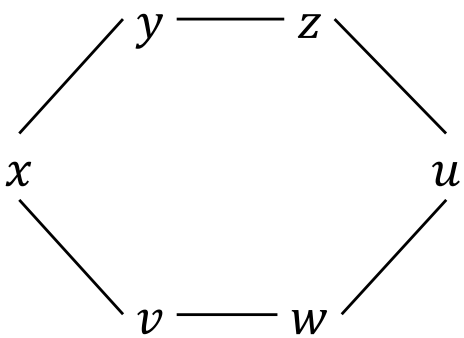
TW = 1

Examples

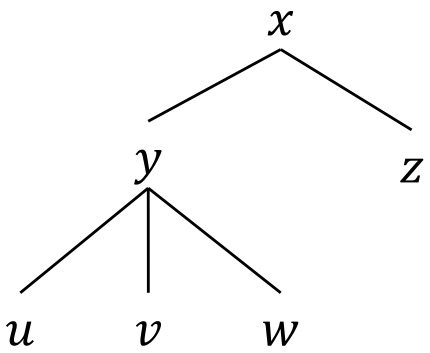


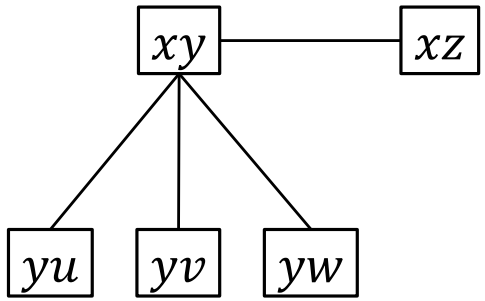


TW = 1

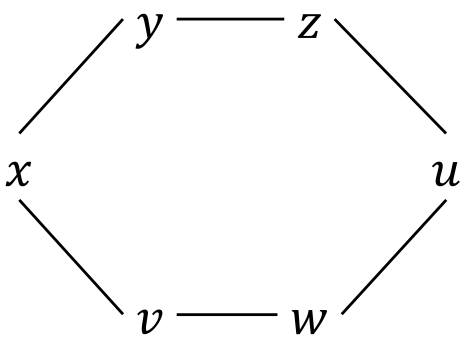


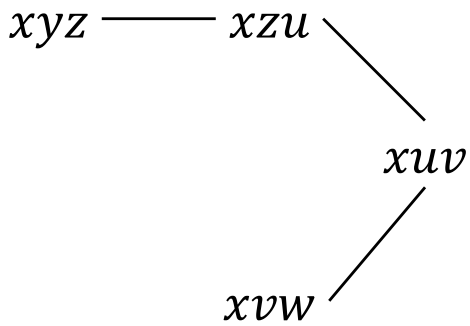
Examples





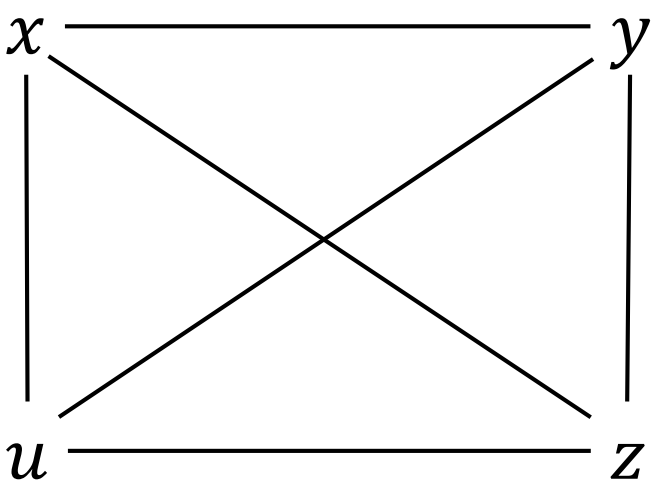
TW = 1



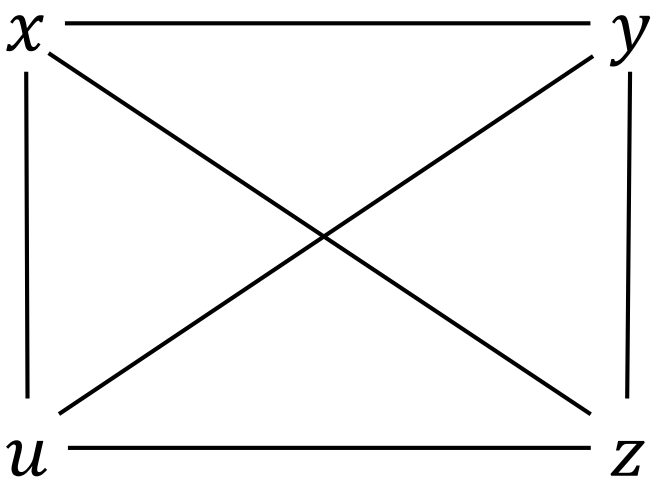


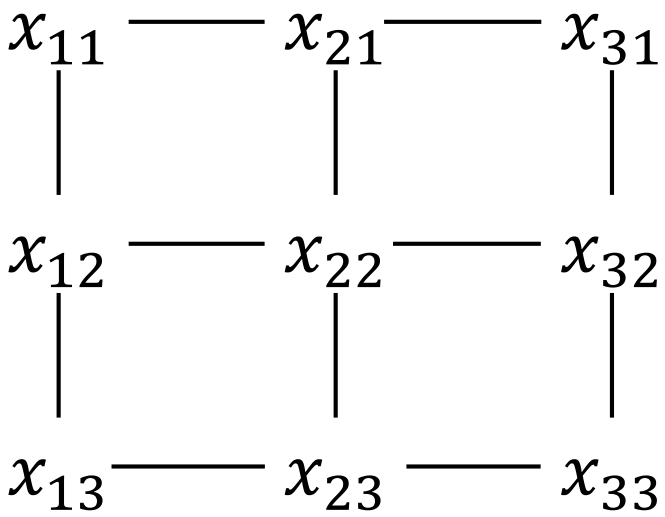
TW = 2

More Examples: What are the TW?

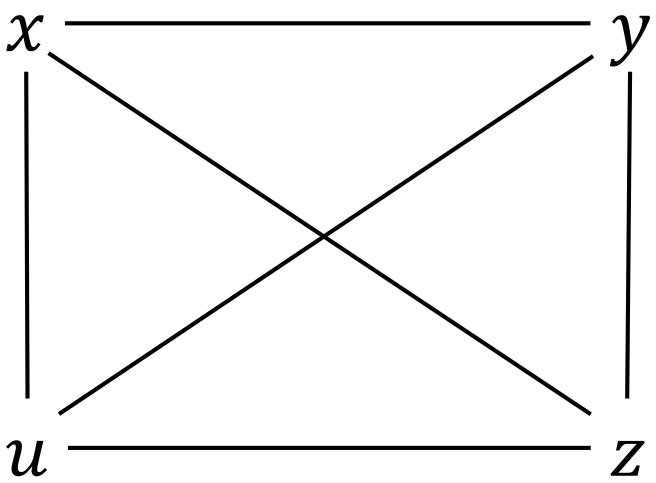


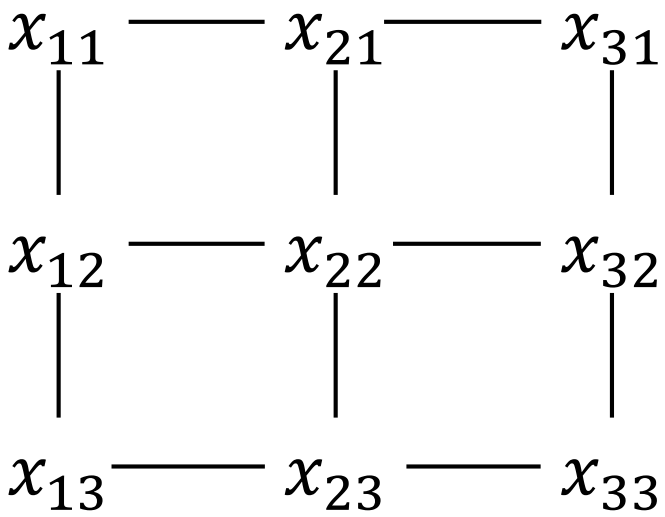
More Examples: What are the TW?

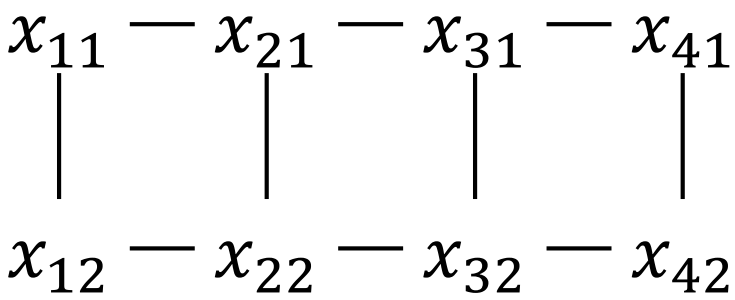




More Examples: What are the TW?







Discussion

- The "bags" are sets, not bags!
- $\text{TW}(G) = 1$ iff G is acyclic (i.e. a tree or a forest).
- If G is a clique, then $\text{TW}(G) = |\texttt{Nodes}(G)| - 1$.
- The problem: "given G, k, check if $\text{TW}(G) \leq k$" is NP-complete.
- For a fixed k, it is in linear time in $|G|$. Algorithm is impractical.
- Some hard problems become easy over graphs with a bounded tree-width.

  E.g. 3-colorability: NP-complete, linear time for graphs with bounded TW.

Hypertree Decomposition

Overview

When Q is acyclic, then we can compute Q in linear time using YA.

When Q is cyclic, then we compute a tree decomposition of Q, then run YA.

Recall that identify Q with a hypergraph $G = (V, E)$. what are V, E?

As usual, we use "query" and "hypergraph" interchangeably.

Hypertree Decomposition

A hypertree decomposition of $G = (V, E)$ is $TD = (T, \chi)$ where:

- Every hyperedge $e \in E$ is contained in some bag $\chi(u)$.
- For every node (variable) x, the bags containing x are connected.

$\text{TW}(TD)$ and $\text{TW}(G)$ are defined before (maximum bag size minus one).

Query Evaluation Algorithm

Query $Q = R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots$

- Compute a "good" hypertree decomposition $TD = (T, \chi)$ of Q.
- For all u, materialize $\chi(u)$ (using any algorithm), store the result in T_u.
- Run Yannakakis algorithm on the acyclic query consisting of all T_u.

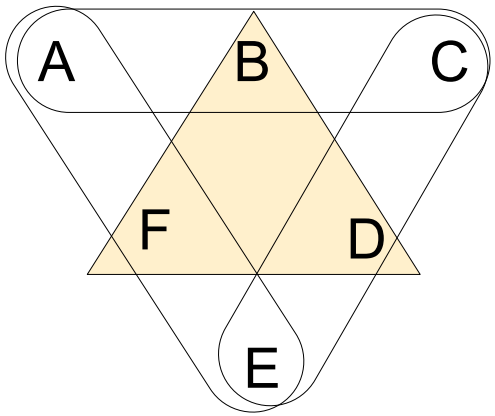
What is the complexity as a function of $N(= \texttt{IN})$ and $\texttt{OUT}$?

Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$

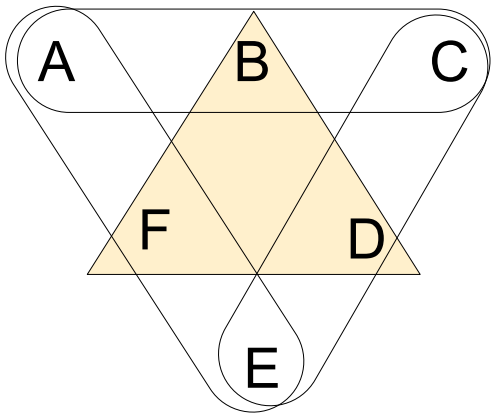
Example

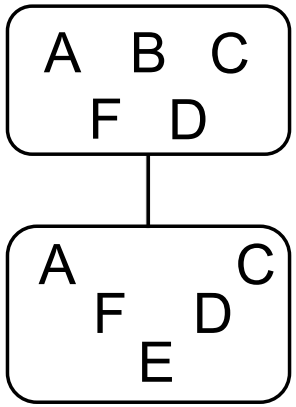
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$



Example

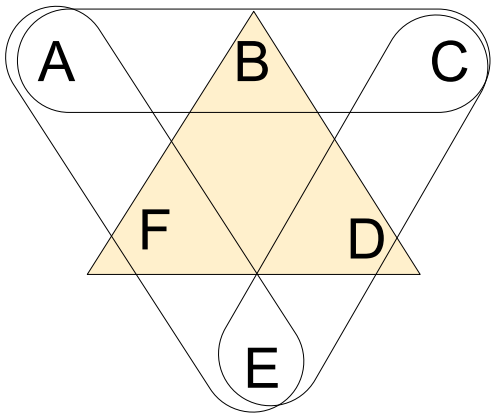
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$

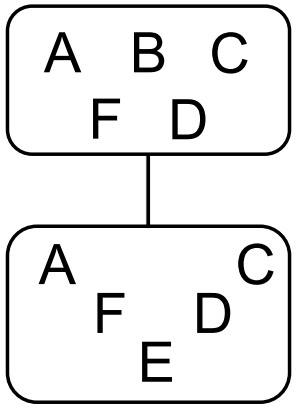


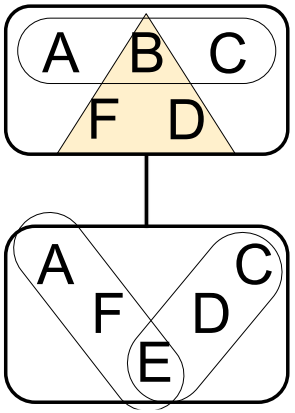


Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$

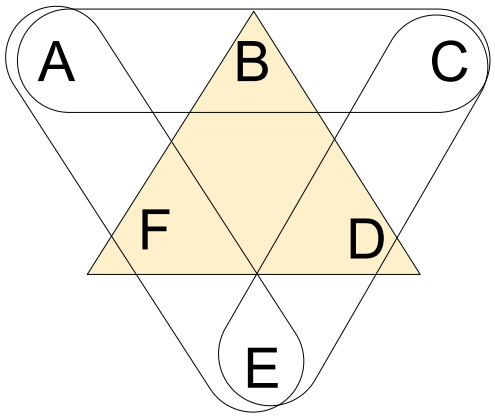


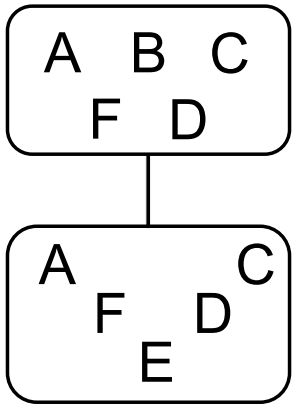


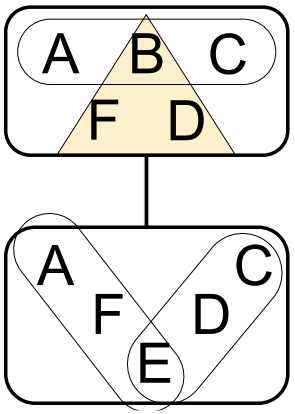


Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$





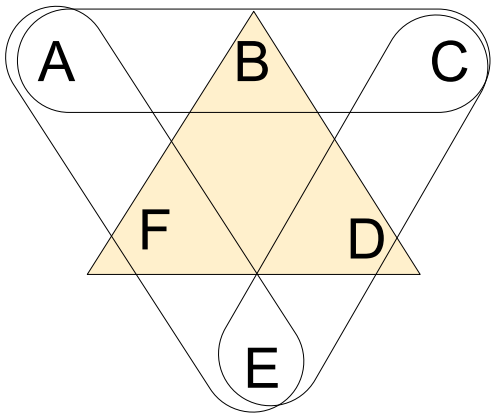


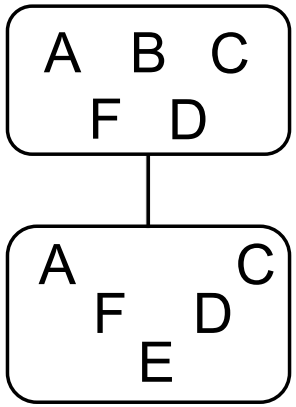
Algorithm:

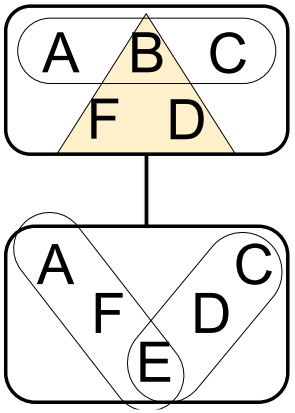
$$T_1(A,B,C,D,F) := R_1(A,B,C) \bowtie R_4(B,D,F)$$
$$T_2(A,C,D,E,F) := R_2(C,D,E) \bowtie R_3(E,F,A)$$
$$Q(A,B,C,D,E,F) := T_1 \bowtie T_2$$

Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$







Algorithm:

$$T_1(A,B,C,D,F) := R_1(A,B,C) \bowtie R_4(B,D,F)$$
$$T_2(A,C,D,E,F) := R_2(C,D,E) \bowtie R_3(E,F,A)$$
$$Q(A,B,C,D,E,F) := T_1 \bowtie T_2$$

Complexity: $O(N^2 + \texttt{OUT})$ Given by the hypertree width (next)

Edge Cover

$G = (V, E)$ hypergraph. An edge cover for G is a subset $E_0 \subseteq E$ s.t.

- Every vertex $x \in V$ is contained in some hyperedge $e \in E_0$.

Edge Cover

$G = (V, E)$ hypergraph. An edge cover for G is a subset $E_0 \subseteq E$ s.t.

- Every vertex $x \in V$ is contained in some hyperedge $e \in E_0$.

Subset $X \subseteq V$. An edge cover for X is a subset $E_0 \subseteq E$ s.t.

- every vertex $x \in X$ is contained in some hyperedge $e \in E_0$.

$\boxed{\rho(X) :=}$ size of the smallest edge cover for X

Hypertree Width

The (hypertree?)width of a tree decomposition $TD = (T, \chi)$ is largest edge cover of any of its bags:

$$\boxed{\text{HTW}(TD) = \max_u (\rho(\chi(u)))}$$

The hypertree width of a hypergraph G (or query) is:

$$\boxed{\text{HTW}(G) = \min_{TD} (\text{HTW}(TD))}$$

Complexity

Query $Q = R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots$

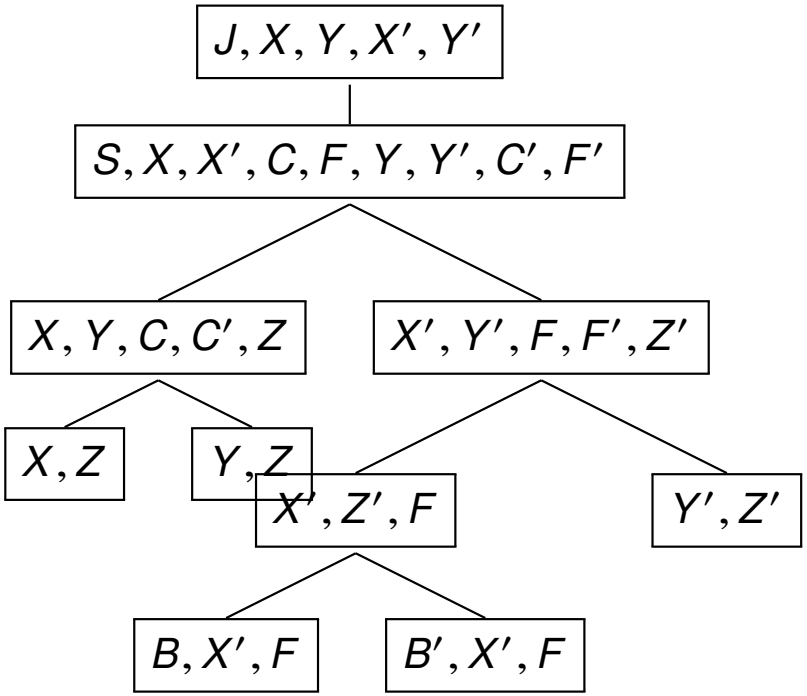
If $|R_1| = |R_2| = \cdots = N$, then Q can be computed in time $O(N^{\text{HTW}(Q)} + \text{OUT})$.

Example from [Pichler, 2023]

$$Q(\ldots) = a(S, X, X', C, F) \wedge b(S, Y, Y', C', F') \wedge c(C, C', Z) \wedge d(X, Z) \wedge e(Y, Z)$$
$$f(F, F', Z') \wedge g(X', Z') \wedge h(Y', Z') \wedge j(J, X, Y, X', Y') \wedge p(B, X', F) \wedge q(B', X', F)$$

Example from [Pichler, 2023]

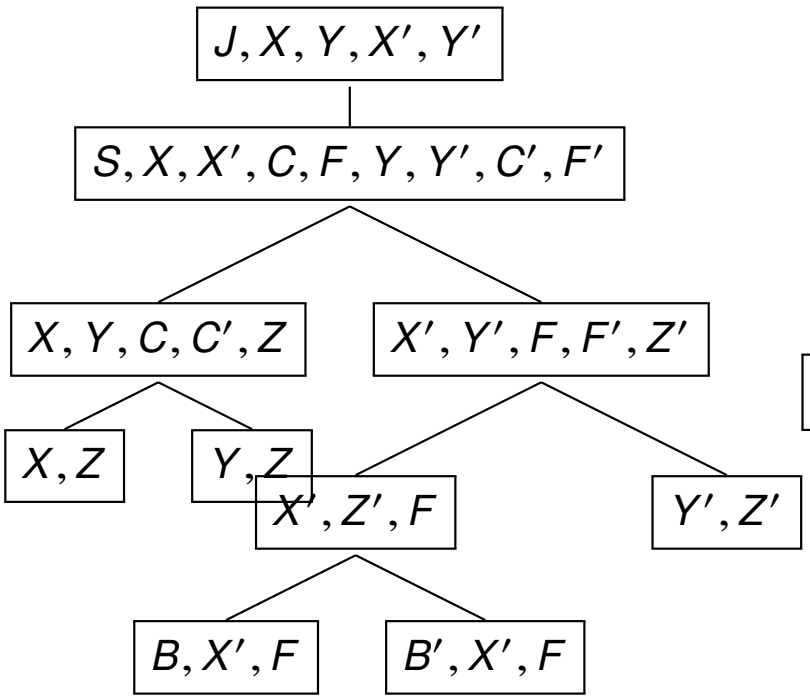
$$Q(\ldots) = a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z)$$
$$f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



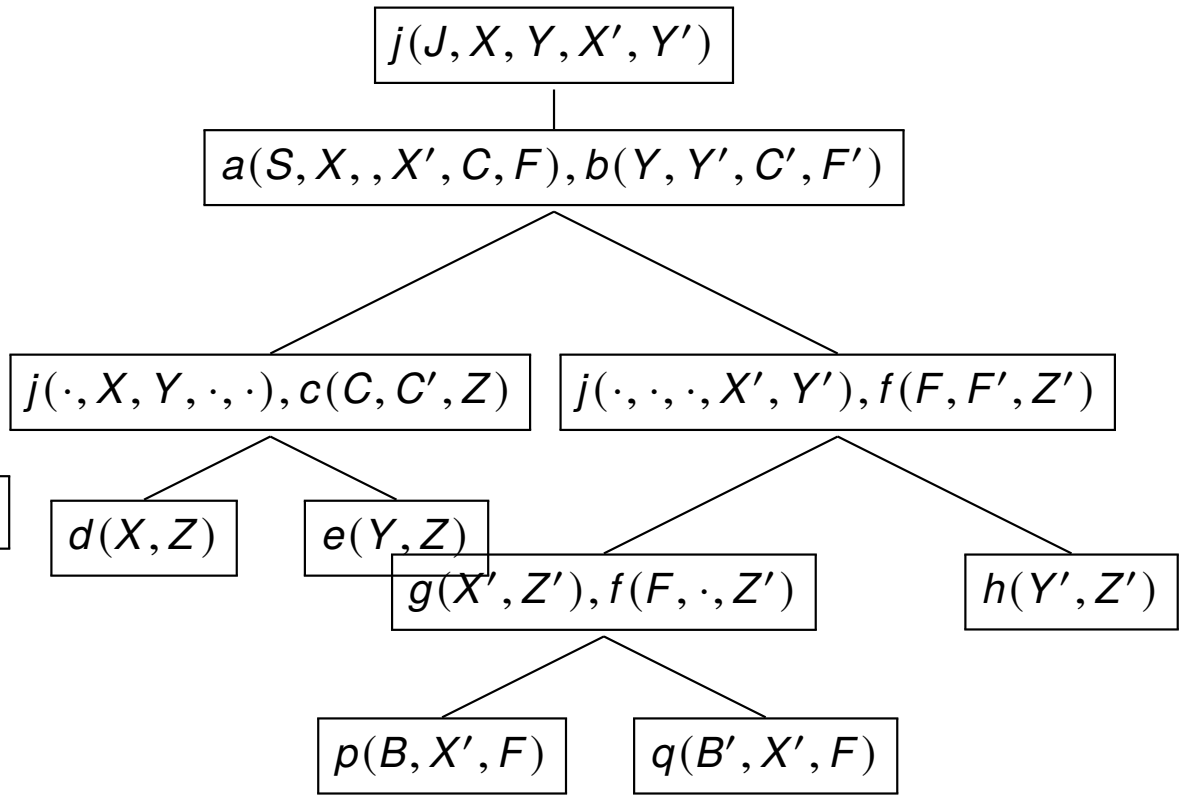
$\text{TW} = 8$

Example from [Pichler, 2023]

$$Q(\ldots) = a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z)$$
$$f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



TW = 8



HTW = 2

Discussion

- Query evaluation in time $O(N^{\text{HTW}} + \text{OUT})$
- "Given Q, check if $\text{HTW}(Q) \leq 3$" is NP-hard [Gottlob et al., 2021]
- Terminology confusion: the term generalized hypertree decomposition is sometimes used in the literature; will explain reason shortly.

Variable Elimination

Overview

Many algorithms in the literature are based on "variable elimination".

This turns out to be equivalent to a tree decomposition.

We will discuss their tight connection next, using [Khamis et al., 2015]

Notation: ∂x = set of hyperedges that contain x: $\boxed{\partial x := \{e \mid x \in e\}}$

Variable Elimination

$G = (V, E)$ hypergraph. A variable elimination order is a total order on V:

$$\sigma = (x_1, \ldots, x_n)$$

Variable Elimination

$G = (V, E)$ hypergraph. A variable elimination order is a total order on V:

$$\sigma = (x_1, \ldots, x_n)$$

σ defines the following sequence of hypergraphs

$$\boxed{G_0 (:= G), G_1, \ldots, G_n}$$

where G_i is obtained from G_{i-1} as follows:

- Remove the variable x_i.
- Remove hyperedges ∂x_i; add hyperedge $U_i - \{x_i\}$, where $\boxed{U_i := \bigcup \partial x_i}$

Variable Elimination

$G = (V, E)$ hypergraph. A variable elimination order is a total order on V:

$$\sigma = (x_1, \ldots, x_n)$$

σ defines the following sequence of hypergraphs

$$\boxed{G_0 (:= G), G_1, \ldots, G_n}$$

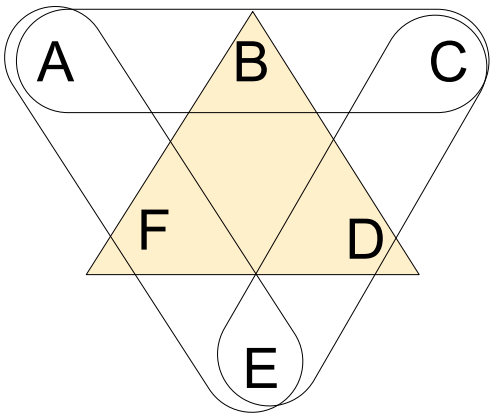
where G_i is obtained from G_{i-1} as follows:

- Remove the variable x_i.
- Remove hyperedges ∂x_i; add hyperedge $U_i - \{x_i\}$, where $\boxed{U_i := \bigcup \partial x_i}$

σ defines the following bags: $\boxed{\mathtt{bags}(\sigma) = \{U_1, \ldots, U_n\}}$

Example

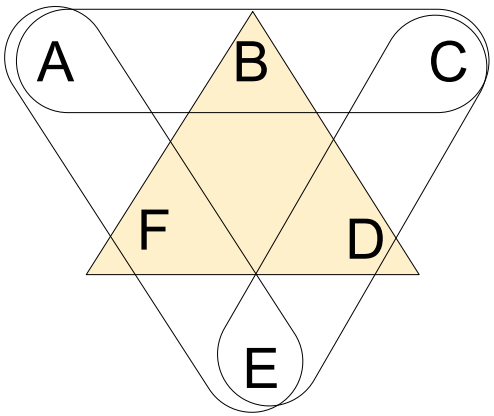
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$



Example

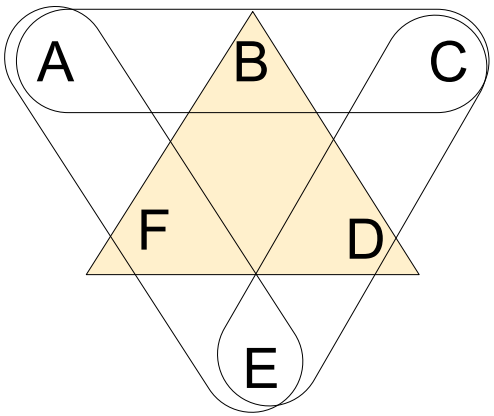
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$

$$G_0 = (\{A,B,C,D,E,F\}, \{ABC, CDE, EFA, BDF\}),$$



Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$

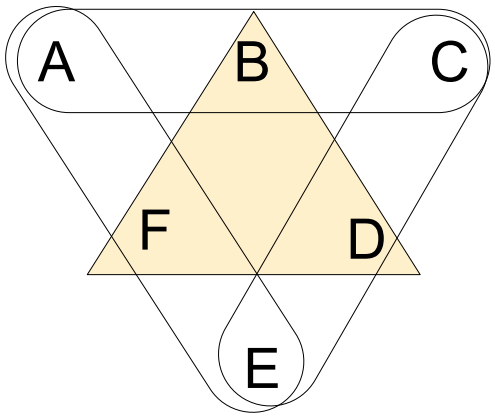


$$G_0 = (\{A,B,C,D,E,F\}, \{ABC, CDE, EFA, BDF\}),$$

$$U_1 = ABCEF, G_1 = (\{B,C,D,E,F\}, \{CDE, BDF, BCEF\}),$$

Example

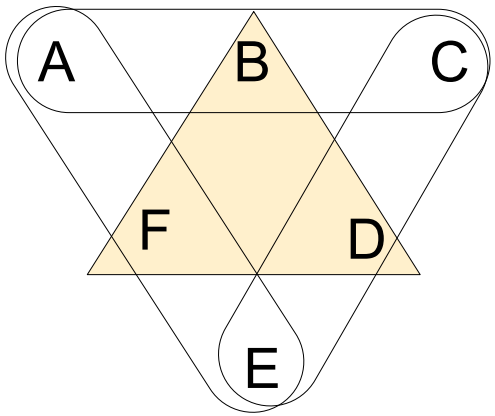
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$



$$G_0 = (\{A,B,C,D,E,F\}, \{ABC, CDE, EFA, BDF\}),$$
$$U_1 = ABCEF, G_1 = (\{B,C,D,E,F\}, \{CDE, BDF, BCEF\}),$$
$$U_2 = BCDEF, G_2 = (\{C,D,E,F\}, \{CDE, CDEF\})$$

Example

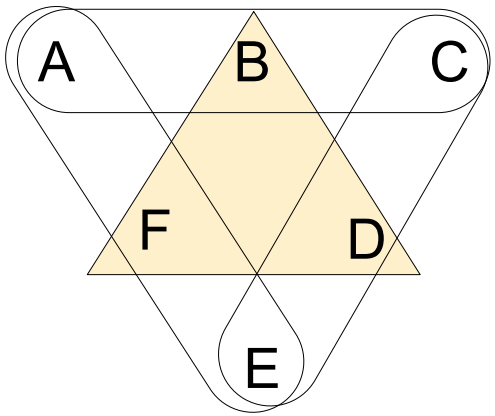
$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$



$$G_0 = (\{A,B,C,D,E,F\}, \{ABC, CDE, EFA, BDF\}),$$
$$U_1 = ABCEF, G_1 = (\{B,C,D,E,F\}, \{CDE, BDF, BCEF\}),$$
$$U_2 = BCDEF, G_2 = (\{C,D,E,F\}, \{CDE, CDEF\})$$
$$U_3 = CDEF, G_3 = (\{D,E,F\}, \{DE, DEF\}),$$

Example

$$Q(A,B,C,D,E,F) = R_1(A,B,C) \wedge R_2(C,D,E) \wedge R_3(E,F,A) \wedge R_4(B,D,F)$$
$$\sigma = (A,B,C,D,E,F)$$



$G_0 = (\{A,B,C,D,E,F\}, \{ABC, CDE, EFA, BDF\})$,

$U_1 = ABCEF$, $G_1 = (\{B,C,D,E,F\}, \{CDE, BDF, BCEF\})$,

$U_2 = BCDEF$, $G_2 = (\{C,D,E,F\}, \{CDE, CDEF\})$

$U_3 = CDEF$, $G_3 = (\{D,E,F\}, \{DE, DEF\})$,

$U_4 = DEF$, $G_4 = (\{E,F\}, \{EF\})$

$U_5 = EF$, $G_5 = (\{F\}, \{F\})$

$U_6 = F$, $G_6 = (\emptyset, \emptyset)$

From Variable Elimination to Tree Decomposition

Theorem For any variable elimination σ with bags $U_1, U_2, \ldots, U_n$ there exists a tree decomposition TD with the same bags.

From Variable Elimination to Tree Decomposition

Theorem For any variable elimination σ with bags $U_1, U_2, \ldots, U_n$ there exists a tree decomposition TD with the same bags.

Proof by induction on n (the number of variables). $n = 1$ is trivial.

From Variable Elimination to Tree Decomposition

Theorem For any variable elimination σ with bags $U_1, U_2, \ldots, U_n$ there exists a tree decomposition TD with the same bags.

Proof by induction on n (the number of variables). $n = 1$ is trivial.

Assume $n > 1$ and let $G_1 = (V_1, E_1)$ be the graph obtained after eliminating x_1.

From Variable Elimination to Tree Decomposition

Theorem For any variable elimination σ with bags $U_1, U_2, \ldots, U_n$ there exists a tree decomposition TD with the same bags.

Proof by induction on n (the number of variables). $n = 1$ is trivial.

Assume $n > 1$ and let $G_1 = (V_1, E_1)$ be the graph obtained after eliminating x_1.

By induction, G_1 has a tree decomposition TD_1 who bags are $U_2, U_3, \ldots, U_n$

One of these bags u_1 contains $U_1 - \{x_1\}$, because this is a hyperedge in G_1.

From Variable Elimination to Tree Decomposition

Theorem For any variable elimination σ with bags $U_1, U_2, \ldots, U_n$ there exists a tree decomposition TD with the same bags.

Proof by induction on n (the number of variables). $n = 1$ is trivial.

Assume $n > 1$ and let $G_1 = (V_1, E_1)$ be the graph obtained after eliminating x_1.

By induction, G_1 has a tree decomposition TD_1 who bags are $U_2, U_3, \ldots, U_n$

One of these bags u_1 contains $U_1 - \{x_1\}$, because this is a hyperedge in G_1.

Extend TD_1 by adding new bag $\chi(u) := U_1$, connect it with an edge to u_1.

From Tree Decomposition to Variable Elimination (1/2)

The correspondence is no exactly 1-1.

Recall the example $G = (\{A, B, C, D, E, F\}, \{ABC, CDE, EFG, BDF\})$:

- $\texttt{bags}(TD) = \{ABCEF, BCDEF\}$
- $\texttt{bags}(\sigma) = \{ABCEF, BCDEF, CDEF, DEF, EF, F\}$

From Tree Decomposition to Variable Elimination (1/2)

The correspondence is no exactly 1-1.

Recall the example $G = (\{A, B, C, D, E, F\}, \{ABC, CDE, EFG, BDF\})$:

- $\texttt{bags}(TD) = \{ABCEF, BCDEF\}$
- $\texttt{bags}(\sigma) = \{ABCEF, BCDEF, CDEF, DEF, EF, F\}$

Given two sets A, B of variables, write $\boxed{A \subseteq^\sharp B}$ if $\boxed{\forall a \in A, \exists b \in B \text{ s.t. } a \subseteq b}$

E.g. $\{ABCEF, BCDEF, CDEF, DEF, EF, F\} \subseteq^\sharp \{ABCEF, BCDEF\}$

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\mathtt{bags}(\sigma) \subseteq^\sharp \mathtt{bags}(TD)$.

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD)$.

Proof Let u = a leaf node, $u_1 = \text{parent}(u)$, $TD_1 = TD - \{u\}$.

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD)$.

Proof Let u = a leaf node, $u_1 = \text{parent}(u)$, $TD_1 = TD - \{u\}$.

- Ear: If $\chi(u) = \chi(u_1)$ then TD_1 is also a t.d. of G. By induction:

$$\exists \sigma : \quad \texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD_1) \subseteq^\sharp \texttt{bags}(TD).$$

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD)$.

Proof Let u = a leaf node, $u_1 = \text{parent}(u)$, $TD_1 = TD - \{u\}$.

- Ear: If $\chi(u) = \chi(u_1)$ then TD_1 is also a t.d. of G. By induction:

$$\exists \sigma : \quad \texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD_1) \subseteq^\sharp \texttt{bags}(TD).$$

- Isolated var: Let x_1 occur only in $\chi(u)$.
  $U_1 := \bigcup \partial x_1$, then: $\boxed{U_1 \subseteq \chi(u)}$

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD)$.

Proof Let u = a leaf node, $u_1 = \text{parent}(u)$, $TD_1 = TD - \{u\}$.

- Ear: If $\chi(u) = \chi(u_1)$ then TD_1 is also a t.d. of G. By induction:

$$\exists \sigma: \quad \texttt{bags}(\sigma) \subseteq^\sharp \texttt{bags}(TD_1) \subseteq^\sharp \texttt{bags}(TD).$$

- Isolated var: Let x_1 occur only in $\chi(u)$.

$$U_1 := \bigcup \partial x_1, \text{ then: } \boxed{U_1 \subseteq \chi(u)}$$

Eliminate x_1 to obtain G_1: it has tree decomposition TD_1. By induction

$$\exists \sigma_1: \quad \boxed{\texttt{bags}(\sigma_1) \subseteq^\sharp \texttt{bags}(TD_1)}$$

From Tree Decomposition to Variable Elimination (2/2)

Theorem For any $TD = (T, \chi)$ there exists σ s.t. $\mathtt{bags}(\sigma) \subseteq^\sharp \mathtt{bags}(TD)$.

Proof Let u = a leaf node, $u_1 = \text{parent}(u)$, $TD_1 = TD - \{u\}$.

- Ear: If $\chi(u) = \chi(u_1)$ then TD_1 is also a t.d. of G. By induction:

$$\exists \sigma: \quad \mathtt{bags}(\sigma) \subseteq^\sharp \mathtt{bags}(TD_1) \subseteq^\sharp \mathtt{bags}(TD).$$

- Isolated var: Let x_1 occur only in $\chi(u)$.

$$U_1 := \bigcup \partial x_1, \text{ then: } \boxed{U_1 \subseteq \chi(u)}$$

Eliminate x_1 to obtain G_1: it has tree decomposition TD_1. By induction

$$\exists \sigma_1: \quad \boxed{\mathtt{bags}(\sigma_1) \subseteq^\sharp \mathtt{bags}(TD_1)}$$

Add x_1 to the front of σ_1 to obtain $\sigma := x_1 : \sigma_1$. Then:

$$\boxed{\mathtt{bags}(\sigma) = \{U_1\} \cup \mathtt{bags}(\sigma_1) \subseteq^\sharp \{\chi(u)\} \cup \mathtt{bags}(TD_1) = \mathtt{bags}(TD)}$$

Hypertree Decomposition v.s. Generalized Hypertree Decomposition

Overview

Even checking if HTW ≤ 3 is NP hard.

Hypertree decomposition was originally introduced by Gottlob's group in the late 90s

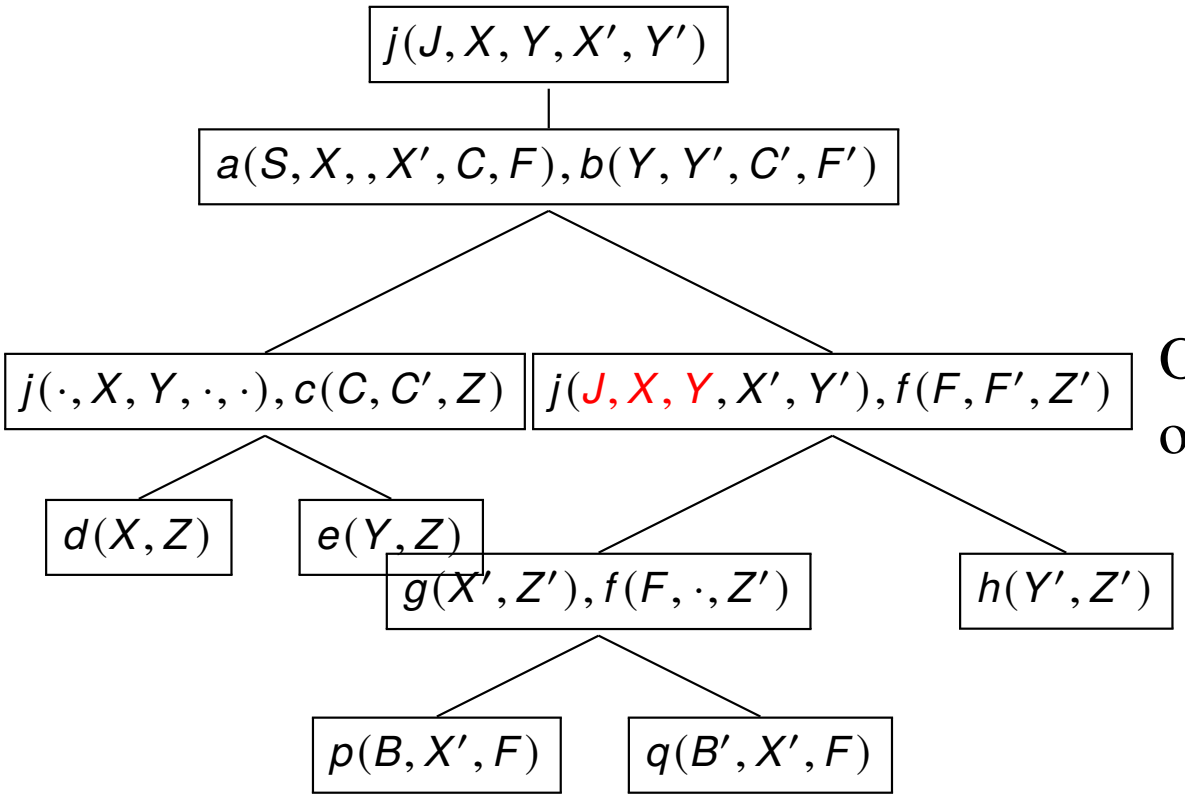
They goal was to also ensure a PTIME algorithm, when HTW is bounded.

They added a special condition to ensure that. Terminology:

- Hypertree decomposition = with special condition
- Generalized hypertree decomposition = without special condition

The Special Condition

If a variable "disappears" in a bag, then it cannot reoccur below.



OK because J, X, Y do not occur below

Effect of the Special Condition

Terminology used in some papers:

$\textsc{htw}(Q)$ = hypertree width with the special condition.
$\textsc{ghtw}(Q)$ = hypertree width without the special condition.

$$\boxed{\textsc{ghtw}(Q) \leq \textsc{htw}(Q) \leq 3 \cdot \textsc{htw}(Q) + 1}$$

Moreover, $\textsc{htw}(Q)$ can be computed in PTIME.

However, this is history. Today: not interested in HTW or GHTW, but in the fractional hypertreewidth FTW (next lecture). Special condition no longer helps.

Bounding the Number of Query Variables

Brief Review of Conjunctive Queries CQ

CQ = the set of Conjunctive Queries:

- Atomic formulas: $R(x, y, z)$
- Conjunctions: $Q_1 \wedge Q_2$
- Existential quantifiers: $\exists x Q$.

E.g. is there a path of length 2?

$$\exists x \exists y (R(x, y) \wedge \exists z (R(y, z))) \equiv \exists x \exists y \exists z (R(x, y) \wedge R(y, z)).$$

CQ^k

CQ^k = conjunctive query expressions using only k variables $x_1, \ldots, x_k$.

E.g. "is there a path of length 4?" in CQ^2:

$$\exists x \exists y (R(x,y) \wedge \exists x (R(y,x) \wedge \exists y (R(x,y) \wedge \exists x R(y,x))))$$

Notice that we reuse variables!

CQ v.s. CQ^k

The standard syntax in CQ uses all quantifiers upfront:

$$\exists x_1 \cdots \exists x_n (R_1(\boldsymbol{X}_1) \wedge R_2(\boldsymbol{X}_2) \wedge \cdots)$$

Every expression in CQ^k is equivalent to standard CQ, by using more vars:

$$\exists x \exists y (R(x,y) \wedge \exists x (R(y,x) \wedge \exists y (R(x,y) \wedge \exists x R(y,x))))$$
$$\equiv \exists x \exists y \exists u \exists v \exists w (R(x,y) \wedge R(y,z) \wedge R(z,u) \wedge R(u,v) \wedge R(v,w))$$

We say that Q is in CQ^k if it is equivalent to an expression in CQ^k.

A free connex TD is one where the free variables form a bag:
$\exists u, \chi(u) = FV(Q)$.

From CQ^k to TW

Theorem If $Q \in \mathsf{CQ}^{k+1}$, then $\mathrm{TW}(Q) \leq k$

Proof Induction on the expression $Q \in \mathsf{CQ}^{k+1}$:

- Atomic formula: $R(x_1, x_2, \ldots, x_\ell)$ with $\ell \leq k+1$.
  $TD :=$ single bag $\{x_1, \ldots, x_k\}$.

- Conjunction: $Q = Q_1 \wedge Q_2$. By induction: $\exists TD_1, TD_2$, with TW $\leq k$.
  Create a new bag with $\chi(u) := FV(Q) = FV(Q_1) \cup FV(Q_2)$.
  Connect u to the bags containing $FV(Q_1)$ and $FV(Q_2)$ respectively.

- Existential quantifier: $Q = \exists x Q_1$. By induction $\exists TD_1$, with TW $\leq k$.
  Create a new bag $\chi(u) := FV(Q) = FV(Q_1) - \{x\}$.
  Connect u to the bag of TD_1 that contains $FV(Q_1)$.

From TW to CQ^k

Theorem If $\text{TW}(Q) \leq k$ then $Q \in \mathsf{CQ}^{k+1}$.

Proof Let TD be s.t. $\text{TW}(TD) \leq k$.

- TD has single bag u; then $|\text{Vars}(Q)| = |\chi(u)| \leq k+1$ and $Q \in \mathsf{CQ}^{k+1}$.
- Otherwise, let v_0 be s.t. $\chi(v_0) = FV(Q)$, and v_1 be a neighbor.

  Edge (v_0, v_1) splits TD into TD_0, TD_1.

  Let Q_0 be the query defined by TD_0, with $FV(Q_0) := \chi(v_0) = FV(Q)$
  Let Q_1 be the query defined by TD_1, with $FV(Q_1) := \chi(v_1)$.

  By induction, $Q_0, Q_1 \in \mathsf{CQ}^{k+1}$. Then $Q = Q_0 \wedge \exists \boldsymbol{Z} Q_1 \in \mathsf{CQ}^{k+1}$
  where $\boldsymbol{Z} = \chi(v_1) - FV(Q)$.

Summary

- (Hyper)Tree decomposition: fundamental technique that allows a graph or query to degenerate gradually from an acyclic graph or query.
- Graph theory is concerned with TW; database theory is concerned with HTW. Actually, it is concerned with FTW, to be discussed on Wednesday.
- No good algorithms exists for finding an efficient tree decomposition!
- Variable elimination order is fundamentally the same as tree decomposition.
- Having a bound on the number of variables is also the same as tree decomposition.


Gottlob, G., Lanzinger, M., Pichler, R., and Razgon, I. (2021).
Complexity analysis of generalized and fractional hypertree decompositions.
J. ACM, 68(5):38:1–38:50.


Khamis, M. A., Ngo, H. Q., and Rudra, A. (2015).
FAQ: questions asked frequently.
CoRR, abs/1504.04044.


Pichler, R. (2023).
Acyclity and notions of ”width” of hypergraphs.
https://berkeley-cs294-248.github.io/lectures/2023-09-18-ReinhardPichler.pdf.