Conjunctive Queries
Complexity & Decomposition Techniques

G. Gottlob
Technical University of Vienna, Austria

This talk reports about joint work with
I. Adler, M. Grohe, N. Leone and F. Scarcello

For papers and further material
see:
http://ulisse.deis.unical.it/~frank/Hypertrees/
Three Problems:

CSP: Constraint satisfaction problem

BCQ: Boolean conjunctive query evaluation

HOM: The homomorphism problem

Important problems in different areas.
All these problems are hypergraph based.

But actually: CSP = BCQ = HOM
CSP

Set of variables \( V = \{X_1, \ldots, X_n\} \), domain \( D \),
Set of constraints \( \{C_1, \ldots, C_m\} \)
where: \( C_i = \langle S_i, R_i \rangle \)

Solution to this CSP: A substitution \( h: V \rightarrow D \) such that \( \forall i: h(S_i \in R_i) \)

Associated hypergraph: \( \{\text{var}(S_i) \mid 1 \leq i \leq m\} \)
Example of CSP: Crossword Puzzle

1h: PARIS PANDA LAURA ANITA

1v: LIMBO LINGO PETRA PAMPA PETER

and so on
Conjunctive Database Queries are CSPs!

DATABASE:

<table>
<thead>
<tr>
<th>Enrolled</th>
<th>Teaches</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Algebra</td>
<td>McLane</td>
</tr>
<tr>
<td>Robert</td>
<td>Logic</td>
<td>Kolaitis</td>
</tr>
<tr>
<td>Mary</td>
<td>DB</td>
<td>Lausen</td>
</tr>
<tr>
<td>Lisa</td>
<td>DB</td>
<td>Rahm</td>
</tr>
<tr>
<td>..........</td>
<td>..........</td>
<td>.......</td>
</tr>
</tbody>
</table>

QUERY: Is there any teacher having a child enrolled in her course?

\[ ans \leftarrow Enrolled(S,C,R) \land Teaches(P,C,A) \land Parent(P,S) \]
Queries and Hypergraphs

\[ \text{ans} \leftarrow \text{Enrolled}(S,C',R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]
Queries, CSPs, and Hypergraphs

Is there a teacher whose child attends some course?

\[ Enrolled(S,C,R) \land Teaches(P,C,A) \land Parent(P,S) \]
The Homomorphism Problem

Given two relational structures

\[ A = (U, R_1, R_2, ..., R_k) \]
\[ B = (V, S_1, S_2, ..., S_k) \]

Decide whether there exists a homomorphism \( h \) from \( A \) to \( B \)

\[ h : U \longrightarrow V \]

such that \( \forall x, \forall i \)

\[ x \in R_i \Rightarrow h(x) \in S_i \]
HOM is NP-complete

(well-known)

Membership: Obvious, guess $h$.

Hardness: Transformation from 3COL.

Graph 3-colourable iff $\text{HOM}(A,B)$ yes-instance.
HOM is NP-complete
(well-known, independently proved in various contexts)

Membership: Obvious, guess \( h \).

Hardness: Transformation from 3COL.

Graph 3-colourable iff \( \text{HOM}(A,B) \) yes-instance.
CSP = BCQ = HOM
Complexity of CSPs

- NP-complete in the general case
  (Bibel, Chandra and Merlin ’77, etc.)
- NP-hard even for fixed constraint relations

- Polynomial in case of acyclic hypergraphs
  (Yannakakis ’81)
- LOGCFL-complete (in NC$_2$)
  (G.L.S. ’98)

Interest in larger tractable classes of CQ/CSP
Acyclic Hypergraphs

\[ \text{ans} \leftarrow \text{Enrolled}(S,C',R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]
\( n^2 \log n \)
d(Y, P)

r(Y, Z, U)

s(Z, U, W)

t(V, Z)

\begin{align*}
d &: 3 \quad 8 \quad 9 \\
   & 9 \quad 3 \quad 8 \\
   & 8 \quad 3 \quad 8 \\
   & 3 \quad 8 \quad 4 \\
   & 3 \quad 8 \quad 3 \\
   & 8 \quad 9 \quad 4 \\
   & 9 \quad 4 \quad 7 \\
\end{align*}
\[d(Y,P)\]

\[r(Y,Z,U)\]

\[s(Z,U,W)\]

\[t(V,Z)\]
$d(Y,P)$

$r(Y,Z,U)$

$s(Z,U,W)$

$t(V,Z)$
d(Y,P)

r(Y,Z,U)

s(Z,U,W)

t(V,Z)
d(Y, P)

r(Y, Z, U)

s(Z, U, W)

t(V, Z)

\[
\begin{array}{c}
3 \ 8 \\
3 \ 7 \\
5 \ 7 \\
6 \ 7 \\
\ldots
\end{array}
\]

\[
\begin{array}{c}
3 \ 8 \ 9 \\
9 \ 3 \ 8 \\
8 \ 3 \ 8 \\
3 \ 8 \ 4 \\
3 \ 8 \ 3 \\
8 \ 9 \ 4 \\
9 \ 4 \ 7
\end{array}
\]

\[
\begin{array}{c}
9 \ 8 \\
9 \ 3 \\
9 \ 5
\end{array}
\]
A solution: Y=3, P=7, Z=8, U=9, W=4, V=9
Computing the result

- The result size can be exponential (even in case of ACQs).

- Even when the result is of polynomial size, it is in general hard to compute.

- In case of acyclic queries, the result can be computed in time polynomial in the result size (i.e., in output-polynomial time).

- This will remain true for the subsequent generalizations of ACQs.

- The result of ACQs can be computed by adding a top-down phase to Yannakakis’ algorithm for ABCQs and by joinoing the partial results.
Theorem [GLS99]: Answering acyclic BCQs is LOGCFL-complete

LOGCFL: class of problems/languages that are logspace-reducible to a CFL

\[ AC_0 \subseteq NL \subseteq \text{LOGCFL} = \text{SAC}_1 \subseteq AC_1 \subseteq NC_2 \subseteq \cdots \subseteq NC = AC \subseteq P \subseteq NP \]

Characterization of LOGCFL [Ruzzo80]:

LOGCFL = Class of all problems solvable with a logspace ATM with polynomial tree-size
ABCQ is in LOGCFL
ABCQ is in LOGCFL
ABCQ is in LOGCFL
Is this query hard?

\[ \text{ans} \leftarrow a(S, X, X', C, F) \land b(S, Y, Y', C', F') \land c(C, C', Z) \land d(X, Z) \land e(Y, Z) \land f(F, F', Z') \land g(X', Z') \land h(Y', Z') \land j(J, X, Y, X', Y') \land p(B, X', F) \land q(B', X', F) \]

- Classical methods worst-case complexity: \( O(n^m) \)

- Despite its appearance, this query is nearly acyclic

It can be evaluated in \( O(m \cdot n^2 \cdot \log n) \)
ans ← \( a(S, X, X', C, F) \land b(S, Y, Y', C', F') \land c(C, C', Z) \land d(X, Z) \land \\
e(Y, Z) \land f(F, F', Z') \land g(X', Z') \land h(Y', Z') \land \\
j(J, X, Y, X', Y') \land p(B, X', F) \land q(B', X', F) \)
Nearly Acyclic Queries & CSPs

- Bounded Treewidth ($tw$)
  - a measure of the cyclicity of graphs
  - for queries: $tw(Q) = tw(G(Q))$

- For fixed $k$:
  - checking $tw(Q) \leq k$
  - Computing a tree decomposition

- Deciding CSP of treewidth $k$:
  \[
  \begin{align*}
  &O(n^k \log n) & \text{ (Chekuri & Rajaraman'97, Kolaitis & Vardi, 98)} \\
  &\text{LOGCFL-complete} & \text{ (G.L.S.'98)}
  \end{align*}
  \]
Primal graphs of CSPs/Queries

\[ \text{ans} \leftarrow \text{Enrolled}(S,C,R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]

Hypergraph \( H(Q) \)  
Primal graph \( G(Q) \)
Example: a cyclic graph
A tree decomposition of width 2

```
ah
  +---+---+
  |   |   |
  +---+---+
  |   |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
  +----+
  |   |
```
Connectedness condition for \( h \)
Game characterization of Treewidth

- A robber and $k$ cops play the game on a graph
- The cops have to capture the robber
- Each cop controls a vertex of the graph
- Each cop, at any time, can fly to any vertex of the graph
- The robber tries to elude her capture, by running arbitrarily fast on the vertices of the graph, but on those vertices controlled by cops
Playing the game
Playing the game
Playing the game
Playing the game
Logical characterization of Treewidth

Logic $L : FO$ based on $\exists, \land$ ($\neg, \lor, \forall$ disallowed)

$L = \exists FO_{\land, +}$ Basic Querying Logic

$TW[k] = L^{k+1}$ (Kolaitis & Vardi '98)

$L^{k+1} : L$ with at most $k + 1$ vars.
Logical characterization of Treewidth

A generalization:

$$NRS\text{-}DATALOG\text{-}TW[k] = FO^{k+1}$$

(Flum, Frick, and Grohe ‘01)
When is the evaluation of conjunctive queries tractable?

In case they are characterized by graphs e.g., through primal or Gaifman graphs:

\[ G \quad : \quad \text{class of graphs} \]

\[ Q(G) \quad : \quad \text{all queries characterized by graphs in } G \]

\[ Q(G) \text{ tractable iff } G \text{ has bounded treewidth} \]

unless \( P = W[1] \) or other collapses occur

(Grohe, Schwentick, and Segoufin, ’01)
Hypergraphs vs Graphs (1)

An acyclic hypergraph

Its cyclic primal graph
There are two cliques. We cannot know where they come from.
Drawbacks of treewidth

Acyclic queries may have unbounded TW!

Example:

$q \leftarrow p_1(X_1, X_2, \ldots, X_n) \land \ldots \land p_m(X_1, X_2, \ldots, X_n)$

is acyclic, obviously polynomial, but has treewidth $n-1$
Beyond treewidth

Bounded Degree of Cyclicity
(Gyssens & Paredaens ’84)

Bounded Query width
(Chekuri & Rajaraman ’97)

Group together query atoms (hyperedges) instead of variables
Query Decomposition

\[ q \leftarrow p_1(X_1, X_2, \ldots, X_n) \land \ldots \land p_m(X_1, X_2, \ldots, X_n) \]

Query width = 1

• Every atom appears in some node
• Connectedness conditions for variables and atoms
Decomposition of cyclic queries

\[ q \leftarrow s(Y,Z,U) \land g(X,Y) \land t(Z,X) \land s(Z,W,X) \land t(Y,Z) \]

Query width = 2

BCQ is polynomial for queries of bounded query width, \textbf{if} a query decomposition is given
Transform a query of bounded width into an acyclic query over a modified database

\[ q \leftarrow s(Y, Z, U) \land g(X, Y) \land t(Z, X) \land s(Z, W, X) \land t(Y, Z) \]
Open Problems by Chekuri & Rajaraman ‘97

Are the following problems solvable in polynomial time for fixed $k$?

- Decide whether $Q$ has query width at most $k$
- Compute a query decomposition of $Q$ of width $k$
A negative answer (G.L.S. ’99)

**Theorem:** Deciding whether a query has query width at most $k$ is NP-complete

**Proof:** Very involved reduction from EXACT COVERING BY 3-SETS
Important Observation

NP-hardness id due to an overly strong condition in the definition of query decomposition

Forbidden!
Important Observation

But the reuse of $p(X,Y,Z)$ is harmless here:

we could added an atom $p(X,Y,Z')$ without changing the query
Hypertree Decompositions

Query atoms can be used “partially” as long as the full atom appears somewhere else

More liberal than query decomposition
Grouping and Reusing Atoms

We group atoms

We use $p(X,Y,Z)$ partially

$p(X,Y,Z), q(U,V,Z)$

$a(X,U,W), b(Y,V,W)$

$p(X,Y,\_), c(T,W)$

$d(X,T)$

$c(Y,T)$
Reusing atoms

We use $p(X,Y,Z)$ partially

$p(X,Y,Z), q(U,V,Z)$

$a(X,U,W), b(Y,V,W)$

$p(X,Y,\_), c(T,W)$

$d(X,T)$

$c(Y,T)$
\[ \text{ans} \leftarrow a(S, X, X', C, F) \land b(S, Y, Y', C', F') \land c(C, C', Z) \land d(X, Z) \land e(Y, Z) \land f(F, F', Z') \land g(X', Z') \land h(Y', Z') \land j(J, X, Y, X', Y') \land p(B, X', F) \land q(B', X', F) \]
Connectedness Condition

\[ j(J, X, Y, X', Y') \]

\[ a(S, X, X', C, F), \ b(S, Y, Y', C', F') \]

\[ j(_, X, Y, _, _), \ c(C, C', Z) \]

\[ j(_, _, X', Y'), \ f(F, F', Z') \]

\[ d(X, Z) \]

\[ e(Y, Z) \]

\[ g(X', Z'), \ f(F, _, Z') \]

\[ h(Y', Z') \]

\[ p(B, X', F) \]

\[ q(B', X', F) \]
Special Condition

Variables omitted at some vertex $v$

Do not reappear in the subtrees rooted at $v$
Special Condition

\[ j(J, X, Y, X', Y') \]
\[ a(S, X, X', C, F), b(S, Y, Y', C', F') \]
\[ j(_, X, Y, _,_), c(C, C', Z) \]
\[ j(J, X, Y, X', Y'), f(F, F', Z') \]
\[ d(X, Z) \]
\[ e(Y, Z) \]
\[ g(X', Z'), f(F, _, Z') \]
\[ h(Y', Z') \]
\[ p(B, X', F) \]
\[ q(B', X', F) \]

Does not appear in the subtrees rooted at \( \nu \)

Variables omitted at some vertex \( \nu \)
Positive Results on Hypertree Decompositions

- For each query $Q$, $hw(Q) \leq qw(Q)$
- In some cases, $hw(Q) < qw(Q)$
- For fixed $k$, deciding whether $hw(Q) \leq k$ is in polynomial time (LOGCFL)
- Computing hypertree decompositions is feasible in polynomial time (for fixed $k$)
Evaluating queries having bounded hypertree width

$k$ fixed

Given:
- a database $db$
- a query $Q$ over $db$ such that $hw(Q) \leq k$
- a width $k$ hypertree decomposition of $Q$

- Deciding whether $Q(db)$ is not empty is in $O(n^{k+1} \log n)$ and complete for LOGCFL

- Computing $Q(db)$ is feasible in output-polynomial time
Comparison results

- Hypertree Decomposition
  - Hinge Decomposition + Tree Clustering
    - Hinge Decomposition
    - Tree Clustering $w^* \equiv$ treewidth
  - Cycle Hypercutset
    - Cycle Cutset

Biconnected Components
Game characterization: Robber and Marshals

- A robber and $k$ marshals play the game on a hypergraph
- The marshals have to capture the robber
- The robber tries to elude her capture, by running arbitrarily fast on the vertices of the hypergraph
Robbers and Marshals: the rules

- Each marshal stays on an edge of the hypergraph and controls all of its vertices at once.

- The robber can go from a vertex to another vertex running along the edges, but she cannot pass through vertices controlled by some marshal.

- The marshals win the game if they are able to monotonically shrink the moving space of the robber, and thus eventually capture her.

- Consequently, the robber wins if she can go back to some vertex previously controlled by marshals.
Step 0: the empty hypergraph
Step 1: first move of the marshals
Step 1: first move of the marshals
Step 2a: shrinking the space
Step 2a: shrinking the space
Step 2a: shrinking the space
A different robber’s choice
Step 2b: the capture
R&M Game and Hypertree Width

Let $H$ be a hypergraph.

**Theorem:** $H$ has hypertree width $\leq k$ if and only if $k$ marshals have a winning strategy on $H$.

**Corollary:** $H$ is acyclic if and only if one marshal has a winning strategy on $H$.

Winning strategies on $H$ correspond to hypertree decompositions of $H$ and vice versa.
Polynomial algorithm: Alternating LOGSPACE

Once I have guessed \( R \), how to guess the next marshal position \( S \)?
Polynomial algorithm: Alternating LOGSPACE

Once I have guessed R, how to guess the next marshal position S?
Polynomial algorithm: Alternating LOGSPACE

Once I have guessed R, how to guess the next marshal position S?
Polynomial algorithm: Alternating LOGSPACE

Once I have guessed R, how to guess the next marshal position S?

Monotonicity: \( \forall E \in \text{edges}(C_R): (P \cap UR) \subseteq US \)

Strict shrinking: \( (US) \cap C_R \neq \emptyset \)

LOGSPACE CHECKABLE
Logical Characterization of Hypertree width

Loosely guarded logic
Guarded Formulas

... $\exists X (g \land \varphi) ...$

Guard atom: $\text{free}(\varphi) \subseteq \var{g}$

$k$-guarded Formulas (loosely guarded):

... $\exists X (g_1 \land g_2 \land \cdots \land g_k \land \varphi) ...$

$k$-guard

GF(FO), GF$_k$(FO) are well-studied fragments of FO (Van Benthem’97, Gradel’99)
Logical Characterization of HW

Theorem: \( \text{HW}_k = \text{GF}_k(L) \)

From this general result, we also get a nice logical characterization of acyclic queries:

Corollary: \( \text{HW}_1 = \text{ACYCLIC} = \text{GF}(L) \)
An Example

\[ \exists X, Y, Z, T, U, W. (p(X, Y, Z) \land q(X, Y, T) \land r(Y, Z, U) \land s(T, W)) \]

Is acyclic:

\[
\begin{align*}
\exists X, Y, Z. (p(X, Y, Z) & \land \exists T. (q(X, Y, T) \land \exists W. s(T, W)) \land \\
& \land \exists U. r(Y, Z, U))
\end{align*}
\]
An Example

\[ \exists X, Y, Z, T, U, W. (p(X, Y, Z) \land q(X, Y, T) \land r(Y, Z, U) \land s(T, W)) \]

Is acyclic:

Indeed, there exists an equivalent guarded formula:

\[ \exists X, Y, Z. (p(X, Y, Z) \land \exists T. (q(X, Y, T) \land \exists W. s(T, W)) \land \exists U. r(Y, Z, U)) \]

Guarded subformula
Alternative view and generalized HT-Decompositions
Tree decomposition of hypergraph

\[ H \]

\[ \text{Tree decomp of } G(H) \]
Hypergraph and the tree decomposition
Hypergraph and the tree decomposition
Hypergraph and the tree decomposition

1, 11, 17, 19

1, 11, 17, 19

1, 2, 3, 4, 5, 6

3, 4, 5, 6, 7, 8

5, 6, 7, 8, 9

7, 9, 10

11, 12, 17, 18, 19

12, 15, 16, 18, 19

12, 13, 14, 15, 18, 19
Hypergraph and the tree decomposition

```
1, 11, 17, 19
1, 2, 3, 4, 5, 6
3, 4, 5, 6, 7, 8
5, 6, 7, 8, 9
7, 9, 10
```
Hypergraph and the tree decomposition
Hypergraph and the tree decomposition
Hypergraph and the tree decomposition

1, 11, 17, 19

1, 2, 3, 4, 5, 6
3, 4, 5, 6, 7, 8
5, 6, 7, 8, 9
7, 9, 10

11, 12, 17, 18, 19
12, 16, 17, 18, 19
12, 15, 16, 18, 19
12, 13, 14, 15, 18, 19
Hypergraph and the tree decomposition
Hypergraph and the tree decomposition

Generalized HT-Decomp of H of width 2
Generalized hypertree decomposition of width 2
Hypertree decomposition of width 3

h10(12,13,19), h12(14,15,18), h14(16,17,18)

h9(11,12,18), h15(1,17,19)

h2(1,4,5,6), h3(3,4,7,8)

h1(1,2,3)

h4(5,7), h5(6,8,9)

h6(7,9,10)
Hypertree decomposition

Hypertree decomposition of width 3
Hypertree decomposition

Hypertree decomposition of width 3

h10(12,13,19), h12(14,15,18), h14(16,17,18)

h9(11,12,18), h15(1,17,19)

h2(1,4,5,6), h3(3,4,7,8)

h1(1,2,3)

h4(5,7), h5(6,8,9)

h6(7,9,10)
Hypertree decomposition

Hypertree decomposition of width 3
Hypertree decomposition

Hypertree decomposition of width 3
Hypertree decomposition

Hypertree decomposition of width 3

h10(12,13,19), h12(14,15,18), h14(16,17,18)

h9(11,12,18), h15(1,17,19)

h2(1,4,5,6), h3(3,4,7,8)

h1(1,2,3)

h4(5,7), h5(6,8,9)

h6(7,9,10)
Hypertree decomposition

Hypertree decomposition of width 3

- $h_{1}(1,2,3)$
- $h_{6}(7,9,10)$
- $h_{4}(5,7)$, $h_{5}(6,8,9)$
- $h_{2}(1,4,5,6)$, $h_{3}(3,4,7,8)$
- $h_{9}(11,12,18)$, $h_{15}(1,17,19)$
- $h_{10}(12,13,19)$, $h_{12}(14,15,18)$, $h_{14}(16,17,18)$
Generalized hypertree decomposition of width 2
QUESTION:

Can we determine in polynomial time whether $g_{hw}(H) < k$ for constant $k$?

Observation: $g_{hw}(H) = h_{w}(H^*)$

Where $H^* = H \cup \{E' \mid \exists E \text{ in } \text{edges}(H): E' \subseteq E\}$

Recent result [AGG]: $g_{hw}(H) \leq 3h_{w}(H)+1$

Connection to the Hypergraph Sandwich problem!
Nasa problem

Part of relations for the Nasa problem

... cid_260(Vid_49, Vid_366, Vid_224),
cid_261(Vid_100, Vid_391, Vid_392),
cid_262(Vid_273, Vid_393, Vid_246),
cid_263(Vid_329, Vid_394, Vid_249),
cid_264(Vid_133, Vid_360, Vid_356),
cid_265(Vid_314, Vid_348, Vid_395),
cid_266(Vid_67, Vid_352, Vid_396),
cid_267(Vid_182, Vid_364, Vid_397),
cid_268(Vid_313, Vid_349, Vid_398),
cid_269(Vid_339, Vid_348, Vid_399),
cid_270(Vid_98, Vid_366, Vid_400),
cid_271(Vid_161, Vid_364, Vid_401),
cid_272(Vid_131, Vid_353, Vid_234),
cid_273(Vid_126, Vid_402, Vid_245),
cid_274(Vid_146, Vid_252, Vid_228),
cid_275(Vid_330, Vid_360, Vid_361),
...

680 relations
579 variables
Nasa problem: hypertree

Part of hypertree for the Nasa problem
Best known hypertree-width for the Nasa problem is 22
Adder circuit examples consist of a certain number of adder cells connected in a line

Adder_1:
6 relations and 8 variables

Adder_2:
11 relations and 15 variables

... 

Adder_99:
496 relations and 694 variables

... 

The basic cell of a adder circuit

The legal values for inputs and outputs in adder should be found

Hypertree width for all examples is 2
Hypertree for the adder circuit with one cell

Constraints
init(ci)
AND1(b, a, x2)
XOR1(b, a, x1)
AND2(ci, x3, x1)
OR(x2, co, x3)
XOR2(ci, s, x1)
Bridge circuit examples consist of a certain number of bridge cells connected in a line

Bridge_1: 11 relations and 11 variables

Bridge_2: 20 relations, 20 variables...

Bridge_99: 893 relations and 893 variables...

The basic cell of a bridge circuit

The legal values for variables (tensions and currents) in circuit should be found

Hypertree width for all examples is 2
The bridge circuit with one cell

Constraints are obtained by Kirchhoff Laws, Ohm's Law and others
Hypertree for the bridge circuit with one cell
The structure of many problems can be described by graphs or hypergraphs.

Many NP-hard problems become tractable for instances whose associated graphs or hypergraphs have bounded treewidth.

An important class of problems is tractable even in case of large treewidth. This class is hypergraph-based.

We described such problems (CQ,CSP,HOM) and developed an appropriate notion of width:

HYPERTREE WIDTH
Conclusions and open questions

- There are some interesting open questions:
  - Special condition

- Hypertree decomposition is a very natural notion
  - Issues of fixed-parameter tractability
  - Nonmonotonic capturing vs monotonic capturing

For papers and further material see:
http://ulisse.deis.unical.it/~frank/Hypertrees/