ILP'10
Firenze, 27-30 June 2010

Structural Decomposition Methods:

Identifying easy instances of hard problems

UNIVERSITÀ DELLA CALABRIA

Francesco Scarcello and **Gianluigi Greco**
University of Calabria, Italy

Outline of the Tutorial

- (NP-hard) Problems
- Identification of "Easy" Classes
- Beyond Tree Decompositions
- Characterizations of Hypertree Width
- Applications

Outline of the Tutorial

- (NP-hard) Problems
- Identification of "Easy" Classes
- Beyond Tree Decompositions
- Characterizations of Hypertree Width
- Applications

+ Appendix

The Knapsack Problem

OBJECT	WEIGHT	VALUE
Silver Plate	5500 g	$1,430.-
Golden Mirror	3200 g	$800.-
Sword	1500 g	$850.-
Painting	3400 g	$680.-
...	...	...

Problem Statement:

Given Instance: List of records $\langle Object, Weight, Value \rangle$, maximum weight G, desired total value W

Question: Is there a set $S \subseteq$ Objects, such that

$$\sum_{x \in S} weight(x) \leq G \quad \text{and} \quad \sum_{x \in S} value(x) \geq W \quad ?$$

The Knapsack Problem

OBJECT	WEIGHT	VALUE
Silver Plate	5500 g	$1,430.-
Golden Mirror	3200 g	$800.-
Sword	1500 g	$850.-
Painting	3400 g	$680.-
...	...	...

Problem Statement:

Given Instance: List of records $\langle Object, Weight, Value \rangle$, maximum weight G, desired total value W

Question: Is there a set $S \subseteq$ Objects, such that

$$\sum_{x \in S} weight(x) \leq G \quad \text{and} \quad \sum_{x \in S} value(x) \geq W \quad ?$$

16 kg $8,000.--

From Decisions to Computations

- **Search Problem**

  Compute a solution S such that

  $$\sum_{x \in S} weight(x) \leq G \quad \text{and} \quad \sum_{x \in S} value(x) \geq W$$

- **Optimization Problem**

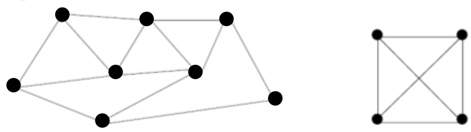
  Compute a solution S such that

  $$\sum_{x \in S} weight(x) \leq G \quad \text{and} \quad \sum_{x \in S} value(x) \text{ is maximized.}$$

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

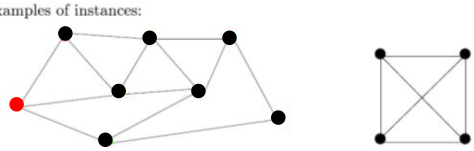


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

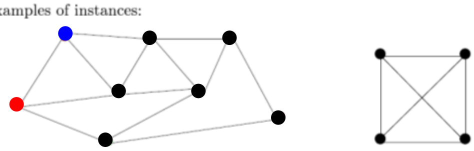


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

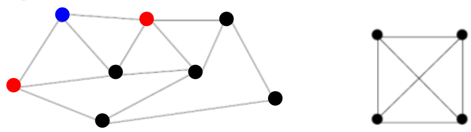


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

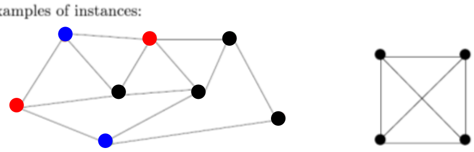


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

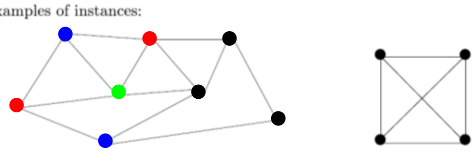


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:



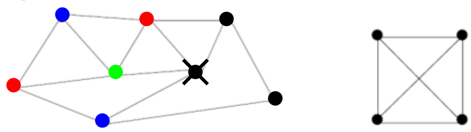
Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

BACKTRACKING!

Examples of instances:

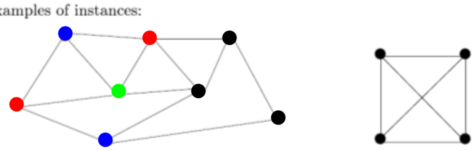


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

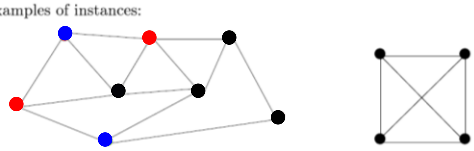


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

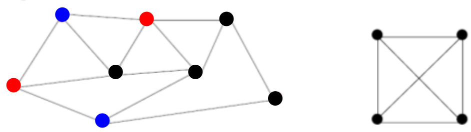


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

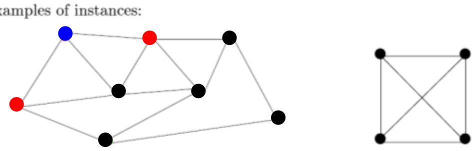


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

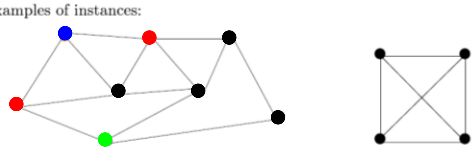


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

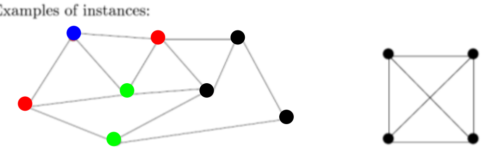


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

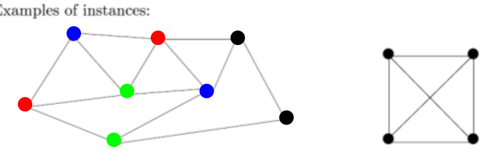


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

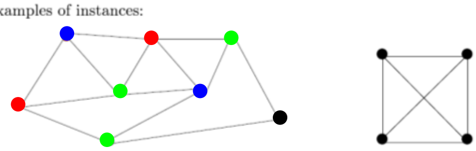


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

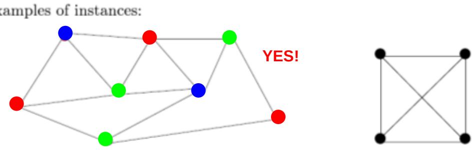


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

Examples of instances:

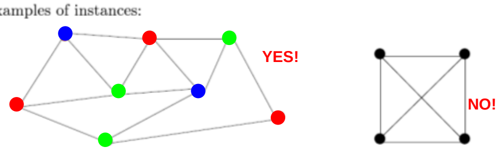


Associated search problem: Compute a correct 3-coloring, if possible.

Graph Three-colorability

Instance: A graph G.
Question: Is G 3-colorable?

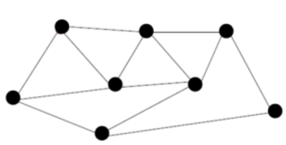
Examples of instances:



Associated search problem: Compute a correct 3-coloring, if possible.

Traveling Salesman Problem (TSP)

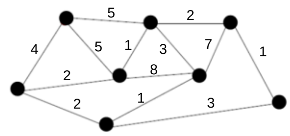
Instance: Road network G with distances, number M.
Question: Is there a "Tour" of total length $\leq M$?



Optimization problem: Compute tour of minimum length.

Traveling Salesman Problem (TSP)

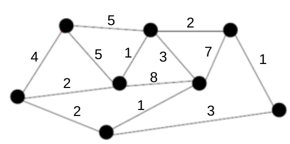
Instance: Road network G with distances, number M.
Question: Is there a "Tour" of total length $\leq M$?



Optimization problem: Compute tour of minimum length.

Traveling Salesman Problem (TSP)

Instance: Road network G with distances, number M.
Question: Is there a "Tour" of total length $\leq M$?

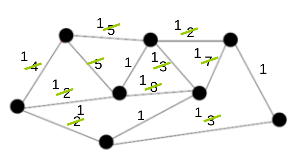


Hamiltonian Cycle

- Does there exist a cycle of n edges going through all n vertices?

Traveling Salesman Problem (TSP)

Instance: Road network G with distances, number M.
Question: Is there a "Tour" of total length $\leq M$? → 8



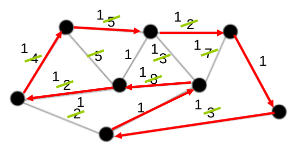
Hamiltonian Cycle

- Does there exist a cycle of n edges going through all n vertices?

Traveling Salesman Problem (TSP)

Instance: Road network G with distances, number M.

Question: Is there a "Tour" of total length $\leq M$? → 8

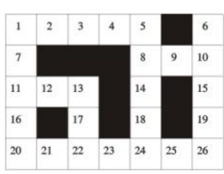


Hamiltonian Cycle

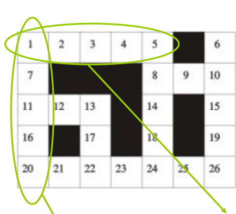
- Does there exist a cycle of n edges going through all n vertices?

G has Hamiltonian circuit ⇔ G' hat Tour of Length 8

Combinatorial Crossword Puzzle



Combinatorial Crossword Puzzle



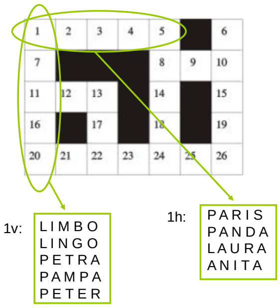
1v:

LIMBO
LINGO
PETRA
PAMPA
PETER

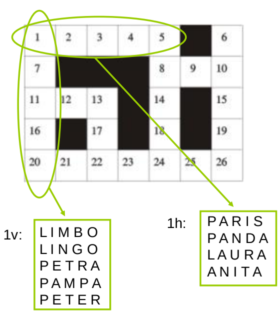
1h:

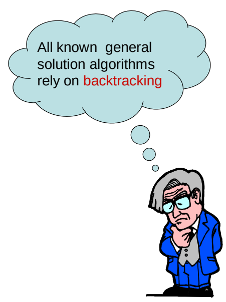
PARIS
PANDA
LAURA
ANITA

Combinatorial Crossword Puzzle



Combinatorial Crossword Puzzle





SATISFIABILITY (SAT)

Instance: A set of Clauses

$$(X1 \text{ or } X2 \text{ or } \overline{X3})$$
$$(\overline{X1} \text{ or } \overline{X2} \text{ or } X3)$$
$$(\overline{X1} \text{ or } \overline{X2} \text{ or } \overline{X3})$$
$$(\overline{X1} \text{ or } X2 \text{ or } X3)$$

Question: Is there a satisfying truth value assignment ?

SATISFIABILITY (SAT)

Instance: A set of Clauses

(X1 or X2 or $\overline{X3}$) ✓
($\overline{X1}$ or $\overline{X2}$ or X3) ✓
($\overline{X1}$ or $\overline{X2}$ or $\overline{X3}$) ✓
($\overline{X1}$ or X2 or X3) ✓

YES, e.g.:

X1=true
X2=false
X3=false

Question: Is there a satisfying truth value assignment ?

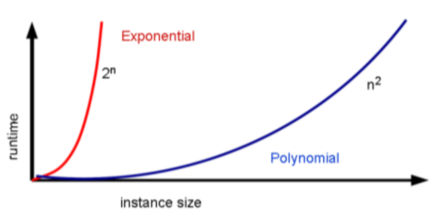
Inherent Problem Complexity

- Problems *decidable* or *undecidable*.
- We concentrate on decidable problems here.
- A problem is as complex as the best possible algorithm which solves it.

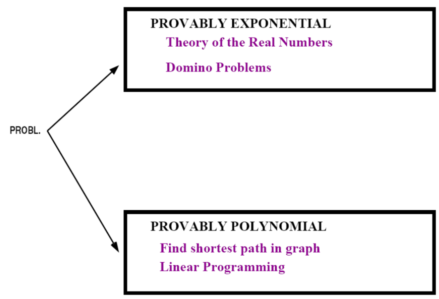
Inherent Problem Complexity

- Problems *decidable* or *undecidable*.
- We concentrate on decidable problems here.
- A problem is as complex as the best possible algorithm which solves it.

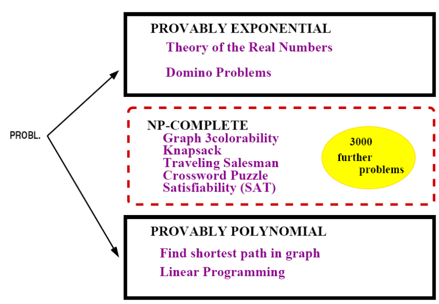
Number of steps it takes for input of size n



Time Complexity



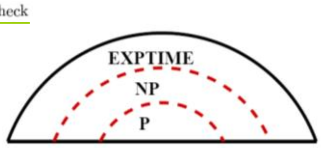
Time Complexity



The class NP

- **NP**: Nondeterministic Polynomial Time

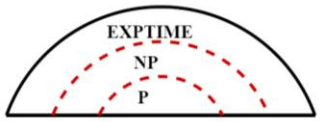
  Paradigm: Guess and Check



The class NP

- **NP**: Nondeterministic Polynomial Time

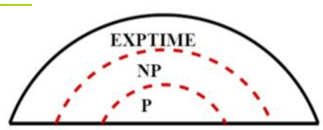
  Paradigm: Guess and Check



The class NP

- **NP**: Nondeterministic Polynomial Time

  Paradigm: Guess and Check



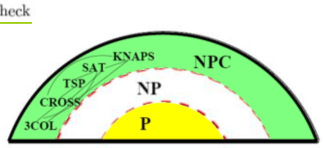
The most important open problem of Theoretical Computer Science!

Clay Mathematical Institute: $1.000.000

The class NP

- **NP**: Nondeterministic Polynomial Time

  Paradigm: Guess and Check



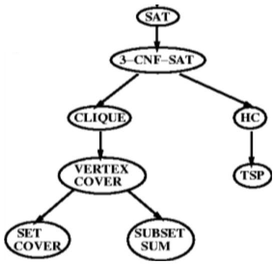
Structure inside NP

NPC: The hardest problems in NP.

All problems in NPC can be polynomially transformed into one another.

One polynomially solvable $\Rightarrow$ all polynomially solvable, i.e. NP=P.

Karp and Cook's Theorem: SAT is NP-Complete [1972]



Approaches for Solving Hard Problems

- NP-complete problems often occur in practice.
- They must be solved by acceptable methods.
- Three approaches:
  - Randomized local search
  - Approximation
  - Identification of easy (=polynomial) subclasses.

Approaches for Solving Hard Problems

- NP-complete problems often occur in practice.
- They must be solved by acceptable methods.
- Three approaches:
  - Randomized local search
  - Approximation
  - Identification of easy (=polynomial) subclasses.

Outline of the Tutorial

(NP-hard) Problems

Identification of “Easy” Classes

Beyond Tree Decompositions

Characterizations of Hypertree Width

Applications

Identification of Polynomial Subclasses

- High complexity arises often in “worst cases” only.
- Intricate structure of worst case problem instances.
- For inputs of simpler structure polynomial algorithms may exist.
- In practice many input instances are *simple*.

  Therefore:

  - Define suitable polynomially solvable subclasses of instances.
  - Prove that membership testing for these subclasses is polynomisl.
  - Develop efficient polynomial algorithms for these classes.

Problems with a Graph Structure

- With graph-based problems, high complexity is mostly due to *cyclicity*.

  Problems restricted to *acyclic* graphs are often trivially solvable (→3COL).
- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

Problems with a Graph Structure

- With graph-based problems, high complexity is mostly due to *cyclicity*.

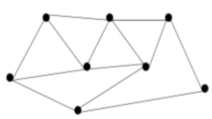
  Problems restricted to *acyclic* graphs are often trivially solvable (→3COL).
- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

How can we measure the degree of cyclicity?

(Three) Early Approaches

(1) **Feedback vertex number**

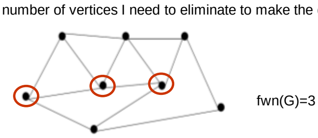
Min. number of vertices I need to eliminate to make the graph acyclic



(Three) Early Approaches

(1) **Feedback vertex number**

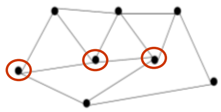
Min. number of vertices I need to eliminate to make the graph acyclic



(Three) Early Approaches

[1] Feedback vertex number

Min. number of vertices I need to eliminate to make the graph acyclic



fwn(G)=3

Is this really a good measure for the "degree of acyclicity" ?

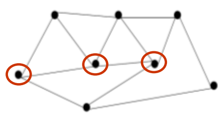
Pro: For fixed k we can check in quadratic time if fwn(G)=k (FPT) .

Con: Very simple graphs can have large FVN:

(Three) Early Approaches

[1] Feedback vertex number

Min. number of vertices I need to eliminate to make the graph acyclic

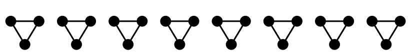


fwn(G)=3

Is this really a good measure for the "degree of acyclicity" ?

Pro: For fixed k we can check in quadratic time if fwn(G)=k (FPT) .

Con: Very simple graphs can have large FVN:



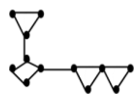
(Three) Early Approaches

[2] Feedback edge number → same problem.

(Three) Early Approaches

[2] Feedback <u>edge</u> number → same problem.

[3] Maximum size of biconnected components



bcw(G)=4

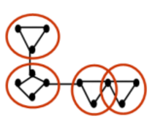
Pro: Actually bcw(G) can be computed in linear time

Con: Adding a single edge may have tremendous effects to bcw(G)

(Three) Early Approaches

[2] Feedback <u>edge</u> number → same problem.

[3] Maximum size of biconnected components



bcw(G)=4

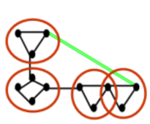
Pro: Actually bcw(G) can be computed in linear time

Con: Adding a single edge may have tremendous effects to bcw(G)

(Three) Early Approaches

[2] Feedback <u>edge</u> number → same problem.

[3] Maximum size of biconnected components



bcw(G)=4

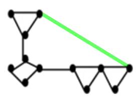
Pro: Actually bcw(G) can be computed in linear time

Con: Adding a single edge may have tremendous effects to bcw(G)

(Three) Early Approaches

[2] Feedback edge number → same problem.

[3] Maximum size of biconnected components



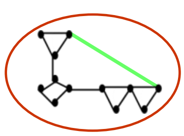
bcw(G)=4

Pro: Actually bcw(G) can be computed in linear time
Con: Adding a single edge may have tremendous effects to bcw(G)

(Three) Early Approaches

[2] Feedback edge number → same problem.

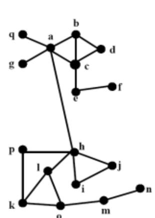
[3] Maximum size of biconnected components



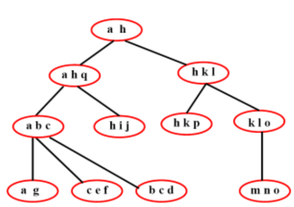
bcw(G)=~~4~~ 12

Pro: Actually bcw(G) can be computed in linear time
Con: Adding a single edge may have tremendous effects to bcw(G)

Tree Decompositions [Robertson & Seymour '86]

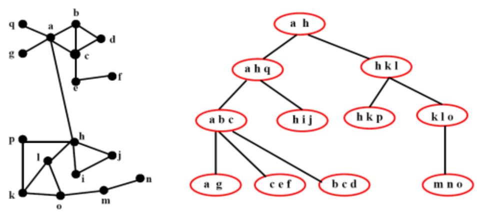


Graph G



Tree decomposition of width 2 of G

Tree Decompositions [Robertson & Seymour '86]

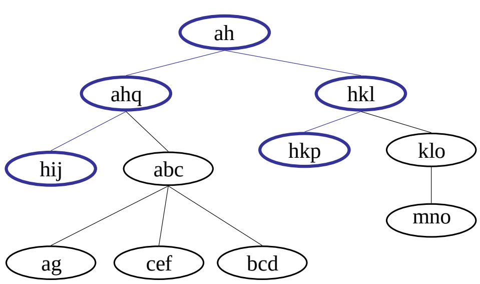


Graph G

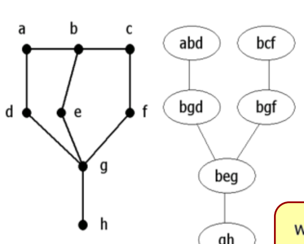
Tree decomposition of width 2 of G

- Every edge realized in some bag
- Connectedness condition

Connectedness condition for *h*



Tree Decompositions and Treewidth



$\text{width}(T,X_i) = \max |X_i| - 1$

$\text{tw}(G) = \min \text{width}(T,X_i)$

Properties of Treewidth

- tw(acyclic graph)=1
- tw(cycle) = 2
- tw(G+v) ≤ tw(G)+1
- tw(G+e) ≤ tw(G)+1
- tw(K_n) = n-1

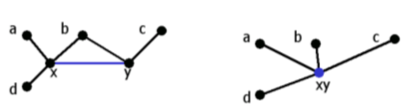
Properties of Treewidth

- tw(acyclic graph)=1
- tw(cycle) = 2
- tw(G+v) ≤ tw(G)+1
- tw(G+e) ≤ tw(G)+1
- tw(K_n) = n-1

[1] tw is preserved under graph minors
[2] tw is a key for tractability
[3] tw is tractable

Graph Minors

- H is a minor of G if it can be obtained by repeatedly applying:
  - Edge deletion
  - Vertex deletion
  - Edge contraction



An important Metatheorem

Courcelle's Theorem [1987]

Let P be a problem on graphs that can be formulated in **Monadic Second Order Logic** (MSO).

Then P can be solved in liner time on graphs of bounded treewidth

An important Metatheorem

Courcelle's Theorem [1987]

Let P be a problem on graphs that can be formulated in **Monadic Second Order Logic** (MSO).

Then P can be solved in liner time on graphs of bounded treewidth

- **Theorem.** (Fagin): Every NP-property over graphs can be represented by an existential formula of Second Order Logic.

  NP=ESO

- Monadic SO (MSO): Subclass of SO, only *set variables*, but no relation variables of higher arity.

  3-colorability $\in$ MSO.

Three Colorability in MSO

$$
\begin{aligned}
(\exists R, G, B) \quad [\quad & (\forall x\, (R(x) \vee G(x) \vee B(x))) \\
\wedge \quad & (\forall x (R(x) \Rightarrow (\neg G(x) \wedge \neg B(x)))) \\
\wedge \quad & \ldots \\
\wedge \quad & \ldots \\
\wedge \quad & (\forall x, y (E(x,y) \Rightarrow (R(x) \Rightarrow (G(x) \vee B(y))))) \\
\wedge \quad & (\forall x, y (E(x,y) \Rightarrow (G(x) \Rightarrow (R(x) \vee B(y))))) \\
\wedge \quad & (\forall x, y (E(x,y) \Rightarrow (B(x) \Rightarrow (R(x) \vee G(y))))) \,]
\end{aligned}
$$

Is Treewidth a Tractable Notion?

- Can we efficiently check for a constant k whether $tw(G) \leq k$?
- Can we construct a tree decomposition efficiently in case ?

Is Treewidth a Tractable Notion?

- Can we efficiently check for a constant k whether $tw(G) \leq k$?
- Can we construct a tree decomposition efficiently in case ?

Yes !

The answer was first given via an amazing theorem of Robertson and Seymour [1986]

Is Treewidth a Tractable Notion?

- Can we efficiently check for a constant k whether $tw(G) \leq k$?
- Can we construct a tree decomposition efficiently in case ?

Yes !

The answer was first given via an amazing theorem of Robertson and Seymour [1986]

Each class of graphs that is closed under taking minors is characterized by a finite set of **forbidden** minors.

- The “obstruction set” of class C.
- For each k and for each class of graphs G for which $tw(G) \leq k$, the obstruction set is a finite set of grids.
- It can be checked in quadratic time whether a fixed graph is a minor of an input graph.
- Linear time algorithm for checking $tw(G) \leq k$ by Bodlaender ‘96

Outline of the Tutorial

Beyond Treewidth

- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are "simple" graphs that are heavily cyclic. For example, a clique.

Beyond Treewidth

- Treewidth is currently the most successful measure of graph cyclicity. It subsumes most other methods.
- However, there are "simple" graphs that are heavily cyclic. For example, a clique.

There are also problems whose structure is better described by **hypergraphs** rather than by graphs…

Three Problems

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

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

Three Problems

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

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

But actually: HOM = BCQ = CSP

The Homomorphism Problem

- Given two relational structures

$$A = (U, R_1, R_2, \ldots, R_k)$$
$$B = (V, S_1, S_2, \ldots, S_k)$$

- Decide whether there exists a homomorphism *h* from *A* to *B*

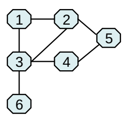
$$h : U \longrightarrow V$$
$$\text{such that} \quad \forall \mathbf{x}, \forall i$$
$$\mathbf{x} \in R_i \Rightarrow h(\mathbf{x}) \in S_i$$

HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h*.

Hardness: Transformation from 3COL.



A

1	2
1	3
2	3
3	4
2	5
4	5
3	6

B

red	green
red	blue
green	red
green	blue
blue	red
blue	green

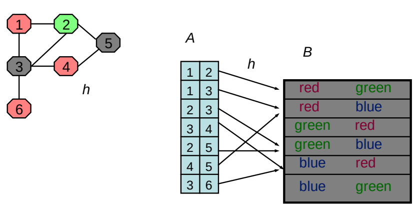
Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess *h*.

Hardness: Transformation from 3COL.



Graph 3-colourable iff HOM(*A*,*B*) yes-instance.

Constraint Satisfaction Problems

- Set of variables V={X_1,...,X_n}, domain D
- Set of constraints {C_1,...,C_m}

  where: C_i= <S_i, R_i>

  scope (X_{j1},...,X_{jr})

  relation

1	6	7	3
1	5	3	9
2	4	7	6
3	5	4	7

- Solution: A substitution h: V→D such that h(S_i)∈R_i holds, for each i

Constraint Satisfaction Problems

- Set of variables $V=\{X_1,\ldots,X_n\}$, domain D
- Set of constraints $\{C_1,\ldots,C_m\}$

where: $C_i = \langle S_i, R_i \rangle$

scope — relation

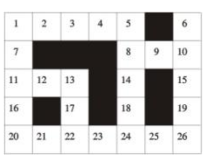
$(X_{j1},\ldots,X_{jr})$

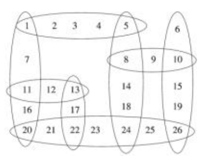
1	6	7	3
1	5	3	9
2	4	7	6
3	5	4	7

- Solution: A substitution h: $V \to D$ such that $h(S_i) \in R_i$ holds, for each i

Associated hypergraph: $\{var(S_i) \mid 1 \leq i \leq m\}$

Example of CSP: Crossword Puzzle





Conjunctive Database Queries

DATABASE:

Enrolled

John	Algebra	2003
Robert	Logic	2003
Mary	DB	2002
Lisa	DB	2003
………	…..	…….

Teaches

McLane	Algebra	March
Kolaitis	Logic	May
Lausen	DB	June
Rahm	DB	May
………	…..	…….

Parent

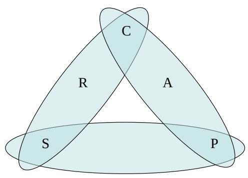
McLane	Lisa
Kolaitis	Robert
Rahm	Mary
………	…..

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

$$ans \leftarrow Enrolled(S,C,R) \wedge Teaches(P,C,A) \wedge Parent(P,S)$$

Queries and Hypergraphs

ans ← Enrolled(S,C,R) ∧ Teaches(P,C,A) ∧ Parent(P,S)



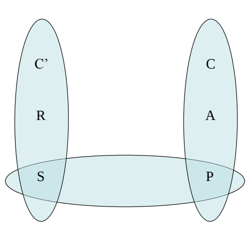
Queries and CSPs

- Database schema (scopes):
  - *Enrolled (Pers#, Course, Reg-Date)*
  - *Teaches (Pers#, Course, Assigned)*
  - *Parent (Pers1, Pers2)*
- Is there any teacher whose child attend **some** course?

ans ← Enrolled(S,C',R) ∧ Teaches(P,C,A) ∧ Parent(P,S)

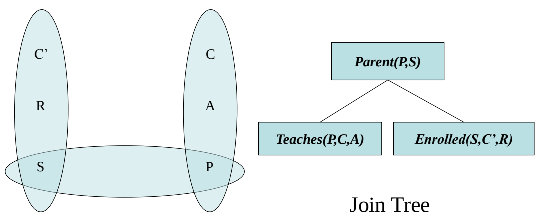
Acyclic Queries

ans ← Enrolled(S,C',R) ∧ Teaches(P,C,A) ∧ Parent(P,S)



Acyclic Queries

ans ← Enrolled(S,C',R) ∧ Teaches(P,C,A) ∧ Parent(P,S)



Complexity of BCQs

- 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)
  (Gottlob, Leone, Scarcello '98)

Properties of Acyclic BCQs

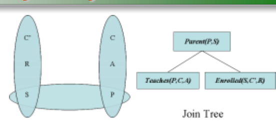
- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

Deciding Hypergraph Acyclicity

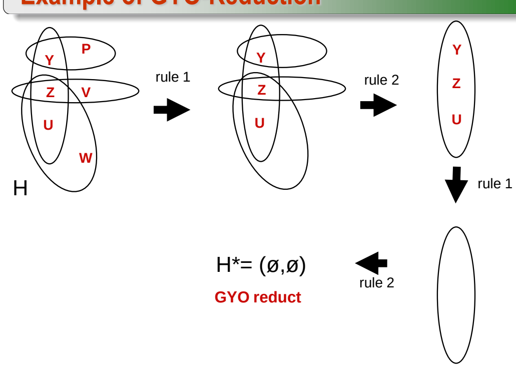
- Can be done in linear time by **GYO-Reduction**



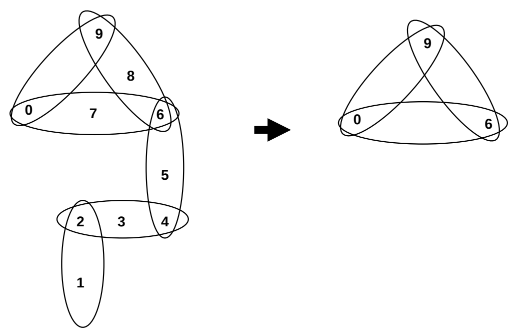
Input: Hypergraph H
Method: Apply the following two rules as long as possible:
(1) Eliminate vertices that are contained in at most one hyperedge
(2) Eliminate hyperedges that are empty or contained in other hyperedges
H is acyclic iff the resulting hypergraph empty

Proof: Easy by considering leaves of join tree

Example of GYO-Reduction



Example of GYO-irreducible Hypergraph



Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

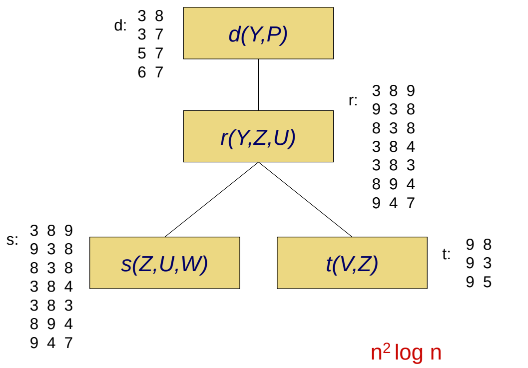
Answering Acyclic Instances

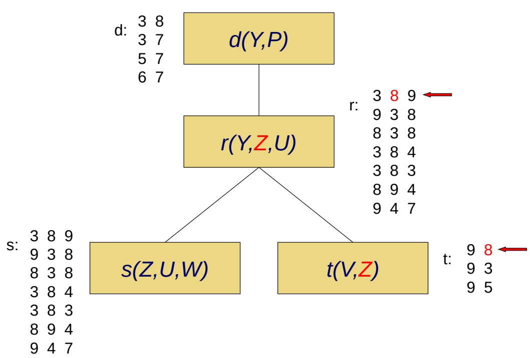
HOM: The homomorphism problem

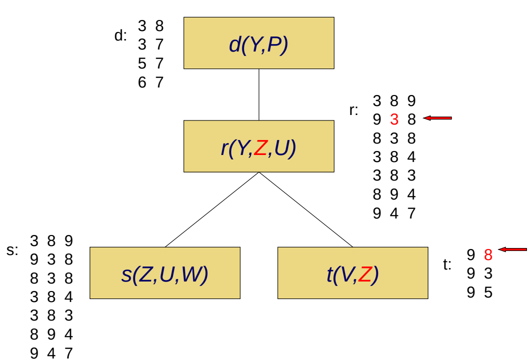
BCQ: Boolean conjunctive query evaluation

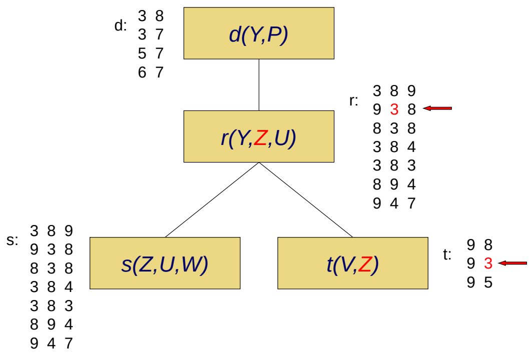
CSP: Constraint satisfaction problem

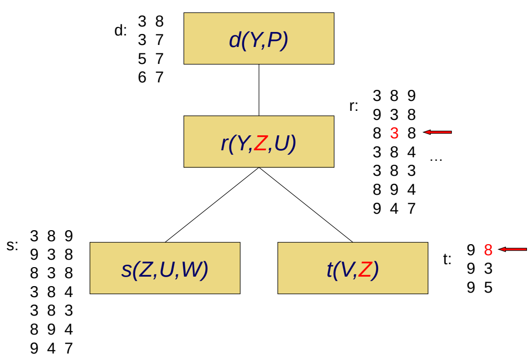
Yannakakis's Algorithm (ABCQs):
Dynamic Programming over a Join Tree

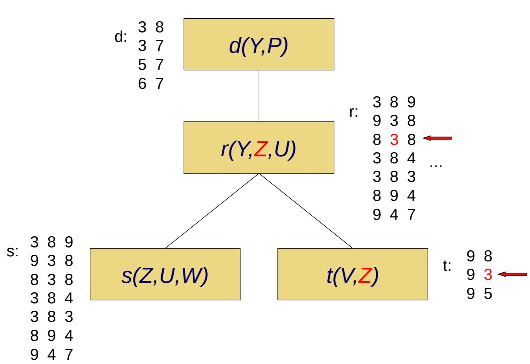

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5
n2 log n

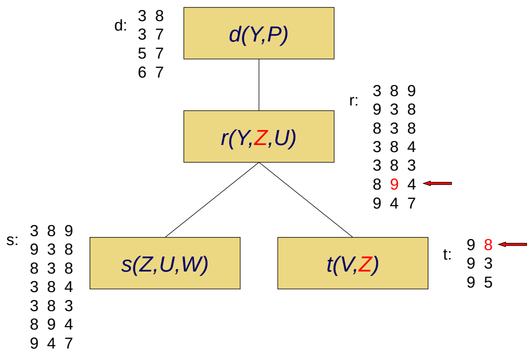

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5

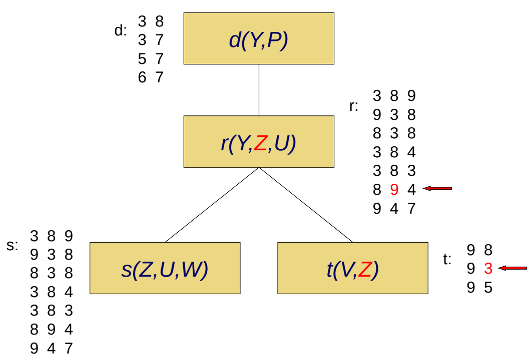

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5

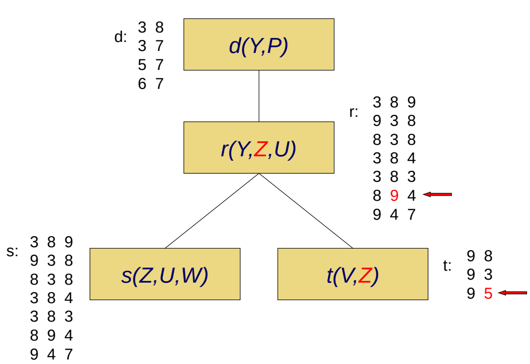

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5

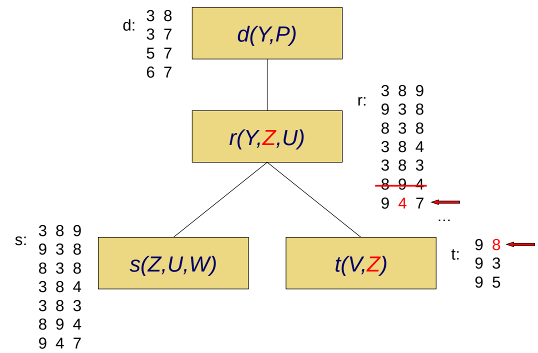

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
...
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5

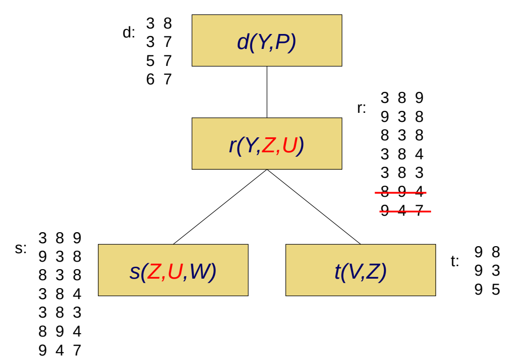

d: 3 8 / 3 7 / 5 7 / 6 7
d(Y,P)
r: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
...
r(Y,Z,U)
s: 3 8 9 / 9 3 8 / 8 3 8 / 3 8 4 / 3 8 3 / 8 9 4 / 9 4 7
s(Z,U,W)
t(V,Z)
t: 9 8 / 9 3 / 9 5

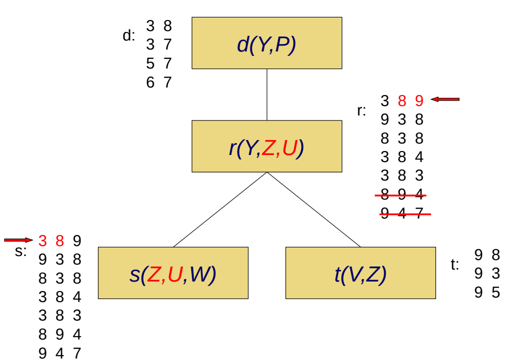

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

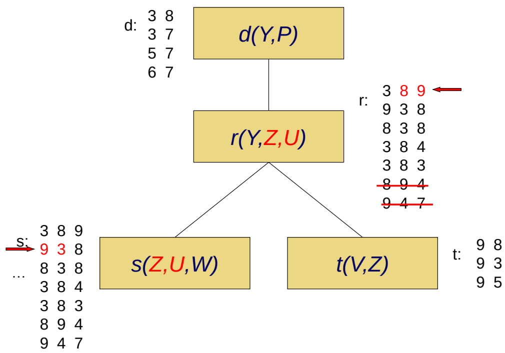

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

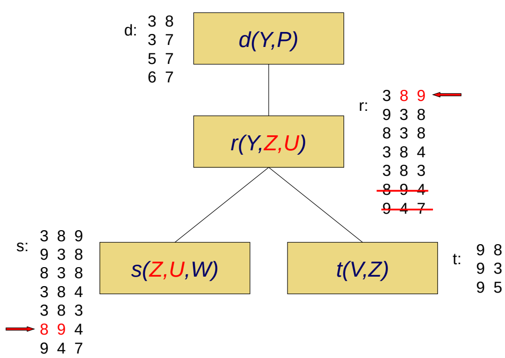

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

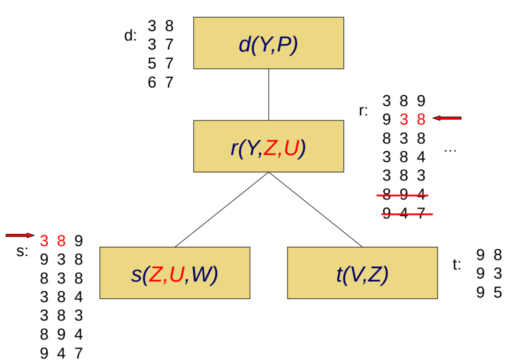

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5

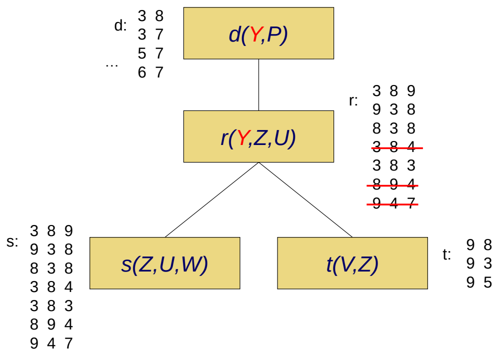

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5

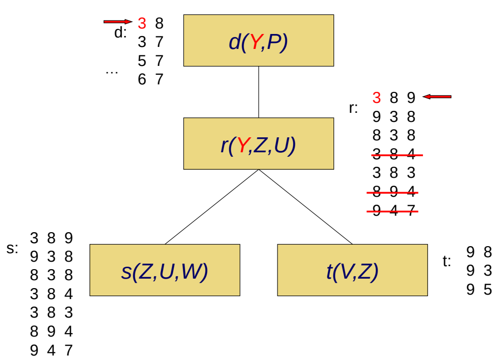

d(Y,P)
r(Y,Z,U)
s(Z,U,W)
t(V,Z)
d:
3 8
3 7
5 7
6 7
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5

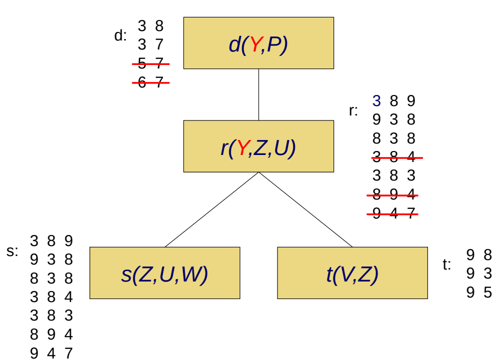

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5

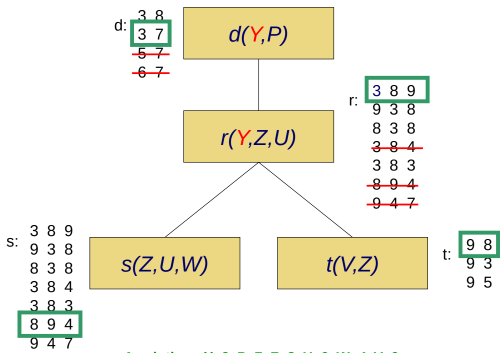

d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
d(Y,P)
r:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
...
r(Y,Z,U)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
s(Z,U,W)
t(V,Z)
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
...
d(Y,P)
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~
s(Z,U,W)
t(V,Z)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5


d:
3 8
3 7
5 7
6 7
...
d(Y,P)
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~
s(Z,U,W)
t(V,Z)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5


d:
3 8
3 7
~~5 7~~
~~6 7~~
d(Y,P)
r(Y,Z,U)
r:
3 8 9
9 3 8
8 3 8
~~3 8 4~~
3 8 3
~~8 9 4~~
~~9 4 7~~
s(Z,U,W)
t(V,Z)
s:
3 8 9
9 3 8
8 3 8
3 8 4
3 8 3
8 9 4
9 4 7
t:
9 8
9 3
9 5



A solution: Y=3, P=7, Z=8, U=9, W=4, V=9

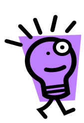
Answering Acyclic Instances

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Yannakakis's Algorithm (ABCQs):
Dynamic Programming over a Join Tree



- Answering ACQs can be done adding a top-down phase to Yannakakis' algorithm for ABCQs
  - obtain a full reducer,
  - join the partial results (or perform a backtrack free visit)

ABCQ is in LOGCFL

Theorem [Gottlob, Leone, Scarcello '99]:

Acyclic CSP-solvability is LOGCFL-complete.
Answering acyclic BCQs is LOGCFL-complete

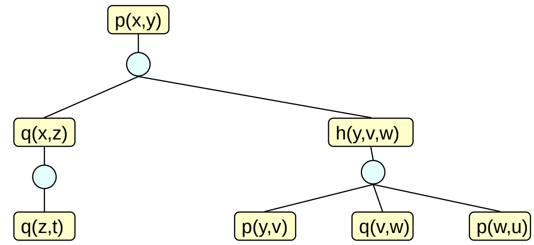
- LOGCFL: class of problems/languages that are logspace-reducible to a CFL
- Admit efficient parallel algorithms

$AC_0 \subseteq NL \subseteq \text{LOGCFL} = SAC_1 \subseteq AC_1 \subseteq NC_2 \subseteq \cdots \subseteq NC = AC \subseteq P \subseteq NP$

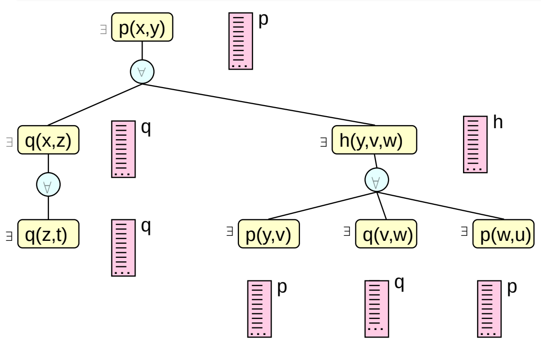
Characterization of LOGCFL [Ruzzo '80]:

LOGCFL = Class of all problems solvable with a logspace ATM with polynomial tree-size

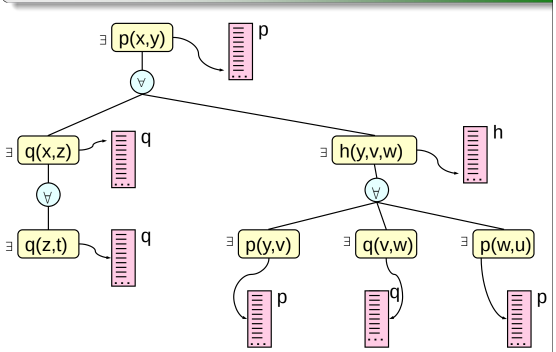
ABCQ is in LOGCFL



ABCQ is in LOGCFL



ABCQ is in LOGCFL



Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

Answering ACQs via Consistency

Method: Enforce pairwise consistency, by taking the join of all pairs of relations until a fixpoint is reached, or some relation becomes empty

d(Y,P) r(Y,Z,U)

s(Z,U,W) t(V,Z)

d:

3	8
3	7
5	7
6	7

r:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

s:

3	8	9
9	3	8
8	3	8
3	8	4
3	8	3
8	9	4
9	4	7

t:

9	8
9	3
9	5

Join Trees or Local Consistency?

- Computing a join tree (in linear time, and logspace-complete [Gottlob, Leone, Scarcello'98+ SL=L]) may be viewed as a clever way to enforce local---and hence---global consistency
- Cost for the computation of the full reducer:

  $O(m\ n^2 \log n)$ vs $O(m\ n \log n)$

- N.B. n is the (maximum) number of tuples in a relation and may be very large

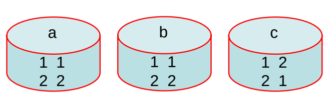
Global and Local Consistency

- An important property of ACQs:
  - Local consistency ➔ Global consistency
  - That is, if all relations are pairwise consistent, then the query is not empty
- Not true in the general case

Global and Local Consistency

- An important property of ACQs:
  - Local consistency ➔ Global consistency
  - That is, if all relations are pairwise consistent, then the query is not empty
- Not true in the general case

$$ans \leftarrow a(X,Y) \wedge b(Y,Z) \wedge c(Z,X)$$



Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

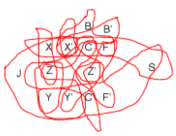
Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

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

n size of the database
m number of atoms in the query

m = 11 !



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

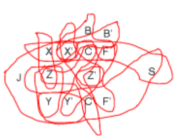
Properties of Acyclic BCQs

- Acyclicity is efficiently recognizable
- Acyclic BCQs (ABCQs) can be efficiently solved
- Local consistency → Global consistency

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

n size of the database
m number of atoms in the query

m = 11 !

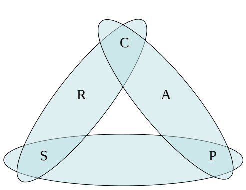


Classical methods worst-case complexity: $O(n^m)$

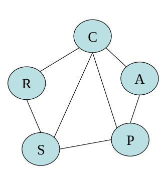
Still, it can be evaluated in $O(m \cdot n^2 \cdot \log n)$

Primal Graphs of Queries

ans ← *Enrolled(S,C,R)* ∧ *Teaches(P,C,A)* ∧ *Parent(P,S)*

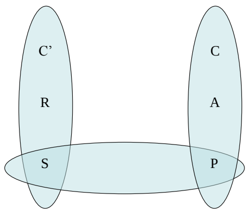


Hypergraph *H(Q)*

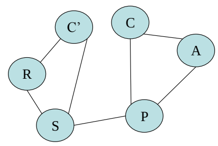


Primal graph *G(Q)*

Hypergraphs vs Graphs

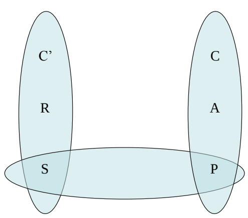


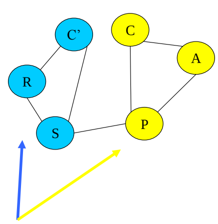
An acyclic hypergraph



Its cyclic primal graph

Hypergraphs vs Graphs





There are two cliques.
We cannot know where they come from

Drawbacks of Treewidth

Acyclic queries may have unbounded TW!

Example:

$q \leftarrow p_1(X_1, X_2,\ldots, X_n, Y_1) \wedge \ldots \wedge p_n(X_1, X_2,\ldots, X_n, Y_n)$

is acyclic, obviously polynomial, but has treewidth *n-1*

Beyond Treewidth

Bounded Degree of Cyclicity (Hinges)
[Gyssens & Paredaens '84, Gyssens, Jeavons, Cohen '94]
Does not generalize bounded treewidth.

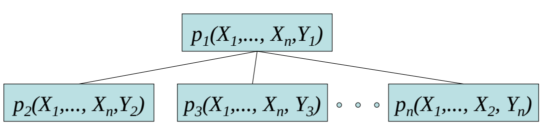
Bounded Query width
[Chekuri & Rajaraman '97]

Group together query atoms (hyperedges) instead of variables

Query Decomposition

$q \leftarrow p_1(X_1, X_2, \ldots, Y_1) \wedge \ldots \wedge p_m(X_1, X_2, \ldots, Y_n)$

Query width = 1 = acyclicity

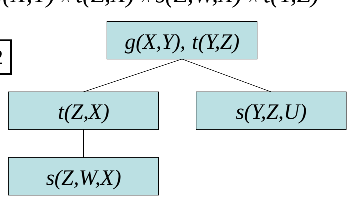


- Every atom/hyperarc appears in some node
- Connectedness conditions for variables and atoms

Decomposition of Cyclic Queries

$q \leftarrow s(Y,Z,U) \wedge g(X,Y) \wedge t(Z,X) \wedge s(Z,W,X) \wedge t(Y,Z)$

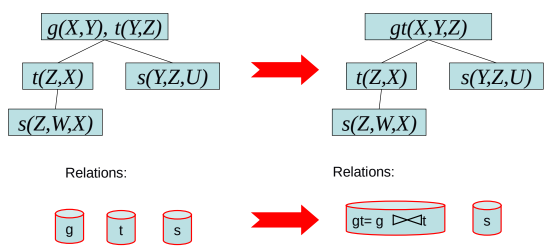
Query width = 2



BCQ is polynomial for queries of bounded query width, **if** a query decomposition is given

From Decompositions to Join Trees

$$q \leftarrow s(Y,Z,U) \wedge g(X,Y) \wedge t(Z,X) \wedge s(Z,W,X) \wedge t(Y,Z)$$



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

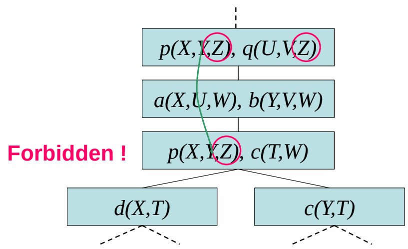
[Gottlob, Leone, Scarcello '99]

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

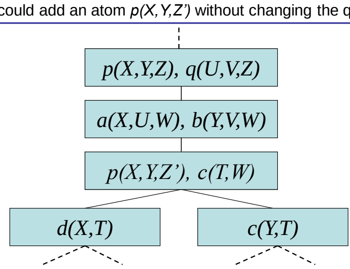
NP-hardness is due to an overly strong condition in the definition of query decomposition



Important Observation

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

we could add an atom *p(X,Y,Z')* without changing the query

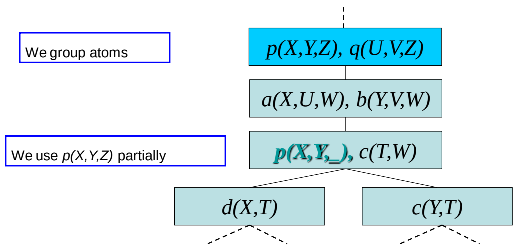


Hypertree Decompositions

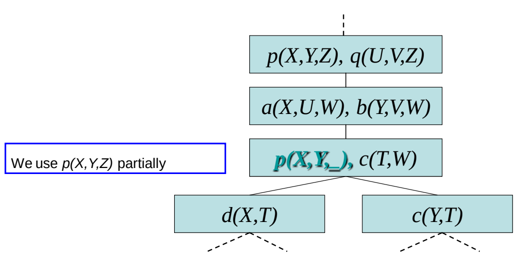
Query atoms can be used “partially” as long as the full atom appears somewhere else

More liberal than query decomposition

Grouping and Reusing Atoms

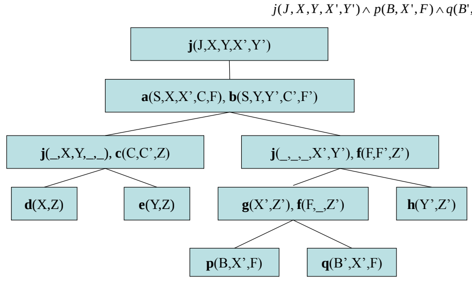


Reusing Atoms



Back to the Example

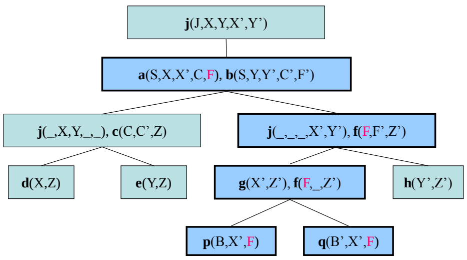
$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



Hypertree of width 2

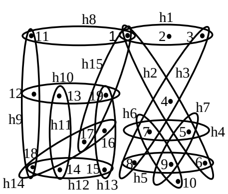
Generalized Hypertree Decomposition

GHD= Hypertree + Connectedness condition

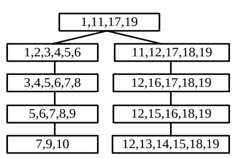


Tree Decomposition of Hypergraphs

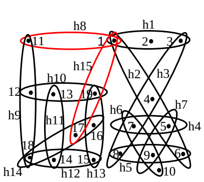
H

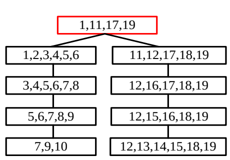


Tree decomp of G(H)

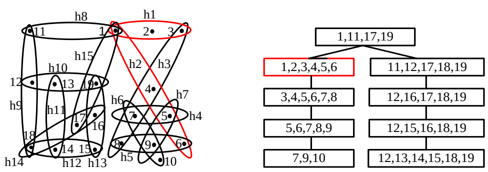


Tree Decomposition of Hypergraphs

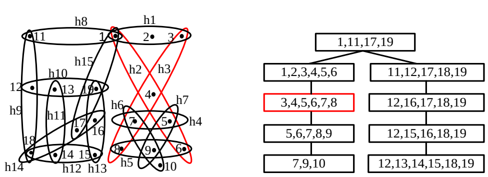




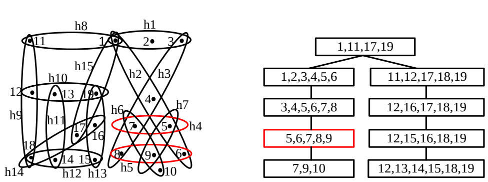
Tree Decomposition of Hypergraphs



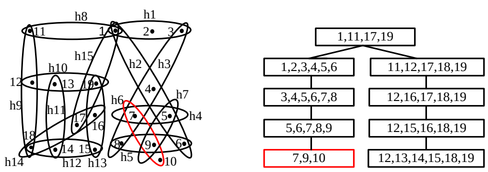
Tree Decomposition of Hypergraphs



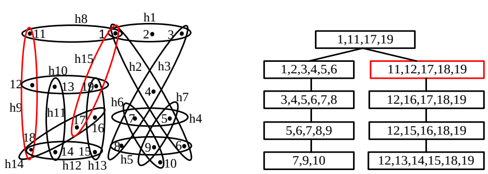
Tree Decomposition of Hypergraphs



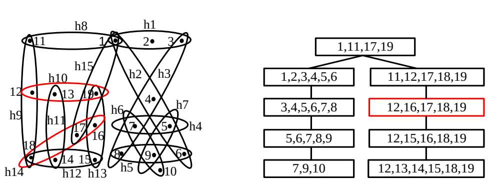
Tree Decomposition of Hypergraphs



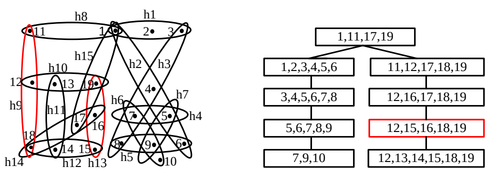
Tree Decomposition of Hypergraphs



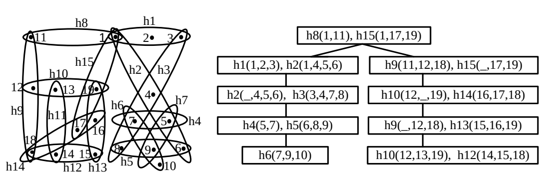
Tree Decomposition of Hypergraphs



Tree Decomposition of Hypergraphs



Generalized Hypertree Decompositions



Generalized hypetree decomposition of width 2

Computational Question

- Can we determine in polynomial time whether ghw(H) < k for constant k ?

Computational Question

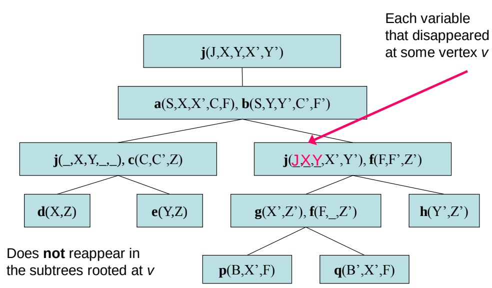
- Can we determine in polynomial time whether ghw(H) < k for constant k ?

Bad news: ghw(H) < 4? NP-complete

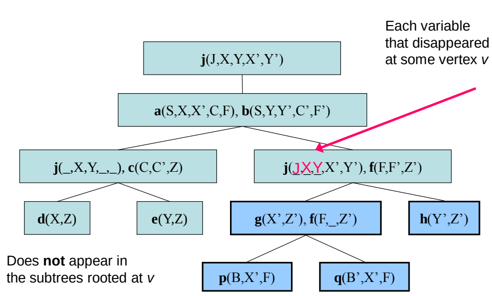
[Schwentick et. al. 06]

Hypertree Decomposition (HTD)

HTD = Generalized HTD +Special Condition



Special Condition



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).

But: FP-intractable wrt k: W[2]-hard.

Evaluating Queries with Bounded (g)hw

k is fixed

Given:
a database db of relations
a CSP Q over db such that $hw(Q) \leq k$ *or* $ghw \leq k$
a width k hypertree decomposition of Q

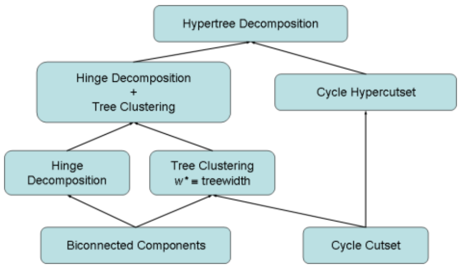
- Deciding whether *(Q,db) solvable* is in $O(n^{k+1} \log n)$ and complete for LOGCFL
- Computing *Q(db)* is feasible in output-polynomial time

Observation

If H has n vertices, then HW(H)≤n/2+1

- Does not hold for TW:
  - $TW(K_n)=n-1$
- Often HW < TW.
  - H-Decomps are interesting in case of bounded arity, too.

Comparison Results



Relationship GHW vs HW

Observation:

ghw(H) = hw(H*)

where H* = H ∪ {E | ∃E in edges(H): E´ ⊆ E}

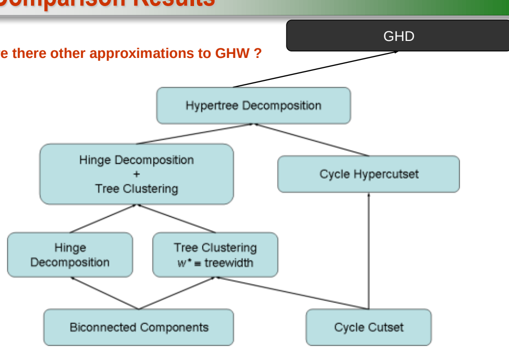
Exponential!

Approximation Theorem [Adler,Gottlob,Grohe ,05] :

ghw(H) <= 3hw(H)+1

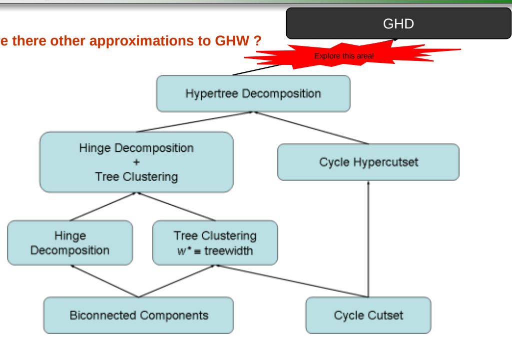
Comparison Results

Are there other approximations to GHW ?



Comparison Results

Are there other approximations to GHW ?



Outline of the Tutorial

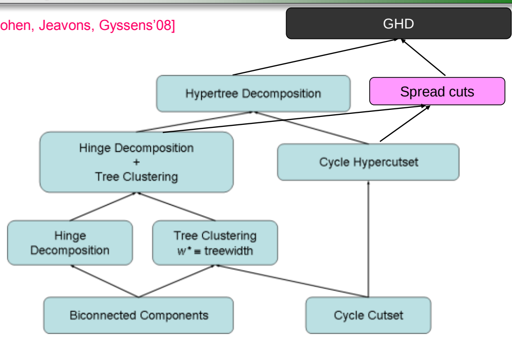
- (NP-hard) Problems
- Identification of "Easy" Classes
- Beyond Tree Decompositions
- Characterizations of Hypertree Width
- Applications

Outline of the Tutorial

- (NP-hard) Problems
- Identification of "Easy" Classes
- Beyond Tree Decompositions, and more!
- Characterizations of Hypertree Width
- Applications

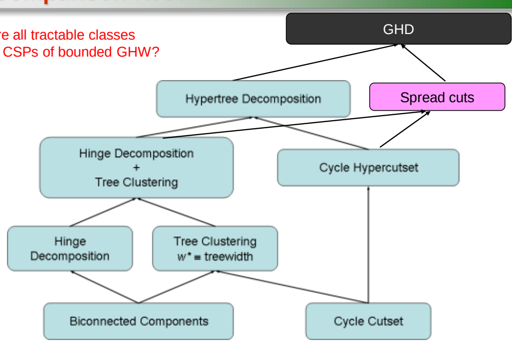
Comparison Results

[Cohen, Jeavons, Gyssens'08]



Comparison Results

Are all tractable classes of CSPs of bounded GHW?



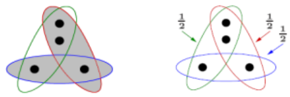
Going Beyond…

- Treewidth and Hypertree width are based on tree-like aggregations of subproblems that are efficiently solvable
- k variables (resp. k atoms) ➔ $||I||^k$ solutions (per subproblem)
- Is there some more general property that makes the number of solutions in any bag polynomial?
- YES!
  [Grohe & Marx '06]

Fractional (edge) Covering

An **edge cover** of a hypergraph is a subset of the edges such that every vertex is covered by at least one edge.
$\varrho(H)$: size of the smallest edge cover.

A **fractional edge cover** is a weight assignment to the edges such that every vertex is covered by total weight at least 1.
$\varrho^*(H)$: smallest total weight of a fractional edge cover.



$\varrho(H) = 2$ $\varrho^*(H) = 1.5$

From Marx's presentation about fractional covers

Edge Covers vs Fractional Edge Covers

Fact: It is NP-hard to determine the edge cover number $\varrho(H)$.
Fact: The fractional edge cover number $\varrho^*(H)$ can be determined in polynomial time using linear progamming.

The gap between $\varrho(H)$ and $\varrho^*(H)$ can be arbitrarily large.

Example:
$\binom{2k}{k}$ vertices: all the possible strings with k 0's and k 1's.
$2k$ hyperedges: edge E_i contains the vertices with 1 at the i-th position.

Edge cover: if only k edges are selected, then there is a vertex that contains 1's only at the remaining k positions, hence not covered $\Rightarrow \varrho(H) \geq k+1$.

Fractional edge cover: assign weight $1/k$ to each edge, each vertex is covered by exactly k edges $\Rightarrow \varrho^*(H) \leq 2k \cdot 1/k = 2$.

From Marx's presentation about fractional covers

Solutions and Fractional Edge Covering

Lemma: If the hypergraph of instance I has edge cover number w, then there are at most $\|I\|^w$ satisfying assignments.
Proof: Assume that $C_1, \ldots, C_w$ cover the instance. Fixing a satisfying assignment for each C_i determines all the variables.

Lemma: If the hypergraph of instance I has fractional edge cover number w, then there are at most $\|I\|^w$ satisfying assignments (and they can be enumerated in polynomial time).
Proof: By Shearer's Lemma.

From Marx's presentation about fractional covers

Shearer's Lemma (Combinatorial Version)

Shearer's Lemma: Let $H = (V, E)$ be a hypergraph, and let $A_1, A_2, \dots, A_p$ be (not necessarily distinct) subsets of V such that each $v \in V$ is contained in at least q of the A_i's. Denote by E_i the edge set of the hypergraph projected to A_i. Then

$$|E| \le \prod_{i=1}^{p} |E_i|^{1/q}.$$

Example:

$$E = \{1, 13, 2, 23, 234, 24\} \quad q = 2$$

$$A_1 = 123 \qquad A_2 = 124 \qquad A_3 = 34$$

$$E_1 = \{1, 13, 2, 23\} \quad E_2 = \{1, 2, 24\} \quad E_3 = \{\emptyset, 3, 4, 34\}$$

$$6 = |E| \le (|E_1| \cdot |E_2| \cdot |E_3|)^{1/q} = (4 \cdot 3 \cdot 4)^{1/2} = 6.928$$

From Marx's presentation about fractional covers

Shearer's Lemma (Entropy Version)

Shearer's Lemma: Assume we have the following random variables:

- $X_1, \dots, X_n$,
- $Y_1, \dots, Y_m$, where each $Y_i = (X_{i_1}, \dots, X_{i_k})$ is a combination of some X_i's,
- $X = (X_1, \dots, X_n)$.

If each X_j appears in at least q of the Y_i's, then $H(X) \le \frac{1}{q} \sum H(Y_i)$.

Entropy: "information content"
$H(X) = -\sum_x P(X = x) \log_2 P(X = x)$

From Marx's presentation about fractional covers

Bounding the Number of Solutions

Lemma: If the hypergraph of instance I has fractional edge cover number w, then there are at most $||I||^w$ satisfying assignments.

Example: Let $C_1(x_1, x_2) \wedge C_2(x_2, x_3) \wedge C_3(x_1, x_3)$ be an instance where each constraint is satisfied by at most n pairs.

Fractonal edge cover number: $3/2 \Rightarrow$ we have to show that there are at most $n^{3/2}$ solutions.

Let $X = (x_1, x_2, x_3)$ be a random variable with uniform distribution over the **satisfying assignments** of the instance.

$Y_1 = (x_1, x_2)$ $Y_2 = (x_2, x_3)$ $Y_3 = (x_1, x_3)$
$H(Y_i) \le \log_2 n$ (has at most n different values)
$H(X) \le \frac{1}{2}(H(Y_1) + H(Y_2) + H(Y_3)) \le \frac{3}{2} \log_2 n$

X has uniform distribution, hence it has $2^{H(X)} = 2^{\frac{3}{2} \log_2 n} = n^{3/2}$ different values.

From Marx's presentation about fractional covers

The Result is Tight!

Theorem *Let $\mathcal{Q}$ be a class of join queries. Then the following statements are equivalent:*

(1) Queries in $\mathcal{Q}$ have answers of polynomial size.
(2) Queries in $\mathcal{Q}$ can be evaluated in polynomial time.
(3) Queries in $\mathcal{Q}$ can be evaluated in polynomial time by an explicit join-project plan.
(4) $\mathcal{Q}$ has bounded fractional edge cover number.

[Atserias, Grohe, Marx '08]

- Note that this tractability result does not cover "tractable" classes of queries as the acyclic queries
- Why that?
- Because acyclic queries may have an exponential number of solutions, but computable efficiently (and with *anytime* algorithms)
- Idea: Combine fractional covers with hypertrees!

Fractional Hypertree Decompositions

In a **fractional hypertree decomposition** of width w, bags of vertices are arranged in a tree structure such that

1. For every edge e, there is a bag containing the vertices of e.
2. For every vertex v, the bags containing v form a connected subtree.
3. A fractional edge cover of weight w is given for each bag.

Fractional hypertree width: width of the best decomposition.

Note: fractional hypertree width $\leq$ generalized hypertree width

[Grohe & Marx '06]

- A query may be solved efficiently, if a fractional hypertree decomposition is given
- FHDs are approximable: If the the width is $\leq w$, a decomposition of width $O(w^3)$ may be computed in polynomial time [Marx '09]

More Beyond?

- A new notion: the submodular width
- Bounded submodular width is a necessary and sufficient condition for fixed-parameter tractability (under a technical complexity assumption)

[Marx '10]

Outline of the Tutorial

- (NP-hard) Problems
- Identification of "Easy" Classes
- Beyond Tree Decompositions, and more!
- Characterizations of Hypertree Width
- Applications

Characterizations of Hypertree Width

- Logical characterization:
  Loosely guarded logic
- Game characterization:
  The robber and marshals game

Guarded Formulas

$$\ldots\ \exists \overline{X}\ (g \wedge \varphi) \ldots$$

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

k-guarded Formulas (loosely guarded):

$$\ldots\ \exists \overline{X}\ (\underbrace{g_1 \wedge g_2 \wedge \cdots \wedge g_k}_{k\text{-guard}} \wedge \varphi) \ldots$$

GF(FO), GF_k(FO) are well-studied fragments of FO (Van Benthem'97, Gradel'99)

Logical Characterization of HW

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

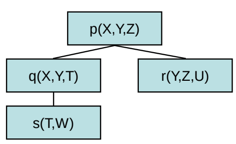
From this general result, we also get a nice logical characterization of acyclic queries:

Corollary: $\mathrm{HW}_1 = \mathrm{ACYCLIC} = \mathrm{GF}(L)$

An Example

$$\exists X,Y,Z,T,U,W.(p(X,Y,Z) \wedge q(X,Y,T) \wedge r(Y,Z,U) \wedge s(T,W))$$

Is acyclic:



Indeed, there exists an equivalent guarded formula:

$$\exists X,Y,Z.(p(X,Y,Z) \wedge \exists T.(q(X,Y,T) \wedge \exists W.s(T,W)) \wedge \wedge \exists U.r(Y,Z,U))$$

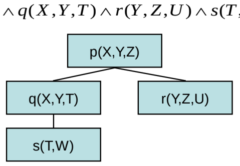
Guard

Guarded subformula

An Example

$$\exists X,Y,Z,T,U,W.(p(X,Y,Z) \wedge q(X,Y,T) \wedge r(Y,Z,U) \wedge s(T,W))$$

Is acyclic:



Indeed, there exists an equivalent guarded formula:

$$\exists X,Y,Z.(p(X,Y,Z) \wedge \exists T.(q(X,Y,T) \wedge \exists W.s(T,W)) \wedge \wedge \exists U.r(Y,Z,U))$$

Guard

Guarded subformula

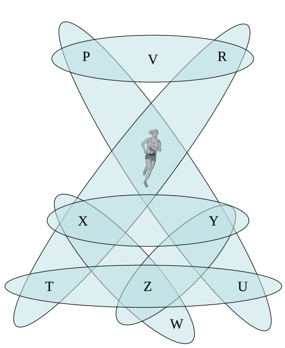
Game Characterization: Robber and Marshals

- 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

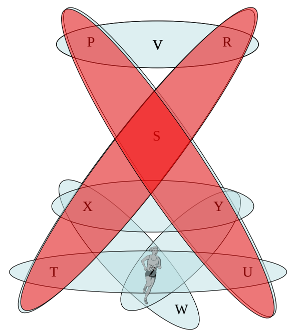
Robbers and Marshals: The Rules

- 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

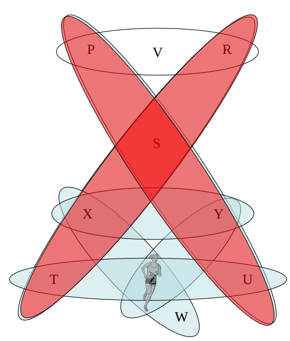
Step 0: the empty hypergraph



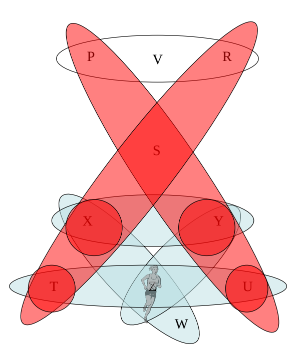
Step 1: first move of the marshals



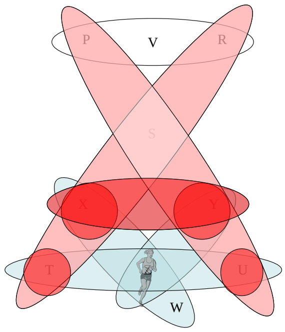
Step 1: first move of the marshals



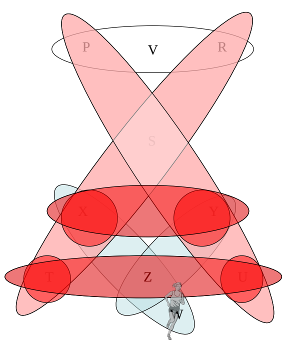
Step 2a: shrinking the space



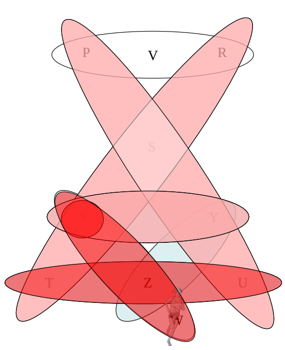
Step 2a: shrinking the space



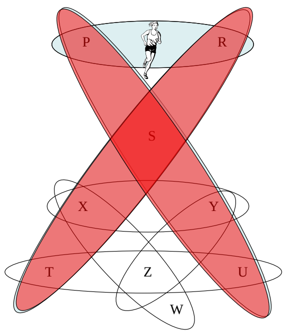
Step 2a: shrinking the space



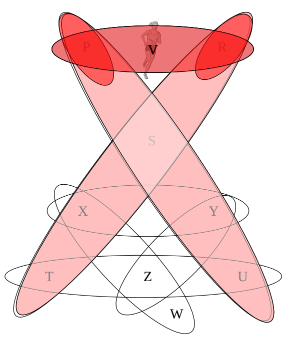
The capture



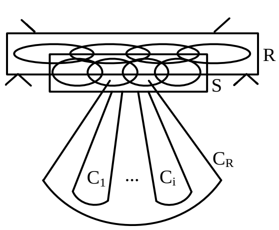
A different robber's choice



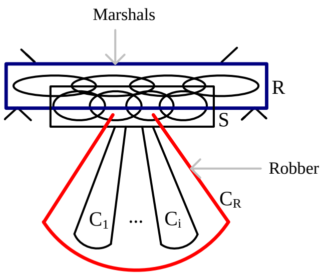
Step 2b: the capture



Marshals…

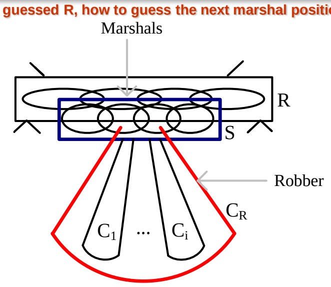


Marshals…



Polynomial algorithm: Alternating LOGSPACE

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



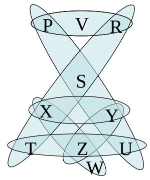
Monotonicity: $\forall$ E$\in$ edges(C_R): (E$\cap$UR) $\subseteq$ US

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

} LOGSPACE CHECKABLE

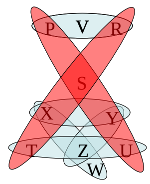
Strategies and Decompositions

$$ans \leftarrow a(S,X,T,R) \wedge b(S,Y,U,P) \wedge c(T,U,Z) \wedge e(Y,Z) \wedge g(X,Y) \wedge f(R,P,V) \wedge \wedge d(W,X,Z)$$



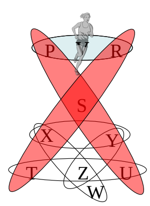
First choice of the two marshals

a(S,X,T,R), **b**(S,Y,U,P)



A possible choice for the robber

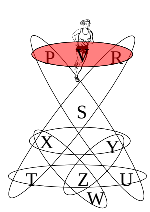
a(S,X,T,R), **b**(S,Y,U,P)



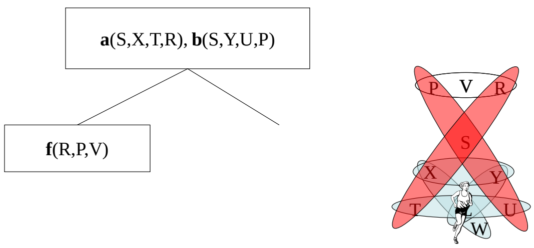
The capture

a(S,X,T,R), **b**(S,Y,U,P)

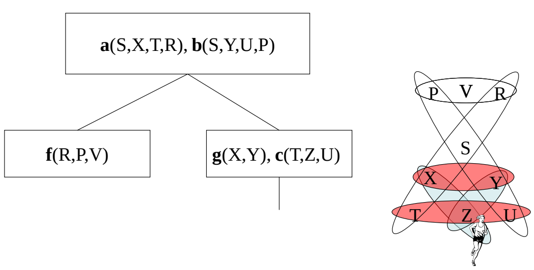
f(R,P,V)



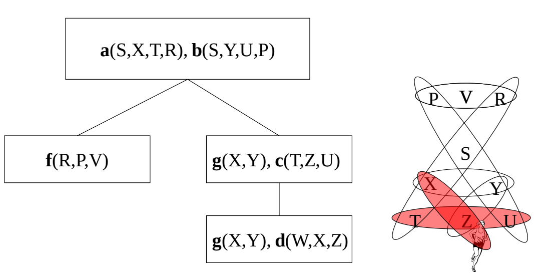
The second choice for the robber



The marshals corner the robber



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.

Outline of the Tutorial

- (NP-hard) Problems
- Identification of “Easy” Classes
- Beyond Tree Decompositions, and more!
- Characterizations of Hypertree Width
- Applications

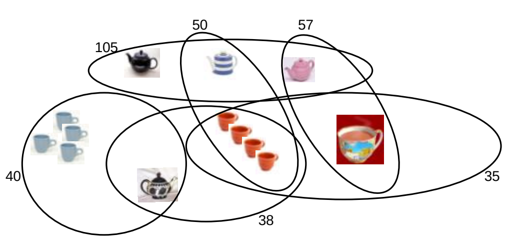
Applications (beyond query answering)

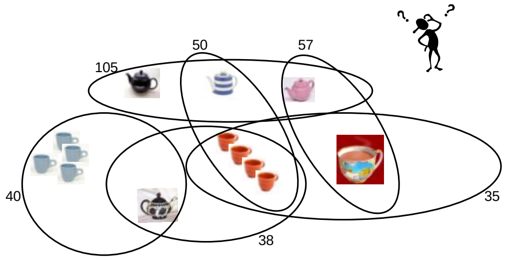
- Query optimization
- Query containment
- Constraint Satisfaction
- Clause subsumption
- Belief Networks
- Diagnosis
- Game Theory
- …

Combinatorial Auctions

Combinatorial Auctions

Combinatorial Auctions

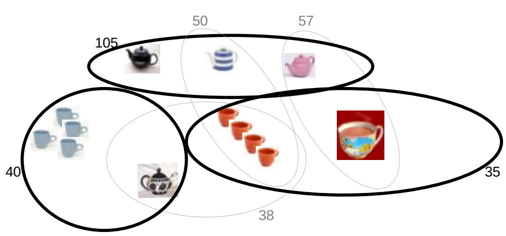




Winner Determination Problem

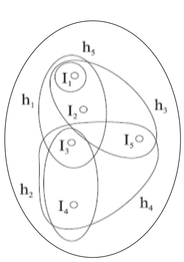
- Determine the outcome that maximizes the sum of accepted bid prices

Combinatorial Auctions



Total £ 180.--

A Negative Result

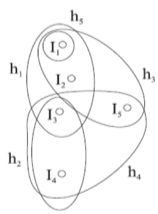


Theorem: The Winner Determination Problem remains NP-hard even in case of acyclic hypergraphs

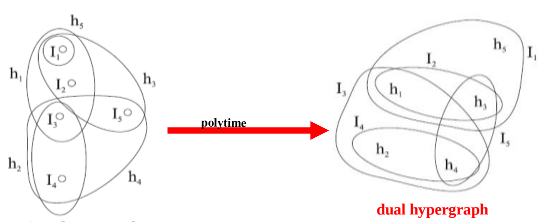
[Gottlob & Greco '07]

Work on the *dual* hypergraph instead

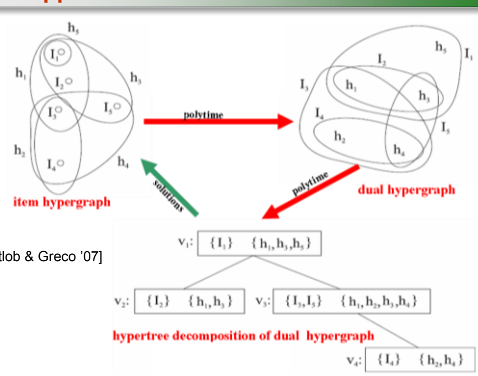
Dual Hypergraph



Dual Hypergraph



The Approach



[Gottlob & Greco '07]

Quantified CSPs

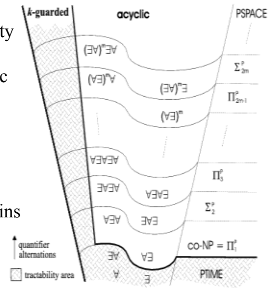
Bad News:

- Even tree-structured QCSPs with prefix ∀∃ are intractable.
- For fixed domains, the tractability of bounded-treewidth QCSPs is optimal: even QCPS with acyclic hypergraphs and bounded treewidth incidence graphs are intractable

Good News:

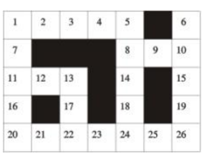
- *k*-guarded QCSPs are tractable, without any restriction on domains or quantified alternations.

[Gottlob, Greco, Scarcello '05]



For further results → [Hubie Chen]

(CSP) Optimization Problems



The puzzle may admit more than one solution...

- E.g., find the solution that **minimizes** the total number of vowels occurring in the words

A Classification for Optimization Problems

CSOP

Each mapping variable-value has a cost.
Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.

1 2 3 4 5
PARIS
PANDA
LAURA
ANITA

A Classification for Optimization Problems

CSOP
- Each mapping variable-value has a cost.
- Then, find an assignment:
  - Satisfying all the constraints, and
  - Having the minimum total cost.

WCSP
- Each tuple has a cost.
- Then, find an assignment:
  - Satisfying all the constraints, and
  - Having the minimum total cost.

1 2 3 4 5
P A R I S
P A N D A
L A U R A
A N I T A

A Classification for Optimization Problems

CSOP
- Each mapping variable-value has a cost.
- Then, find an assignment:
  - Satisfying all the constraints, and
  - Having the minimum total cost.

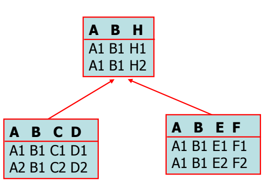
WCSP
- Each tuple has a cost.
- Then, find an assignment:
  - Satisfying all the constraints, and
  - Having the minimum total cost.

MAX-CSP
- Each constraint relation has a cost.
- Then, find an assignment:
  - Minimizing the cost of violated relations.

1 2 3 4 5
P A R I S
P A N D A
L A U R A
A N I T A

Tractability of CSOP Instances

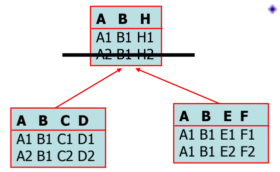
- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**



Tractability of CSOP Instances

- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**

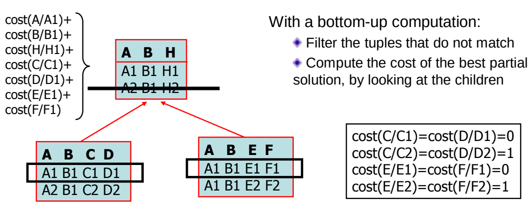
With a bottom-up computation:
- Filter the tuples that do not match



Tractability of CSOP Instances

- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**

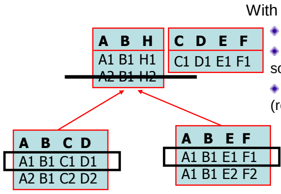
With a bottom-up computation:
- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children



Tractability of CSOP Instances

- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**

With a bottom-up computation:
- Filter the tuples that do not match
- Compute the cost of the best partial solution, by looking at the children
- Propagate the best partial solution (resolving ties arbitrarily)



Tractability of CSOP Instances

- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**
- Over "nearly-acyclic" instances…

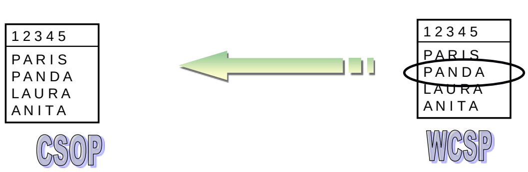
Tractability of CSOP Instances

- Over acyclic instances, adapt the dynamic programming approach in **(Yannakakis'81)**
- Over "nearly-acyclic" instances…

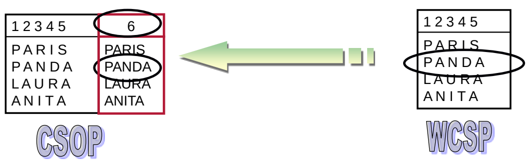
Apply "acyclicization" via decomposition methods

Bounded Hypertree Width Instances are Tractable

Tractability of WCSP Instances

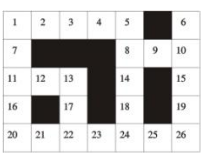


Tractability of WCSP Instances

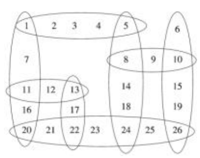


The mapping:
- Is feasible in linear time
- Preserves the solutions
- Preserves the Hypertree Width

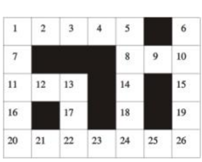
In-Tractability of MAX-CSP Instances



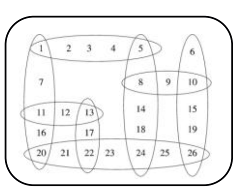
- Maximize the number of words placed in the puzzle



In-Tractability of MAX-CSP Instances



- Maximize the number of words placed in the puzzle

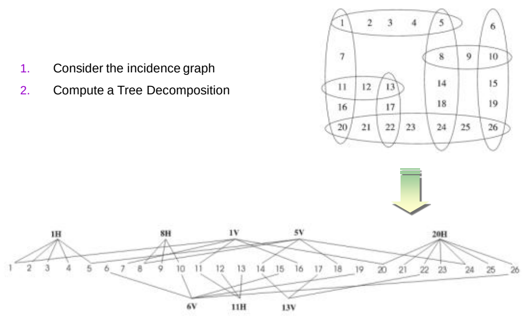


- Add a “big” constraint with no tuple

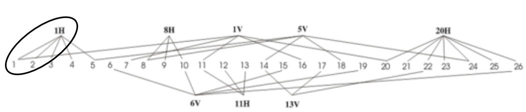
The puzzle is satisfiable ↔ exactly one constraint is violated in the **acyclic** MAX-CSP

Tractability of MAX-CSP Instances

1. Consider the incidence graph
2. Compute a Tree Decomposition



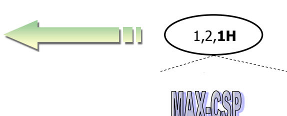
Tractability of MAX-CSP Instances

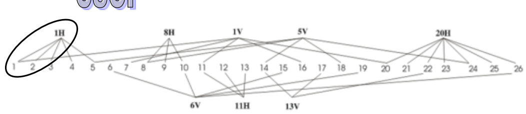


Tractability of MAX-CSP Instances

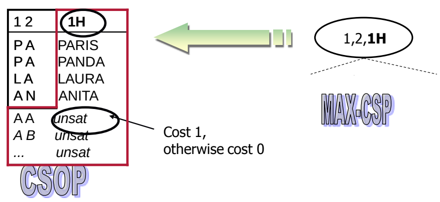
1 2	1H
P A	PARIS
P A	PANDA
L A	LAURA
A N	ANITA
A A	*unsat*
A B	*unsat*
...	*unsat*

Cost 1,
otherwise cost 0





Tractability of MAX-CSP Instances



The mapping:

- Is feasible in time exponential in the width
- Preserves the solutions
- Leads to an Acyclic CSOP Instance

Weighted Hypertree Decompositions

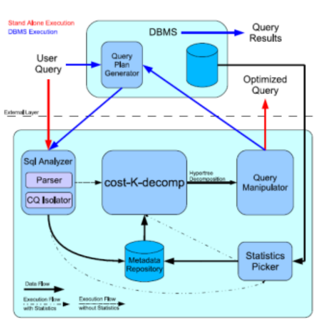
- Hypertree decompositions having k-bounded width are not always equivalent
- We want to find the best ones
- We need a way for weighting decompositions according to a given criterium

Hypertree Weighting Functions

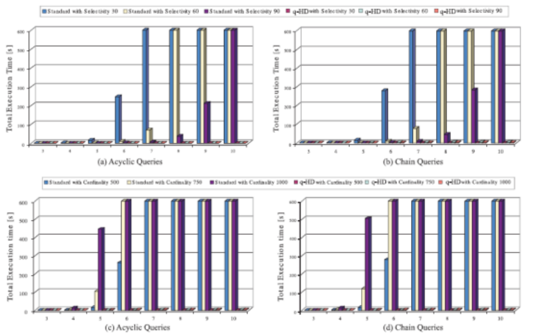
Let $\mathcal{H}$ be a hypergraph, $\omega_{\mathcal{H}}$ is any polynomial-time function that maps each hypertree decomposition HD = <T,χ, λ> of $\mathcal{H}$ to a real number, called the weight of HD.

Example: $\omega_{\mathcal{H}}(\text{HD}) = \max_{p \in vertices(T)} |\lambda(p)|$

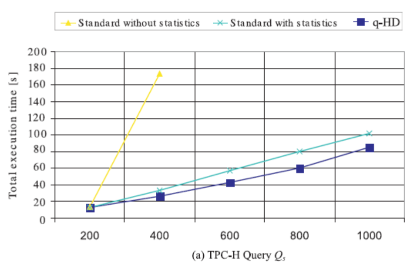
Practice of Weighted Hypertree Decompositions



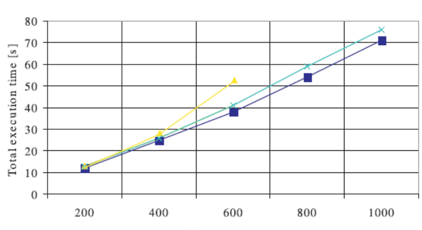
Hypertrees vs a well-known commercial DBMS



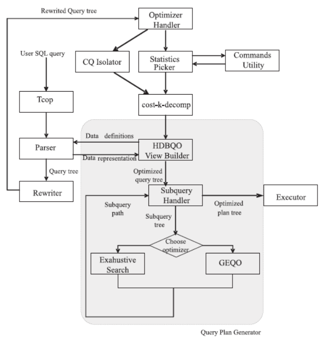
TPC-H queries



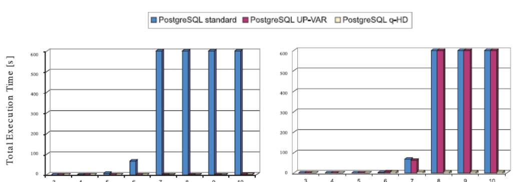
TPC-H queries



Inside PostgreSQL



Results Inside PostgreSQL



(a) Acyclic Queries

(b) Chain Queries

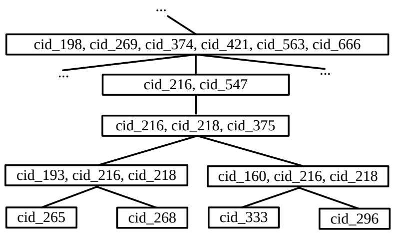
Nasa Problem

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),
...

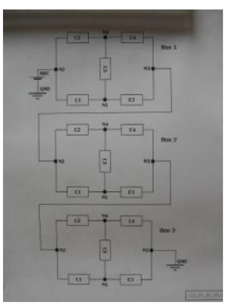
- 680 relations
- 579 variables

Nasa Problem: Hypertree

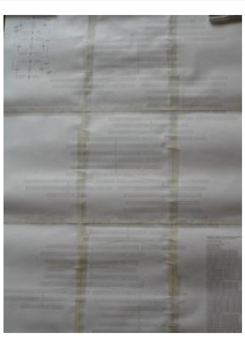


Part of hypertree for the Nasa problem
Best known hypertree-width for the Nasa problem is 22

Electric Circuits



Low hypertree width



Outline of the Tutorial

- (NP-hard) Problems
- Identification of “Easy” Classes
- Beyond Tree Decompositions
- Characterizations of Hypertree Width
- Applications

References

1. I. Adler. Tree-Related Widths of Graphs and Hypergraphs. SIAM Journal Discrete Mathematics, 22(1), pp. 102–123, 2008.
2. A. Atserias, A. Bulatov, and V. Dalmau. On the Power of k-Consistency, In Proc. of ICALP’07, pp. 279–290, 2007.
3. P.A. Bernstein and N. Goodman. The power of natural semijoins. SIAM Journal on Computing, 10(4), pp. 751–771, 1981.
4. A. Bulatov, V. Dalmau, M. Grohe, and D. Marx. Enumerating Homomorphism. In Proc. of STACS’09, pp. 231–242, 2009.
5. H. Chen and V. Dalmau. Beyond Hypertree Width: Decomposition Methods Without Decompositions. In Proc. of CP’05, pp. 167–181, 2005.
6. D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction problems. Journal of Computer and System Sciences, 74(5): 721-43, 2008.
7. R.G. Downey and M.R. Fellows. Parameterized Complexity. Springer, New York, 1999.
8. N. Goodman and O. Shmueli. The tree projection theorem and relational query processing. Journal of Computer and System Sciences, 29(3), pp. 767–786, 1984.
9. G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. Artificial Intelligence, 124(2): 243–282, 2000.
10. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. Journal of Computer and System Sciences, 64(3), pp. 579–627, 2002.
11. G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. J. of Computer and System Sciences, 66(4), pp. 775–808, 2003.
12. G. Gottlob, Z. Mikl´os, and T. Schwentick. Generalized hypertree decompositions: NPhardness and tractable variants. Journal of the ACM, 56(6), 2009.
13. G. Greco and F. Scarcello. The Power of Tree Projections: Local Consistency, Greedy Algorithms, and Larger Islands of Tractability. in Proc. of PODS’10.

References

14. M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In Proc. of STOC’01, pp. 657–666, 2001.
15. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. Journal of the ACM, 54(1), 2007.
16. M. Grohe and D. Marx. Constraint solving via fractional edge covers. In Proc. of SODA’06, pp. 289–298, 2006.
17. D. Marx. Approximating fractional hypertree width. In Proc. of SODA’09, pp. 902–911, 2008. 18. D. Marx. Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. To appear in of STOC’10.
19. N. Robertson and P.D. Seymour. Graph minors III: Planar tree-width. Journal of Combinatorial Theory, Series B, 36, pp. 49-64, 1984.
20. N. Robertson and P.D. Seymour. Graph minors V: Excluding a planar graph. Journal of Combinatorial Theory, Series B, 41, pp. 92-114, 1986.
21. Y. Sagiv and O Shmueli. Solving Queries by Tree Projections. ACM Transaction on Database Systems, 18(3), pp. 487–511, 1993.
22. F. Scarcello, G. Gottlob, and G. Greco. Uniform Constraint Satisfaction Problems and Database Theory. In Complexity of Constraints, LNCS 5250, pp. 156–195, Springer-Verlag, 2008.

Thank you!