Structural Decomposition Methods and Islands of Tractability for NP-hard Problems

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello
Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties
Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width
Outline of PART I

Introduction to Decomposition Methods

Tree Decompositions

Applications of Tree Decompositions
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

PROVABLY EXPONENTIAL
- Theory of the Real Numbers
- Domino Problems

PROVABLY POLYNOMIAL
- Find shortest path in graph
- Linear Programming
Time Complexity

**PROVABLY EXPONENTIAL**
- Theory of the Real Numbers
- Domino Problems

**NP-COMPLETE**
- Graph 3-colorability
- Knapsack
- Traveling Salesman
- Crossword Puzzle
- Satisfiability (SAT)

**PROVABLY POLYNOMIAL**
- Find shortest path in graph
- Linear Programming
Graph Three-colorability

\{\textit{Instance:} \ A graph \( G \). \}

\textit{Question:} \ Is \( G \) 3-colorable?

Examples of instances:
Graph Three-colorability

\[\begin{align*}
\text{Instance:} & \quad \text{A graph } G. \\
\text{Question:} & \quad \text{Is } G \text{ 3-colorable?}
\end{align*}\]

Examples of instances:

\[\text{YES!}\]
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.
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.
Identification of Polynomial Subclasses

- High complexity often arises in “rare” worst case instances
- Worst case instances exhibit intricate structures
- In practice, many input instances have simple structures
- Therefore, our goal is to
  - Define polynomially solvable subclasses (possibly, the largest ones)
  - Prove that membership testing is tractable for these classes
  - Develop efficient algorithms for instances in these classes
Graph and Hypergraph Decompositions

The evil in Computer science is hidden in (vicious) cycles.

We need to get them under control!

Decompositions: Tree-Decomposition, path decompositions, hypertree decompositions,…

Exploit bounded degree of cyclicity.
**Graph Three-colorability**

**Instance:** A graph $G$.

**Question:** Is $G$ 3-colorable?

Examples of instances:
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 ($\rightarrow 3\text{COL}$).

- Moreover, many graph problems are polynomially solvable if restricted to instances of *low cyclicity*.

**How can we measure the degree of cyclicity?**
How much “cyclicity” in this graph?

Suggest a measure of distance from an acyclic graph
Three Early Approaches

Feedback vertex set

Set of vertices whose deletion makes the graph acyclic
The feedback vertex number

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

\[
\text{fwn}(G) = 3
\]
Feedback vertex number

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

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

Pro: For fixed $k$ we can check efficiently whether $fwn(G) \leq k$

What does it mean efficiently when parameter $k$ is fixed?
In many problems there exists some part of the input that are quite small in practical applications.

- Natural parameters
- Many NP-hard problems become easy if we fix such parameters (or we assume they are below some fixed threshold)

Positive examples: $k$-vertex cover, $k$-feedback vertex set, $k$-clique, …

Negative examples: $k$-coloring, $k$-CNF, …
Parameterized Complexity

Initiated by Downey and Fellows, late ‘80s

\[ n = \text{input size} \]

Typical assumption: \( \text{FPT} \neq \text{W}[1] \)
W[1]-hard problems: k-clique

- k-clique is hard w.r.t. fixed parameter complexity!

**INPUT:** A graph $G=(V,E)$

**PARAMETER:** Natural number $k$

Does $G$ have a clique over $k$ vertices?
FPT races

http://fpt.wikidot.com/

<table>
<thead>
<tr>
<th>Problem</th>
<th>f(k)</th>
<th>vertices in kernel</th>
<th>Reference/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex Cover</td>
<td>$1.2738^k$</td>
<td>$2k$</td>
<td>1</td>
</tr>
<tr>
<td>Connected Vertex Cover</td>
<td>$2^k$</td>
<td>no $k^{O(1)}$</td>
<td>26, randomized algorithm</td>
</tr>
<tr>
<td>Multiway Cut</td>
<td>$2^k$</td>
<td>not known</td>
<td>21</td>
</tr>
<tr>
<td>Directed Multiway Cut</td>
<td>$2^k$</td>
<td>no $k^{O(1)}$</td>
<td>34</td>
</tr>
<tr>
<td>Almost-2-SAT (VC-PM)</td>
<td>$4^k$</td>
<td>not known</td>
<td>21</td>
</tr>
<tr>
<td>Multicut</td>
<td>$2^{O(k^2)}$</td>
<td>not known</td>
<td>22</td>
</tr>
<tr>
<td>Pathwidth One Deletion Set</td>
<td>$4.65^k$</td>
<td>$O(k^2)$</td>
<td>28</td>
</tr>
<tr>
<td>Undirected Feedback Vertex Set</td>
<td>$3.83^k$</td>
<td>$4k^2$</td>
<td>2, deterministic algorithm</td>
</tr>
<tr>
<td>Undirected Feedback Vertex Set</td>
<td>$3^k$</td>
<td>$4k^2$</td>
<td>23, randomized algorithm</td>
</tr>
<tr>
<td>Subset Feedback Vertex Set</td>
<td>$2^{O(k \log k)}$</td>
<td>not known</td>
<td>29</td>
</tr>
<tr>
<td>Directed Feedback Vertex Set</td>
<td>$4^k$</td>
<td>not known</td>
<td>27</td>
</tr>
<tr>
<td>Odd Cycle Transversal</td>
<td>$3^k$</td>
<td>$k^{O(1)}$</td>
<td>24, randomized kernel</td>
</tr>
<tr>
<td>Edge Bipartization</td>
<td>$2^k$</td>
<td>$k^{O(1)}$</td>
<td>25, randomized kernel</td>
</tr>
<tr>
<td>Planar Dis</td>
<td>$2^{11.98\sqrt{7}}$</td>
<td>$6^{7/4}k$</td>
<td>3</td>
</tr>
<tr>
<td>1-Sided Crossing Min</td>
<td>$2^{O(\sqrt{k \log k})}$</td>
<td>$O(k^2)$</td>
<td>4</td>
</tr>
<tr>
<td>Max Leaf</td>
<td>$3.72^k$</td>
<td>$3.75k$</td>
<td>5</td>
</tr>
<tr>
<td>Directed Max Leaf</td>
<td>$3.72^k$</td>
<td>$O(k^2)$</td>
<td>6</td>
</tr>
<tr>
<td>Set Splitting</td>
<td>$1.5213^k$</td>
<td>$k$</td>
<td>7</td>
</tr>
<tr>
<td>Nonblocker</td>
<td>$2.5154^k$</td>
<td>$5k/3$</td>
<td>8</td>
</tr>
<tr>
<td>Edge Dominating Set</td>
<td>$2.9147^k$</td>
<td>$2k^2 + 2k$</td>
<td>10</td>
</tr>
<tr>
<td>k-Path</td>
<td>$4^k$</td>
<td>no $k^{O(1)}$</td>
<td>11a, deterministic algorithm</td>
</tr>
<tr>
<td>k-Path</td>
<td>$1.66^k$</td>
<td>no $k^{O(1)}$</td>
<td>11b, randomized algorithm</td>
</tr>
<tr>
<td>Convex Recolouring</td>
<td>$4^k$</td>
<td>$O(k^3)$</td>
<td>12</td>
</tr>
<tr>
<td>VC-max degree 3</td>
<td>$1.1616^k$</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>Clique Cover</td>
<td>$2^k$</td>
<td>$2k$</td>
<td>14</td>
</tr>
<tr>
<td>Clique Partition</td>
<td>$2^{k^2}$</td>
<td>$k^2$</td>
<td>15</td>
</tr>
<tr>
<td>Cluster Editing</td>
<td>$1.62^k$</td>
<td>$2k$</td>
<td>16, weighted and unweighted</td>
</tr>
<tr>
<td>Steiner Tree</td>
<td>$2^k$</td>
<td>no $k^{O(1)}$</td>
<td>17</td>
</tr>
<tr>
<td>3-Hitting Set</td>
<td>$2.077^k$</td>
<td>$O(k^3)$</td>
<td>18</td>
</tr>
</tbody>
</table>
INPUT: A graph $G=(V,E)$

PARAMETER: Natural number $k$

Does $G$ has a feedback vertex set of $k$ vertices?

Naïve algorithm: $O(n^{k+1})$  Not good!

Solvable in $O((2k+1)^kn^2)$ [Downey and Fellows ‘92]

A practical randomized algorithm runs in time: $O(4^kkn)$ [Becker et al 2000]
Feedback vertex number

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

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

**Pro:** For fixed $k$ we can check in quadratic time if $\text{fwn}(G) = k$ \hspace{1cm} (FPT).

**Con:** Very simple graphs can have large FVN:
Feedback edge number → same problem.
Any idea for further techniques?
Well known graph properties:

- A biconnected component is a maximal subgraph that remains connected after deleting any single vertex.
- In any graph, its biconnected components form a tree.
Maximum size of biconnected components

Pro: Actually $bcw(G)$ can be computed in linear time
Drawbacks of BiComp

Maximum size of biconnected components

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

Con: Adding a single edge may have tremendous effects to $bcw(G)$
Maximum size of biconnected components

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

Con: Adding a single edge may have tremendous effects to $bcw(G)$
Can we do better?

Hint:
- why should clusters of vertices be of this limited kind?

Use arbitrary (possibly small) sets of vertices!
- How can we arrange them in some tree-shape?
- What is the key property of tree-like structures (in most applications)?

Information on the rightmost vertex is no longer necessary (quite often)

Information propagation
Can we do better?

Hint:
- why should clusters of vertices be of this limited kind?

Use arbitrary (possibly small) sets of vertices!
- How can we arrange them in some tree-shape?
- What is the key property of tree-like structures, in applications?

Information is still necessary to take decisions about the yellow vertex
Tree Decompositions [Robertson & Seymour ‘86]
Tree Decompositions [Robertson & Seymour ‘86]
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 $h$

- ah
  - ahq
    - hij
    - abc
    - ag
    - cef
    - bcd
  - hkl
    - hkp
    - klo
    - mno
Tree Decompositions and Treewidth

\[
\text{width}(T, X_i) = \max |X_i| - 1 \\
\text{tw}(G) = \min \text{width}(T, X_i)
\]
Playing the Robber & Cops Game
Playing the Robber & Cops Game
Playing the Robber & Cops Game
Playing the Robber & Cops Game
Properties of Treewidth

- $\text{tw}(\text{acyclic graph}) = 1$
- $\text{tw}(\text{cycle}) = 2$
- $\text{tw}(G+v) \leq \text{tw}(G)+1$
- $\text{tw}(G+e) \leq \text{tw}(G)+1$
- $\text{tw}(K_n) = n-1$
- $\text{tw}$ is fixed-parameter tractable (parameter: treewidth)
Outline of PART I

Introduction to Decomposition Methods

Tree Decompositions

Applications of Tree Decompositions
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
   b) Fixed-Parameter tractability: $f(w) \cdot O(n^k)$

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
   b) Prove all yes-instances to have small width
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
      constraint satisfaction = conjunctive database queries
   b) Fixed-Parameter tractability: $f(w) \cdot O(n^k)$
      multicut problem

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
      finding even cycles in graphs – ESO over graphs
   b) Prove all yes-instances to have small width
      the Partner Unit Problem
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
   b) Fixed-Parameter tractability: $f(w) \cdot O(n^k)$

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
   b) Prove all yes-instances to have small width
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
   b) Fixed-Parameter tractability: $f(w) \cdot O(n^k)$

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
   b) Prove all yes-instances to have small width
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 linear 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 linear 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 ∈ MSO.
(\exists R, G, B) \left[ (\forall x (R(x) \lor G(x) \lor B(x))) \right.
\left. \land (\forall x(R(x) \Rightarrow (\neg G(x) \land \neg B(x)))) \right.
\left. \land \ldots \right.
\left. \land \ldots \right.
\left. \land (\forall x, y(E(x, y) \Rightarrow (R(x) \Rightarrow (G(x) \lor B(y))))) \right.
\left. \land (\forall x, y(E(x, y) \Rightarrow (G(x) \Rightarrow (R(x) \lor B(y))))) \right.
\left. \land (\forall x, y(E(x, y) \Rightarrow (B(x) \Rightarrow (R(x) \lor G(y))))) \right]
Courcelle's Theorem: Problems expressible in MSO$^2$ are solvable in linear time on structures of bounded treewidth

...and in LOGSPACE [Elberfeld, Jacoby, Tantau]

Example – Graph Coloring

$\exists P \forall x \forall y : (E(x,y) \rightarrow (P(x) \neq P(y)))$
Arnborg, Lagergren, Seese '91:

Optimization version of Courcelle's Theorem:
Finding an optimal set $P$ such that $G \models \Phi(P)$ is FP-linear over inputs $G$ of bounded treewidth.

Example:

Given a graph $G=(V,E)$

Find a smallest $P$ such that

$$\forall x \forall y : (E(x,y) \Rightarrow (P(x) \neq P(y)))$$
Find minimum-cardinality vertex set separating $S_i$ from $T_j$ for each tuple $<S_i,T_j>$ in relation $H$.
Unrestricted Vertex Multicut Problems

H:

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>T1</th>
<th>S2</th>
<th>T3</th>
<th>S2</th>
<th>T2</th>
</tr>
</thead>
</table>

© Seffi Naor
Unrestricted Vertex Multicut Problems

Results

[Guo et al. 06] UVMC FPT if $|S|$, $|C|$ and tree-width fixed

[G. & Tien Lee] UVMC FPT if overall structure has bounded tw. using master theorem by Arnborg, Lagergren and Seese.
PROOF

**Definition 8.** On structures $\mathcal{A} = (V, E, H)$ as above, let $\text{connects}(S, x, y)$ be defined as follows:

$$S(x) \land S(y) \land \forall P\left( (P(x) \land \neg P(y)) \rightarrow (\exists v \exists w (S(v) \land S(w) \land P(v) \land \neg P(w) \land E(v, w))) \right).$$

$$uvmc(X) \equiv \forall x \forall y \left( H(x, y) \rightarrow \forall S (\text{connects}(S, x, y) \rightarrow \exists v (X(v) \land S(v))) \right)$$

Minimize $X$ in $uvmc$

$X$ intersects each set that connects $x$ and $y$
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
   b) Fixed-Parameter tractability: $f(w) \cdot O(n^k)$

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
   b) Prove all yes-instances to have small width
The Generalized Even Cycle Problem

**INPUT:** A graph G, a constant k.

**QUESTION:** Decide whether G has a cycle of length 0 (mod k)

In the past century, this was an open problem for a long time.

Carsten Thomassen in 1988 proved it polynomial for all graphs using treewidth as a tool.
Proof Idea

**Small Treewidth ($\leq c$)**

"cycle of length 0 (mod k)" can be expressed un MSO

example

- k=4

→ Courcelle's Theorem

(but was not known then…)

**Large Treewidth ($> c$)**

∀k ∃c: each graph $G$ with $\text{tw}(G) > c$ contains a subdivision of the $f(k)$-grid. [for suitable $f$]

∀n > $f(k)$, each subdivision of $f(k)$-grid contains a cycle of length 0 (mod k).
Determine the complexity of SO fragments over finite structures.

Finite structures: words (strings), graphs, relational databases

Known: SO=PH; ESO = NP

Which SO-fragments can be evaluated in polynomial time?

Which SO-fragments express regular languages on strings?

More modestly: What about prefix classes?
Every room should be equipped with a computer.

If a printer is not present in a room, then one should be available in an adjacent room.

No room with a printer should be a meeting room.

Every room is at most 5 rooms distant from a meeting room.
Given an office layout as a graph, decide whether the facility placement constraints are satisfiable.

\[ \exists P \exists M \ldots \forall x \exists y ((P(x) \lor E(x,y) \land P(y)) \land \ldots \]

Observe that this is an \( E_1^{*ae} \) formula

This leads to the questions:

Are formulas of the type \( E_1^{*ae} \) or even \( E^{*ae} \) polynomially verifiable over graphs?

What about other fragments of ESO or SO?
This motivates the following question:

Can formulas in classes such as $E_2(ae_2)$ or even $ESO(e^*ae^*)$ be evaluated in polynomial time over strings?

More generally:

Which ESO-fragments admit polynomial-time model checking over strings?

A similar, even more important question can be asked for graphs and general finite structures:

Which ESO-fragments admit polynomial-time model checking over graphs or arbitrary finite structures?
Directed graphs (or undirected graphs with self-loops):

Undirected graphs w/o self-loops:
The Saturation Problem

Pattern graph P1

Graph G

Saturation of G via P1:

In PTIME!
The Saturation Problem

Relating $E_{1}^{*}ae$ to the Saturation Problem

Pattern graph P2

Graph G

Saturation of $G$ via P2 impossible!

No cycle of length 0 (mod 4) in $G$. 
The Saturation Problem

Pattern graph P1

Graph G

Saturation of G via P1:
Relating $E_1^*ae$ to the Saturation Problem

$$\exists \quad P_1, P_2 \forall x \exists y$$

$$[(E(x, y) \land P_1(x) \land P_2(x) \land P_1(y) \land \neg P_2(y)) \lor$$

$$\quad (E(x, y) \land P_1(x) \land \neg P_2(x) \land \neg P_1(y) \land \neg P_2(y)) \lor$$

$$\quad (\neg E(x, y) \land \neg P_1(x) \land \neg P_2(x) \land P_1(y) \land P_2(y))]$$
1. Prove Tractability of bounded-width instances
   a) Genuine tractability: $O(n^{f(w)})$-bounds
   b) Fixed-Parameter tractability: $f(w) \times O(n^k)$

2. Tool for proving general tractability
   a) Prove tractability for both large & small width
   b) Prove all yes-instances to have small width
Partner Units Scenario

- Track People in Buildings
- Sensors on Doors, Rooms Grouped into Zones
Partner Units Solution

- Assigning Sensors and Zones to Control Units
- Respect Adjacency Constraints
The Partner-Unit Problem

Bipartite graph $G=(V,E) \ V = V_a \cup V_b$;
$V_a = \{a_1, \ldots, a_r\}$,
$V_b = \{b_1, \ldots, b_s\}$,
$E$: edges btw. $V_a$ and $V_b$
The Partner-Unit Problem

Replace connections by connections to units
The Partner-Unit Problem

Bipartite graph $G=(V,E)$ $V=V_a \cup V_b$; $V_a = \{a_1, \ldots, a_r\}$, $V_b = \{b_1, \ldots, b_s\}$, $E$: edges btw. $V_a$ and $V_b$

Replace connections by connections to units

OR
The Partner-Unit Problem

Bipartite graph $G=(V,E)$ $V=V_a \cup V_b$; $V_a=\{a_1,\ldots,a_r\}$, $V_b=\{b_1,\ldots,b_s\}$, $E$: edges btw. $V_a$ and $V_b$

Replace connections by connections to units

OR

$ai \rightarrow u \leftarrow bj$
The Partner-Unit Problem

CONSTRAINTS:
- Each $a_i$ or $b_i$ is connected to exactly 1 unit.
- Each unit connected to:
  - at most 2 other units,
  - at most 2 elements from $V_a$,
  - at most 2 elements from $V_b$,
- If $a_i$ connected to $b_j$ in $G$, then $\text{dist}(a_i,b_i) \leq 3$ in $G^*$

$U=\{u_1,u_2,u_3,u_4\}$
Assume one node $a$ is connected to 7 nodes $b_1, \ldots, b_7$ in $G$. Then instance $G$ is unsolvable.

Thus, no vertex can have more than 6 neighbours in $G$. 
The PU Problem(s)

- **PU DECISION PROBLEM (PUDP):**
  Given G, is there a G* satisfying the constraints? (Number of units irrelevant.)

- **PU SEARCH PROBLEM (PUSP)**
  Given G, find a suitable G* whenever possible.

- **PU OPTIMIZATION PROBLEM (PUOP)**
  Given G, find a suitable G* with minimum number of units |U| (whenever possible).
ASSUMPTION: G is connected.

Note: This assumption can be made wlog, because the PUDP can be otherwise decomposed into a conjunction of independent PUDPs, one for each component.

Lemma 1: If G is connected and solvable, then there exists a solution G* in which the unit-graph UG=G*[U] is connected.
**Lemma 2:** If $G$ is connected and solvable, then there exists a solution $G^*$ whose unit graph is a cycle.

**Note:** We still don’t know $|U|$, but we may just try all cycles of length $\max(|Va|,|Vb|)/2$ to length $|Va|+|Vb|$. There are only linearly many! (Guessable in logspace)
Theorem:
Assume $G$ is solvable through solution $G^*$ with $|U|=n$ and having
unit function $f$. Then:

1. $pw(G) \leq 11$
2. $tw(G) \leq 5$
3. There is a path decomposition $T=(W,A)$ that can be locally check to witness PUDP solution $G^*$
Example

$G^*$

$U = \{u_1, u_2, u_3, u_4\}$

$T$

$T_1: a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4$

$T_2: a_1, a_2, a_3, a_4, a_5, b_1, b_2, b_3, b_4, b_5, b_6$

$T_3: a_1, a_2, a_5, a_6, b_1, b_2, b_3, b_5, b_6, b_7$
Note: We cannot do better, thus the bound 11 is actually tight!
We now show (2)
Strip off the $V_b$-elements and put them into separate bags.

Note: Other examples show, we cannot do better, thus the bound 5 is actually tight.
Example for lower bound 5

\[ \text{tw} = 5 \]
... and this G is actually solvable:
Theorem : PUDP is in polynomial time and is solvable by dynamic programming techniques.

QED
## Partner Units Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Sensors</th>
<th>Zones</th>
<th>Edges</th>
<th>Cost</th>
<th>CSP</th>
<th>DECPUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbl-20</td>
<td>28</td>
<td>20</td>
<td>56</td>
<td>14</td>
<td>*</td>
<td>0.01</td>
</tr>
<tr>
<td>dbl-40</td>
<td>58</td>
<td>40</td>
<td>116</td>
<td>29</td>
<td>*</td>
<td>0.05</td>
</tr>
<tr>
<td>dbl-60</td>
<td>88</td>
<td>60</td>
<td>176</td>
<td>44</td>
<td>*</td>
<td>0.08</td>
</tr>
<tr>
<td>dblv-30</td>
<td>28</td>
<td>30</td>
<td>92</td>
<td>15</td>
<td>*</td>
<td>65.49</td>
</tr>
<tr>
<td>dblv-60</td>
<td>58</td>
<td>60</td>
<td>192</td>
<td>30</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>triple-30</td>
<td>40</td>
<td>30</td>
<td>78</td>
<td>20</td>
<td>*</td>
<td>0.50</td>
</tr>
<tr>
<td>triple-34</td>
<td>40</td>
<td>34</td>
<td>93</td>
<td>/</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>grid-90</td>
<td>50</td>
<td>68</td>
<td>97</td>
<td>34</td>
<td>*</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Case N>2

For constant N totally open. Could well be NP-hard.
In fact, Unit Graph does not need to have bounded treewidth!

If N is not-constant, then NP-complete:

For Siemens, it seems that very small values of N are relevant.
Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties
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…
Database queries

Database schema (scopes):
- Enrolled (Pers#, Course, Reg-Date)
- Teaches (Pers#, Course, Assigned)
- Parent (Pers1, Pers2)

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

\[ \text{ans} \leftarrow \text{Enrolled}(S,C,R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]
QUERY: Is there any teacher having a child enrolled in her course?

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

\[ \text{Ans} \leftarrow \text{Enrolled}(S,C,R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]
Database schema (scopes):
- *Enrolled* (Pers#, Course, Reg-Date)
- *Teaches* (Pers#, Course, Assigned)
- *Parent* (Pers1, Pers2)

Is there any teacher whose child attend some course?

\[ \text{Ans} \leftarrow \text{Enrolled}(S,C',R) \land \text{Teaches}(P,C,A) \land \text{Parent}(P,S) \]
A more intricate query

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

Crossword puzzle

1h: PARIS
    PANDA
    LAURA
    ANITA

1v: LIMBO
    LINGO
    PETRA
    PAMPA
    PETER

and so on
Constraint Satisfaction Problems

- Set of variables \{X_1, \ldots, X_{26}\}
- Set of constraint scopes

- Set of constraint relations

- \(r_{1h}(X_1, X_2, X_3, X_4, X_5)\)
- \(r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})\)

- Paris PANDA Laura Anita

- Limbo LINGO Petra Pampa Peter
Problems on Electric Circuits

[Diagram of electrical circuits with labeled nodes and connections]

- Box 1: Nodes N1, N2, N3, N4
- Box 2: Nodes N1, N2, N3, N4
- Box 3: Nodes N1, N2, N3, N4

Connections and references...

- SRC: Source
- GND: Ground

- Problem statements and conditions...
A problem from Nasa

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 constraints
- 579 variables
Configuration problems (Renault example)

- Renault Megane configuration
  [Amilhastre, Fargier, Marquis AIJ, 2002]
  Used in CSP competitions and as a benchmark problem

- Variables encode type of engine, country, options like air cooling, etc.

- 99 variables with domains ranging from 2 to 43.

- 858 constraints, which can be compressed to 113 constraints.

- The maximum arity is 10 (hyperedge cardinality/size of constraint scopes)

- Represented as extensive relations, the 113 constraints comprise about 200 000 tuples

- $2.84 \times 10^{12}$ solutions.
Further examples...

In the third part
Representing Hypergraphs via Graphs

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.
Further Graph Representations

- **Dual Graph**: Seidel, 81
- **Incidence Graph**: Seidel, 81 (Hidden variable encoding)
- **Decther, 92**:
α-acyclic Hypergraphs

Note the connectedness condition for α

Acyclic hypergraphs may contain cycles
Again on the simplest query

\[ \text{Ans} \iff \text{Enrolled}(S, C', R) \land \text{Teaches}(P, C, A) \land \text{Parent}(P, S) \]

\[ \alpha \text{-acyclic hypergraph} \quad \text{Join Tree} \]
Deciding Hypergraph Acyclicity

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

[Yu and Özsoyoğlu, IEEE Compsac’79; see also Graham, Tech Rep’79]

---

*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 (α-)acyclic iff the resulting hypergraph empty*

*Proof*: Easy by considering leaves of join tree
Example of GYO-Reduction

\[ H^* = (\emptyset, \emptyset) \]

GYO reduct
Example of GYO-irreducible Hypergraph
Tree decompositions as Join trees

- Tree decomposition as a way of clustering vertices to obtain a join tree (acyclic hypergraph)
- Implicitly defines an equivalent acyclic instance
The “degree of cyclcity” is the treewidth (maximum number of vertices in a cluster -1)

In this example, the treewidth is 2

That’s ok! We started with a cyclic graph…
Not good for hypergraph-based problems

Here the input instance is acyclic (hence, easy)

However, its treewidth is 2!
(similar troubles for all graph representations)
A different notion of “width”

- Exploit the fact that a single hyperedge covers many vertices
- Degree of cyclicity: maximum number of hyperedges needed to cover every cluster

Input: acyclic instance
One hyperedge covers each cluster: width 1
Generalizing acyclicity and treewidth

- Tree decomposition as a way of clustering vertices to obtain a join tree (acyclic hypergraph)
- Implicitly defines an equivalent acyclic instance
- Width of a decomposition: maximum number of hyperedges needed to cover each bag of the tree decomposition

**Generalized Hypertree Width** (ghw): minimum width over all possible decompositions [Gottlob, Leone, Scarcello, JCSS’03]
- also known as (acyclic) cover width

Generalizes both acyclicity and treewidth:
- Acyclic hypergraphs are precisely those having ghw = 1
- The “covering power” of a hyperedge is always greater than the covering power of a vertex (used in the treewidth)
Tree Decomposition of a Hypergraph

H

Tree decomp of G(H)

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
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag

\[ \{1,11,17,19\} \]

\[ \{1,2,3,4,5,6\} \]
\[ \{11,12,17,18,19\} \]

\[ \{3,4,5,6,7,8\} \]
\[ \{12,16,17,18,19\} \]

\[ \{5,6,7,8,9\} \]
\[ \{12,15,16,18,19\} \]

\[ \{7,9,10\} \]
\[ \{12,13,14,15,18,19\} \]
2 hyperedges suffice for each bag
2 hyperedges suffice for each bag
Generalized Hypertree Decomposition

Notation:
- label decomposition vertices by hyperedges
- omit hyperedge elements not used for bag covering (hidden elements are replaced by “_”)

Generalized hypetree decomposition of width 2
Generalized Hypertree Decompositions

\[ a(S, X, X', C, F) \quad b(S, Y, Y', C', F') \quad c(C, C', Z) \quad d(X, Z) \]
\[ e(Y, Z) \quad f(F, F', Z') \quad g(X', Z') \quad h(Y', Z') \]
\[ j(J, X, Y, X', Y') \quad p(B, X', F) \quad q(B', X', F) \]
Basic Conditions (1/3)

Original (direct) definition

We group edges

We group edges
Basic Conditions\(^{(2/3)}\)

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

*Edges can partially be used*
Connectedness Condition \(3/3\)

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

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

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

\[ d(X, Z) \]
\[ e(Y, Z) \]

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

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

\[ p(B, X', F) \]
\[ q(B', X', F) \]
Can we determine in polynomial time whether $\text{ghw}(H) < k$ for constant $k$?
Can we determine in polynomial time whether $\text{ghw}(H) < k$ for constant $k$?

Bad news: $\text{ghw}(H) < 4$? NP-complete

[Gottlob, Miklós, and Schwentick, J.ACM‘09]
Hypertree Decomposition (HTD)

HTD = Generalized HTD + Special Condition

[Gottlob, Leone, Scarcello, PODS'99; JCSS'02]

Does not appear in the subtrees rooted at \( v \)
Special Condition

Each variable not used at some vertex $v$

$\text{j}(J,X,Y,X',Y')$

$\text{a}(S,X,X',C,F), \text{b}(S,Y,Y',C',F')$

$\text{j}(_,X,Y,_,_), \text{c}(C,C',Z)$

$\text{j}(LXY,Y',X'), \text{f}(F,F',Z')$

$\text{d}(X,Z)$

$\text{e}(Y,Z)$

$\text{g}(X',Z'), \text{f}(F,_,Z')$

$\text{h}(Y',Z')$

$\text{p}(B,X',F)$

$\text{q}(B',X',F)$

Does not appear in the subtrees rooted at $v$
Thus, e.g., all available variables in the root must be used.
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 (\textsc{LOGCFL})
- Computing hypertree decompositions is feasible in polynomial time (for fixed $k$).

But: \text{FP-intractable wrt } k: \text{ \textsc{W[2]}-hard.}
Relationship  GHW vs HW

Observation:
\[ \text{ghw}(H) = \text{hw}(H^*) \]

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

Approximation Theorem [Adler,Gottlob,Grohe,05]:
\[ \text{ghw}(H) \leq 3\text{hw}(H)+1 \]

GHW and HW identify the same set of classes having bounded width
Game Characterization: Robber and Marshals
Marshals block hyperedges
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
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 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
\[ \text{ans} \leftarrow a(S, X, T, R) \land b(S, Y, U, P) \land c(T, U, Z) \land e(Y, Z) \land g(X, Y) \land f(R, P, V) \land d(W, X, Z) \]
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

\[ a(S, X, T, R), \ b(S, Y, U, P) \]

\[ f(R, P, V) \]
The marshals corner the robber

\[ a(S, X, T, R), \ b(S, Y, U, P) \]

\[ f(R, P, V) \]

\[ g(X, Y), \ c(T, Z, U) \]
The capture

\[ a(S,X,T,R), b(S,Y,U,P) \]

\[ f(R,P,V) \]
\[ g(X,Y), c(T,Z,U) \]
\[ g(X,Y), d(W,X,Z) \]
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.

[Gottlob, Leone, Scarcello, PODS’01, JCSS’03]
A Useful Tool: Alternating Turing Machines

- Generalization of non-deterministic Turing machines
- There are two special states: and
- Acceptation: Computation tree
- $\text{ALOGSPACE} = \text{PTIME}$
ATMs and LOGCFL

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

\[ \text{AC}_0 \subseteq \text{NL} \subseteq \text{LOGCFL} = \text{SAC}_1 \subseteq \text{AC}_1 \subseteq \text{NC}_2 \subseteq \cdots \subseteq \text{NC} = \text{AC} \subseteq \text{P} \subseteq \text{NP} \]

Characterization of LOGCFL [Ruzzo ‘80]:

LOGCFL = Class of all problems solvable with a logspace ATM with polynomial tree-size
Coming back to Marshals...
A polynomial algorithm: ALOGSPACE
Actually, LOGCFL

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

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

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

- LOGSPACE checkable
- Polynomial proof-tree
Outline of PART II

Beyond Tree Decompositions

Applications to Databases and CSPs

Structural and Consistency Properties
Some hypergraph based 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.

[e.g., Kolaitis & Vardi, JCSS’98]
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 \( B \)

\[ h : U \longrightarrow V \]

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

\[ x \in R_i \implies h(x) \in S_i \]
Example: graph colorability

\[ \begin{array}{c|c}
1 & 2 \\
1 & 3 \\
2 & 3 \\
3 & 4 \\
2 & 5 \\
4 & 5 \\
3 & 6 \\
\end{array} \]

\[ \begin{array}{c|c}
\text{red} & \text{green} \\
\text{red} & \text{blue} \\
\text{green} & \text{red} \\
\text{green} & \text{blue} \\
\text{blue} & \text{red} \\
\text{blue} & \text{green} \\
\end{array} \]
Example: graph colorability

A homomorphism $h$ maps the vertices of graph $A$ to the vertices of graph $B$ such that if two vertices are connected in $A$, their images are also connected in $B$. The table illustrates the mapping of colors:

<table>
<thead>
<tr>
<th>A Vertex</th>
<th>B Vertex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>green</td>
</tr>
<tr>
<td>3</td>
<td>red</td>
</tr>
<tr>
<td>4</td>
<td>blue</td>
</tr>
<tr>
<td>5</td>
<td>green</td>
</tr>
<tr>
<td>6</td>
<td>blue</td>
</tr>
</tbody>
</table>

The colors assigned to each vertex in graph $B$ are as follows:

- Vertex 1 is red
- Vertex 2 is green
- Vertex 3 is red
- Vertex 4 is blue
- Vertex 5 is green
- Vertex 6 is blue
Complexity: HOM is NP-complete

(well-known, independently proved in various contexts)

Membership: Obvious, guess $h$.

Hardness: Transformation from 3COL.

Graph 3-colourable if and only if $\text{HOM}(A,B)$ is a yes-instance.
Conjunctive Database Queries

DATABASE:

Enrolled

- John: Algebra, 2003
- Robert: Logic, 2003
- Mary: DB, 2002
- Lisa: DB, 2003

Teaches

- McLane: Algebra, March
- Verdi: Logic, May
- Lausen: DB, June
- Rahm: DB, May

Parent

- McLane: Lisa
- Verdi: Robert
- Rahm: Mary

QUERY:

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

ans ← Enrolled(S,C,R) ⋀ Teaches(P,C,A) ⋀ Parent(P,S)
**Conjunctive Database Queries**

**DATABASE:**

<table>
<thead>
<tr>
<th>Enrolled</th>
<th>Teaches</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>McLane</td>
<td>McLane</td>
</tr>
<tr>
<td>Algebra</td>
<td>Algebra</td>
<td>Lisa</td>
</tr>
<tr>
<td>Robert</td>
<td>Verdi</td>
<td>Verdi</td>
</tr>
<tr>
<td>Logic</td>
<td>Logic</td>
<td>Robert</td>
</tr>
<tr>
<td>Mary</td>
<td>Lausen</td>
<td>Rahm</td>
</tr>
<tr>
<td>DB</td>
<td>DB</td>
<td>Mary</td>
</tr>
<tr>
<td>2002</td>
<td>June</td>
<td></td>
</tr>
<tr>
<td>Lisa</td>
<td>Rahm</td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>May</td>
<td></td>
</tr>
</tbody>
</table>

... ...

**Enrolled** \((S,C,R)\), **Teaches** \((P,C,A)\), **Parent** \((P,S)\)

*homomorphism*
CSPs as Homomorphism Problems

- Set of variables \{X_1, \ldots, X_{26}\}
- Set of constraint scopes

\[ r_{1h}(X_1, X_2, X_3, X_4, X_5) \]

1 2 3 4 5 6
7
8 9 10
11 12 13 14 15
16 17 18 19
20 21 22 23 24 25 26

\[ r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20}) \]

Paris, Panda, Laura, Anita
Limbo, Lingo, Petra, Pampa, Peter

- Set of (finite) constraint relations
CSPs as Homomorphism Problems

\[ r_{1h}(X_1, X_2, X_3, X_4, X_5) \]

\[ r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20}) \]

\( r_{1h}: \) PARIS PANDA LAURA ANITA

\( r_{1v}: \) LIMBO LINGO PETRA PAMPA PETER

\( \ell \)-structure \( \mathbb{A} \)

\( r \)-structure \( \mathbb{B} \)
CSPs as Homomorphism Problems

$$r_{1h}(X_1, X_2, X_3, X_4, X_5)$$

$$r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20})$$

$$\ell\text{-structure } A$$

$$r\text{-structure } B$$

homomorphism
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other
Endomorphisms and cores

- Sometimes the two structures coincide
- Core: minimal substructure to which there is an endomorphism
- Cores are isomorphic to each other
Cores and equivalent instances

- Can be used to simplify problems
- There is a homomorphism from \( A \) to \( B \) if and only if there is a homomorphism from a/any core of \( A \) to \( B \)
- Sometimes terrific simplifications:

This undirected grid is equivalent to a single edge. That is, it is equivalent to an acyclic instance!
Structurally Restricted CSPs

\[ H_A \]
Structurally Restricted CSPs

The hypergraph is acyclic
We have seen that Acyclicity is efficiently recognizable.

We shall see that Acyclic CSPs can be efficiently solved.
Basic Question

INPUT: CSP instance \((A, B)\)

Is there a homomorphism from \(A\) to \(B\)?
Basic Question (on Acyclic Instances)

**INPUT:** CSP instance \((A, B)\)

- Is there a homomorphism from \(A\) to \(B\)?

- Feasible in polynomial time \(O(n^2 \times \log n)\)
- LOGCFL-complete
Basic Question (on Acyclic Instances)

**INPUT:** CSP instance \((A, B)\)

- Is there a homomorphism from \(A\) to \(B\)?

- Feasible in polynomial time \(O(n^2 \times \log n)\)

- LOGCFL-complete

[Yannakakis, VLDB’81]
Basic Question (on Acyclic Instances)

**INPUT:** CSP instance \((A, B)\)

Is there a homomorphism from \(A\) to \(B\) ?

Feasible in polynomial time \(O(n^2 \times \log n)\)

LOGCFL-complete

---

[Gottlob, Leone, Scarcello, J.ACM’00]
A Polynomial-time Algorithm

HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Yannakakis’s Algorithm (Acyclic structures):

• Dynamic Programming over a Join Tree, where each vertex contains the relation associated with the corresponding hyperedge

• Therefore, if there are more constraints over the same relation, it may occur (as a copy) at different vertices
\[ d(Y, P) \]
\[ r(Y, Z, U) \]
\[ s(Z, U, W) \]
\[ t(V, Z) \]
\[ d(Y, P) \]

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

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

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

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

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

\[t(V, Z)\]
\[ d(Y, P) \]
\[ r(Y, Z, U) \]
\[ s(Z, U, W) \]
\[ t(V, Z) \]
\[ d(Y, P) \]

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

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

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

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

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

\[t(V, Z)\]
HOM: The homomorphism problem

BCQ: Boolean conjunctive query evaluation

CSP: Constraint satisfaction problem

Yannakakis’s Algorithm (Acyclic structures):
Dynamic Programming over a Join Tree

Solutions can be computed by adding a top-down phase to Yannakakis’ algorithm for acyclic instances
A solution: Y=3, P=7, Z=8, U=9, W=4, V=9
The result size can be exponential (even in the acyclic case).

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

In case of acyclic instances, the result can be computed in time polynomial in the result size (and **with polynomial delay**: first solution, if any, in polynomial time, and each subsequent solution within polynomial time from the previous one).

This will remain true for the subsequent generalizations of acyclicitiy.

Add a top-down phase to Yannakakis’ algorithm for acyclic instances, thus obtaining a full reducer, and join the partial results (or perform a backtrack free visit)
Decomposition Methods

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ H_A \]
Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition
Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition
(Generalized) Hypertree Decompositions

Transform the hypergraph into an acyclic one:
- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition

Each cluster can be seen as a subproblem
(Generalized) Hypertree Decompositions

Each cluster can be seen as a subproblem.

Relations:

\{1,2,3,4,5,20,21,22,23,24,25,26\} \{1H,20H\}

\{1,7,11,16,20,22\} \{1V,20H\}

\{11,12,13,17,22\} \{11H,13V\}

\{5,8,14,18,24,26\} \{5V,20H\}

\{8,9,10,6,15,19,26\} \{8H,6V\}

Relations:

\{1V,20H\} = 1V \bowtie 20H
Toward an equivalent acyclic instance

- Each cluster can be seen as a subproblem
- Associate each subproblem with a fresh constraint
Toward an equivalent acyclic instance

- Each cluster can be seen as a **subproblem**
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)
The structure of the equivalent instance

- Each cluster can be seen as a **subproblem**
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)

A join tree of the new instance
An acyclic equivalent instance

- Each cluster can be seen as a subproblem
- Compute solutions for subproblems (exponential dependency on the width)
- Associate each subproblem with a fresh constraint
- Get an equivalent problem (all original constraints are there…)

Solve the acyclic instance with any known technique
Tree Projection (idea)

- Generalization where suproblems are arbitrary (not necessarily clusters of k edges or vertices)

Structure

Sandwich acyclic hypergraph (Tree Projection)

Available Subproblems

More information in the appendix
Hypertrees for Databases

Weighted HDs, which exploit quantitative data, too.
Some experiments

(a) Acyclic Queries

(b) Chain Queries
Large width example: 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

Best known hypertree-width for the Nasa problem is 22
Further Structural Methods

Many proposals in the literature, besides (generalized) hypertree width (see [Gottlob, Leone, Scarcello. Art. Int.’00])

For the binary case, the method based on tree decompositions (first proposed as heuristics in [Dechter and Pearl. Art.Int.’88 and Art.Int.’89]) is the most powerful [Grohe. J.ACM’07]

Let us recall some recent proposals for the general (non-binary) case:
- Fractional hypertree width [Grohe and Marx. SODA’06]
- Spread-cut decompositions [Cohen, Jeavons, and Gyssens. J.CSS’08]
- Component Decompositions [Gottlob, Miklòs, and Schwentick. J.ACM’09]
- Greedy tree projections [Greco and Scarcello, PODS’10, ArXiv’12]

Computing a width-k decomposition is in PTIME for all of them (for any fixed k>0).

If we relax the above requirement, we can consider fixed-parameter tractable methods. If the size of the hypergraph structure is the fixed parameter, the most powerful is the Submodular width [Marx. STOC’10]
1. Computing decompositions
   • Heuristics to get variants of (hyper)tree decompositions

2. Evaluating instances
   • Computing all solutions of the subproblems involved at each node may be prohibitive
   • Memory explosion

Solution: combine with other techniques. E.g., in CSPs,
   use (hyper)tree decompositions for bounding the search space [Otten and Dechter. UAI‘08]
   use (hyper)tree decompositions for improving the performance of consistency algorithms (hence, speeding-up propagations) [Karakashian, Woodward, and Choueiry. AAAI’13]
   …
Alternative constraint encodings

Most results hold on constraint encodings where allowed tuples are listed as finite relations

Alternative encodings make sense

For instance,

- constraint satisfaction with succinctly specified relations
  [Chen and Grohe. J.CSS’10]
- see also [Cohen, Green, and Houghton. CP’09]
Local (pairwise) consistency

- For every relation/constraint:
  each tuple matches some tuple in every other relation
- Can be enforced in polynomial time:
  take the join of all pairs of relations/constraints until a fixpoint is reached, or some relation becomes empty
Enforcing pairwise consistency
Enforcing local consistency

d(Y,P)

s(Z,U,W)

t(V,Z)

r(Y,Z,U)

\[
\begin{align*}
&d: 3 & 8 \\
& & 3 & 7 \\
& & 5 & 7 \\
& & 6 & 7 \\
&s: 3 & 8 & 9 \\
& & 9 & 3 & 8 \\
& & 8 & 3 & 8 \\
& & 3 & 8 & 4 \\
& & 3 & 8 & 3 \\
& & 8 & 9 & 4 \\
& & 9 & 4 & 7 \\
& & 9 & 8 \\
& & 9 & 3 \\
& & 9 & 5 \\
\end{align*}
\]
Enforcing local consistency
Enforcing pairwise consistency

Further steps are useless, because the instance is now locally consistent.

On acyclic instances, same result as Yannakakis’ algorithm on the join tree!

\[ d(Y,P) \quad r(Y,Z,U) \quad s(Z,U,W) \quad t(V,Z) \]

\[
\begin{array}{c}
\text{(3 8 9)} \\
\text{(9 3 8)} \\
\text{(9 5 7)} \\
\text{(6 7 4)}
\end{array}
\]

\[
\begin{array}{c}
\text{(3 8 9)} \\
\text{(9 3 8)} \\
\text{(8 3 9)} \\
\text{(6 7 4)}
\end{array}
\]
Computing a join tree (in linear time, and logspace-complete [GLS'98+ SL=L]) may be viewed as a clever way to enforce pairwise consistency.

Cost for the computation of the full reducer:  
\[ \text{O}(m \ n^2 \ \log n) \text{ vs } \text{O}(m \ n \ \log n) \]

N.B. \( n \) is the (maximum) number of tuples in a relation and may be very large (esp. in database applications)
Global and pairwise Consistency

Yannakakis’ algorithm actually solves acyclic instances because of their following crucial property:

- Local (pairwise) consistency $\Rightarrow$ Global consistency [Beeri, Fagin, Maier, and Yannakakis. J.ACM’83]

- Global consistency: Every tuple in each relation can be extended to a full (global) solution

- In particular, if all relations/constraints are pairwise consistent, then the result is not empty

Not true in the general case:

$\text{ans}:- \ a(X, Y) \land b(Y, Z) \land c(Z, X)$
Huge number of works in the database and constraint satisfaction literature about different kinds (and levels) of consistencies (e.g., recall the seminal paper [Mackworth. Art. Int., 1977] or [Beeri, Fagin, Maier, and Yannakakis. J.ACM’83] and [Dechter and van Beek. TCS’97])

Most theoretical papers in the database community

Also practical papers in the constraint satisfaction community:
- Local consistencies are crucial for filtering domains and constraints
- Allow tremendous speed-up in constraint solvers
- Sometimes allow backtrack-free computations
Global consistency (GC): Every tuple in each relation can be extended to a full (global) solution
[Beeri, Fagin, Maier, and Yannakakis. J.ACM’83]

For instances with $m$ constraints, it is also known as
- m-wise consistency [Gyssens. TODS’86]
- relational (i;m)-consistency [Dechter and van Beek. TCS’97]
- $R^{(*)}_m$ [Karakashian, Woodward, Reeson, Choueiry and Bessiere. AAAI’10]

... 

Remark:
In the CSP literature, “global consistent network” sometimes means “strongly n-consistent network”, which is a different notion (see, e.g., [Constraint Processing, Dechter, 2003]).
If an instance is globally consistent, we can immediately read partial solutions from the constraint/database relations. Full solutions are often computed efficiently and can be exploited in heuristics by constraint solvers. For a very recent example, see [Karakashian, Woodward, and Choueiry. AAAI’13]: enforce global consistency on groups of subproblems (tree-like arranged) for bolstering propagations.
When pairwise consistency entails GC

- We have seen that it happens in acyclic instances…
- Is it the case that this condition is also necessary?
- What is the real power of local consistency?
  i.e., relational **arc-consistency** (more precisely, arc-consistency on the dual graph)
  
  Also known as
  - pairwise consistency [Janssen, Jégou, Nougier, and Vilarem. IEEE WS Tools for AI’89],
  - 2-wise consistency [Gyssens. TODS’86],
  - R(*,2)C [Karakashian, Woodward, Reeson, Choueiry and Bessiere. AAAI’10]
  - …
When pairwise consistency entails GC

- We have seen that it happens in acyclic instances…
- The classical result that this is also necessary [Beeri, Fagin, Maier, and Yannakakis. J.ACM’83] actually holds only if relations cannot be used in more than one constraint/query atoms
- In fact, it works even on some cyclic instances
- We now have a precise structural characterization of the instances where local consistency entails global consistency
  - it applies to the binary case, too
  - it applies to the more general case where pairwise consistency is enforced between each pair of arbitrary defined subproblems (see appendix)!

[Greco and Scarcello. PODS’10]
The Power of Pairwise Consistency

Let us describe when local (pairwise) consistency (LC) entails global consistency (GC), on the basis of the constraint structure.

That is, we describe the condition such that:

- whenever it holds, LC entails GC for every possible CSP instance (i.e., no matter on the constraint relations)
- if it does not hold, there exists an instance where LC fails

If we are interested only in the decision problem (is the CSP satisfiable?) than this condition is the existence of an acyclic core [Atserias, Bulatov, and Dalmau. ICALP’07]
The Power of Pairwise Consistency

Does pairwise consistency entail global consistency in this case?

Constraints

- e(A,B)
- e(A,C)
- e(D,C)
- e(D,B)
The Power of Pairwise Consistency

Does pairwise consistency entail global consistency in this case?

Yes! No matter of the tuples in the constraint relation \( e \)

Every constraint is a core of the instance

Constraints

- \( e(A,B) \)
- \( e(A,C) \)
- \( e(D,C) \)
- \( e(D,B) \)
Does pairwise consistency entail global consistency in this case?

Yes! No matter of the tuples in the constraint relation $e$

Every constraint is a core of the instance
The constraint $e(X,Y)$ is **tp-covered** in an acyclic hypergraph if,

- add a fresh constraint $e'(X,Y)$ (where $e'$ is a fresh relational symbol),
- a core of the new instance has an acyclic hypergraph

Intuitively the “coloring” of $e(X,Y)$ forces the core of the new structure to deal with the ordered pair $(X,Y)$

- Indeed, every core must contain $e'(X,Y)$

Instead, the usual notion of the core does not preserve the meaning of variables

- this is crucial for computing solutions, but not for the decision problem
The Power of Pairwise Consistency

The constraint $e(X,Y)$ is **tp-covered** in an acyclic hypergraph if,

- add a fresh constraint $e'(X,Y)$ (where $e'$ is a fresh relational symbol),
- a core of the new instance has an acyclic hypergraph

Local (pairwise) consistency entails Global consistency if and only if every constraint is tp-covered in an acyclic hypergraph.
The constraint \( e(X,Y) \) is \( tp \)-covered in an acyclic hypergraph if,

- add a fresh constraint \( e'(X,Y) \) (where \( e' \) is a fresh relational symbol),
- a core of the new instance has an acyclic hypergraph.

Note that \( e(F,C) \) does not occur in any core.
The constraint $e(X,Y)$ is tp-covered in an acyclic hypergraph if,

- add a fresh constraint $e'(X,Y)$ (where $e'$ is a fresh relational symbol),
- a core of the new instance has an acyclic hypergraph
Here pairwise consistency solves the satisfaction problem.

The structure of any core is an undirected acyclic graph.
The power of Pairwise Consistency

Here pairwise consistency solves the satisfaction problem.

The structure of any core is an undirected acyclic graph.

However, it does not entail global consistency.

There is an instance that is pairwise consistent but $e(A,B)$ contains wrong tuples.

$e(A,B)$ is not $tp$-covered: the core of the new structure is cyclic.
A generalization: Local k-consistency

Consider subproblems of k constraints

*Local k-consistency*: pairwise consistency over such (k-constraints) subproblems

Equivalent to *relational k-consistency* [Dechter and van Beek. TCS’97]

Local k-consistency entails Global consistency if and only if every constraint is tp-covered in a hypergraph having Generalized Hypertree width k [Greco and Scarcello. PODS’10]

See the appendix for a further generalization to arbitrary subproblems in the general framework of tree projections
Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width
Constraint Optimization Problems

Classically, CSP: Constraint Satisfaction Problem

However, sometimes a solution is enough to “satisfy” (constraints), but not enough to make (users) “happy”

Hence, several variants of the basic CSP framework:
- E.g., fuzzy, probabilistic, weighted, lexicographic, penalty, valued, semiring-based, …
Classical CSPs

- Set of variables \{X_1, \ldots, X_{26}\}
- Set of constraint scopes
- Set of constraint relations
E.g., find the solution that minimizes the total number of vowels occurring in the words.

The puzzle in general admits more than one solution...
Each mapping variable-value has a cost. Then, find an assignment:

- Satisfying all the constraints, and
- Having the minimum total cost.
Each mapping variable-value has a cost.

Then, find an assignment:
- Satisfying all the constraints, and
- Having the minimum total cost.

Each tuple has a cost.

Then, find an assignment:
- Satisfying all the constraints, and
- Having the minimum total cost.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARIS</td>
<td>PANDA</td>
<td>Laura</td>
<td>Anita</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Classification for Optimization Problems

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

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

Each constraint relation has a cost.
Then, find an assignment:
- Minimizing the cost of violated relations.
Adapt the dynamic programming approach in (Yannakakis’81)

[CSOP: Tractability of Acyclic Instances]

- [A1 B1 H1 A2 B1 H2]
- [A1 B1 C1 D1 A1 B1 C2 D2]
- [A1 B1 E1 F1 A1 B1 E2 F2]

[Gottlob & Greco, EC’07]
Adapt the dynamic programming approach in (Yannakakis’81) with a bottom-up computation:

- Filter the tuples that do not match
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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>C1</td>
<td>D1</td>
</tr>
<tr>
<td>A1</td>
<td>B1</td>
<td>C2</td>
<td>D2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>E1</td>
<td>F1</td>
</tr>
<tr>
<td>A1</td>
<td>B1</td>
<td>E2</td>
<td>F2</td>
</tr>
</tbody>
</table>

- cost(C/C1) = cost(D/D1) = 0
- cost(C/C2) = cost(D/D2) = 1
- cost(E/E1) = cost(F/F1) = 0
- cost(E/E2) = cost(F/F2) = 1
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

Cost calculations:
- \( \text{cost}(C/C1) = \text{cost}(D/D1) = 0 \)
- \( \text{cost}(C/C2) = \text{cost}(D/D2) = 1 \)
- \( \text{cost}(E/E1) = \text{cost}(F/F1) = 0 \)
- \( \text{cost}(E/E2) = \text{cost}(F/F2) = 1 \)
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

\[
\begin{align*}
\text{cost}(A/A1) + \\
\text{cost}(B/B1) + \\
\text{cost}(C/C1) + \\
\text{cost}(D/D1) + \\
\text{cost}(E/E1) + \\
\text{cost}(F/F1)
\end{align*}
\]

\[
\begin{align*}
\text{cost}(C/C1) = \text{cost}(D/D1) = 0 \\
\text{cost}(C/C2) = \text{cost}(D/D2) = 1 \\
\text{cost}(E/E1) = \text{cost}(F/F1) = 0 \\
\text{cost}(E/E2) = \text{cost}(F/F2) = 1
\end{align*}
\]
Adapt the dynamic programming approach in (Yannakakis’81)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>H1</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
<td>H2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>D1</td>
<td>E1</td>
<td>F1</td>
</tr>
</tbody>
</table>

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)
**WCSP: Tractability of Acyclic Instances**

<table>
<thead>
<tr>
<th>CSOP</th>
<th>1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARIS</td>
<td>PANDA</td>
</tr>
<tr>
<td>LAURA</td>
<td>ANITA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WCSP</th>
<th>1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PANDA</td>
<td>LAURA</td>
</tr>
<tr>
<td>ANITA</td>
<td></td>
</tr>
</tbody>
</table>

[Gottlob, Greco, and Scarcello, ICALP'09]
WCSP: Tractability of Acyclic Instances

The mapping:

- Is feasible in linear time
- Preserves the solutions
- Preserves acyclicity
In-tractability of MAX-CSP Instances

- Maximize the number of words placed in the puzzle

[Gottlob, Greco, and Scarcello, ICALP'09]
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 $\iff$ 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

<table>
<thead>
<tr>
<th>1 2</th>
<th>1H</th>
</tr>
</thead>
<tbody>
<tr>
<td>P A</td>
<td>PARIS</td>
</tr>
<tr>
<td>P A</td>
<td>PANDA</td>
</tr>
<tr>
<td>L A</td>
<td>LAURA</td>
</tr>
<tr>
<td>A N</td>
<td>ANITA</td>
</tr>
<tr>
<td>A A</td>
<td>unsat</td>
</tr>
<tr>
<td>A B</td>
<td>unsat</td>
</tr>
<tr>
<td>...</td>
<td>unsat</td>
</tr>
</tbody>
</table>

Cost 1, otherwise cost 0

1,2,1H

MAX-CSP

CSOP
The mapping:

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

Cost 1, otherwise cost 0
Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width
Game Theory (in a Nutshell)

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Which actions have to be performed?

**Solution Concepts**

- Nash equilibria
- Strong equilibria
- Pareto equilibria
- ... Kernel
- Nucleolus
- Core
- Bargaining set
- ... Shapley value
- Stable sets
Game Theory (in a Nutshell)

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Which actions have to be performed?

Solution Concepts
Non-Cooperative Games

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- Interacts with other players
- Is **rational**

Payoff maximization problem

<table>
<thead>
<tr>
<th>Bob</th>
<th>John goes <em>out</em></th>
<th>John stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>John</th>
<th>Bob goes <em>out</em></th>
<th>Bob stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Non-Cooperative Games (2/3)

Payoff maximization problem

Nash equilibria

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

John goes **out**

Bob goes **out**

<table>
<thead>
<tr>
<th>Bob</th>
<th>John goes out</th>
<th>John stays at home</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>home</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Bob goes **out**

Bob stays at **home**

<table>
<thead>
<tr>
<th>John</th>
<th>Bob goes out</th>
<th>Bob stays at home</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>home</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Non-Cooperative Games(2/3)

Payoff maximization problem

Nash equilibria

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

<table>
<thead>
<tr>
<th>Bob</th>
<th>John goes <em>out</em></th>
<th>John stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>John</th>
<th>Bob goes <em>out</em></th>
<th>Bob stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Non-Cooperative Games

Payoff maximization problem

Nash equilibria

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

<table>
<thead>
<tr>
<th>Bob</th>
<th>John goes <em>out</em></th>
<th>John stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bob goes <em>out</em></th>
<th>Bob stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>1</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
</tr>
</tbody>
</table>
Non-Cooperative Games

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Payoff maximization problem

Nash equilibria

<table>
<thead>
<tr>
<th></th>
<th>Bob goes <em>out</em></th>
<th>Bob stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>John goes <em>out</em></th>
<th>John stays at <em>home</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>out</em></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><em>home</em></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Payoff maximization problem

Nash equilibria

Every game admits a **mixed Nash equilibrium**, where players chose their strategies according to probability distributions
Succinct Game Representations

Players:
- Maria, Francesco

Choices:
- movie, opera

If 2 players, then size = $2^2$
Succint Game Representations

Players:
- Maria, Francesco, Paola

Choices:
- movie, opera

If 2 players, then size = $2^2$
If 3 players, then size = $2^3$

<table>
<thead>
<tr>
<th>Maria</th>
<th>F_{movie} and P_{movie}</th>
<th>F_{movie} and P_{opera}</th>
<th>F_{opera} and P_{movie}</th>
<th>F_{opera} and P_{opera}</th>
</tr>
</thead>
<tbody>
<tr>
<td>movie</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>opera</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Succint Game Representations

Players:
- Maria, Francesco, Paola, Roberto, and Giorgio

Choices:
- movie, opera

If 2 players, then size = $2^2$
If 3 players, then size = $2^3$
...  
If N players, then size = $2^N$

<table>
<thead>
<tr>
<th>Maria</th>
<th>$F_{\text{movie}}$</th>
<th>$P_{\text{movie}}$</th>
<th>$R_{\text{movie}}$</th>
<th>$G_{\text{movie}}$</th>
<th>……………………</th>
</tr>
</thead>
<tbody>
<tr>
<td>movie</td>
<td>2</td>
<td>………</td>
<td>………</td>
<td>………</td>
<td>………</td>
</tr>
<tr>
<td>opera</td>
<td>0</td>
<td>………</td>
<td>………</td>
<td>………</td>
<td>………</td>
</tr>
</tbody>
</table>
Succinct Game Representations

- Players:
  - Francesco, Paola, Roberto, Giorgio, and Maria

- Choices:
  - movie, opera
Succinct Game Representations

Players:
- Francesco, Paola, Roberto, Giorgio, and Maria

Choices:
- movie, opera

<table>
<thead>
<tr>
<th></th>
<th>$P_mR_m$</th>
<th>$P_mR_o$</th>
<th>$P_oR_m$</th>
<th>$P_oR_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$P_mF_m$</th>
<th>$P_mF_o$</th>
<th>$P_oF_m$</th>
<th>$P_oF_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R_m$</th>
<th>$R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Pure Equilibria

Players:
- Francesco, Paola, Roberto, Giorgio, and Maria

Choices:
- movie, opera

<table>
<thead>
<tr>
<th></th>
<th>$P_m R_m$</th>
<th>$P_m R_o$</th>
<th>$P_o R_m$</th>
<th>$P_o R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$P_m F_m$</th>
<th>$P_m F_o$</th>
<th>$P_o F_m$</th>
<th>$P_o F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R_m$</th>
<th>$R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
### Pure Equilibria

**Players:**
- Francesco, Paola, Roberto, Giorgio, and Maria

**Choices:**
- movie, opera

#### Payoff Matrices

**Players:** Francesco, Paola, Roberto, Giorgio, and Maria

**Choices:** movie, opera

<table>
<thead>
<tr>
<th></th>
<th>P&lt;sub&gt;m&lt;/sub&gt;R&lt;sub&gt;m&lt;/sub&gt;</th>
<th>P&lt;sub&gt;m&lt;/sub&gt;R&lt;sub&gt;o&lt;/sub&gt;</th>
<th>P&lt;sub&gt;o&lt;/sub&gt;R&lt;sub&gt;m&lt;/sub&gt;</th>
<th>P&lt;sub&gt;o&lt;/sub&gt;R&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P&lt;sub&gt;m&lt;/sub&gt;F&lt;sub&gt;m&lt;/sub&gt;</th>
<th>P&lt;sub&gt;m&lt;/sub&gt;F&lt;sub&gt;o&lt;/sub&gt;</th>
<th>P&lt;sub&gt;o&lt;/sub&gt;F&lt;sub&gt;m&lt;/sub&gt;</th>
<th>P&lt;sub&gt;o&lt;/sub&gt;F&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>F&lt;sub&gt;m&lt;/sub&gt;</th>
<th>F&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>F&lt;sub&gt;m&lt;/sub&gt;</th>
<th>F&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R&lt;sub&gt;m&lt;/sub&gt;</th>
<th>R&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
### Pure Equilibria

**Players:**
- Francesco, Paola, Roberto, Giorgio, and Maria

**Choices:**
- movie, opera

#### Payoff Tables

**Players:** Francesco, Paola, Roberto, Giorgio, and Maria

**Choices:** movie, opera

<table>
<thead>
<tr>
<th></th>
<th>$P_mR_m$</th>
<th>$P_mR_o$</th>
<th>$P_oR_m$</th>
<th>$P_oR_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$P_mF_m$</th>
<th>$P_mF_o$</th>
<th>$P_oF_m$</th>
<th>$P_oF_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R_m$</th>
<th>$R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Pure Nash Equilibria and Easy Games

Nash Equilibrium Existence

Constraint Satisfaction Problem

Solve CSP in polynomial time using known methods

[Gottlob, Greco, and Scarcello, JAIR’05]
# Encoding Games in CSPs

<table>
<thead>
<tr>
<th>$F$</th>
<th>$P_mR_m$</th>
<th>$P_mR_o$</th>
<th>$P_oR_m$</th>
<th>$P_oR_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$G$</th>
<th>$P_mF_m$</th>
<th>$P_mF_o$</th>
<th>$P_oF_m$</th>
<th>$P_oF_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R$</th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$</th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M$</th>
<th>$R_m$</th>
<th>$R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

\[\text{Encoding Diagrams}\]

\[\text{Table Representation}\]
Encoding Games in CSPs

<table>
<thead>
<tr>
<th>F</th>
<th>P_mR_m</th>
<th>P_mR_o</th>
<th>P_oR_m</th>
<th>P_oR_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G</th>
<th>P_mF_m</th>
<th>P_mF_o</th>
<th>P_oF_m</th>
<th>P_oF_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>F_m</th>
<th>F_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>F_m</th>
<th>F_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>R_m</th>
<th>R_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Encoding Games in CSPs

<table>
<thead>
<tr>
<th></th>
<th>$P_m R_m$</th>
<th>$P_m R_o$</th>
<th>$P_o R_m$</th>
<th>$P_o R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$P_m F_m$</th>
<th>$P_m F_o$</th>
<th>$P_o F_m$</th>
<th>$P_o F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$o$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$F_m$</th>
<th>$F_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R_m$</th>
<th>$R_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$o$</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

$\tau_F$: $m$ $m$ $m$ $m$

$\tau_G$: $m$ $m$ $m$ $m$

$\tau_R$: $m$ $o$ $m$

$\tau_P$: $m$ $m$

$\tau_M$: $m$ $m$
The interaction structure of a game $G$ can be represented by:
- the dependency graph $G(G)$ according to $\text{Neigh}(G)$
- a hypergraph $H(G)$ with edges: $H(p) = \text{Neigh}(p) \cup \{p\}$
Interaction Among Players: Friends

This is the same structure as the one of the associated CSP

$H(\text{FRIENDS})$
Interaction Among Players: Friends

This is the same structure as the one of the associated CSP

On (nearly)-Acyclic Instances, Nash equilibria are easy

$H(\text{FRIENDS})$
Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width
Game Theory (in a Nutshell)

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

Which actions have to be performed?

Solution Concepts

- Kernel
- Nucleolus
- Core
- Bargaining set
- ... Shapley value
- Stable sets
Cooperative Game Theory (1/2)

To perform some task
Utility distribution, if the task is performed
Jointly perform the task (with some cost)

Each player:
- Has a goal to be achieved
- Has a set of possible actions
- Interacts with other players
- Is rational
Cooperative Game Theory (1/2)

To perform some task
Utility distribution, if the task is performed
Jointly perform the task (with some cost)

Each player:
- Has a goal to be achieved
- Has a set of possible actions
- Interacts with other players
- Is rational

- Players get 9$, if they enforce connectivity
- Enforcing connectivity over an edge as a cost
Cooperative Game Theory (1/2)

To perform some task
Utility distribution, if the task is performed
Jointly perform the task (with some cost)

Each player:
- Has a goal to be achieved
- Has a set of possible actions
- Interacts with other players
- Is rational

- Players get 9$, if they enforce connectivity
- Enforcing connectivity over an edge as a cost

Coalition \{F,P,R,M\} gets 9$, and pays 6$

worth \ v(\{F,P,R,M\}) = 9$ - 6$
Cooperative Game Theory (1/2)

To perform some task

Utility distribution, if the task is performed

Jointly perform the task (with some cost)

Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- **Is rational**

<table>
<thead>
<tr>
<th>coalition</th>
<th>worth</th>
</tr>
</thead>
<tbody>
<tr>
<td>{F}</td>
<td>0</td>
</tr>
<tr>
<td>{G,F,P,R,M}</td>
<td>3</td>
</tr>
<tr>
<td>{G,F,P,R,M}</td>
<td>4</td>
</tr>
</tbody>
</table>

How to distribute 9\$, based on such worths?
Each player:
- Has a **goal** to be achieved
- Has a set of possible **actions**
- **Interacts** with other players
- Is **rational**

How to distribute 9$, based on such worths?
Cooperative Game Theory (2/2)

Each player:
- Has a goal to be achieved
- Has a set of possible actions
- Interacts with other players
- Is rational

How to distribute 9\$, based on such worths?

<table>
<thead>
<tr>
<th>coalition</th>
<th>worth</th>
</tr>
</thead>
<tbody>
<tr>
<td>{F}</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>{G,P,R,M}</td>
<td>0</td>
</tr>
<tr>
<td>{F,P,R,M}</td>
<td>3</td>
</tr>
<tr>
<td>{G,F,P,R,M}</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>value</th>
<th>excess</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
</tbody>
</table>
Find the distribution(s) that:
- Each coalition has a positive excess
- Lexicographically maximizes the excess vector
- Is immune against deviations

How to distribute 9\$, based on such worths?
The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

\[ \mathcal{G} = \langle N, \nu \rangle, \quad \nu : 2^N \mapsto \mathbb{R} \]

Outcomes belong to the imputation set \( X(\mathcal{G}) \)

\[ x \in X(\mathcal{G}) \]

- **Efficiency**
  \[ x(N) = \nu(N) \]

- **Individual Rationality**
  \[ x_i \geq \nu(\{i\}), \quad \forall i \in N \]
The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

\[ \mathcal{G} = \langle N, \nu \rangle, \quad \nu : 2^N \rightarrow \mathbb{R} \]

**Solution Concepts** characterize outcomes in terms of
- Fairness
- Stability
The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

\[ G = \langle N, v \rangle, \ v : 2^N \rightarrow \mathbb{R} \]

**Solution Concepts** characterize outcomes in terms of

- Fairness
- Stability

\[ 0 \geq e(S, x) = v(S) - \sum_{i \in S} x_i \]

**The Core:** \[ \forall S \subseteq N, x(S) \geq v(S); \]
\[ x(N) = v(N) \]
Complexity of Solution Concepts

- Nucleolus
- Kernel
- Bargaining Set
- Stable Sets

Graph games:
- Succinct specification
- Core existence is coNP-complete

<table>
<thead>
<tr>
<th>coalition</th>
<th>worth</th>
</tr>
</thead>
<tbody>
<tr>
<td>{F}</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>{G,P,R,M}</td>
<td>0</td>
</tr>
<tr>
<td>{F,P,R,M}</td>
<td>5</td>
</tr>
<tr>
<td>{G,F,P,R,M}</td>
<td>6</td>
</tr>
</tbody>
</table>
Complexity of Solution Concepts

- Nucleolus
- Kernel
- Bargaining Set
- Stable Sets

**Succinct games:**
- Nucleolus is \( P^{NP} \)-complete
- Kernel is \( P^{NP} \)-complete
- Bargaining set is \( coNP^{NP} \)-complete
- Stable sets is still open

Reductions for graph games

Ellipsoid method + NP separation oracles

[Greco, Malizia, Palopoli, Scarcello, AIJ‘11]
Consider the sentence, 
over the graph where *N* is the set of nodes and *E* the set of edges:

\[
\text{proj}(X, Y) \equiv X \subseteq N \land \exists c, c'(Y(c, c') \rightarrow X(c) \land x(c')) \land \\
\forall c, c'(X(c) \land X(c') \land E(c, c') \rightarrow Y(c, c'))
\]
The Core: \( \forall S \subseteq N, x(S) \geq v(S); \)
\[ x(N) = v(N) \]

Consider the sentence,
over the graph where \( N \) is the set of nodes and \( E \) the set of edges:

\[ proj(X, Y) \equiv X \subseteq N \land \]
\[ \forall c, c' \left( Y(c, c') \rightarrow X(c) \land x(c') \right) \land \]
\[ \forall c, c' \left( X(c) \land X(c') \land E(c, c') \rightarrow Y(c, c') \right) \]

…it tells that \( Y \) is the set of edges covered by the nodes in \( X \)
Membership in the Core on Graph Games

The Core: \( \forall S \subseteq N, x(S) \geq v(S); \)
\( x(N) = v(N) \)

Let \( \text{proj}(X, Y) \) be the formula stating that \( Y \) is the set of edges covered by the nodes in \( X \)

Define the following weights:
\( w_E(c, c') = -w(c, c'); \quad w_N(c) = x_c \)

Value of the edge (negated)  Value at the imputation
Membership in the Core on Graph Games

Let $\text{proj}(X, Y)$ be the formula stating that $Y$ is the set of edges covered by the nodes in $X$.

Define the following weights: $w_E(c, c') = -w(c, c')$, $w_N(c) = x_c$.

Find the “minimum-weight" $X$ and $Y$ such that $\text{proj}(X, Y)$ holds.

The Core: $\forall S \subseteq N, x(S) \geq v(S)$; $x(N) = v(N)$.
Membership in the Core on Graph Games

Let $\text{proj}(X, Y)$ be the formula stating that $Y$ is the set of edges covered by the nodes in $X$.

Define the following weights:

$w_E(c, c') = -w(c, c'); \quad w_N(c) = x_c$

The Core: $\forall S \subseteq N, x(S) \geq v(S);$ $x(N) = v(N)$

Let $\text{proj}(X, Y)$ be the formula stating that $Y$ is the set of edges covered by the nodes in $X$.

Find the “minimum-weight” $X$ and $Y$ such that $\text{proj}(X, Y)$ holds.

Max (value of edges – value of the imputation), i.e., $\max_{S \subseteq N} e(S, x)$
Outline of Part III

- Applications to Optimization Problems
  - Application: Nash Equilibria
  - Application: Coalitional Games
  - Application: Combinatorial Auctions
- Appendix: Beyond Hypertree Width
Example: Combinatorial Auctions
Example: Combinatorial Auctions
Example: Combinatorial Auctions
Winner Determination Problem

Determine the outcome that maximizes the sum of accepted bid prices
Winner Determination Problem

Determine the outcome that maximizes the sum of accepted bid prices
Other applications [Cramton, Shoham, and Steinberg, ‘06]
- airport runway access
- trucking
- bus routes
- industrial procurement
Example: Combinatorial Auctions

Winner Determination is NP-hard
Structural Properties

item hypergraph
The Winner Determination Problem remains NP-hard even in case of acyclic hypergraphs
Dual Hypergraph

item hypergraph
Dual Hypergraph

item hypergraph

dual hypergraph

I_1

<table>
<thead>
<tr>
<th></th>
<th>h_1</th>
<th>h_3</th>
<th>h_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

polytime
The Approach

[Gottlob & Greco, EC’07]

item hypergraph

solutions

polytime

dual hypergraph

hypertree decomposition of dual hypergraph
Outline of Part III

Applications to Optimization Problems

Application: Nash Equilibria

Application: Coalitional Games

Application: Combinatorial Auctions

Appendix: Beyond Hypertree Width
Treewidth and Hypertree width are based on tree-like aggregations of subproblems that are efficiently solvable.

$k$ variables (resp. $k$ atoms) $\Rightarrow ||I||^k$ solutions (per subproblem)

Is there some more general property that makes the number of solutions in any bag polynomial?

**YES!**

[Grohe & Marx ’06]
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

A query may be solved efficiently, if a fractional hypertree decomposition is given.

FHDs are approximable: If the width is $\leq w$, a decomposition of width $O(w^3)$ may be computed in polynomial time.
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]
Revisiting Decomposition Methods

Each cluster can be seen as a subproblem

Relations:

1V  20H  1H

{1V,20H} = 1V ⊳⊳ 20H
Revisiting Decomposition Methods

Each cluster can be seen as a subproblem

Relations:

\{1V,20H\} = 1V \iff 20H
Revisiting Decomposition Methods

Relations:

\{1V,20H\} = 1V \rightarrow\leftarrow 20H
Revisiting Decomposition Methods

CSP instance \((A, B)\)

\[ A_{\gamma} = \ell - \text{DM}(A) \quad B_{\gamma} = r - \text{DM}(A, B) \]

Relations:

\{1V,20H\} = 1V \bowtie 20H
Revisiting Decomposition Methods

CSP instance \((A, B)\)

\[ A^\wedge = \ell\text{-DM}(A) \quad B^\wedge = r\text{-DM}(A, B) \]

Scopes \quad Solutions

Work on subproblems

Relations:

\(\{1V, 20H\} = 1V \rightarrow 20H\)
Revisiting Decomposition Methods

CSP instance \((A, B)\)

\[ A_\gamma = \ell\text{-DM}(A), \quad B_\gamma = r\text{-DM}(A, B) \]

Scopes

Solutions

Work on subproblems

Generalized hypertree width: take all views that can be computed by joining at most \(k\) atoms (\(k\) query views)
Revisiting Decomposition Methods

CSP instance \((\mathbb{A}, \mathbb{B})\)

\[ A_\mathcal{V} = \ell\text{-DM}(\mathbb{A}) \quad B_\mathcal{V} = r\text{-DM}(\mathbb{A}, \mathbb{B}) \]

**Generalized hypertree width:**
- take all views that can be computed by joining at most \(k\) atoms (\(k\) query views)

Relations:

\{1V,20H\} = 1V \Leftrightarrow 20H
1. Every constraint is associated with a base subproblem
2. Further subproblems can be defined

1. Every subproblem is not more restrictive than the full problem
2. Every base subproblem is at least restrictive as the corresponding constraint

CSP instance \((A, B)\)

\[ A = \ell -\text{DM}(A) \quad B = r -\text{DM}(A, B) \]
Acyclicity in Decomposition Methods

CSP instance $(A, B)$

$A_V = \ell\text{-DM}(A)$
$B_V = r\text{-DM}(A, B)$

Working on subproblems is not necessarily beneficial…
Can some and/or portions of them be selected such that:
- They still cover $A$, and
- They can be arranged as a tree

Working on subproblems is not necessarily beneficial…
Tree Projections (by Example)

\[ \Delta : r_1(A, B, C) r_2(A, F) r_3(C, D) r_4(D, E, F) \\
r_5(E, F, G) r_6(G, H, I) r_7(I, J) r_8(J, K) \]

Structure of the CSP
Tree Projections (by Example)

\[ \Delta : \quad r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \]
\[ r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

Structure of the CSP

Available Views
Tree Projections (by Example)

\[ \Delta : r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \]
\[ r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

Structure of the CSP

Tree Projection

Available Views
Tree Projections (by Example)

\[ \Delta : \ r_1(A, B, C') \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \]
\[ r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

\[ \mathcal{H}_A \]

\[ \mathcal{H}_\alpha \]

\[ \mathcal{H}_{AV} \]

Structure of the CSP

Tree Projection

Available Views
(Noticeable) Examples

- **Treewidth**: take all views that can be computed with at most $k$ variables
- **Generalized hypertree width**: take all views that can be computed by joining at most $k$ atoms ($k$ query views)
- **Fractional hypertree width**: take all views that can be computed through subproblems having fractional cover at most $k$ (or use Marx’s $O(k^3)$ approximation to have polynomially many views)
Tree Decomposition

\[ \mathcal{H}_A \]

\[ \mathcal{H}_{Aa} \]

\[ \nu = \ell \cdot tw_2(A) \]
A General Framework, but

- Decide the existence of a tree projection is \textbf{NP-hard}

[Gottlob, Miklos, and Schwentick, JACM‘09]
A General Framework, but

Decide the existence of a tree projection is **NP-hard**

Hold on generalized hypertree width too.

[Gottlob, Miklos, and Schwentick, JACM'09]
The core of a query \(Q\) is a query \(Q'\) s.t.:

1. \(\text{atoms}(Q') \subseteq \text{atoms}(Q)\)

2. There is a mapping \(h: \text{var}(Q) \rightarrow \text{var}(Q')\) s.t., \(\forall r(X) \in \text{atoms}(Q), r(h(X)) \in \text{atoms}(Q')\)

3. There is no query \(Q''\) satisfying 1 and 2 and such that \(\text{atoms}(Q'') \subset \text{atoms}(Q')\)
The core of a query $Q$ is a query $Q'$ s.t.:

1. $\text{atoms}(Q') \subseteq \text{atoms}(Q)$

2. There is a mapping $h: \text{var}(Q) \rightarrow \text{var}(Q')$ s.t., $\forall r(X) \in \text{atoms}(Q), \ r(h(X)) \in \text{atoms}(Q')$

3. There is no query $Q''$ satisfying 1 and 2 and such that $\text{atoms}(Q'') \subset \text{atoms}(Q')$

Example:
A Source of Complexity: The Core

Cores are isomorphic The “Core”

Cores are equivalent to the query

Example:

\[ Q \]

\[
\begin{array}{ccc}
1 & 2 & 5 \\
3 & 4 & \\
6 & & \\
\end{array}
\]

\[ Q' \]

\[
\begin{array}{ccc}
1 & 2 \\
3 & & \\
\end{array}
\]
Example

\[ Q : \ r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \]
\[ Q : \quad r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \]
Cores and Tree Projections

Structure of the CSP

Tree Projection

Available Views
Cores and Tree Projections

Structure of the CSP

Tree Projection

Available Views
Cores and Tree Projections

Structure of the CSP

Tree Projection

Available Views
Cores and Tree Projections

Structure of the CSP

Tree Projection

Available Views
CORE is NP-hard

Deciding whether $Q'$ is the core of $Q$ is NP-hard

For instance, let 3COL be the class of all 3-colourable graphs containing a triangle

Clearly, deciding whether $G \in 3\text{COL}$ is NP-hard

It is easy to see that $G \in 3\text{COL} \iff K_3$ is the core of $G$

Example:

\[ Q \]

\[ Q' \]

![Graphs](image-url)
Enforcing Local Consistency (Acyclic)
Enforcing Local Consistency (Acyclic)
Enforcing Local Consistency (Decomposition)

CSP instance \((A, B)\)

\[ A_y = \ell \text{-} DM(A), \quad B_y = r \text{-} DM(A, B) \]
If there is a tree projection, then enforcing local consistency over the views solves the decision problem.

\[ A_\mathcal{V} = \ell\text{-DM}(A) \quad B_\mathcal{V} = r\text{-DM}(A, B) \]

[Sagiv & Smueli, ‘93]
Enforcing Local Consistency

If there is a tree projection, then enforcing local consistency over the views solves the decision problem.

\[ A_N = l\text{-}DM(A), \quad B_N = r\text{-}DM(A, B) \]

Does not need to be computed

[Sagiv & Smueli, ‘93]
Even Better

There is a polynomial-time algorithm that:
- either returns that there is no tree projection,
- or solves the decision problem
There is a polynomial-time algorithm that:
- either returns that there is no tree projection,
- or solves the decision problem
The followings are equivalent:

- Local consistency solves the decision problem
- There is a core of the query having a tree projection

[Greco & Scarcello, PODS‘10]
The followings are equivalent
- Local consistency solves the decision problem
- There is a core of the query having a tree projection

\[ Q : r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \]
The Precise Power of Local Consistency

The followings are equivalent

- Local consistency solves the decision problem
- There is a core of the query having a tree projection

\[ Q : r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \]

a core with TP

a core without TP
A Relevant Specialization (not immediate)

- The followings are equivalent
  - Local consistency solves the decision problem
  - There is a core of the query having a tree projection

- The CSP has generalized hypertreewidth $k$ at most

Over all union of $k$ atoms

[Greco & Scarcello, CP’11]
Back on the Result

The followings are equivalent

- Local consistency solves the decision problem
- There is a core of the query having a tree projection

«Promise» tractability

- There is no polynomial time algorithm that
  - either solves the decision problem
  - or disproves the promise
The followings are equivalent:

- Local consistency entails «views containing variables \( O \) are correct»
- The set of variables \( O \) is tp-covered in a tree projection.

\[
Q : \ r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \land \text{atoms}\{A,E\}
\]

\( \{A,E\} \) is tp-covered

\( w(A,B,C) \) is an additional available view.

A core with a TP.
Local consistency for computing solutions

The followings are equivalent

- Local consistency entails «views containing variables O are correct»
- The set of variables O is tp-covered in a tree projection

\[ Q : \ r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \land \text{atoms}([A,F]) \]

\{A,F\} is not tp-covered

\( TP \)
The followings are equivalent

- Local consistency entails global consistency
- Every query atom/constraint is tp-covered in a tree projection

\[ Q : \ r(A, B) \land r(B, C) \land r(A, C) \land r(D, C) \land r(D, B) \land r(A, E) \land r(F, E), \land \text{atoms}\{D, B\} \]
Thank you!