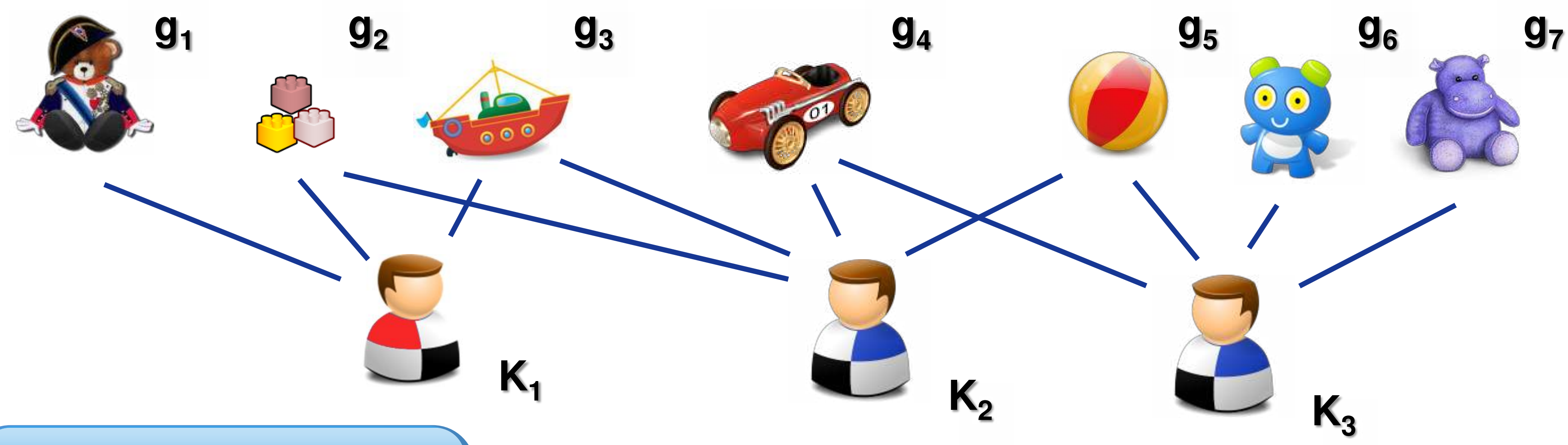
Constraint Satisfaction and Fair Multi-Objective Optimization Problems: Foundations, Complexity, and Islands of Tractability

Gianluigi Greco and Francesco Scarcello

IJCAI-13

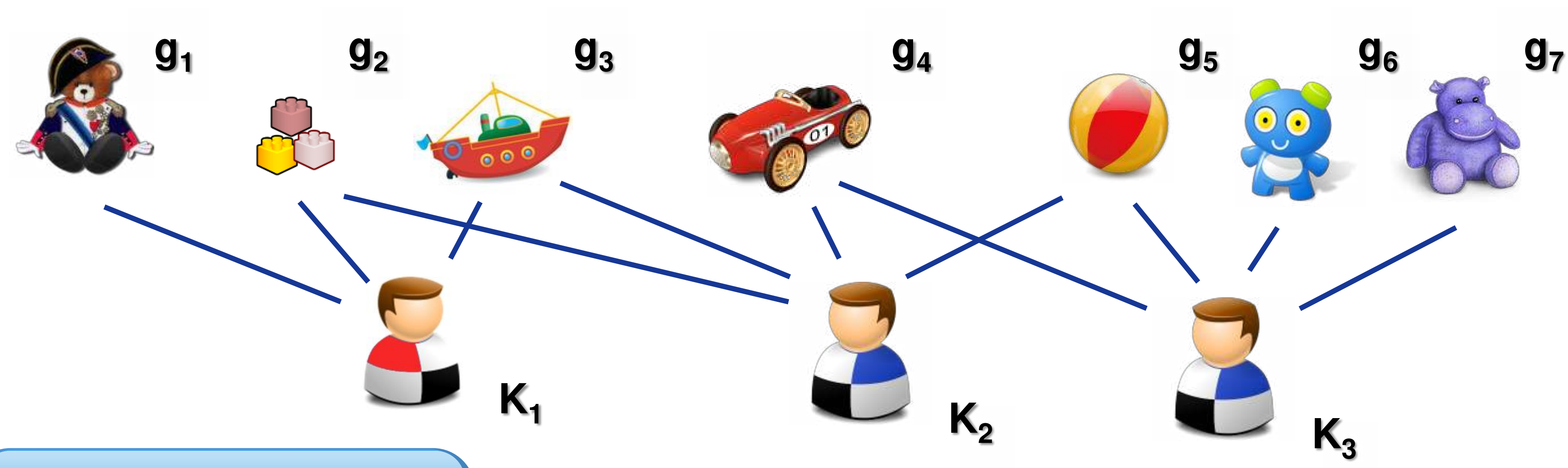
Campus di Arcavacata
UNIVERSITÀ DELLA CALABRIA

Constraint Satisfaction Problems



- Distribute the goods/ presents to the kids
- Goods are indivisible

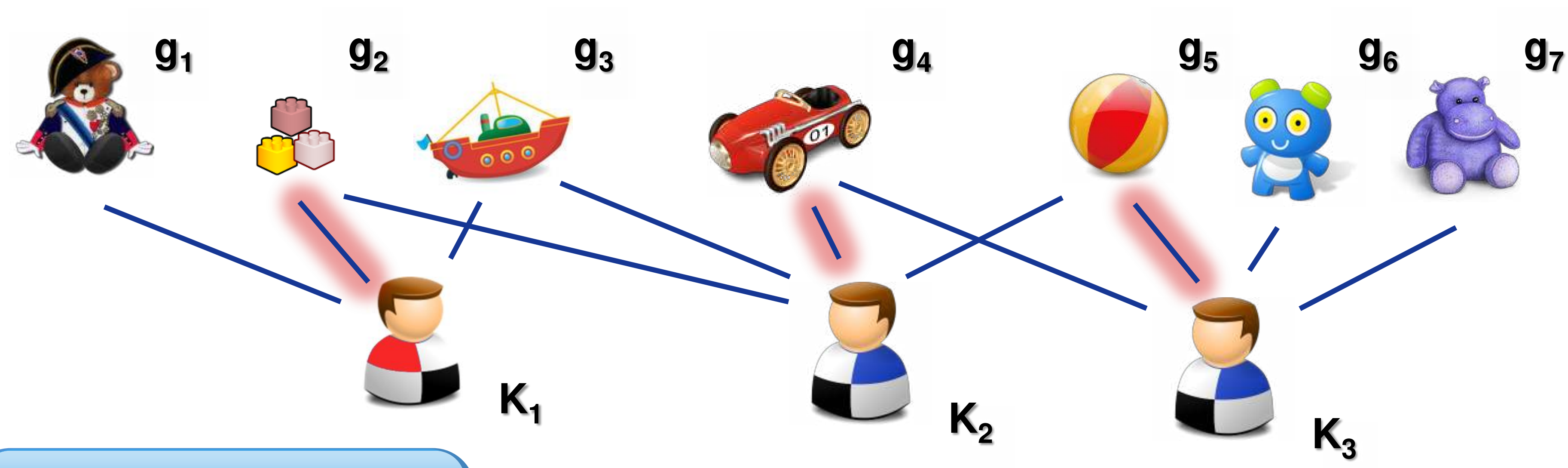
Constraint Satisfaction Problems



- Distribute the goods/ presents to the kids
- Goods are indivisible

- **Variables**
  - K_1, K_2, and K_3
- **Domain**
  - $g_1, g_2, \ldots, g_7$
- **Constraints**
  - $K_1 \in \{g_1, g_2, g_3\}$
  - $K_2 \in \{g_2, g_3, g_4, g_5\}$
  - $K_3 \in \{g_4, g_5, g_6, g_7\}$
  - $K_1 \neq K_2$, $K_2 \neq K_3$

Constraint Satisfaction Problems

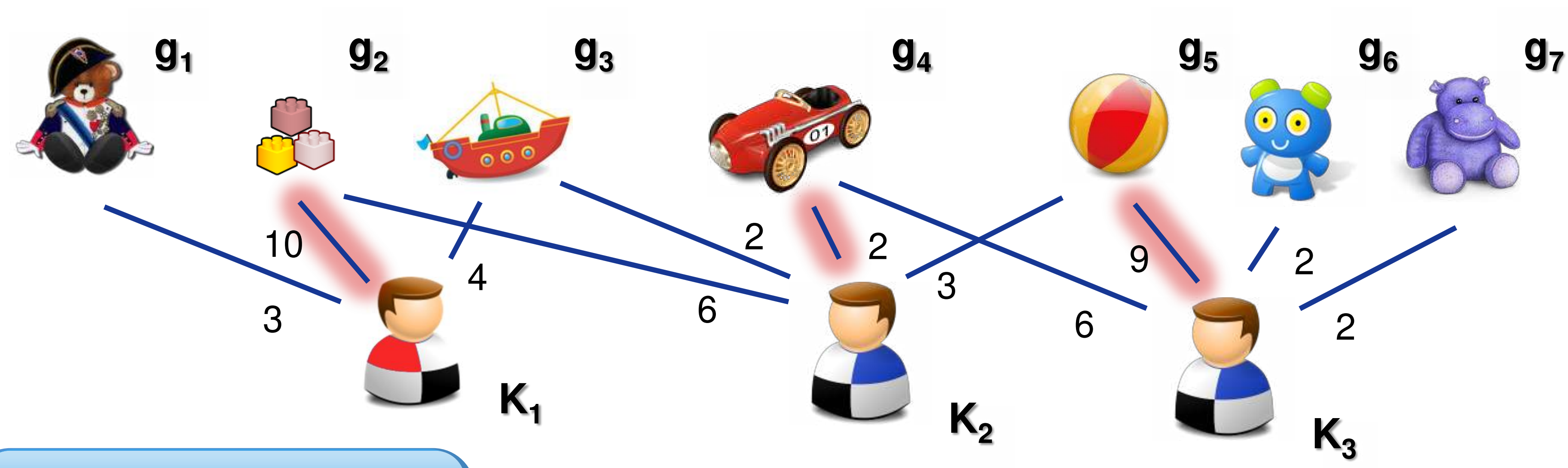


- Distribute the goods/ presents to the kids
- Goods are indivisible

- **Variables**
  - K_1, K_2, and K_3
- **Domain**
  - $g_1, g_2, \ldots, g_7$
- **Constraints**
  - $K_1 \in \{g_1, g_2, g_3\}$
  - $K_2 \in \{g_2, g_3, g_4, g_5\}$
  - $K_3 \in \{g_4, g_5, g_6, g_7\}$
  - $K_1 \neq K_2, K_2 \neq K_3$

- **Solution**
  - $K_1 \rightarrow g_2$
  - $K_2 \rightarrow g_4$
  - $K_3 \rightarrow g_6$

Optimization Functions in CSPs

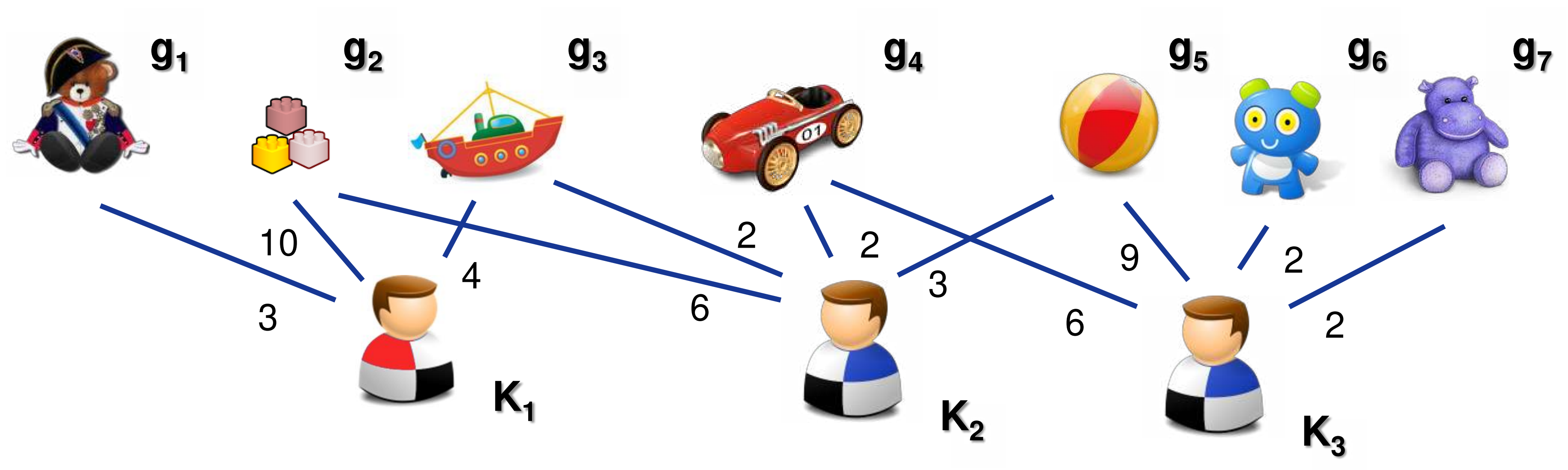


- Distribute the goods/ presents to the kids
- Goods are indivisible

- Kids have preferences over the presents

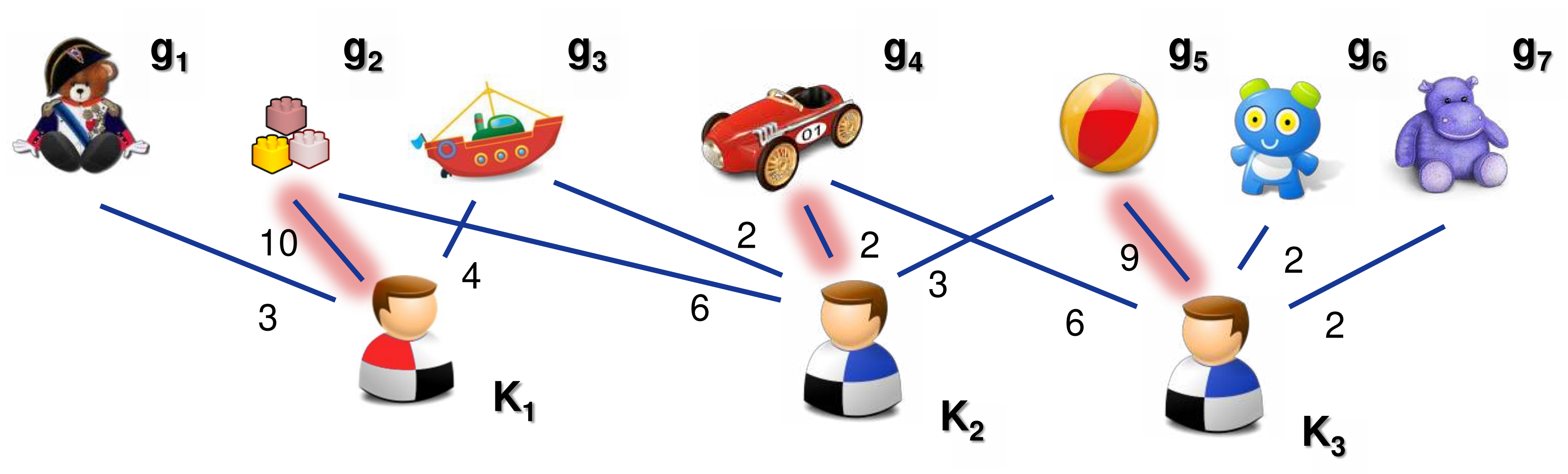
- **Valuation Function:** $\mathcal{F} = \langle w, + \rangle$
  - $w(K_1/g_1) = 3$
  - $w(K_1/g_2) = 10$
  - …
- **Value of the Solution**
  - $w(K_1/g_1) + w(K_2/g_4) + w(K_3/g_6) = 21$
- **Optimal (MAX) Solution**
  - Maximizes the *social welfare*

Multi-Objective Optimization



- **Different Valuations:** $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$
  - $\mathcal{F}_h$ is defined on K_h
- **Combination Strategies:**

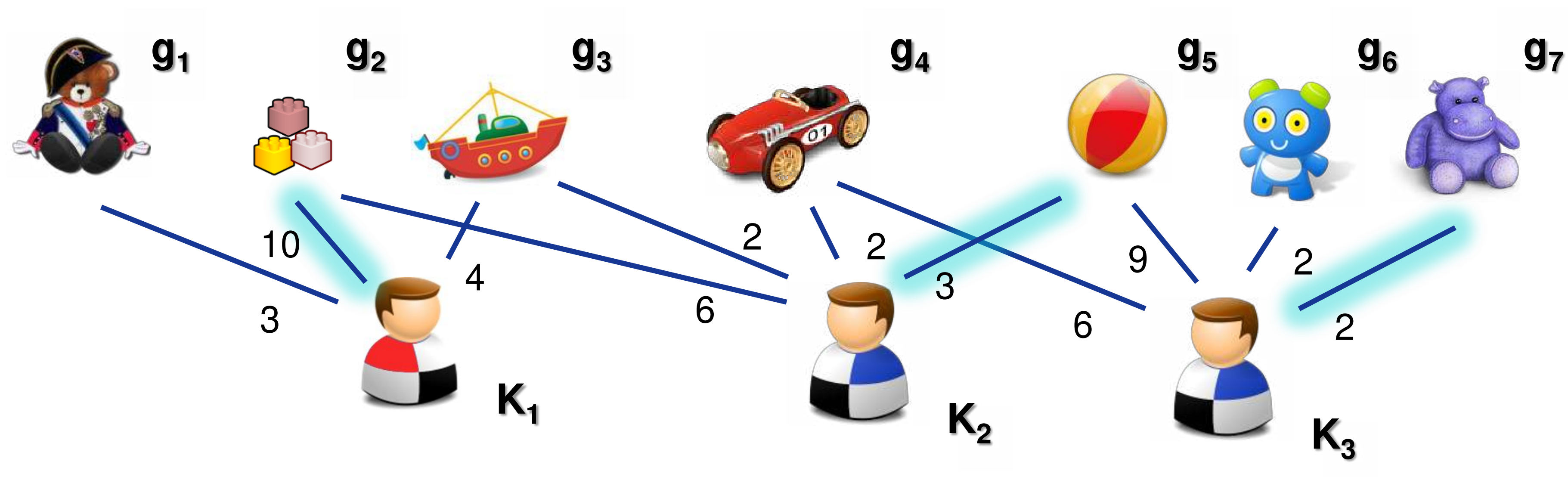
Multi-Objective Optimization



- **Different Valuations:** $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$
  - $\mathcal{F}_h$ is defined on K_h
- **Combination Strategies:**

	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$
MAX-SUM (*social welfare*)	**10**	**2**	**9**

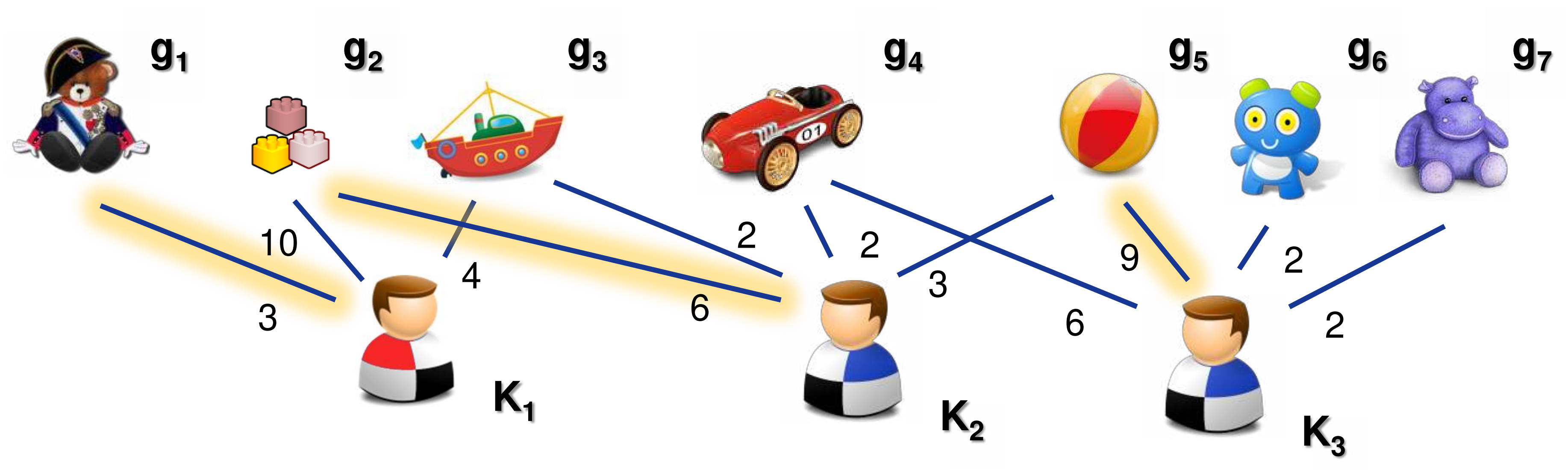
Multi-Objective Optimization



- **Different Valuations:** $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$
  - $\mathcal{F}_h$ is defined on K_h
- **Combination Strategies:**

	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$
MAX-SUM (*social welfare*)	**10**	**2**	**9**
LEX	**10**	**3**	**2**

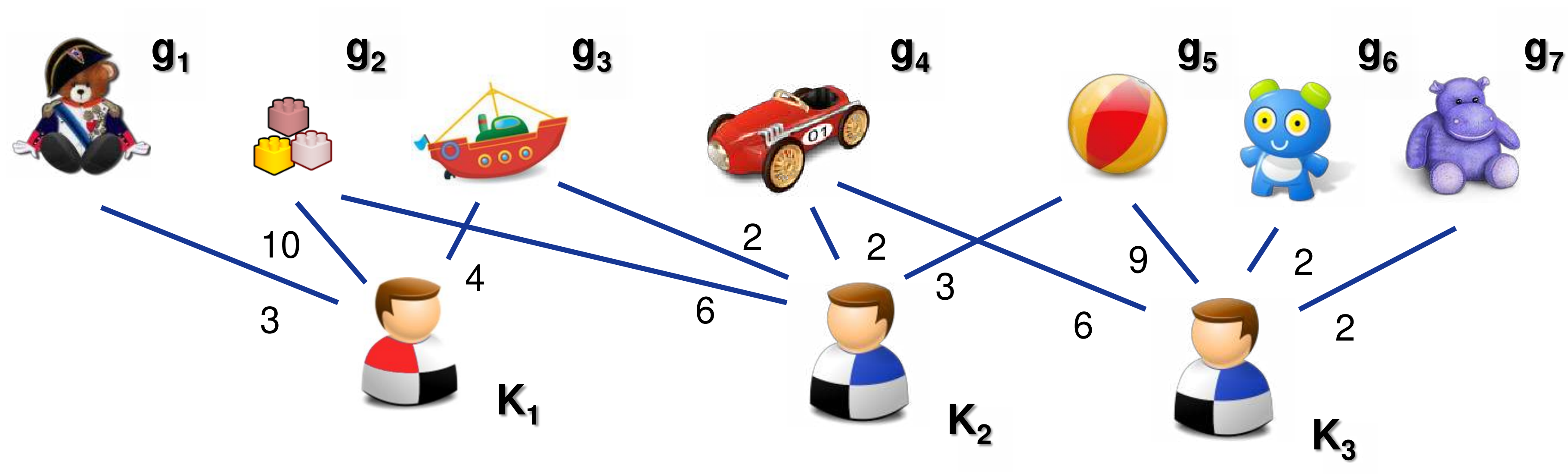
Multi-Objective Optimization



- **Different Valuations:** $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$
  - $\mathcal{F}_h$ is defined on K_h
- **Combination Strategies:**

	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$
MAX-SUM (*social welfare*)	10	2	9
LEX	10	3	2
PARETO *e.g.*	3	6	9

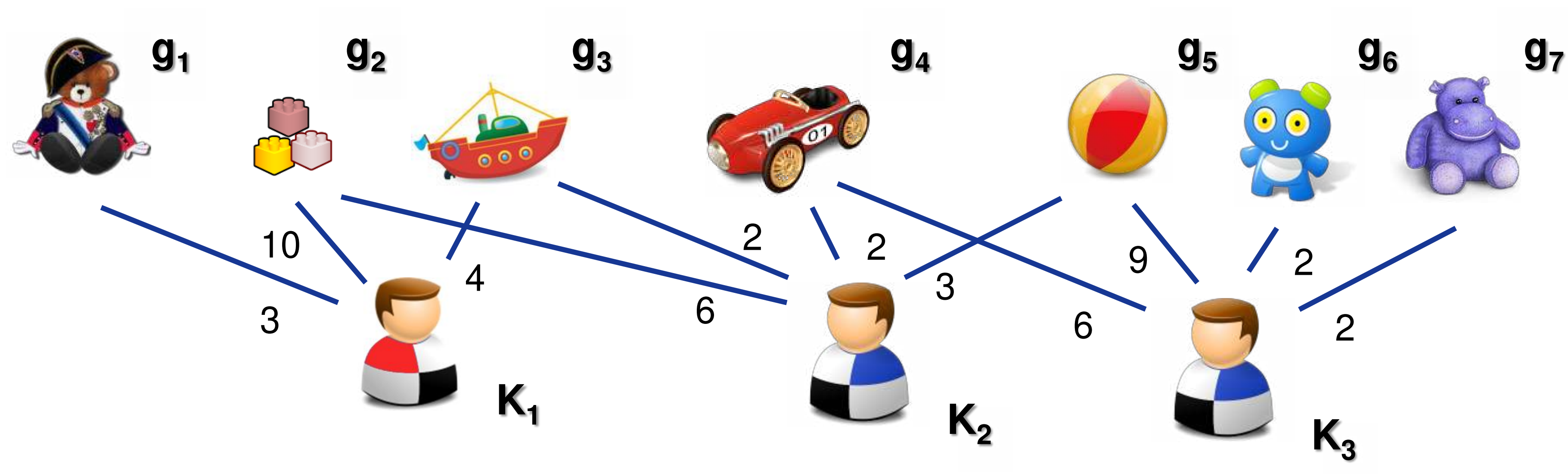
Related Literature



- **Different Valuations:** $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$
  - $\mathcal{F}_h$ is defined on K_h
- **Combination Strategies:**

MAX-SUM (*social welfare*)	10	2	9	[Bistarelli *et al.*, Rossi *et al.*]
LEX	10	3	2	[Freuder *et al.*]
PARETO *e.g.*	3	6	9	[Torrens and Faltings]

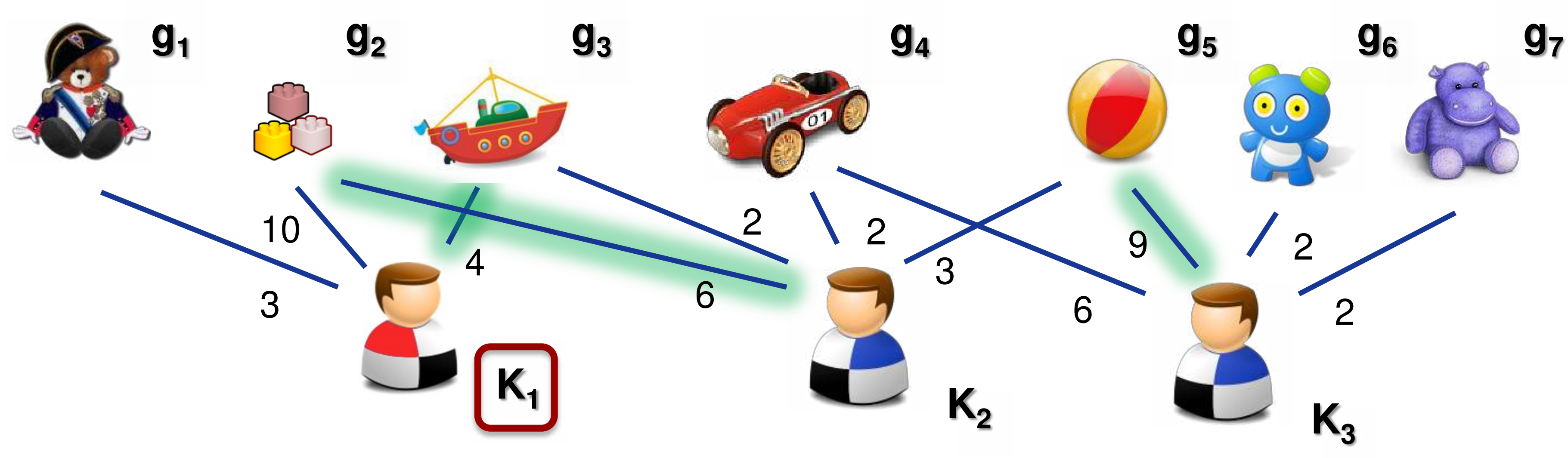
Related Literature



The Santa Claus Problem:

Santa's goal is to distribute presents in a way that the least lucky kid is as happy as possible.

Related Literature

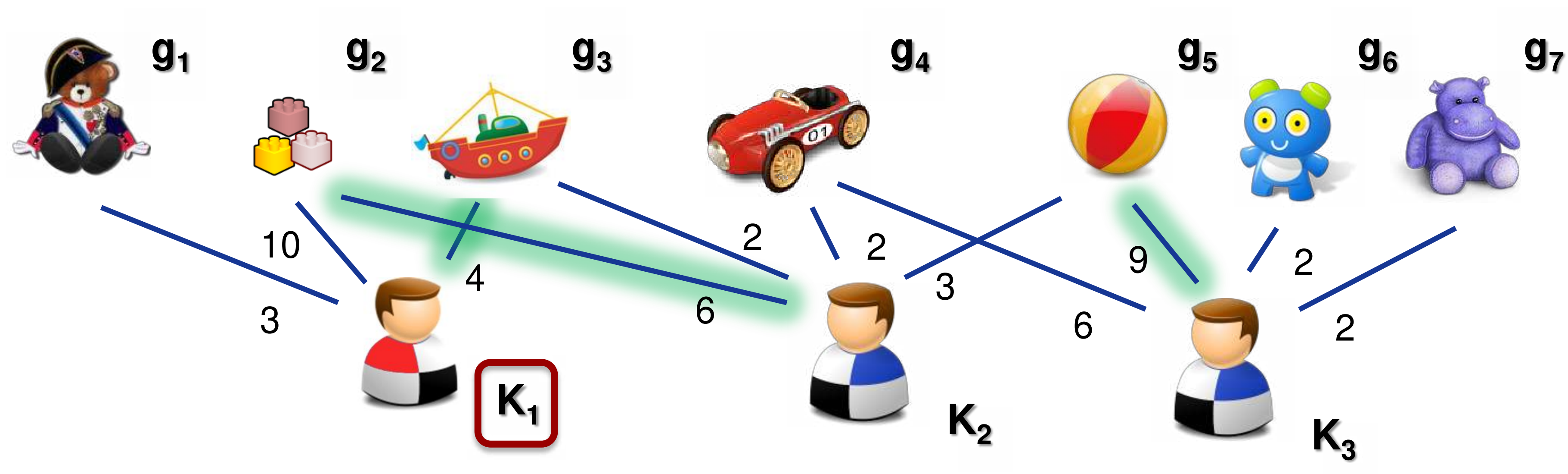


The Santa Claus Problem:

Social welfare = 19 (max 21)

Santa's goal is to distribute presents in a way that the least lucky kid is as happy as possible.

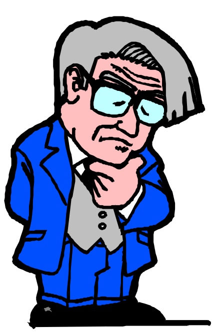
Related Literature



The Santa Claus Problem:

Social welfare = 19 (max 21)

Santa's goal is to distribute presents in a way that the least lucky kid is as happy as possible.

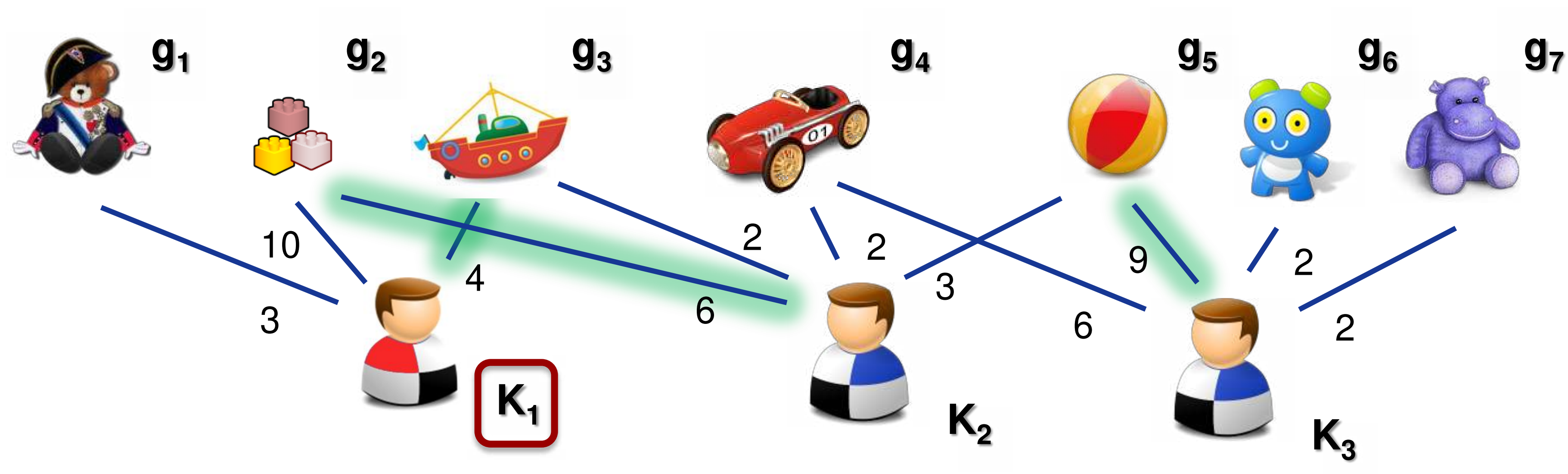


FAIR OPTIMIZATION

- MAX-MIN

[Snow and Freuder, Dubois and Fortemps, Bouveret and Lematre]

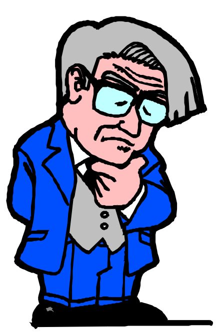
Related Literature



The Santa Claus Problem:

Social welfare = 19 (max 21)

Santa's goal is to distribute presents in a way that the least lucky kid is as happy as possible.



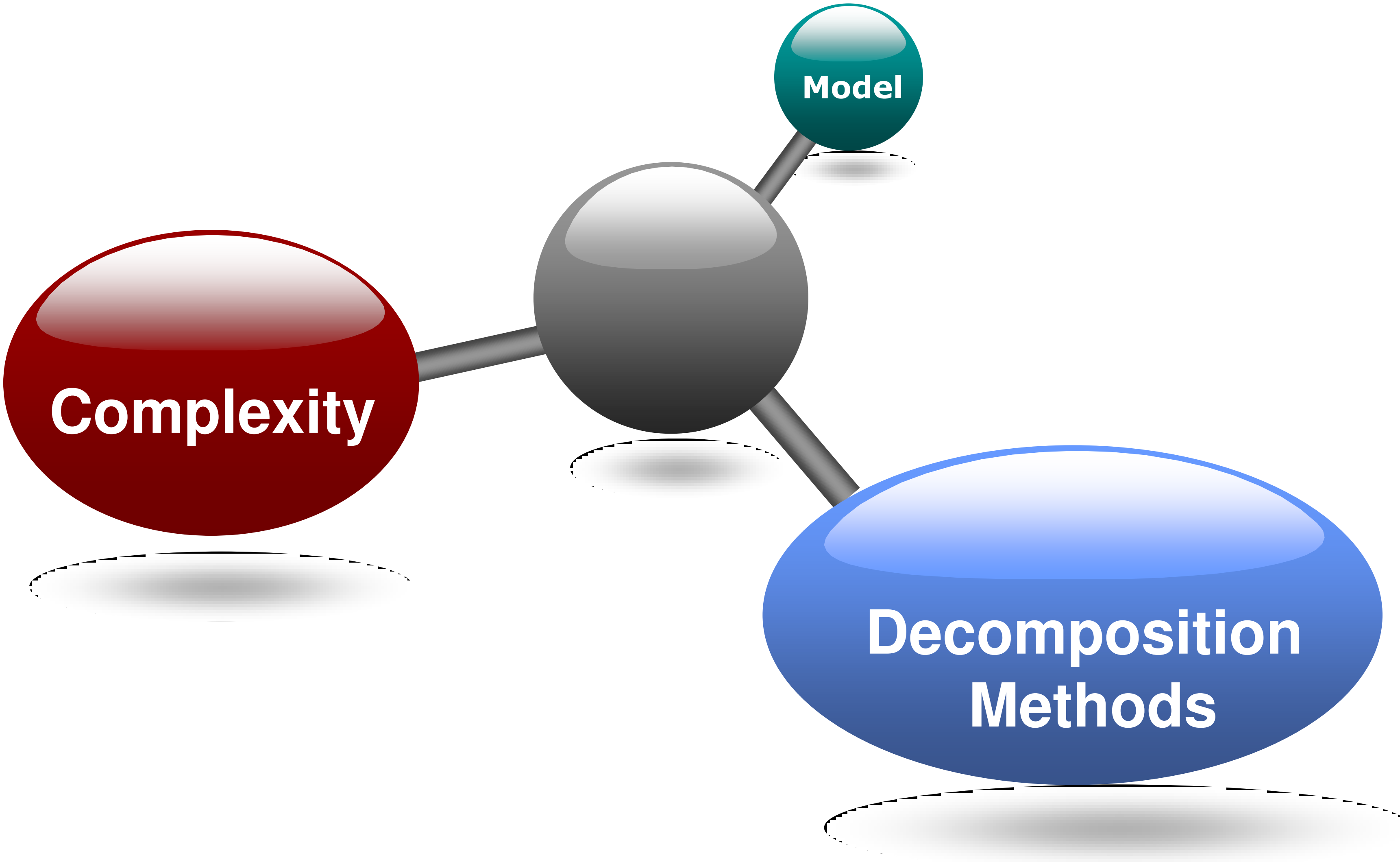
FAIR OPTIMIZATION

- MAX-MIN

[Snow and Freuder, Dubois and Fortemps, Bouveret and Lematre]

- ***Limited expressiveness***
  - *functions on one variable/constraint*
- ***No complexity analysis***

Overview



The Model

- $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$ **is a set of valuation functions**
- $\mathcal{F}_i = \langle w_i, \bigoplus_i \rangle$ **is such that**
  - $w_i : \bar{X}_i \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X}_i \subseteq Var$
  - $\bigoplus_i$ is a *commutative*, *associative*, and *closed* binary operator
- $\mathcal{F}_i(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}_i\}} w_i(X, u)$

$$\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$$

The Model

- $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ **is a set of valuation functions**
- $\mathcal{F}_i = \langle w_i, \bigoplus_i \rangle$ **is such that**
  - $w_i : \bar{X}_i \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X}_i \subseteq Var$
  - $\bigoplus_i$ is a *commutative*, *associative*, and *closed* binary operator
- $\mathcal{F}_i(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}_i\}} w_i(X, u)$

The Santa Claus Problem:

(possible solutions)

$\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$

10	2	9
10	3	2
3	6	9
4	6	6
4	6	9
⋮	⋮	⋮

$\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

The Model

- $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$ **is a set of valuation functions**
- $\mathcal{F}_i = \langle w_i, \bigoplus_i \rangle$ **is such that**
  - $w_i : \bar{X}_i \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X}_i \subseteq Var$
  - $\bigoplus_i$ is a *commutative*, *associative*, and *closed* binary operator
- $\mathcal{F}_i(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}_i\}} w_i(X, u)$

The Santa Claus Problem:

(possible solutions)

$\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$

10	2	9
10	3	2
3	6	9
4	6	6
4	6	9
⋮	⋮	⋮

$\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

The Model

- $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$ **is a set of valuation functions**
- $\mathcal{F}_i = \langle w_i, \bigoplus_i \rangle$ **is such that**
  - $w_i : \bar{X}_i \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X}_i \subseteq Var$
  - $\bigoplus_i$ is a *commutative*, *associative*, and *closed* binary operator
- $\mathcal{F}_i(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}_i\}} w_i(X, u)$

The Santa Claus Problem:

(possible solutions)

$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$
10	2	9
10	3	2
3	6	9
4	6	6
4	6	9
⋮	⋮	⋮

$\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

$\text{lex}\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

The Model

- $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ **is a set of valuation functions**
- $\mathcal{F}_i = \langle w_i, \bigoplus_i \rangle$ **is such that**
  - $w_i : \bar{X}_i \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X}_i \subseteq Var$
  - $\bigoplus_i$ is a *commutative*, *associative*, and *closed* binary operator
- $\mathcal{F}_i(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}_i\}} w_i(X, u)$

The Santa Claus Problem:

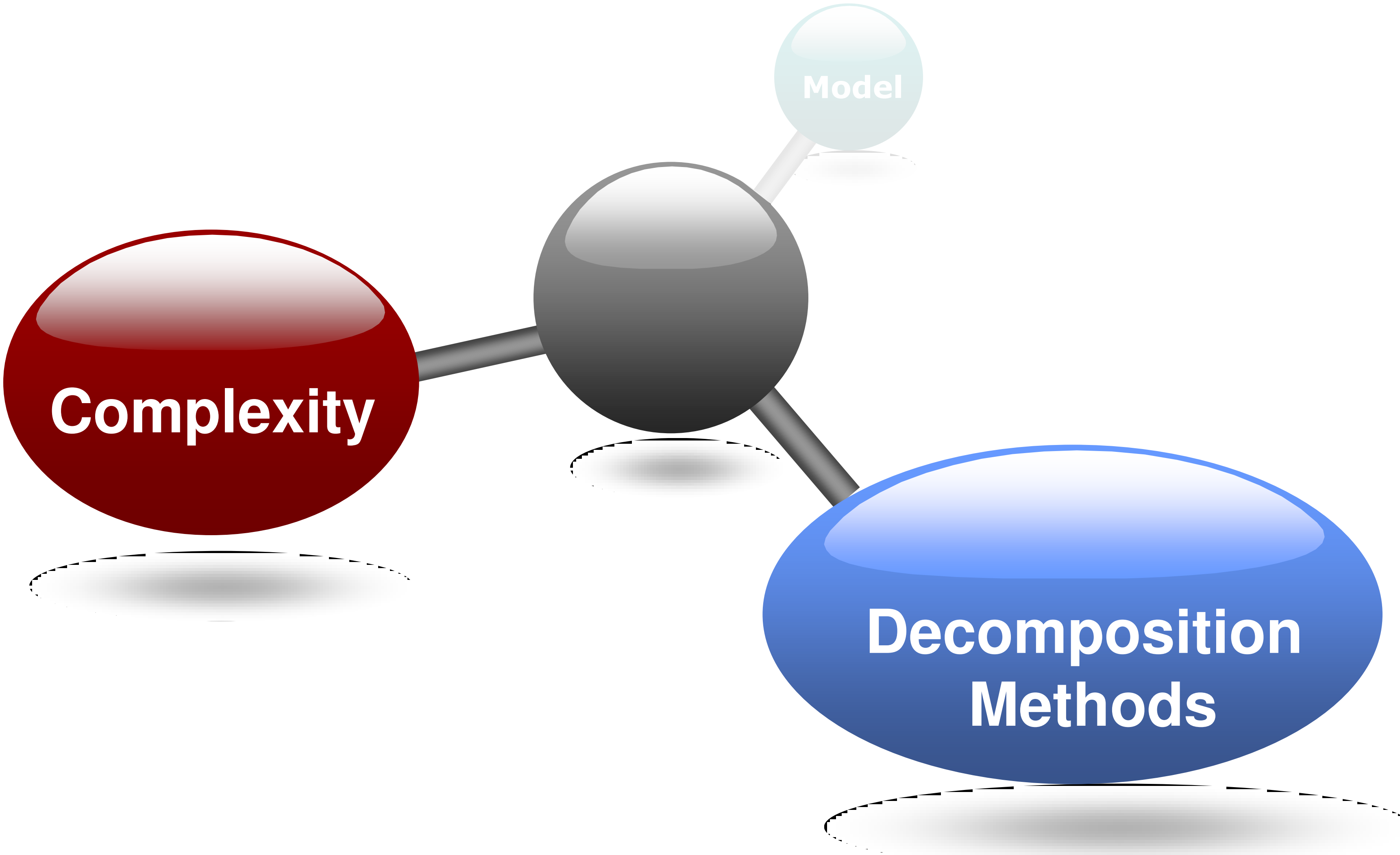
(possible solutions)

$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$
10	2	9
10	3	2
3	6	9
4	6	6
4	6	9
⋮	⋮	⋮

$\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

$\mathbf{lex}\max_\theta \ \min_{\mathcal{F} \in L} \mathcal{F}(\theta)$

Overview



Complexity of (LEX)MAX-MIN Solutions

- **Constraint satisfaction is NP-hard**
  - Even without optimization functions…
- **Tractable classes of CSPs**
  - Based on the values in the constraint relations
  - <u>Based on the structure of the constraint scopes</u>

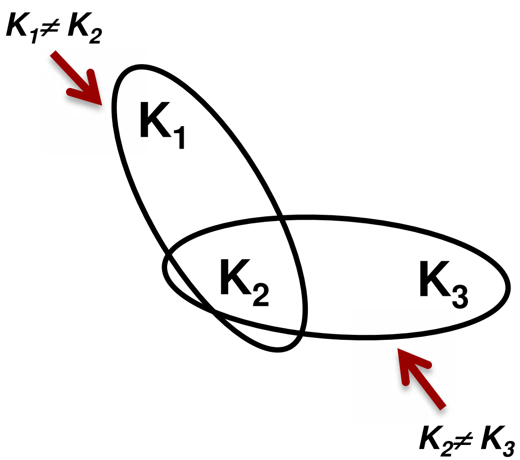
Complexity of (LEX)MAX-MIN Solutions

- **Constraint satisfaction is NP-hard**
  - Even without optimization functions…
- **Tractable classes of CSPs**
  - Based on the values in the constraint relations
  - <u>Based on the structure of the constraint scopes</u>
    - Treewidth [Dechter & Pearl]

Complexity of (LEX)MAX-MIN Solutions

- **Constraint satisfaction is NP-hard**
  - Even without optimization functions…
- **Tractable classes of CSPs**
  - Based on the values in the constraint relations
  - <u>Based on the structure of the constraint scopes</u>

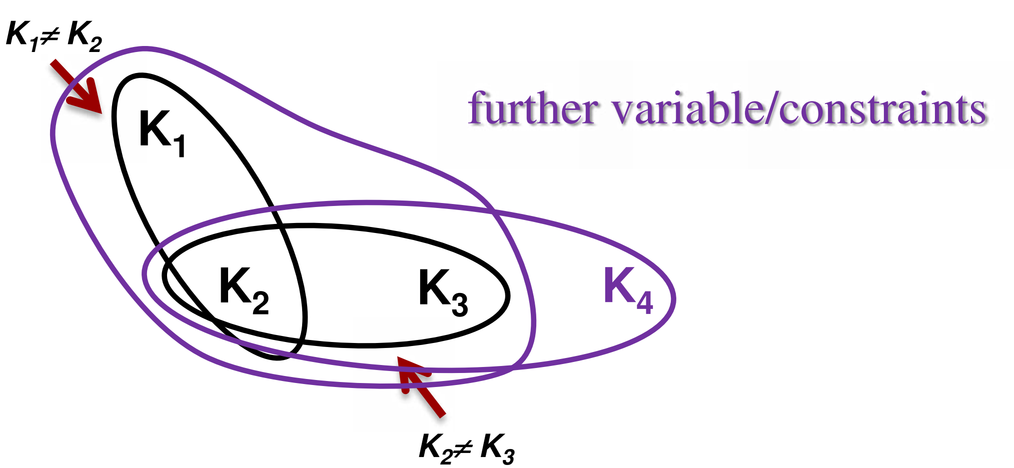
- **CONSTRAINT HYPERGRAPH**



Complexity of (LEX)MAX-MIN Solutions

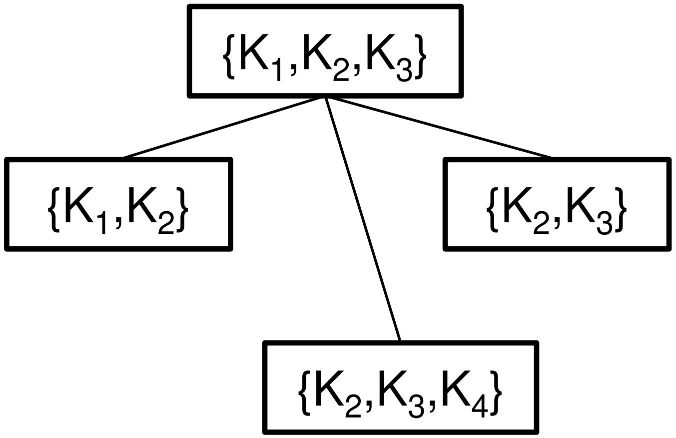
- **Constraint satisfaction is NP-hard**
  - Even without optimization functions…
- **Tractable classes of CSPs**
  - Based on the values in the constraint relations
  - <u>Based on the structure of the constraint scopes</u>

- **CONSTRAINT HYPERGRAPH**

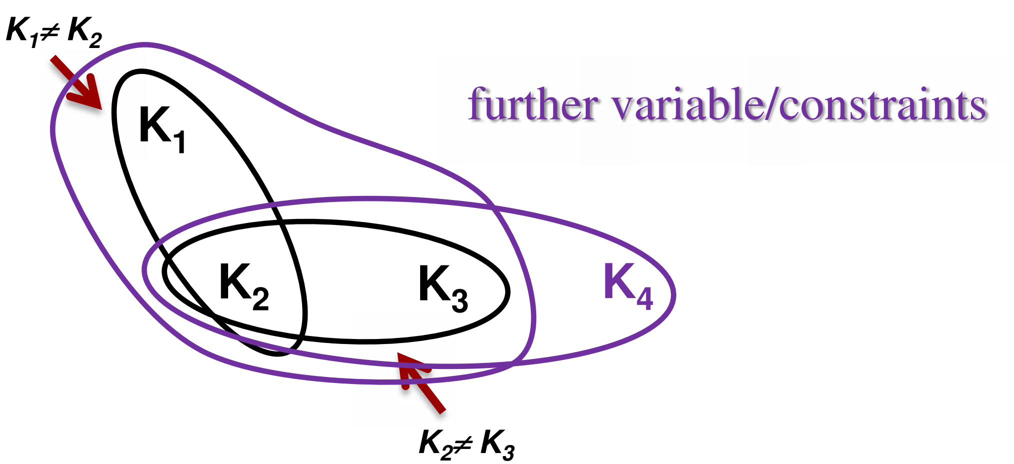


Complexity of (LEX)MAX-MIN Solutions

- **JOIN TREE**
  - Vertices correspond to the hyperedges
  - Each variable induces a connected subtree

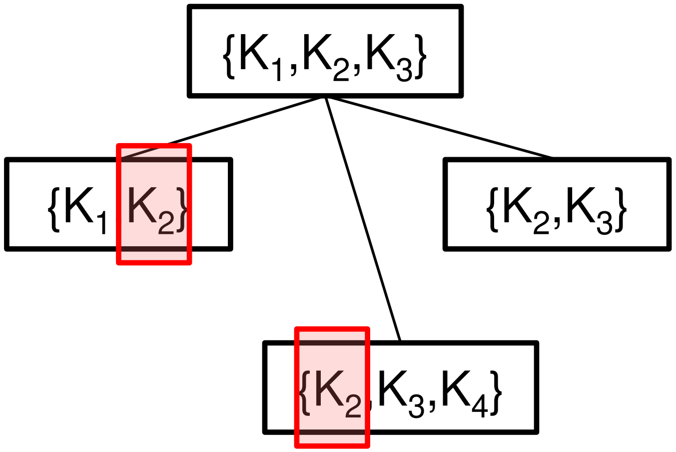


- **CONSTRAINT HYPERGRAPH**

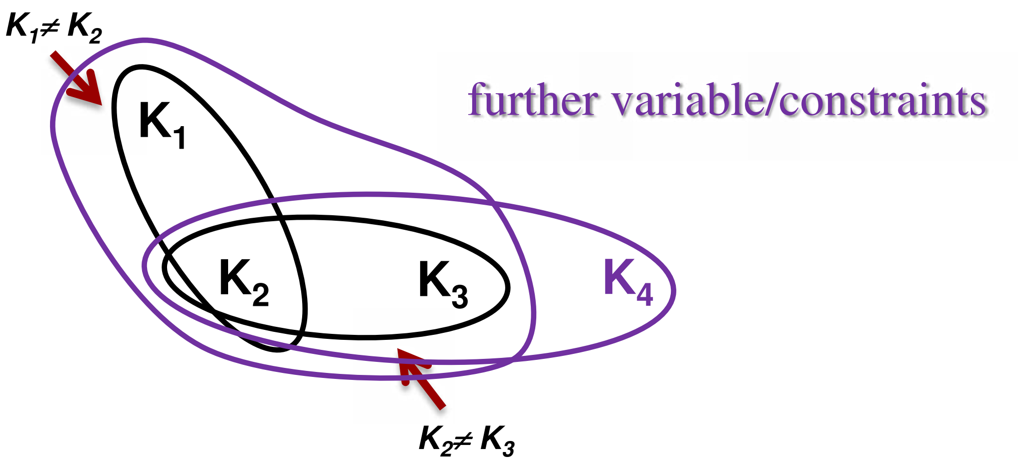


Complexity of (LEX)MAX-MIN Solutions

- **JOIN TREE**
  - Vertices correspond to the hyperedges
  - Each variable induces a connected subtree



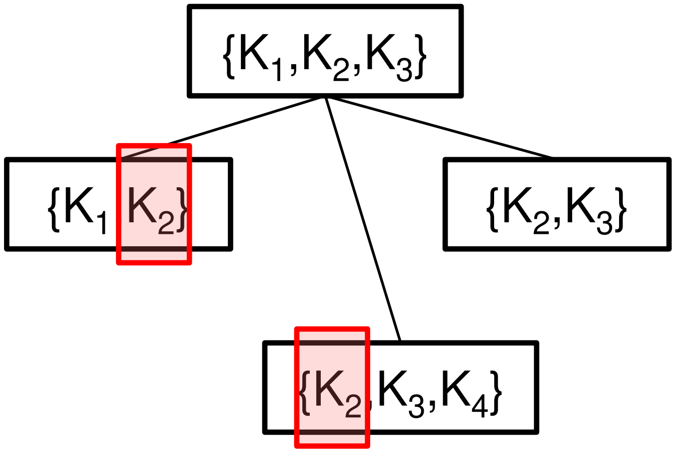
- **CONSTRAINT HYPERGRAPH**



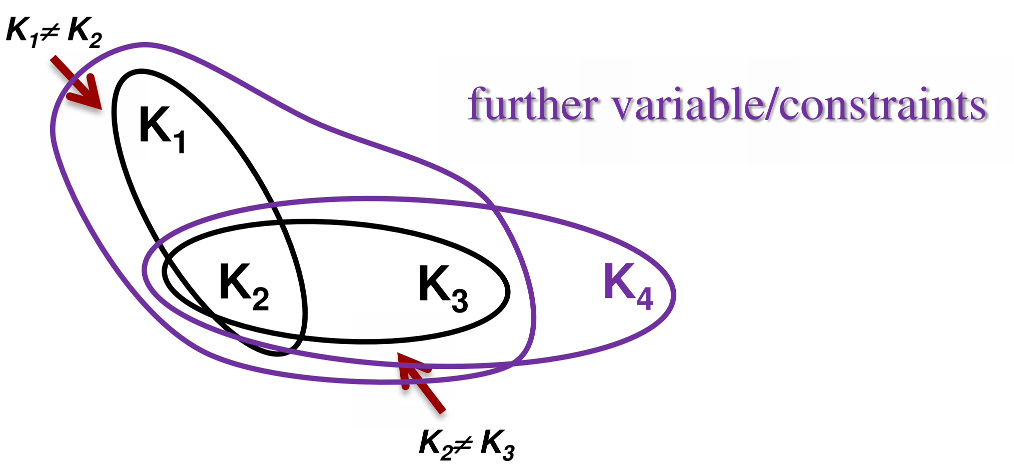
Complexity of (LEX)MAX-MIN Solutions

- **JOIN TREE**
  - Vertices correspond to the hyperedges
  - Each variable induces a connected subtree

ACYCLIC CSP



- **CONSTRAINT HYPERGRAPH**



Complexity of Acyclic Instances

[D, F]	1	h	∞
1			
k			
∞			

- **Restrictions on** $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

[D, F]	1	h	∞
1			
k			
∞	in **P**		

[Gottlob *et al.*]

- **Restrictions on** $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\text{dom}(\mathcal{F})| \leq$ D
  - $|L| \leq$F

Complexity of Acyclic Instances

$[\mathrm{D}, \mathrm{F}]$	1	h	∞
1	in **P**		
k	in **P**		
∞	in **P**		

[Gottlob *et al.*]

- **Restrictions on** $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

$[\mathrm{D}, \mathrm{F}]$	1	h	∞
1	in **P**		in **P**
k	in **P**		
∞	in **P**		

[Gottlob *et al.*]

- **Restrictions on** $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

$[\mathrm{D},\mathrm{F}]$	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		
∞	in **P**		

[Gottlob *et al.*]

- **Restrictions on** $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$
  - $\max_{\mathcal{F}\in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		
∞	in **P**	weakly **NP-hard**	

[Gottlob *et al.*]

- Reduction from «Partition»
- Pseudo-polynomial

- **Restrictions on** $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

Reduction from «Set Packing»

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	

[Gottlob *et al.*]

- Reduction from «Partition»
- Pseudo-polynomial

- **Restrictions on** $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

Reduction from «Set Packing»

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

[Gottlob *et al.*]

- Reduction from «Partition»
- Pseudo-polynomial

- **Restrictions on** $L = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Complexity of Acyclic Instances

Dynamic programming

Reduction from «Set Packing»

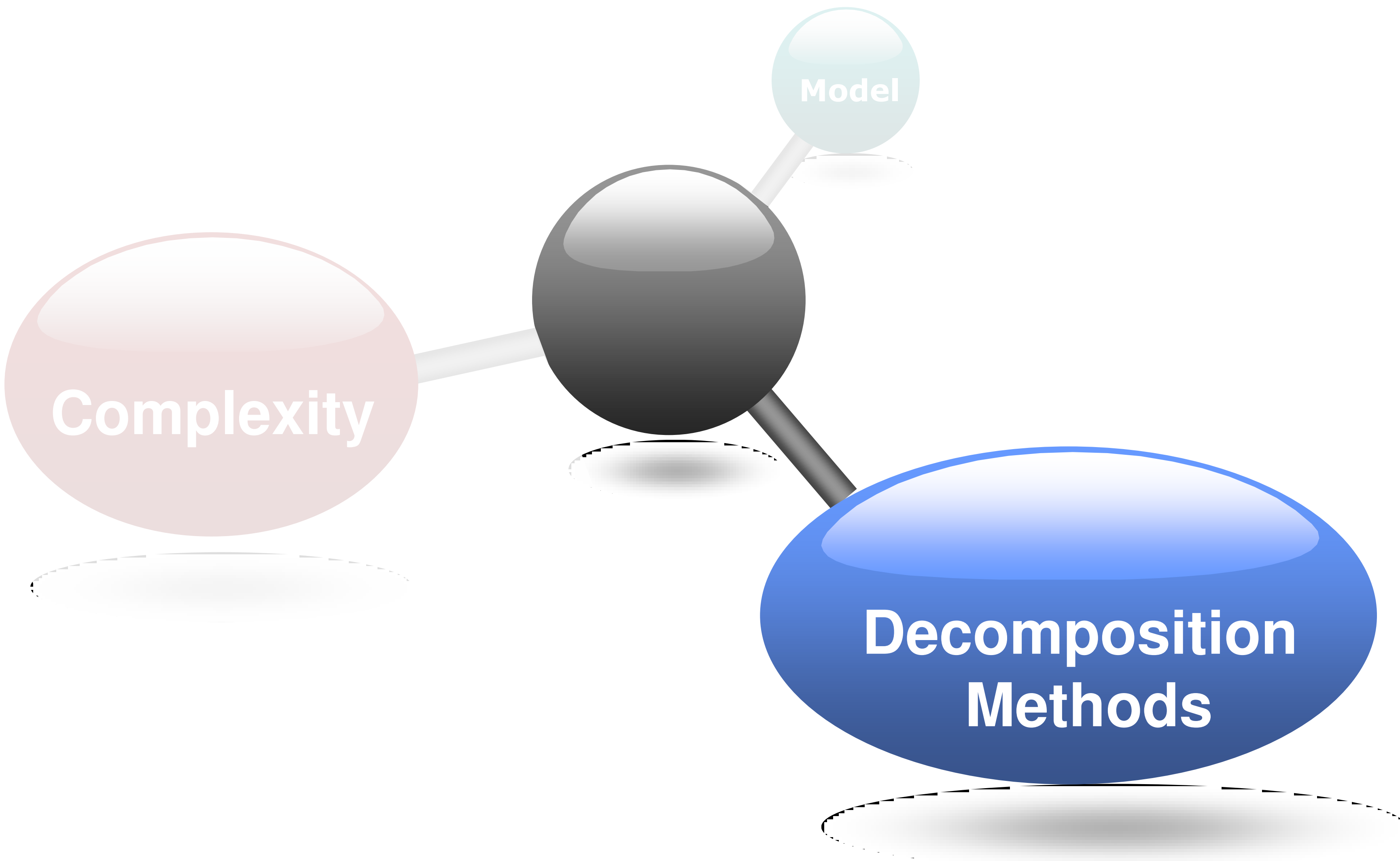
[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**	***a novel machinery is needed***	**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

[Gottlob *et al.*]

- Reduction from «Partition»
- Pseudo-polynomial

- **Restrictions on** $L = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$
  - $\max_{\mathcal{F} \in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$
  - $|L| \leq \mathrm{F}$

Overview



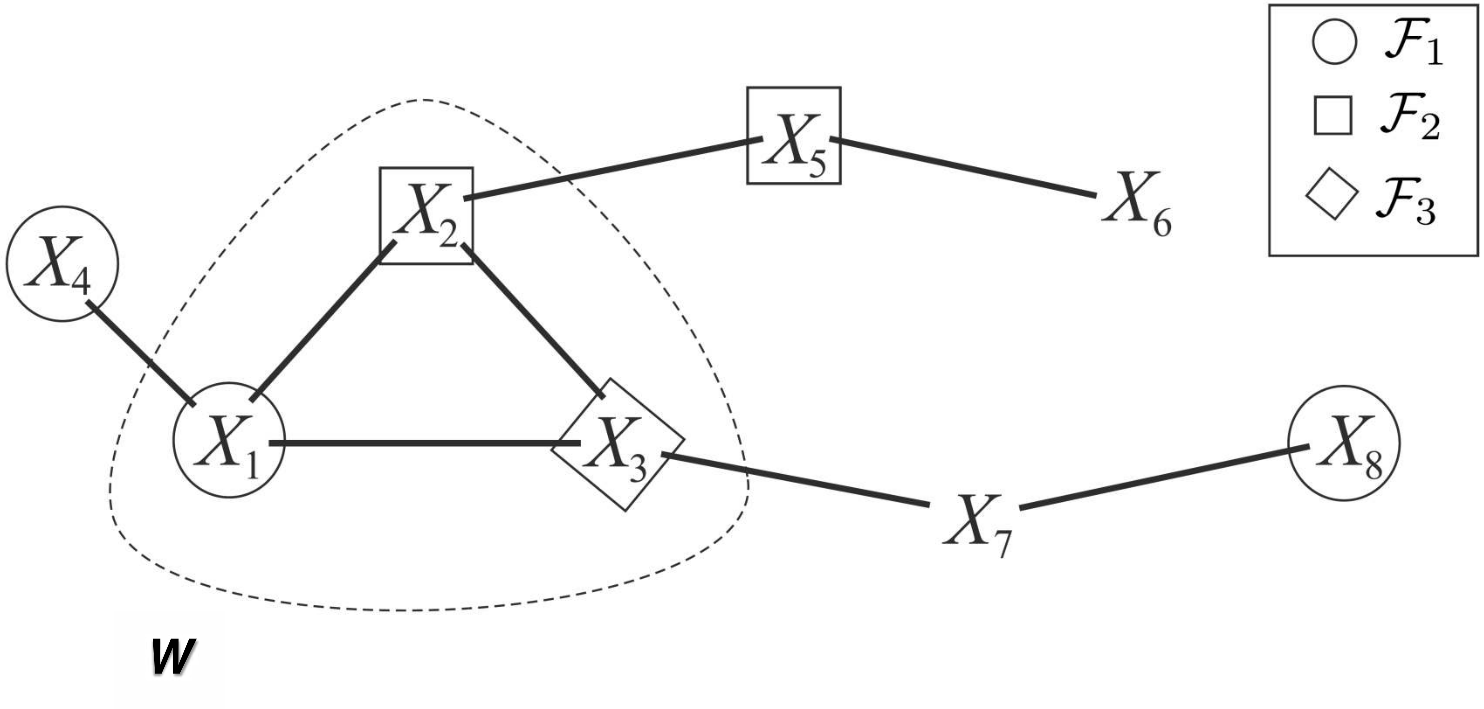
Key Ideas

Guards for Valuation Functions

+

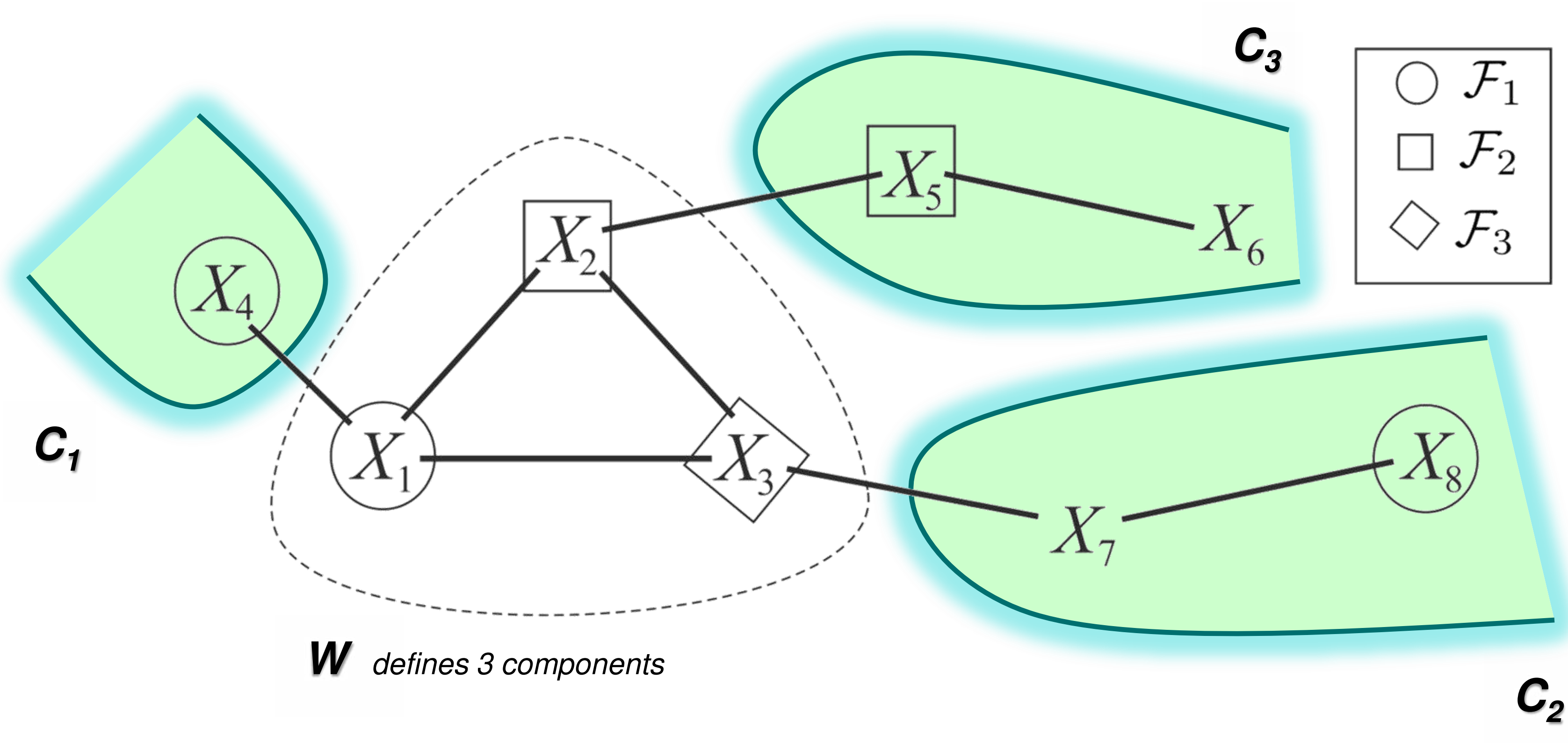
Decomposition Methods

Guards for Valuation Functions



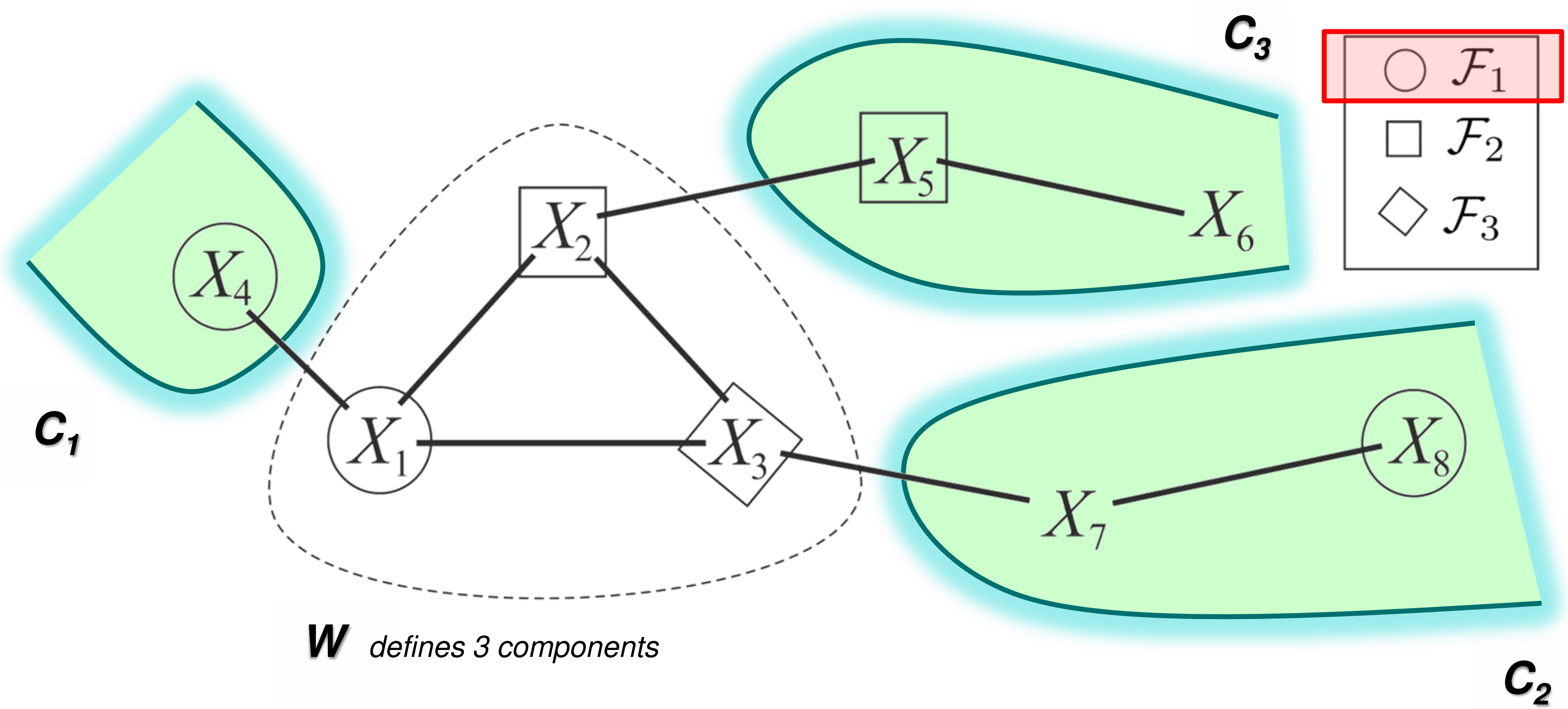
- **A set of variables *W* is a *guard* for a valuation function if**
  - separates the hypergraph in components where its domain variables do not occur together with any variable occurring in other valuation functions

Guards for Valuation Functions



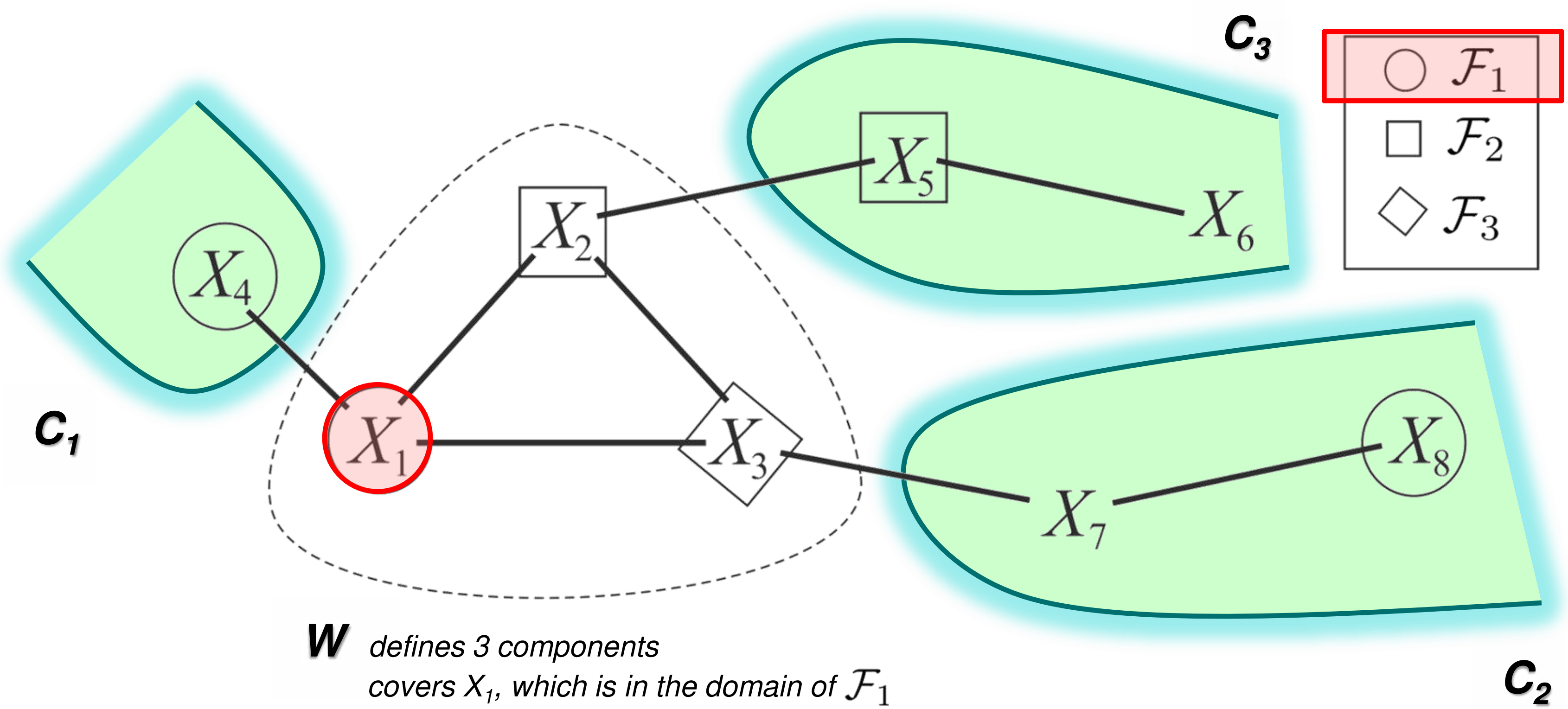
- **A set of variables *W* is a *guard* for a valuation function if**
  - separates the hypergraph in components where its domain variables do not occur together with any variable occurring in other valuation functions

Guards for Valuation Functions



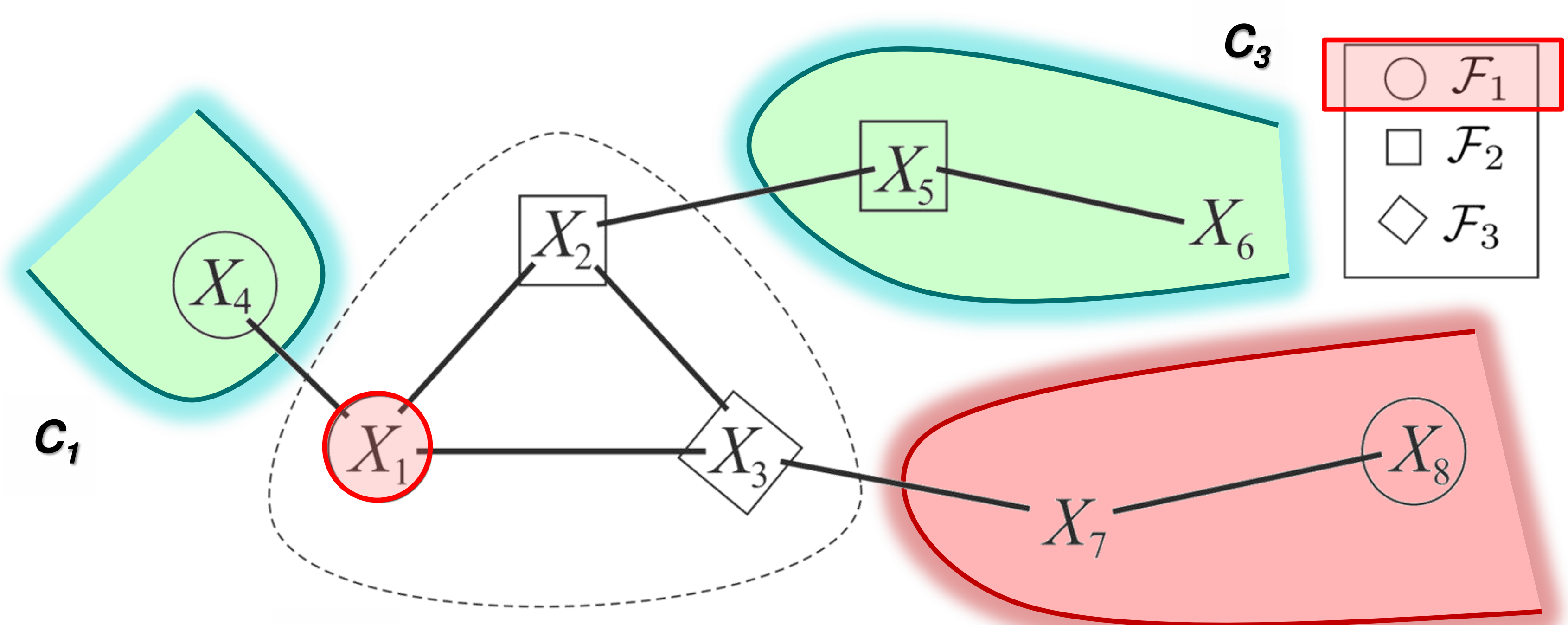
- **A set of variables *W* is a *guard* for a valuation function if**
  - separates the hypergraph in components where its domain variables do not occur together with any variable occurring in other valuation functions

Guards for Valuation Functions



- **A set of variables *W* is a *guard* for a valuation function if**
  - separates the hypergraph in components where its domain variables do not occur together with any variable occurring in other valuation functions

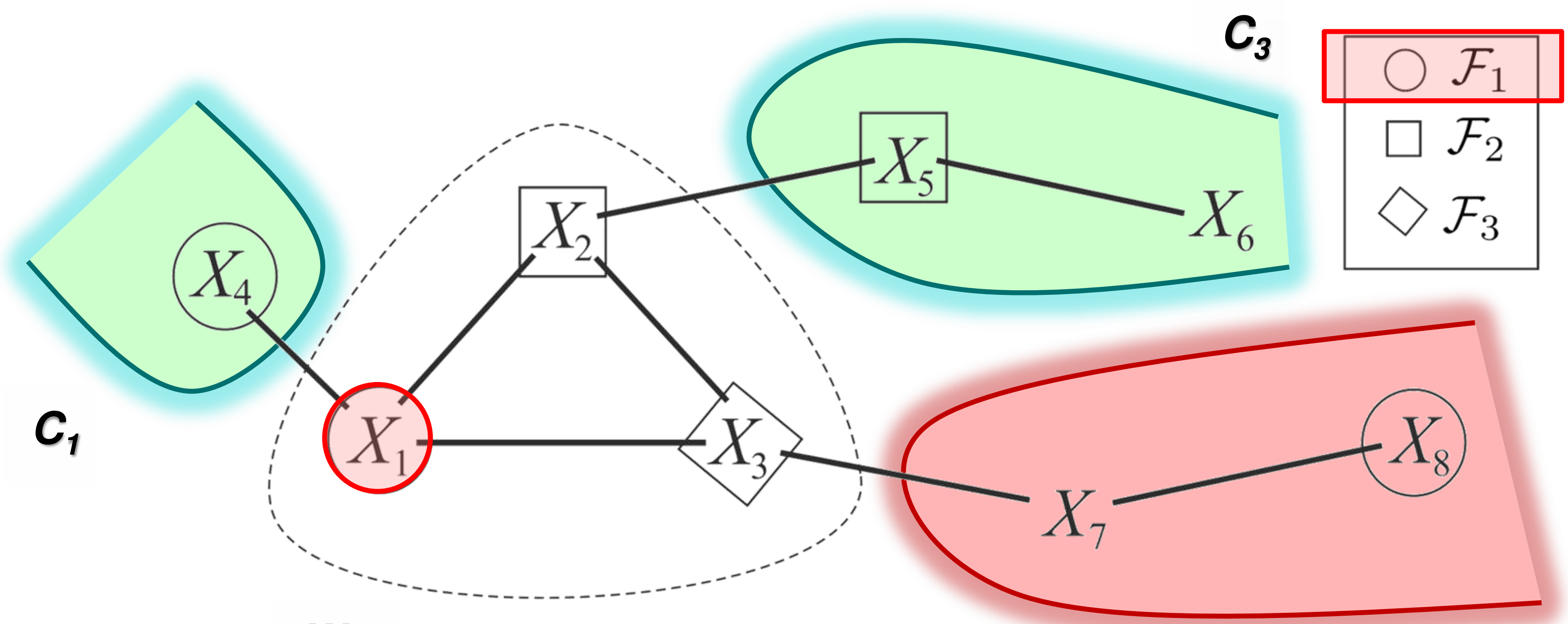
Guards for Valuation Functions



W *defines 3 components*
covers X_1, which is in the domain of $\mathcal{F}_1$
C_2 (and C_1) does not contain variables in the domain of $\mathcal{F}_2$ and $\mathcal{F}_3$

- **A set of variables *W* is a *guard* for a valuation function if**
  - separates the hypergraph in components where its domain variables do not occur together with any variable occurring in other valuation functions

Guards for Valuation Functions



W *defines 3 components*

covers X_1, which is in the domain of $\mathcal{F}_1$

C_2 *(and* ***C_1****) does not contain variables in the domain of $\mathcal{F}_2$ and $\mathcal{F}_3$*

is a guard for $\mathcal{F}_1$; in fact, it is also a guard for the other functions

Key Ideas

Guards for Valuation Functions

+

Decomposition Methods

Decomposition Methods

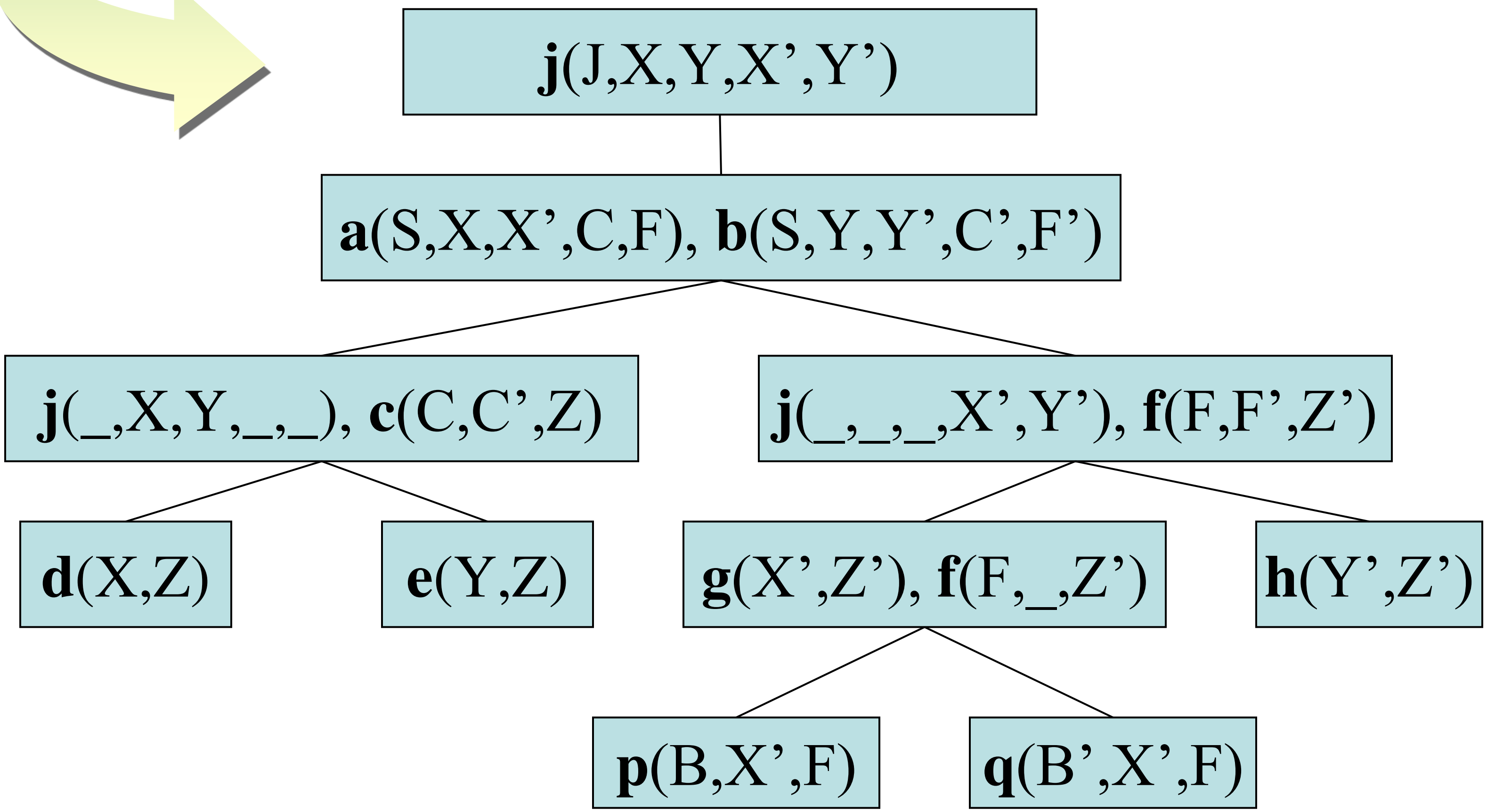
- **Common Ideas**
  - Generalize the notion of graph or hypergraph acyclicity
  - Associate a width to each instance, expressing its degree of cyclicity
  - Polynomial time algorithms for bounded-width CSP instances, running in $O(n\ w+1 \cdot \log n)$
  - Bounded-width CSP instances can be recognized in polynomial time
  - Bounded-width decompositions can be computed in polynomial time
- **Noticeable Examples**
  - Tree decompositions
  - (Generalized) Hypertree decompositions

Generalized Hypertree Decompositions

$$a(S,X,X',C,F) \quad b(S,Y,Y',C',F') \quad c(C,C',Z) \quad d(X,Z)$$

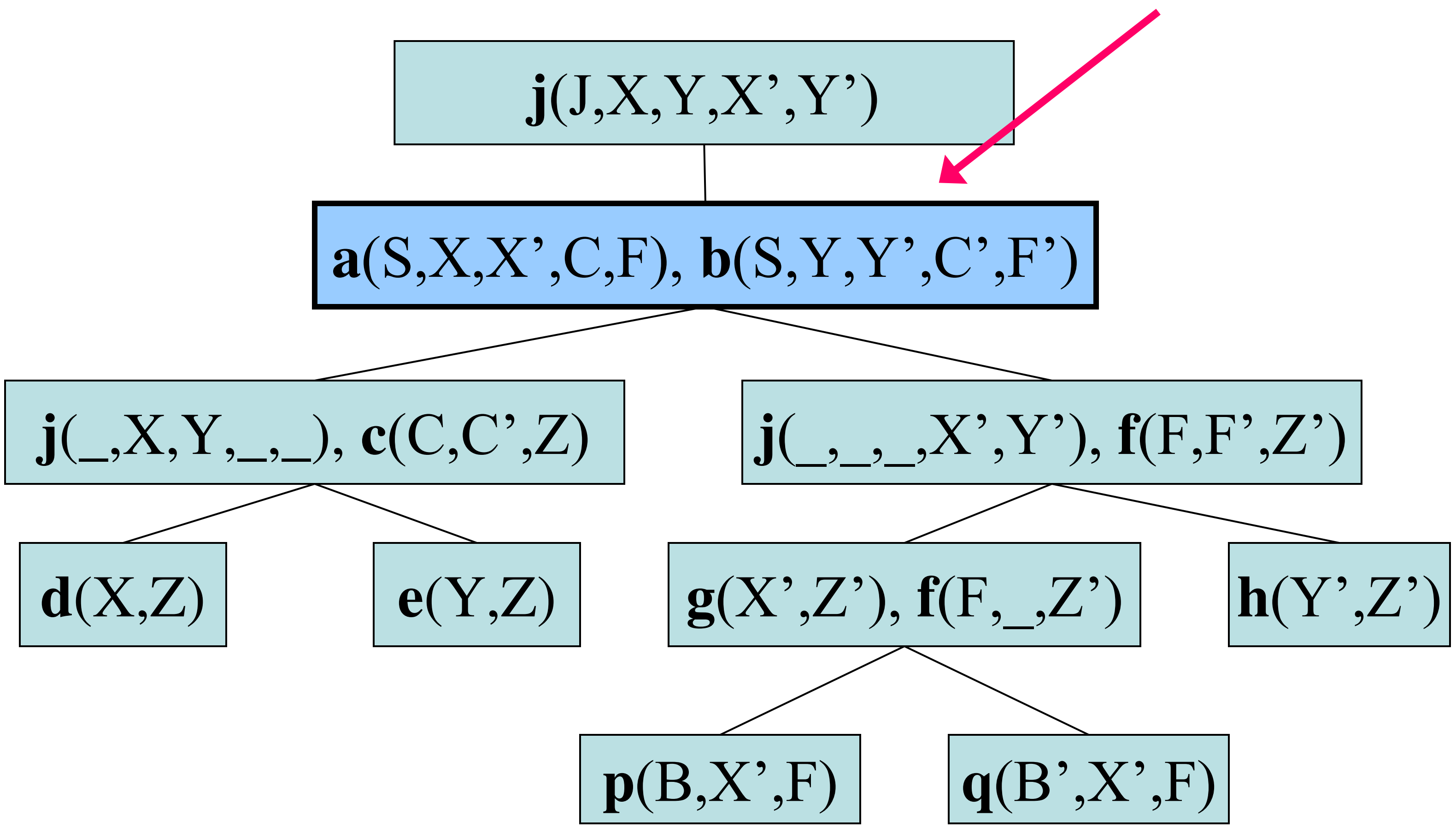
$$e(Y,Z) \quad f(F,F',Z') \quad g(X',Z') \quad h(Y',Z')$$

$$j(J,X,Y,X',Y') \quad p(B,X',F) \quad q(B',X',F)$$



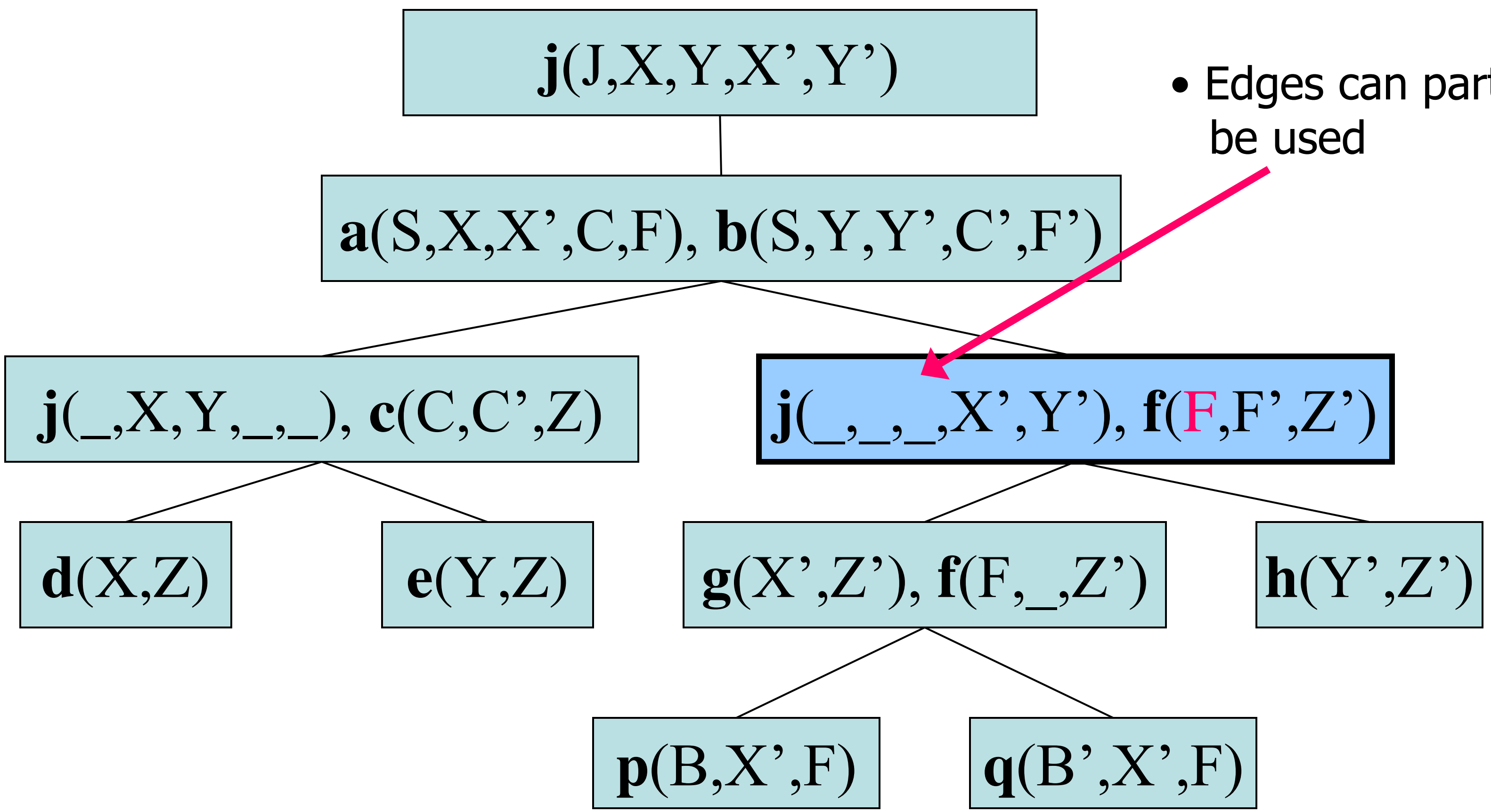
Basic Conditions$_{(1/2)}$

- We group edges

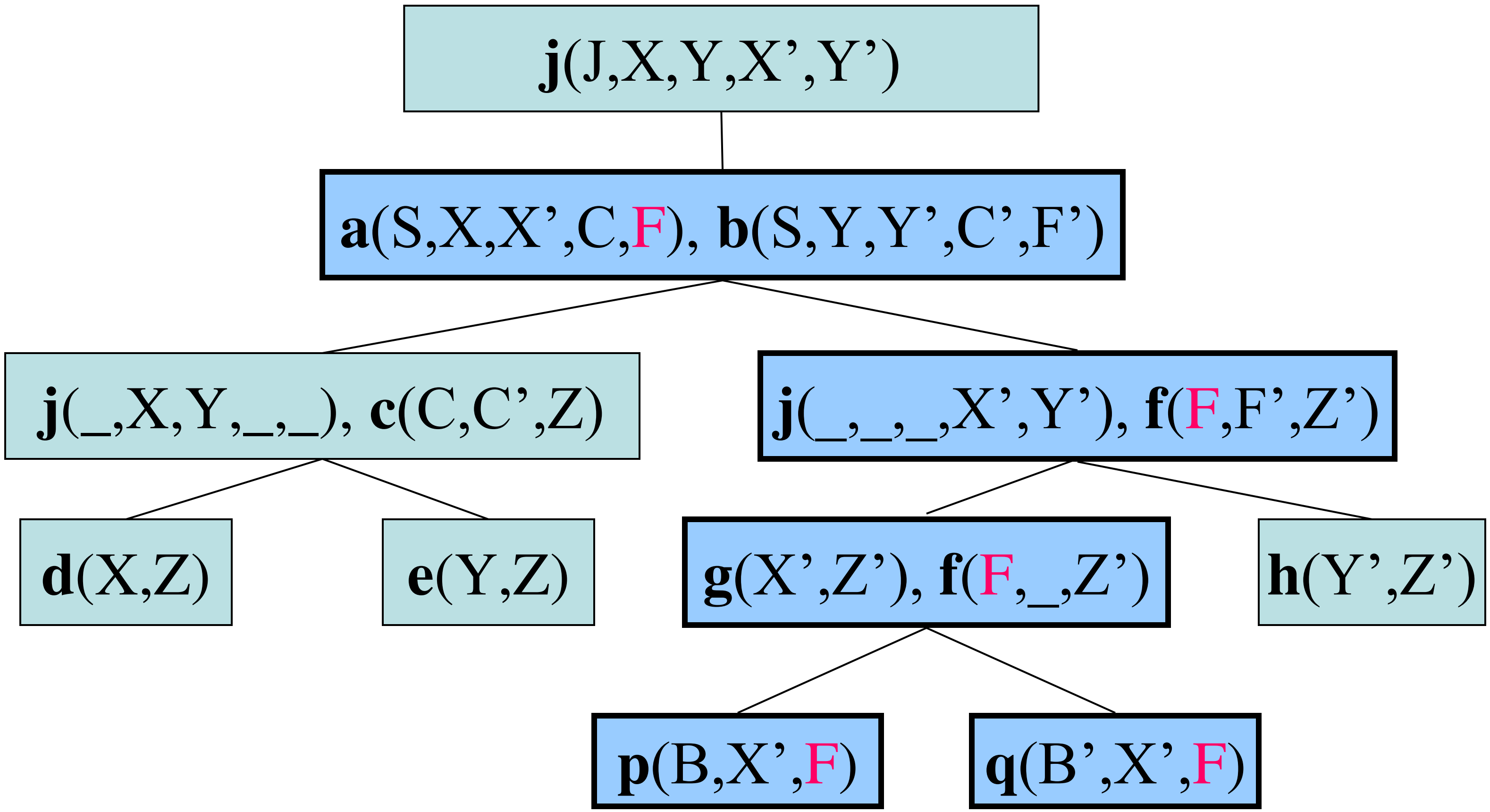


Basic Conditions$_{(2/2)}$

- Edges can partially be used



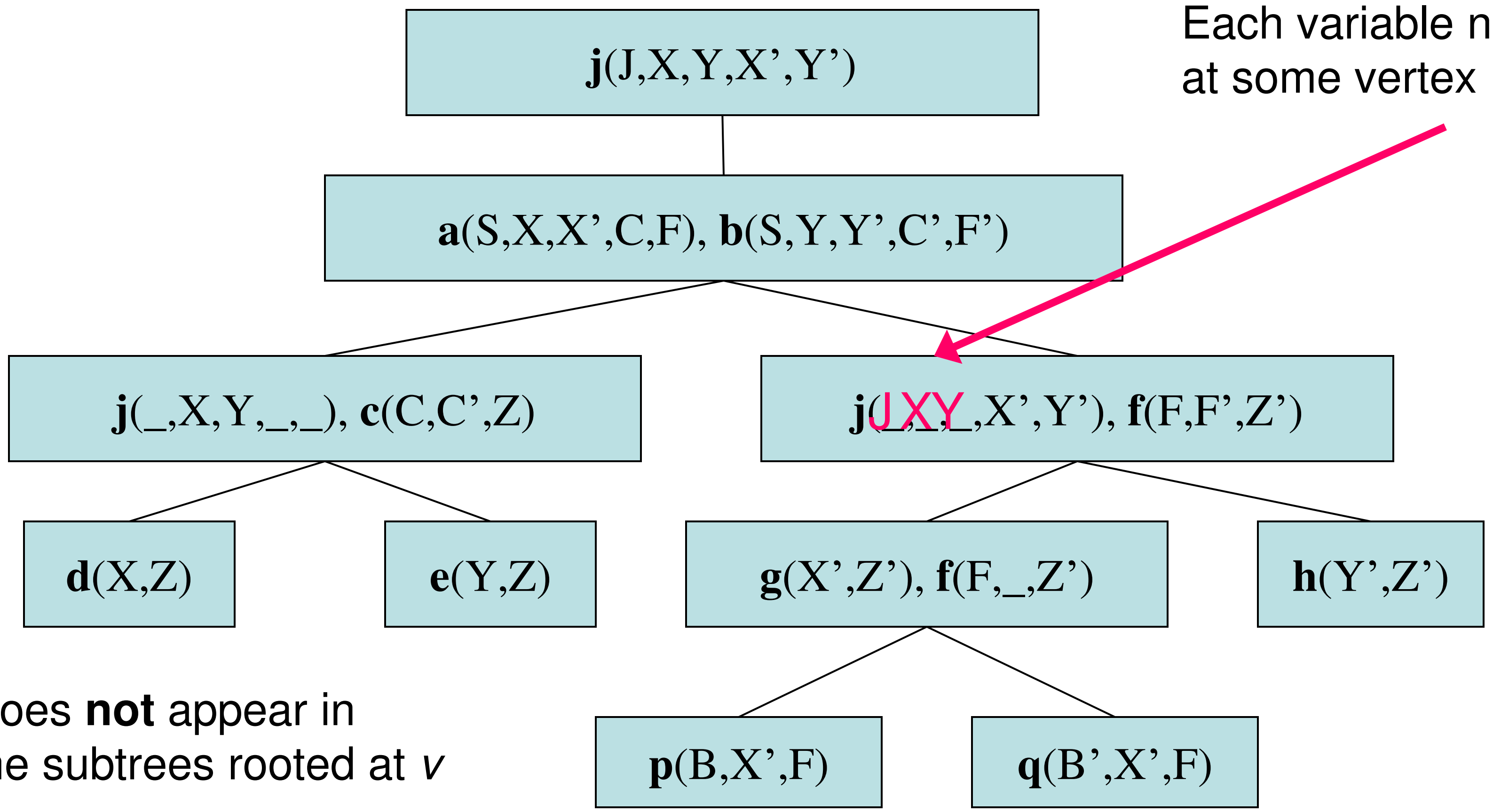
Connectdness Condition



Hypertree Decompositions (HTD)

HTD = Generalized HTD +Special Condition

Each variable not used
at some vertex *v*



Does **not** appear in
the subtrees rooted at *v*

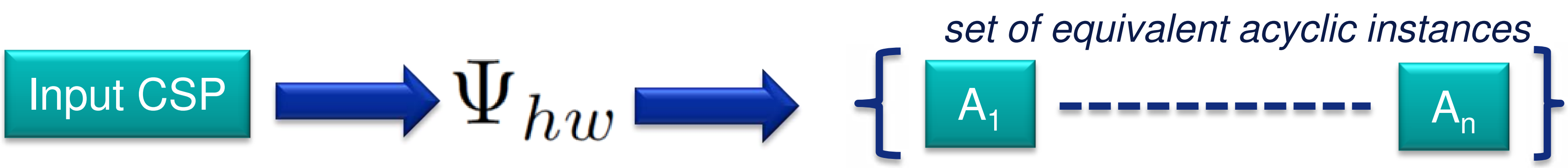
Key Ideas

Guards for Valuation Functions

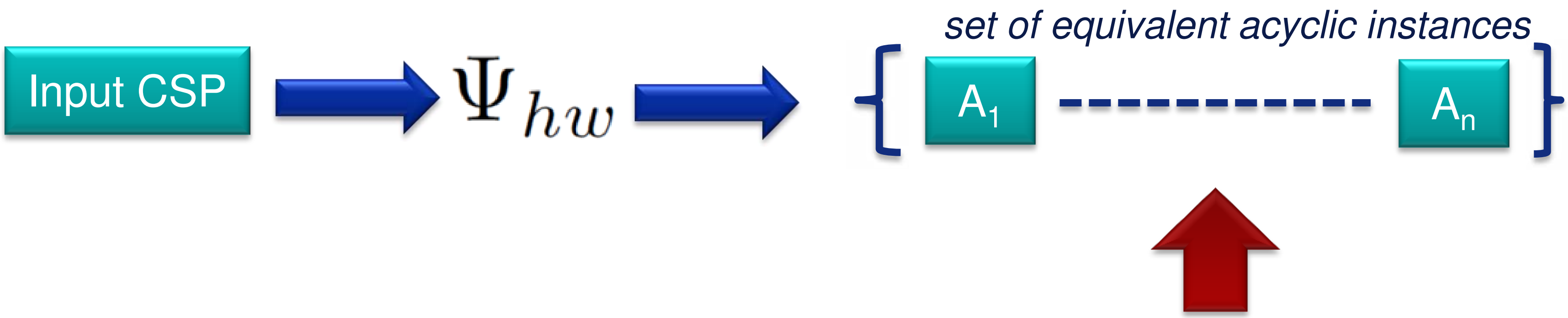
+

Decomposition Methods

Decomposition Methods and Guards

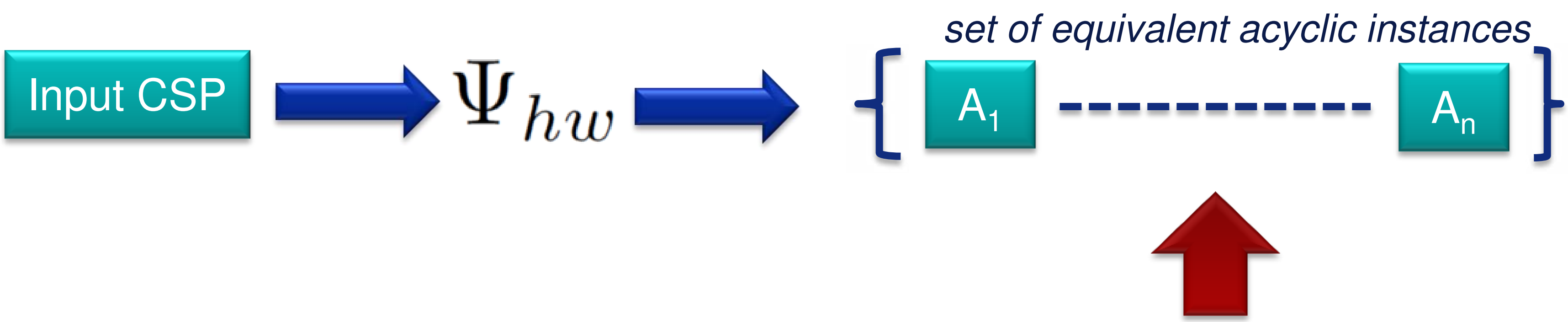


Decomposition Methods and Guards

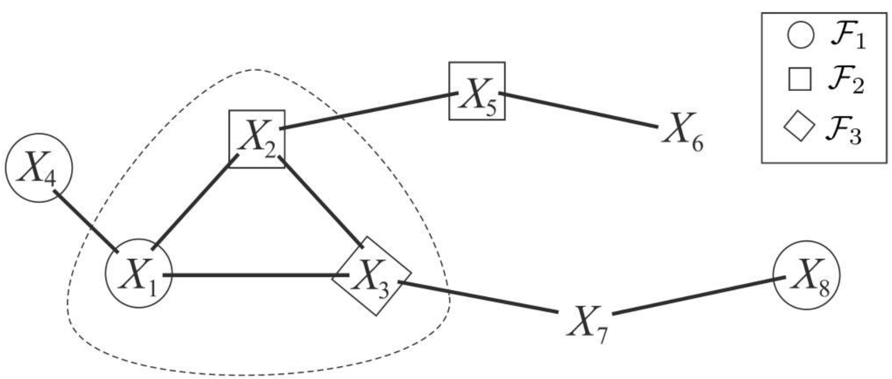


An instance is guarded via the given method if there is an output A_i such that each valution function is guarded by some hyperedge

Decomposition Methods and Guards



An instance is guarded via the given method if there is an output A_i such that each valuation function is guarded by some hyperedge



is guarded via hypertree decomposition (width k=3)

Main Results

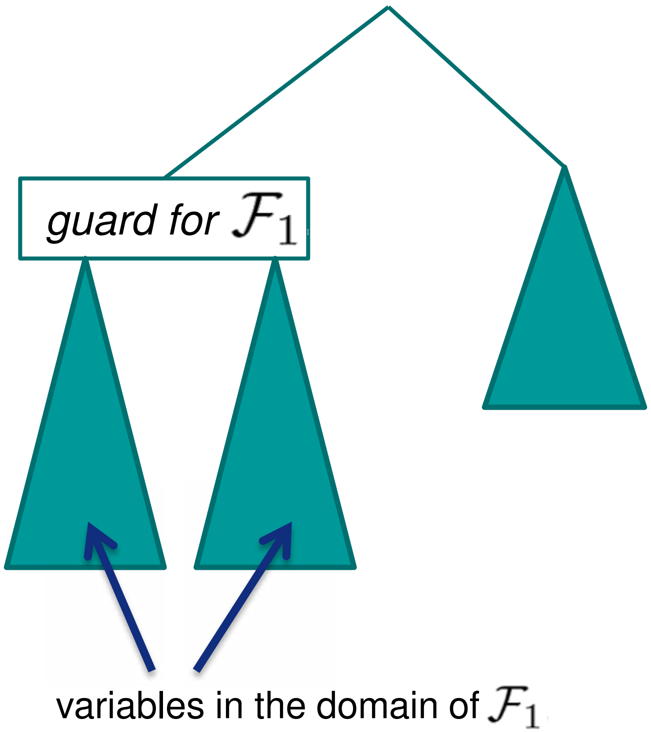
$[\mathrm{D}, \mathrm{F}]$	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

Main Results

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

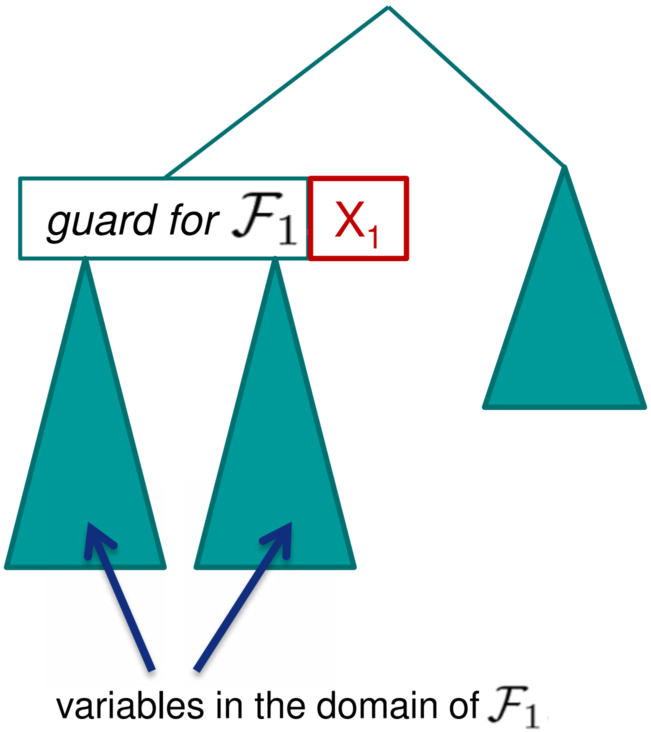
in **P**, if guarded via Ψ_{hw}

Proof Idea



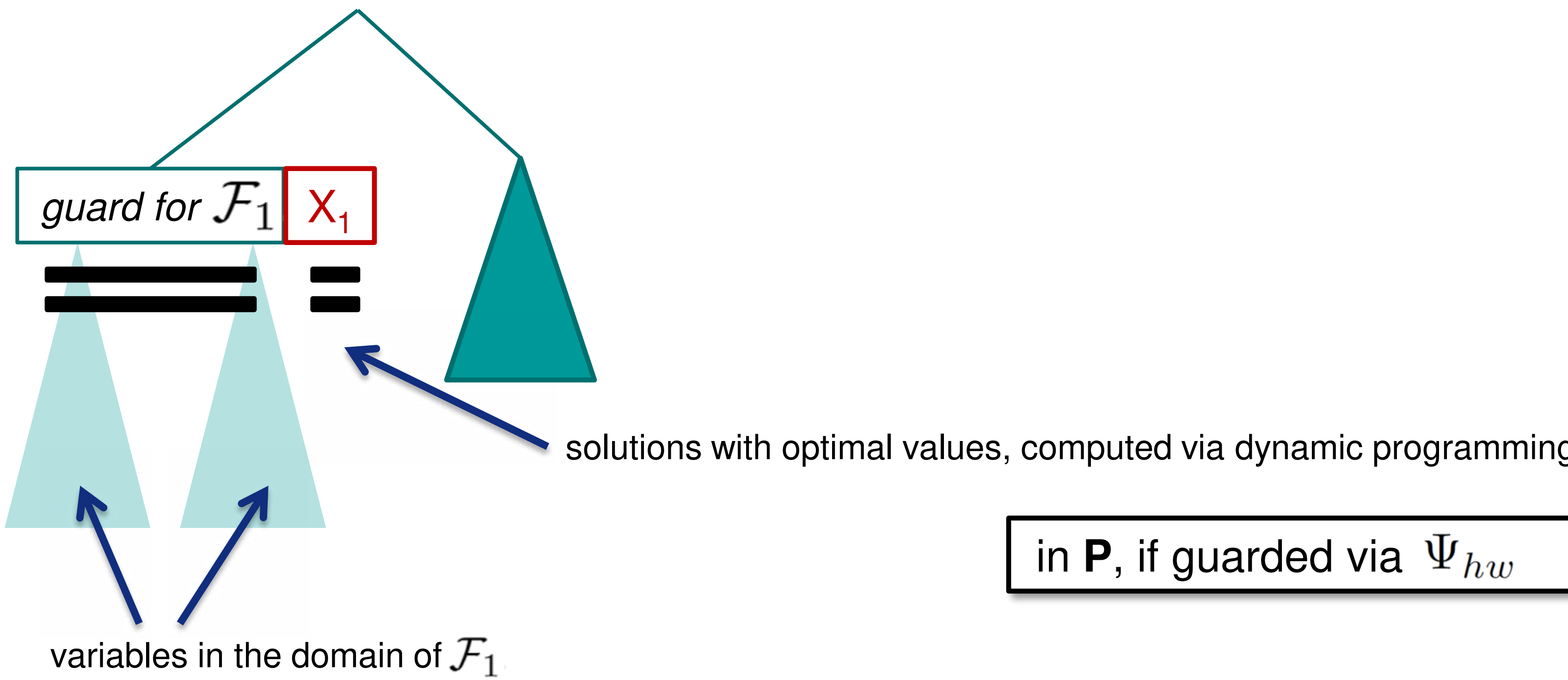
in **P**, if guarded via Ψ_{hw}

Proof Idea



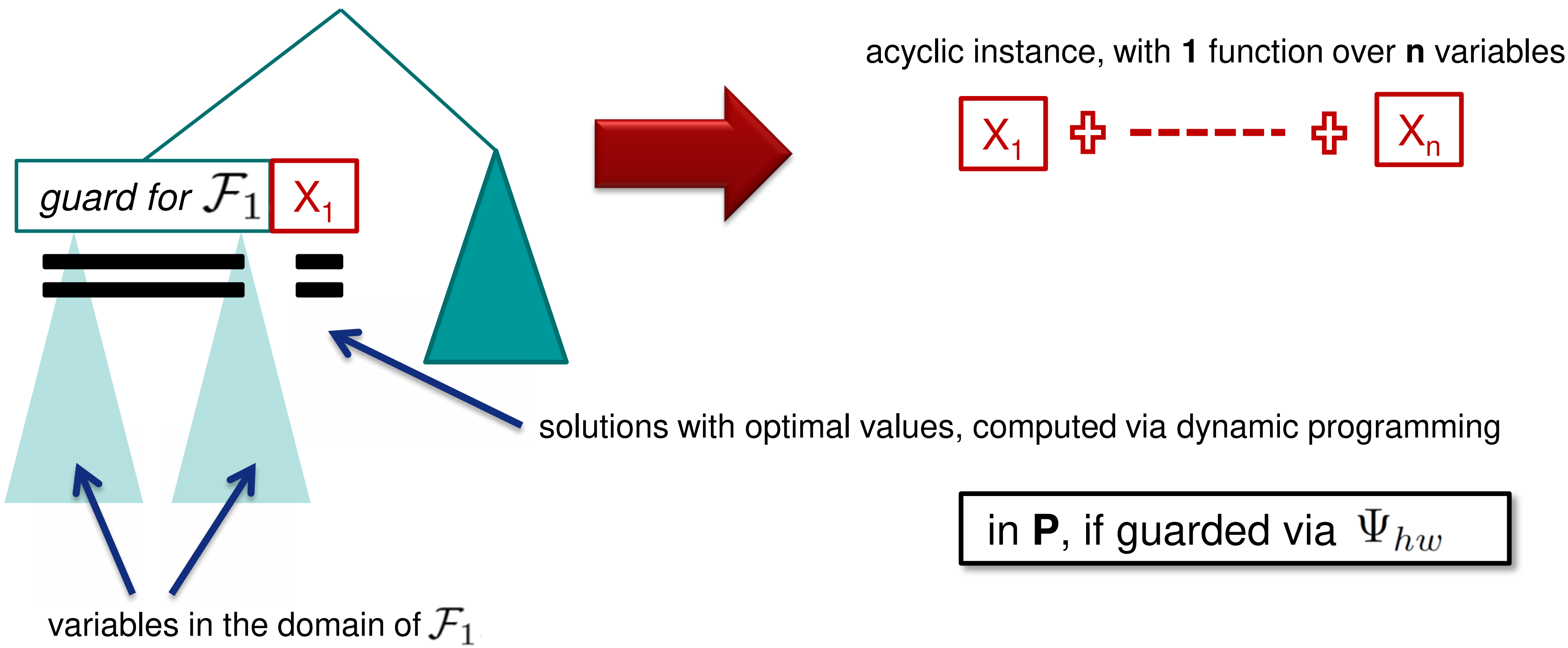
in **P**, if guarded via Ψ_{hw}

Proof Idea



in **P**, if guarded via Ψ_{hw}

Proof Idea


guard for $\mathcal{F}_1$
X_1
acyclic instance, with **1** function over **n** variables
X_1 + - - - - - - + X_n
solutions with optimal values, computed via dynamic programming
in **P**, if guarded via Ψ_{hw}
variables in the domain of $\mathcal{F}_1$

Main Results

$[\mathrm{D}, \mathrm{F}]$	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

in **P**, if guarded via Ψ_{hw}

Main Results

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**		**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

in **P**, if guarded via Ψ_{hw}

(acyclic) instances of this kind are always guarded via Ψ_{hw} width: h x k+1

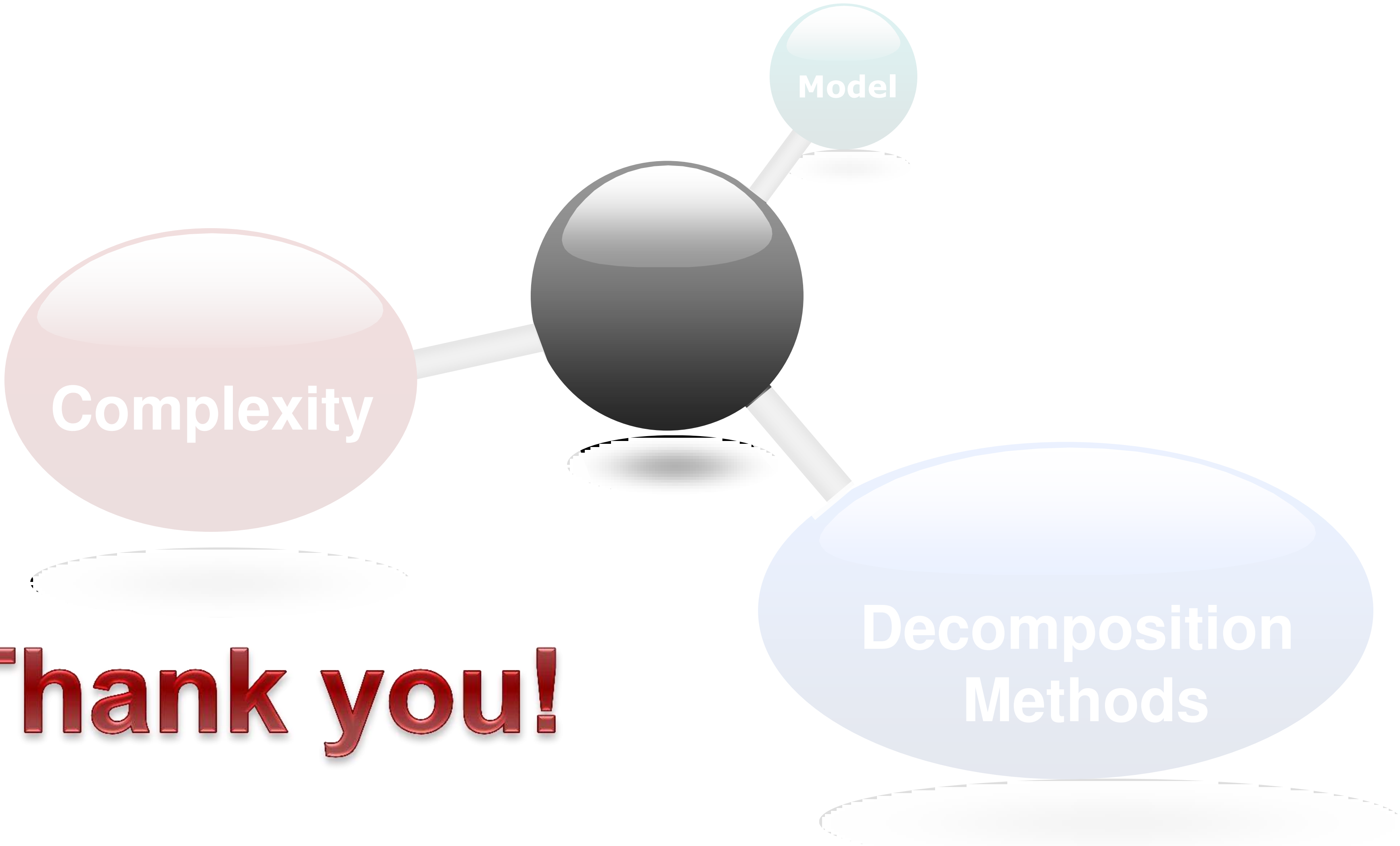
Main Results

[D, F]	1	h	∞
1	in **P**	in **P**	in **P**
k	in **P**	in **P**	**NP-hard**
∞	in **P**	weakly **NP-hard**	**NP-hard**

in **P**, if guarded via Ψ_{hw}

(acyclic) instances of this kind are always guarded via Ψ_{hw} width: h x k+1

Overview



Thank you!