Living with Computational Complexity

Georg Gottlob

Rende/Cosenza
25 maggio 2017

My Connection to Calabria

1982,1989, Mimmo Saccá

Georg Gottlob, **Nicola Leone**, Helmut Veith:
Second Order Logic and the Weak Exponential Hierarchies. MFCS 1995: 66-81

Thomas Eiter, Georg Gottlob, **Nicola Leone**:
Semantics and Complexity of Abduction from Default Theories. IJCAI (1) **1995**: 870-877; AIJ 1997

Georg Gottlob, **Nicola Leone, Francesco Scarcello**:
On the Complexity of Some Inductive Logic Programming Problems. ILP 1997: 17-32;

Thomas Eiter, Georg Gottlob, **Nicola Leon**e:
On the Indiscernibility of Individuals in Logic Programming. J. Log. Comput. 7(6): 805-824 (1997)

Georg Gottlob, **Nicola Leone, Francesco Scarcello:**
Hypertree Decompositions and Tractable Queries. PODS 1999: 21-32

Francesco **Buccafurri**, Thomas Eiter, Georg Gottlob, Nicola Leone:
On ACTL Formulas Having Linear Counterexamples. J. Comput. Syst. Sci. 62(3): 463-515 (2001)

Georg Gottlob, **Gianluigi Greco**:
Decomposing combinatorial auctions and set packing problems. J. ACM 60(4): 24:1-24:39 (2013)

Georg Gottlob, **Marco Manna**, Andreas Pieris:
Polynomial Combined Rewritings for Linear Existential Rules and DL-Lite with n-ary Relations. Description Logics 2015

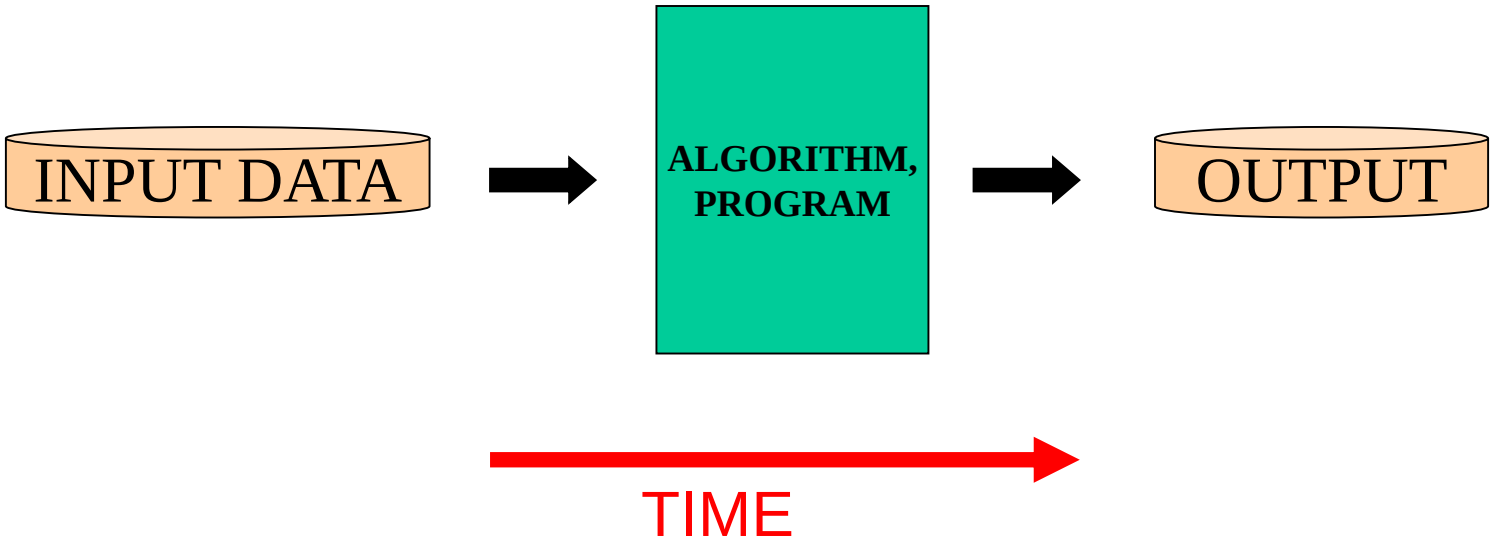
+ ca 15; Bettina Fazzinga…..

Grazie a tutti i miei amici e co-autori calbresi!

E grazie a

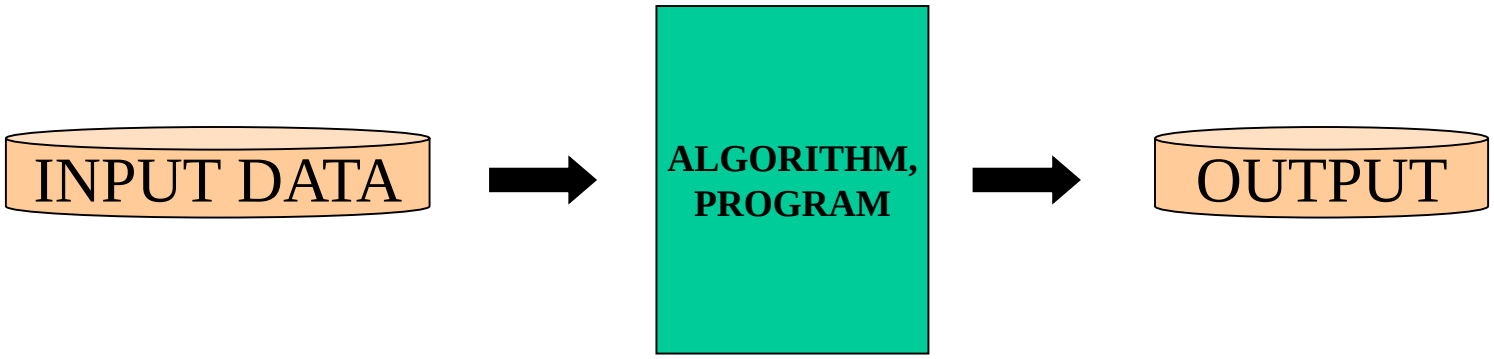
Computational Complexity

How long does a computer take to solve a problem?



Computational Complexity

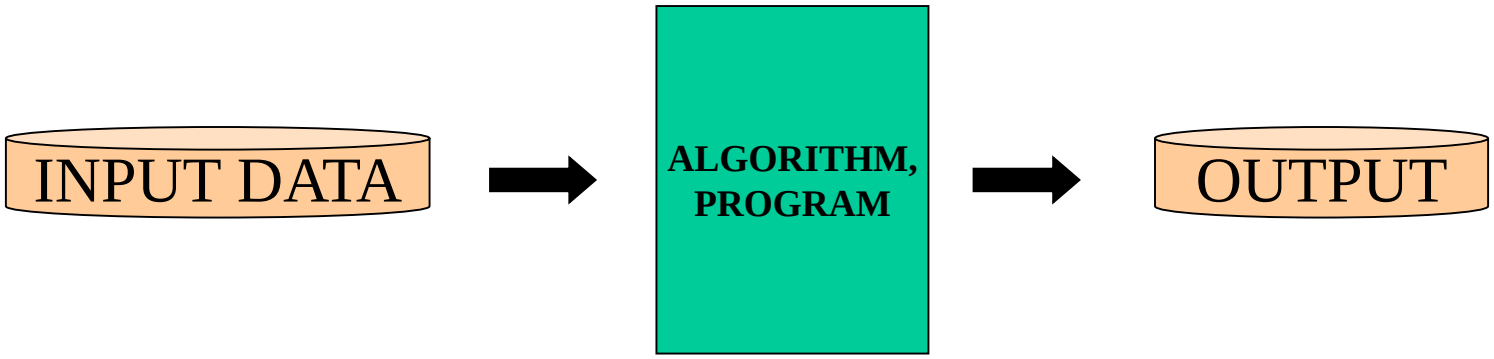
How long does it take a computer to solve a problem?

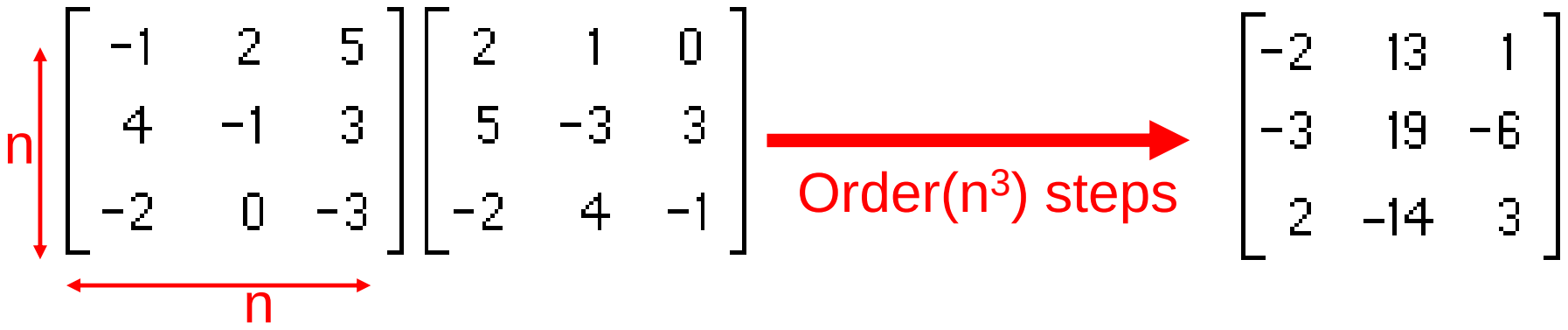


$$\begin{bmatrix} -1 & 2 & 5 \\ 4 & -1 & 3 \\ -2 & 0 & -3 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 5 & -3 & 3 \\ -2 & 4 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} -2 & 13 & 1 \\ -3 & 19 & -6 \\ 2 & -14 & 3 \end{bmatrix}$$

Computational Complexity

How long does it take a computer to solve a problem?





Computational Complexity of a Problem

= **Complexity of the best algorithm solving the problem.**

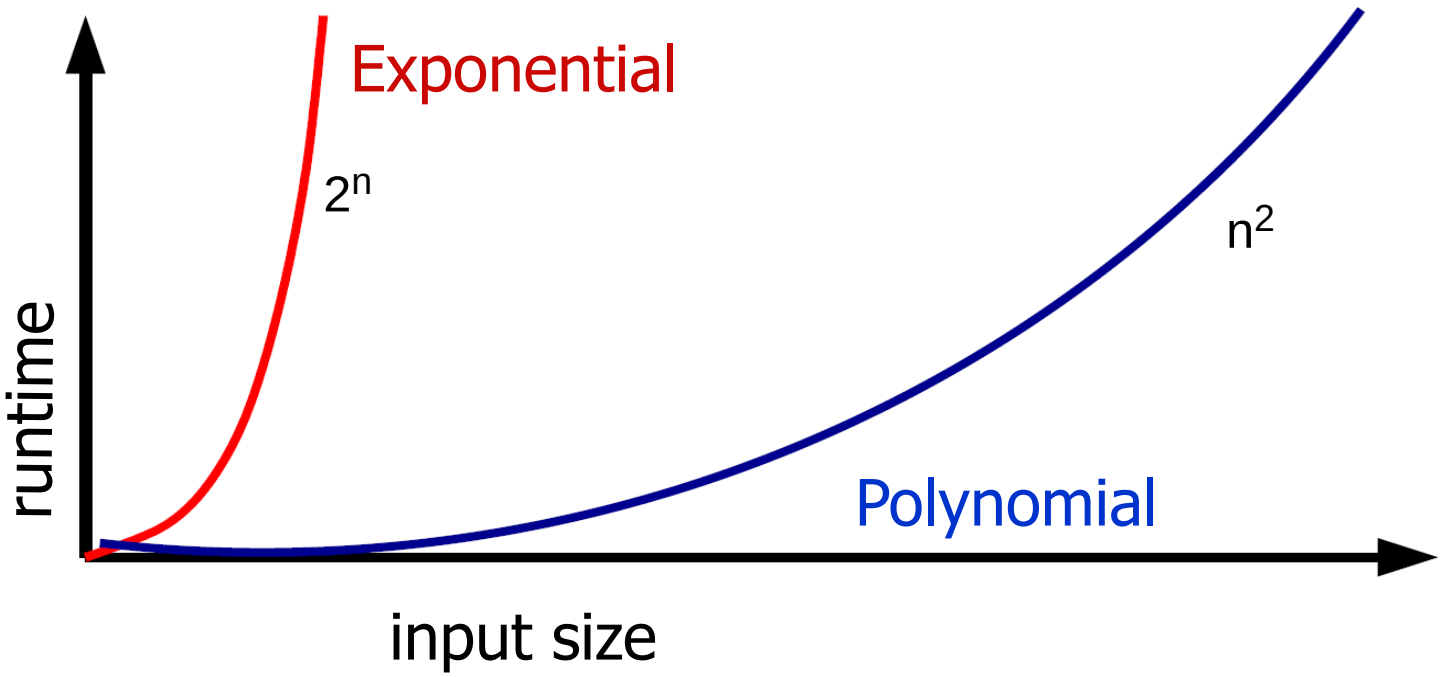
Tractable problems: Polynomial time

Intractable problems: - Exponential time

- Presumably exponential (e.g. NP-Complete)

TIME COMPLEXITY OF ALGORITHM:

Number of steps it takes for input of size n



Classification of decidable problems

PROVABLY INTRACTABLE PROBLEMS

- Theory of the Real Numbers
- Many tasks in automated program verification

TRACTABLE PROBLEMS (polynomial time)

- Matrix multiplication
- Shortest path
- Linear programming

Classification of decidable problems

PROVABLY INTRACTABLE PROBLEMS

- Theory of the Real Numbers
- Many tasks in automated program verification

PRESUMABLY INTRACTABLE PROBLEMS

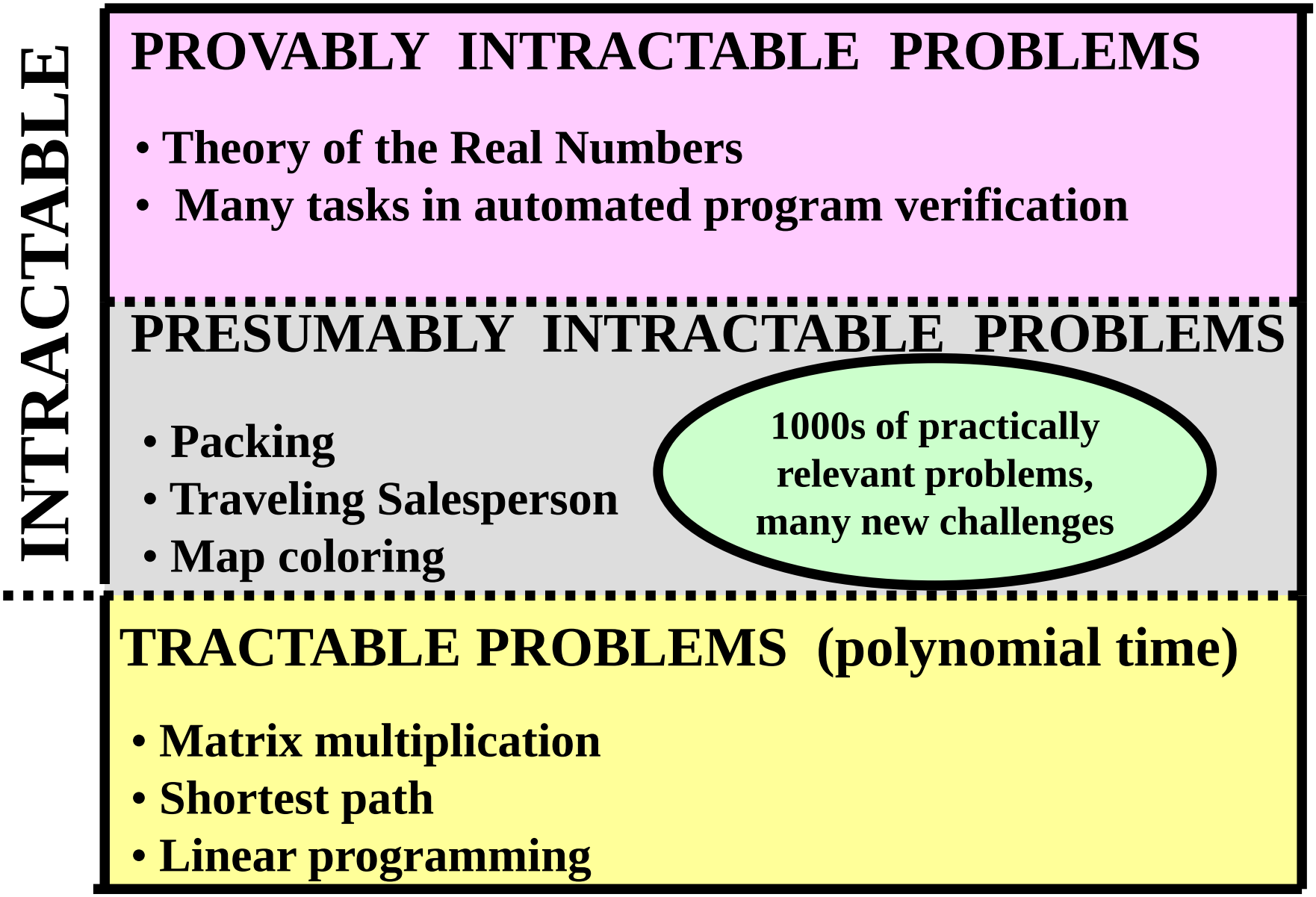
- Packing
- Traveling Salesperson
- Map coloring

NP-COMPLETE

TRACTABLE PROBLEMS (polynomial time)

- Matrix multiplication
- Shortest path
- Linear programming

Classification of decidable problems



Living with Computational Complexity

➔ **Challenges for computer scientists and mathematicians**

- **Tractable problems:**

  - Find faster algorithms

- **New Problems of unknown complexity:**

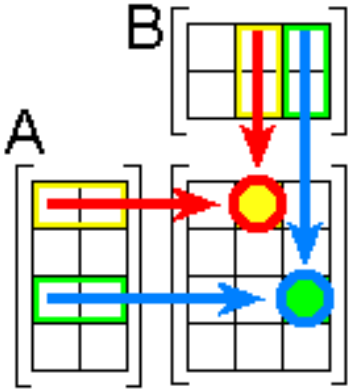
  - Find polynomial-time algorithms if possible.

Faster Algorithms for tractable problems

Example: Matrix Multiplication, a continuing race

Naïve method: $O(n^3)$

V. Strassen 1969: $O(n^{2.807})$



Coppersmith & Winograd 1990: $O(n^{2.376})$

Cohn, Kleinberg, Szegedy, Umans 2005:

Existence of certain groups → $O(n^2)$ multiplications

Living with Computational Complexity

Intractable problems:

Essentially 3 approaches:

1. Find large tractable classes, e.g., by using problem decomposition methods.

2. Use heuristics or randomized techniques.

3. Find approximate solutions efficiently

Tractable instances of hard problems

"Intractability" refers to the worst case.

In practice, problems are often easier.

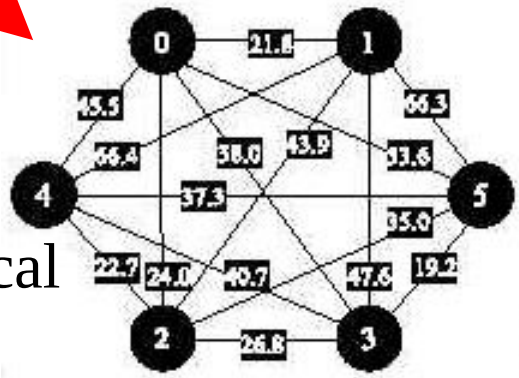
How to automatically recognise it (AI) ???

→ Use problem decomposition techniques


PROBLEM
INSTANCE

PROBLEM INSTANCE

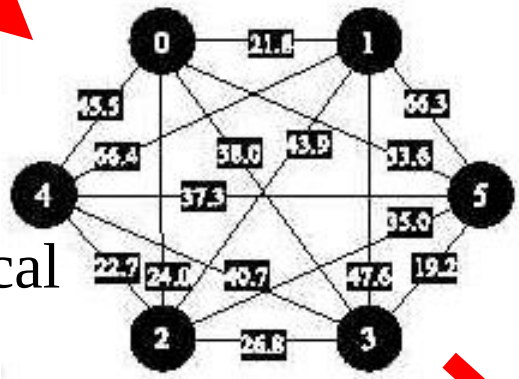
mathematical model



(b)

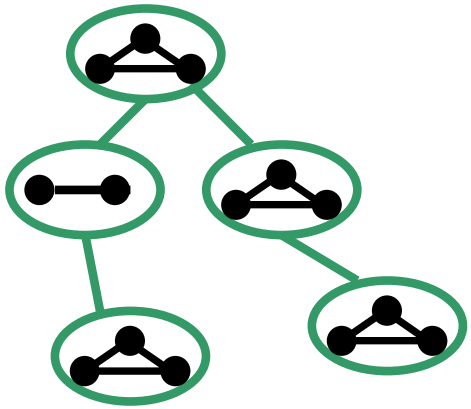
PROBLEM INSTANCE

mathematical model



(b)

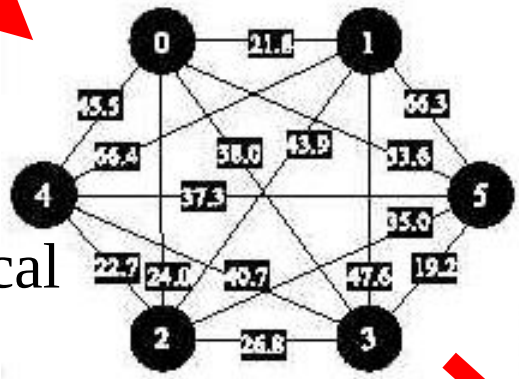
Divide and conquer
divide et impera



decomposition

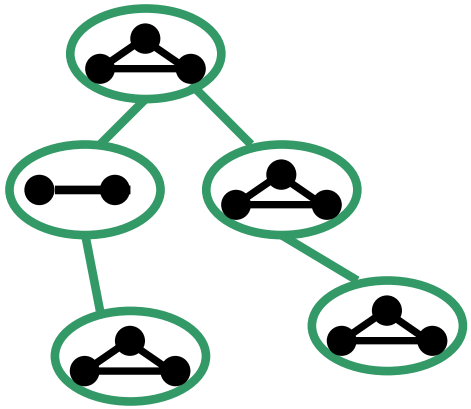
PROBLEM INSTANCE

mathematical model



(b)

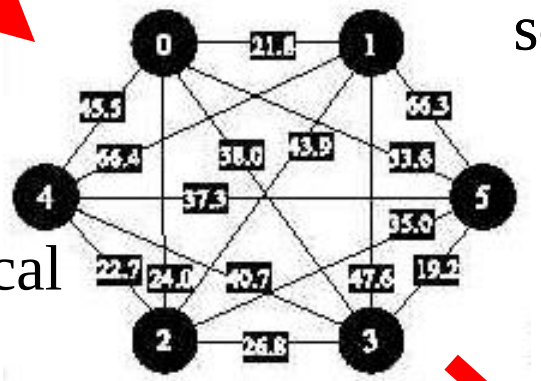
solution



decomposition

PROBLEM INSTANCE

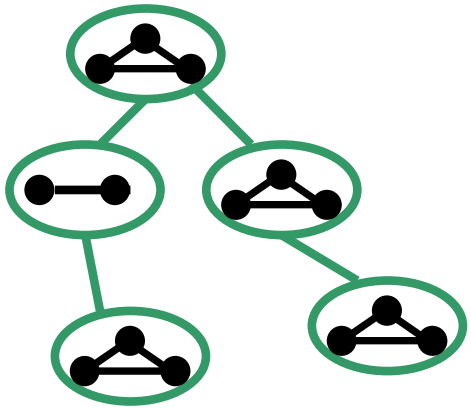
mathematical model



(b)

solution

solution



decomposition

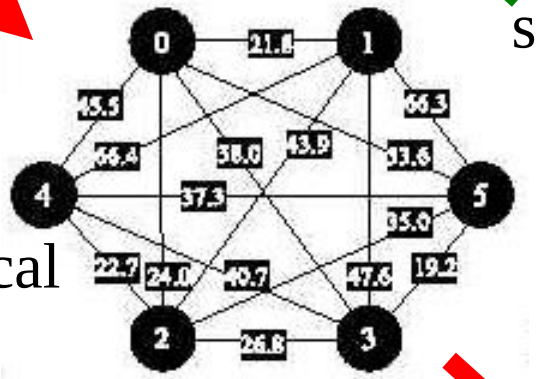
PROBLEM INSTANCE

solution

solution

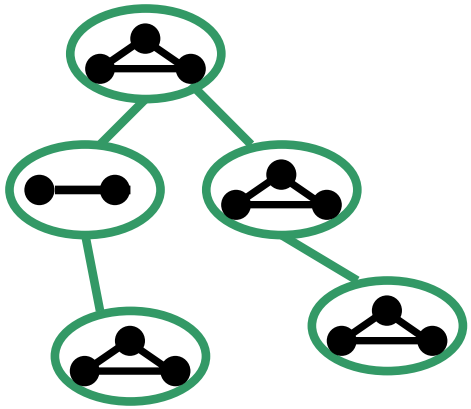
solution

mathematical model



(b)

decomposition



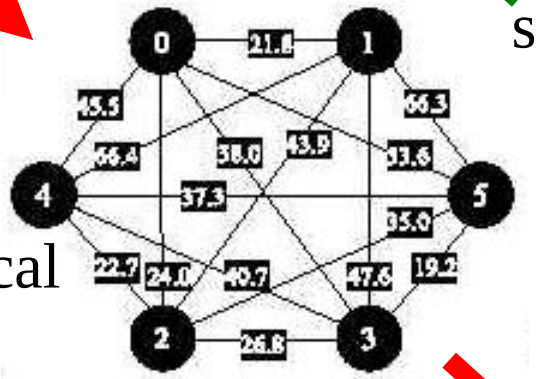
PROBLEM INSTANCE

solution

AI
constraint solving

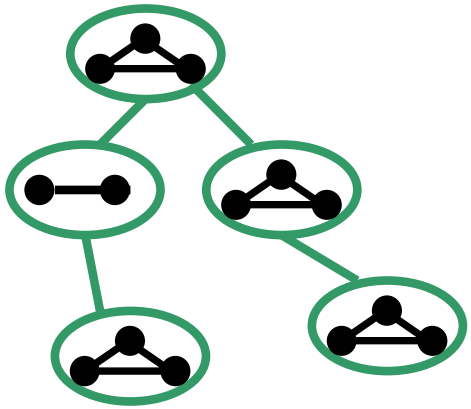
solution

mathematical model

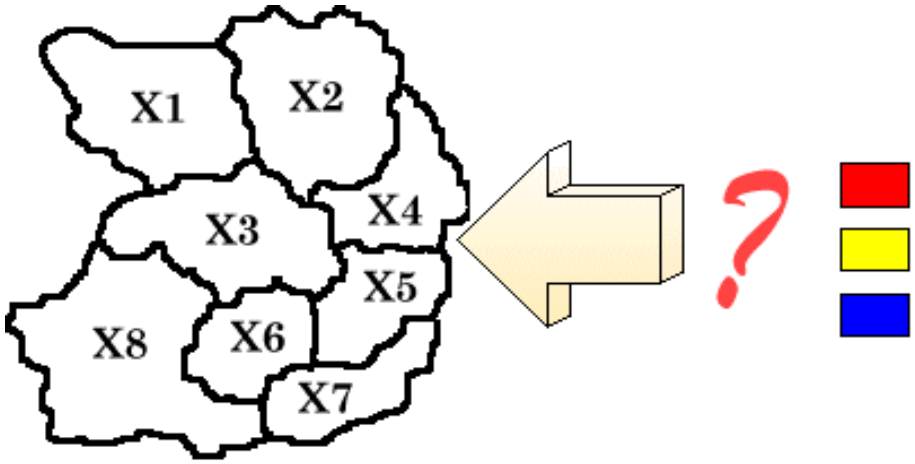


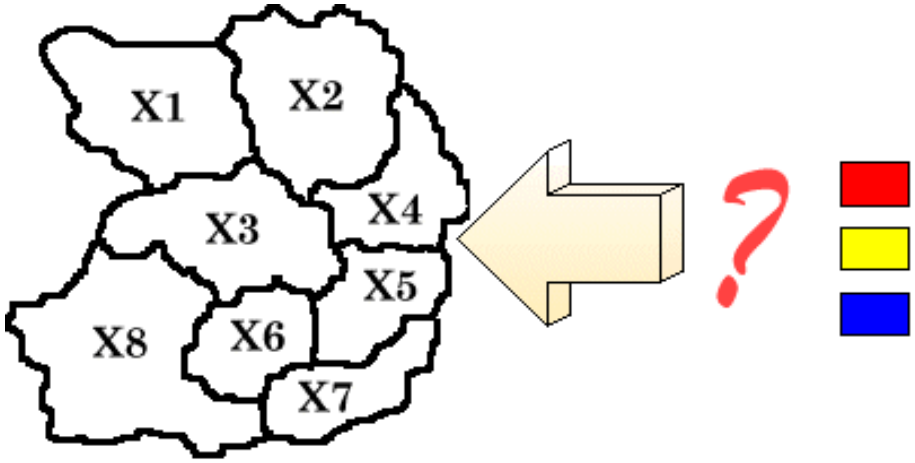
(b)

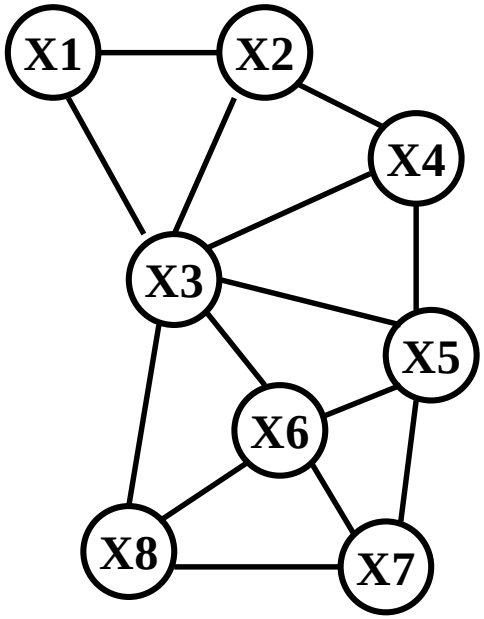
solution



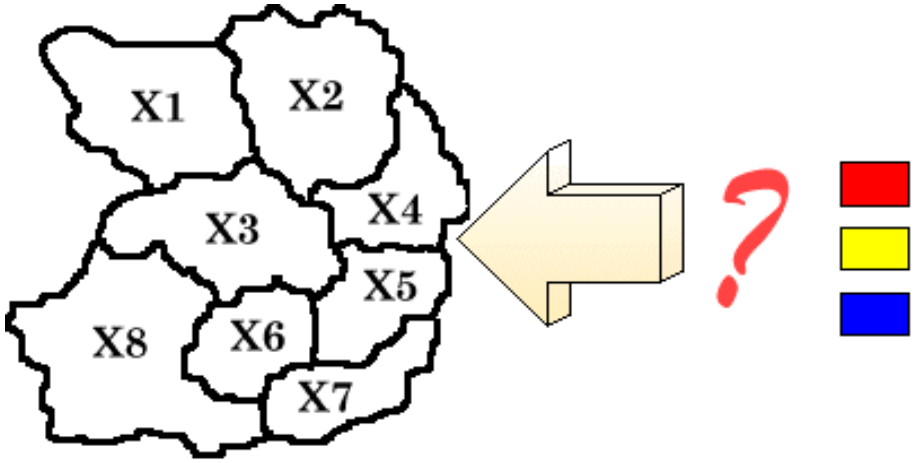
decomposition

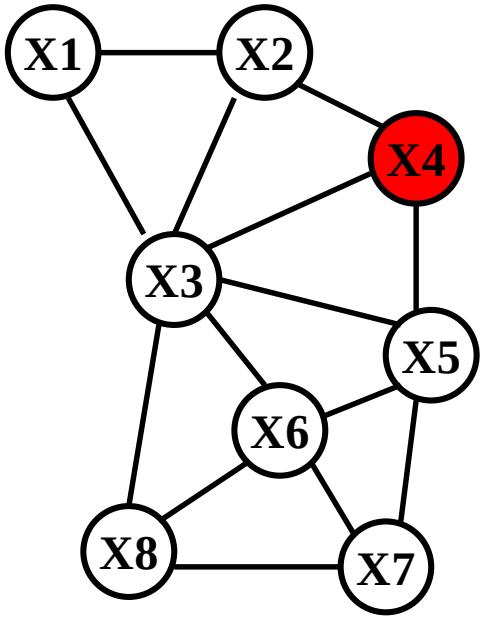

X1
X2
X3
X4
X5
X6
X7
X8
?

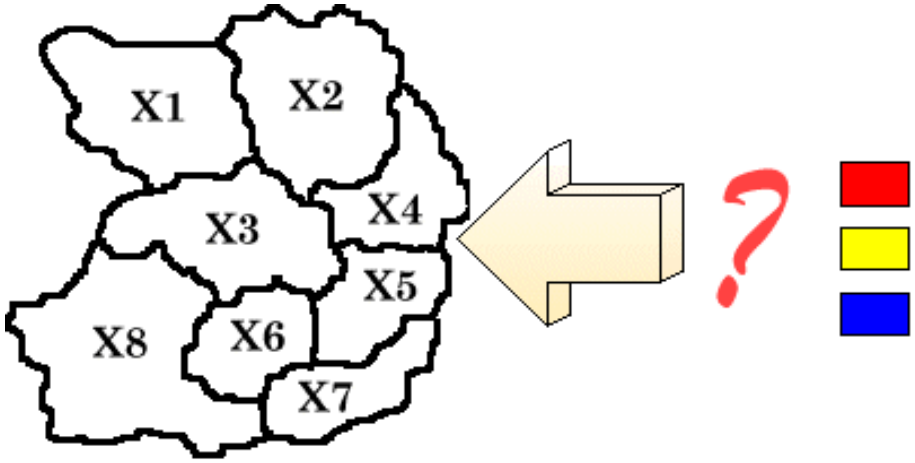


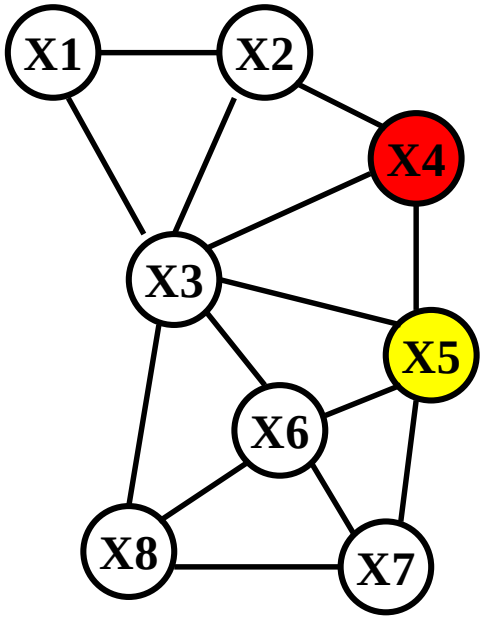


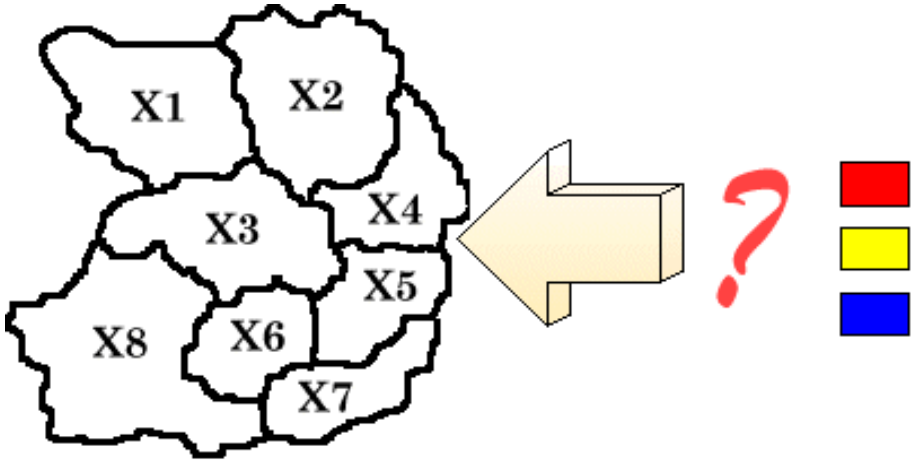
Problem representation by mathematical structure: **Graph**

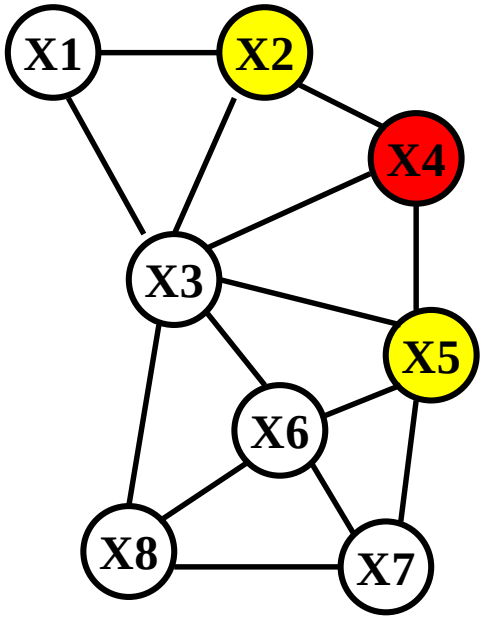

X1
X2
X3
X4
X5
X6
X7
X8
?

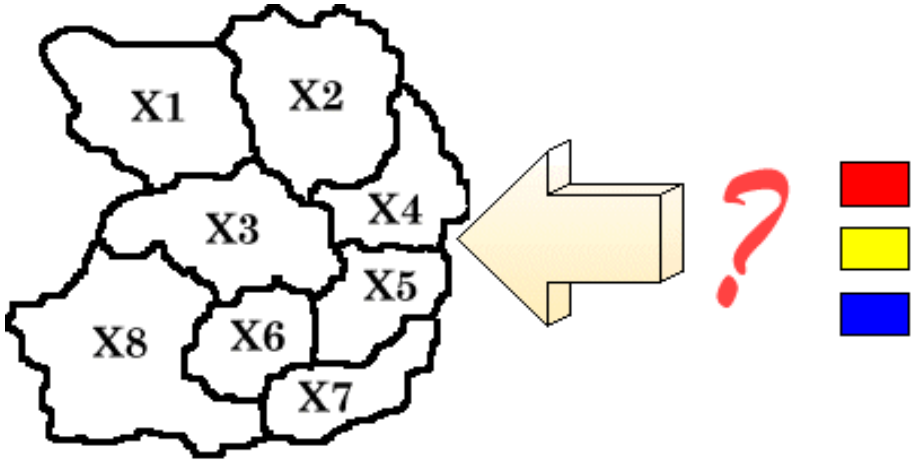

X1
X2
X4
X3
X5
X6
X8
X7

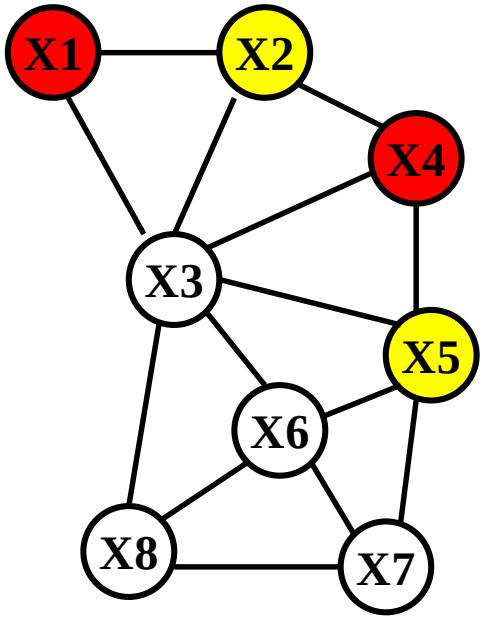

X1
X2
X3
X4
X5
X6
X7
X8
?

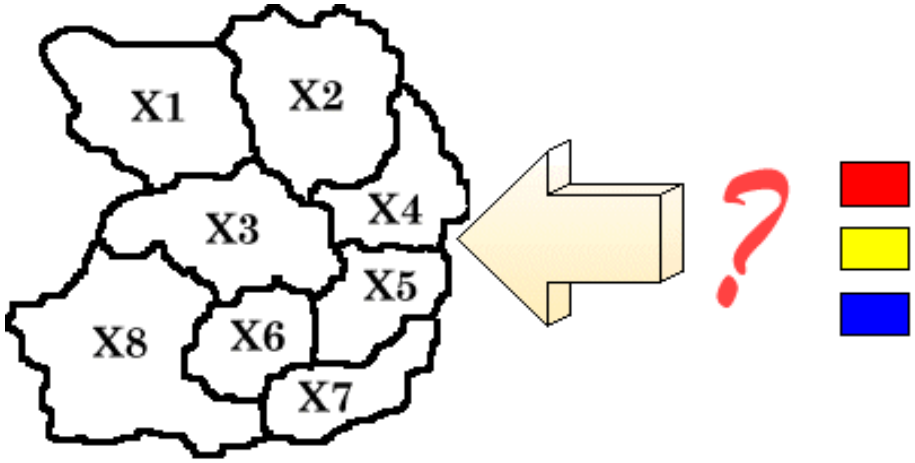

X1
X2
X4
X3
X5
X6
X8
X7

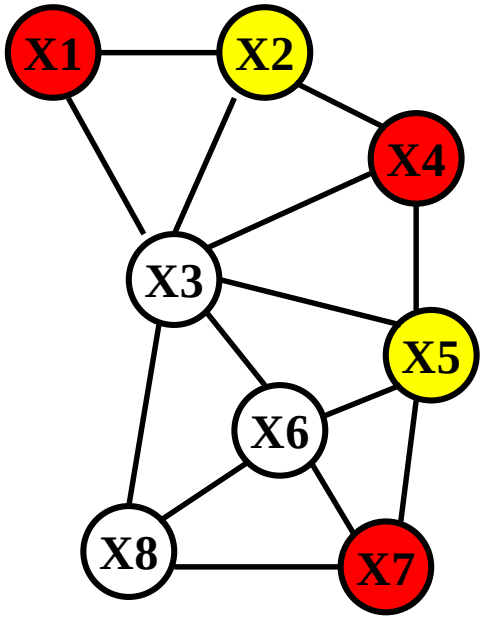

X1
X2
X3
X4
X5
X6
X7
X8
?

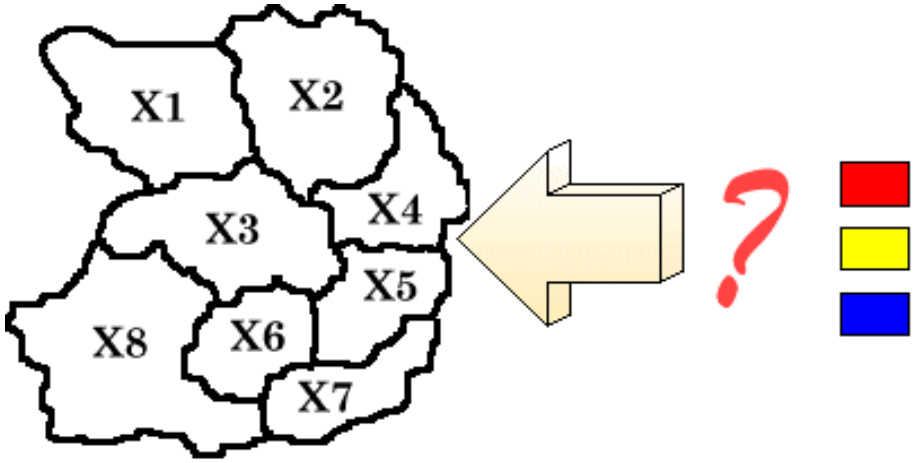

X1
X2
X4
X3
X5
X6
X8
X7

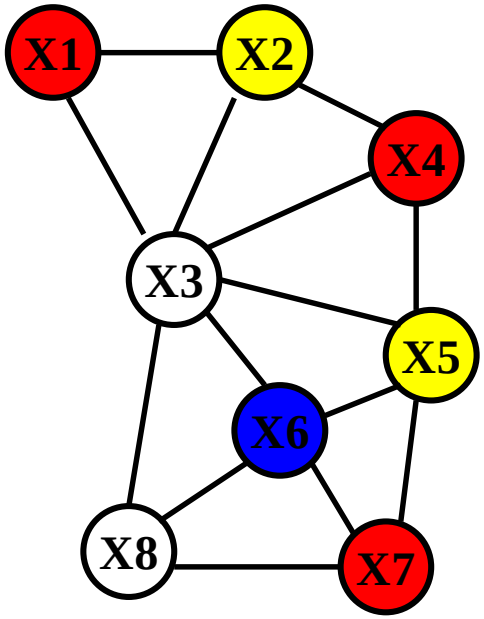

X1
X2
X3
X4
X5
X6
X7
X8
?

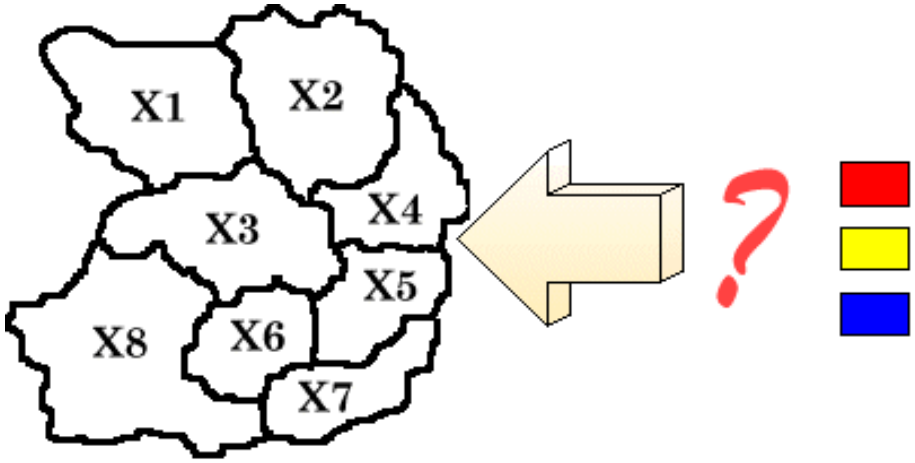

X1
X2
X4
X3
X5
X6
X8
X7

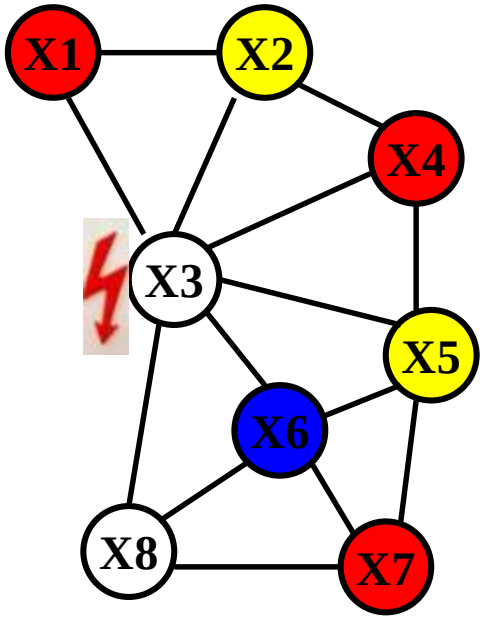

X1
X2
X3
X4
X5
X6
X7
X8
?


X1
X2
X4
X3
X5
X6
X8
X7


X1
X2
X3
X4
X5
X6
X7
X8
?


X1
X2
X4
X3
X5
X6
X8
X7

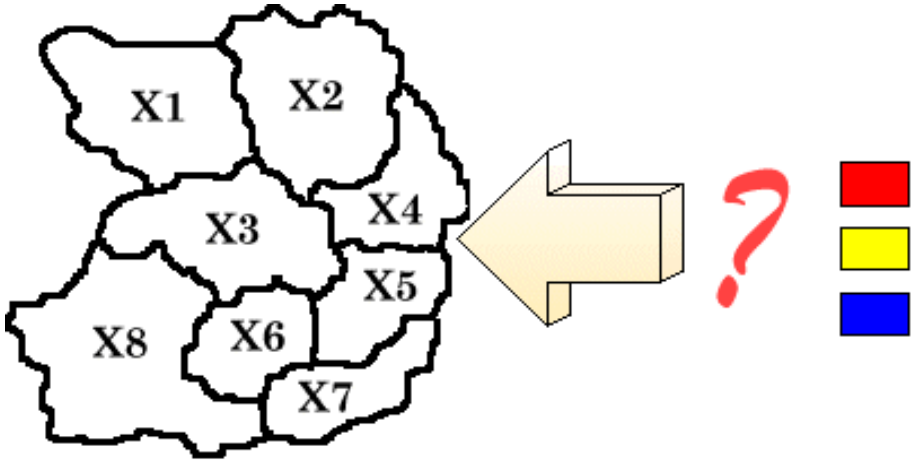


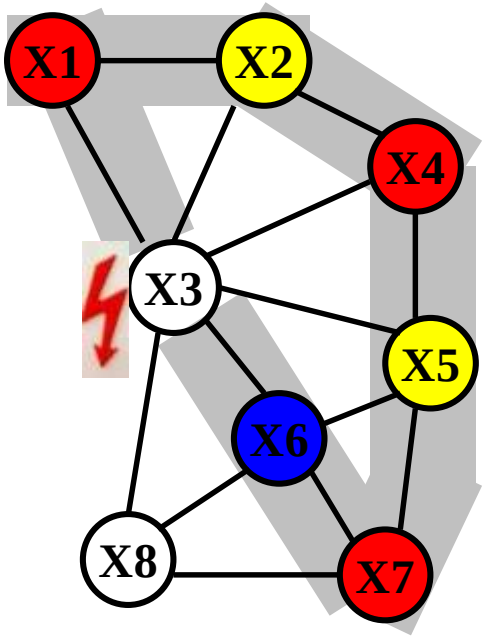


NO CORRECT COLORING OF X3 POSSIBLE!

BACKTRACK & FIND OTHER SOLUTIONS...

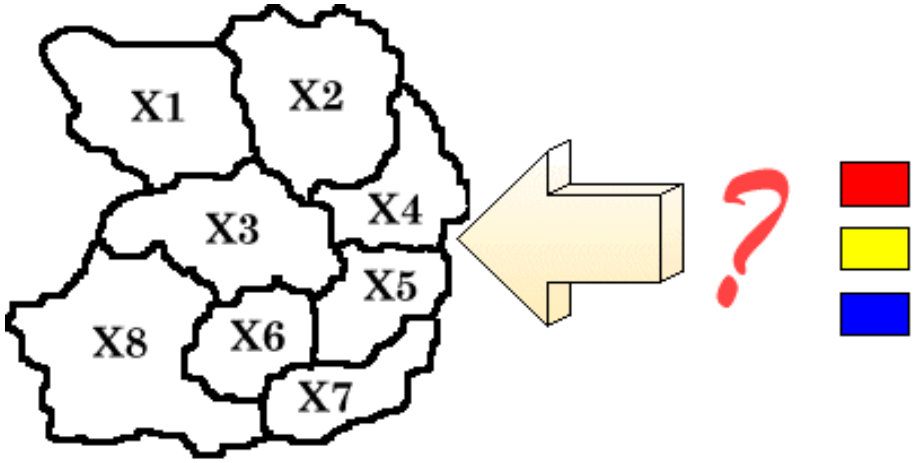
MAY REQUIRE EXPONENTIAL TIME!

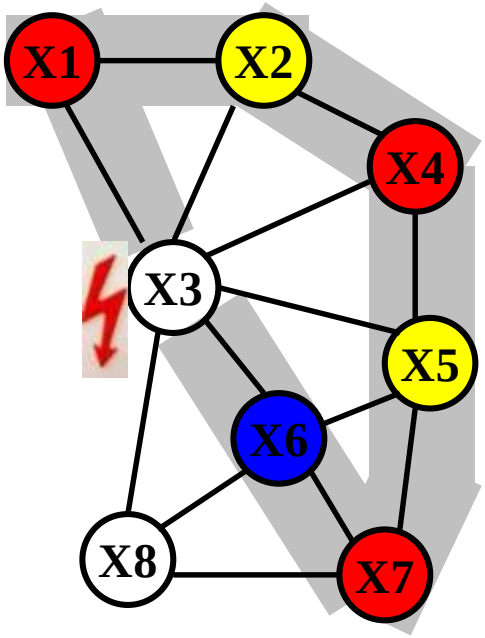




The reason for the bad computational behaviour is cyclicity!

Large cycles reflect non-locality!

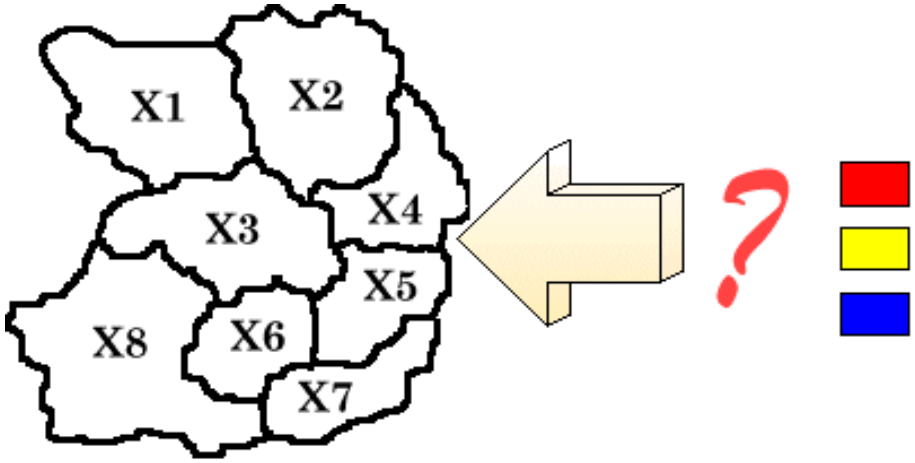


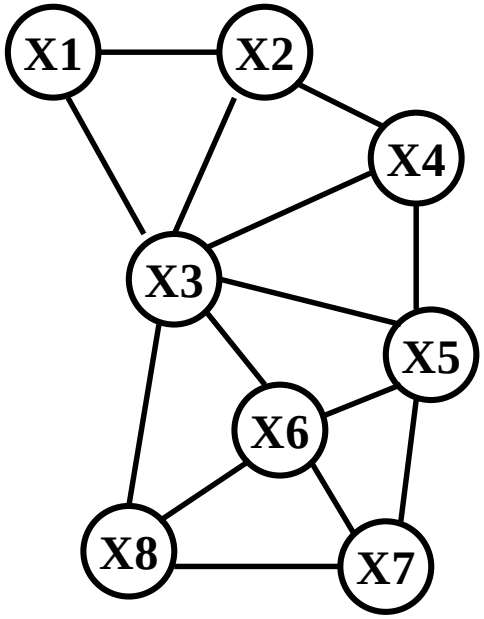


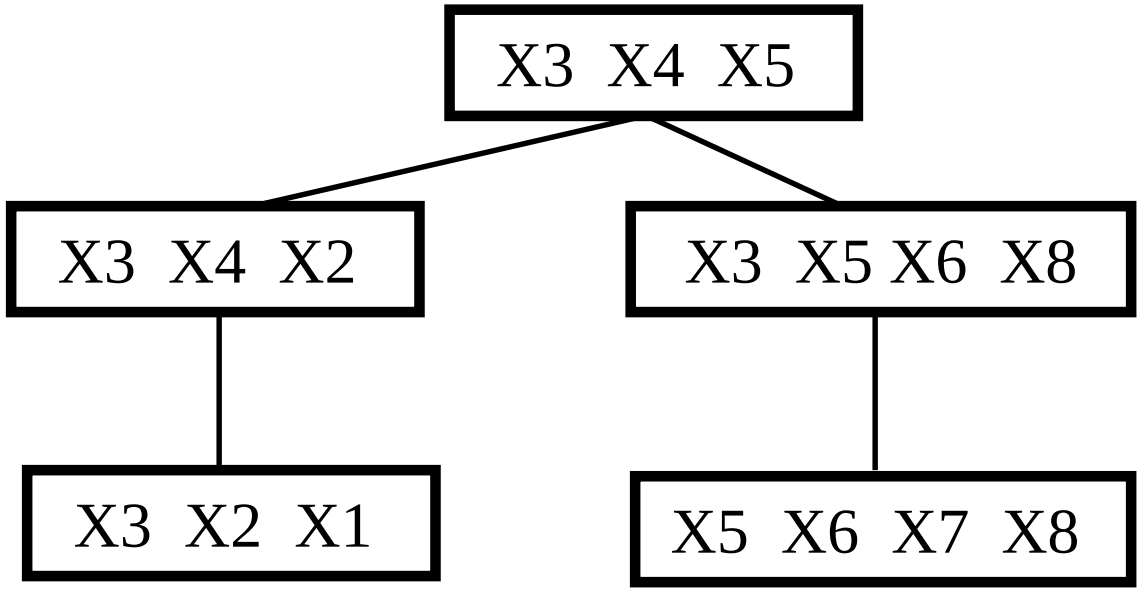
We may attempt a problem decomposition by clustering vertices and eliminating cycles:

Tree decomposition

[Seymour & Robertson 1986]

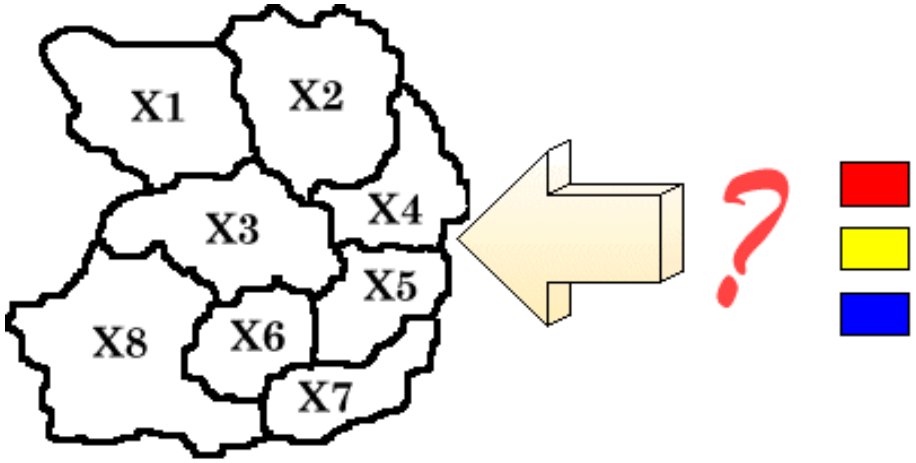


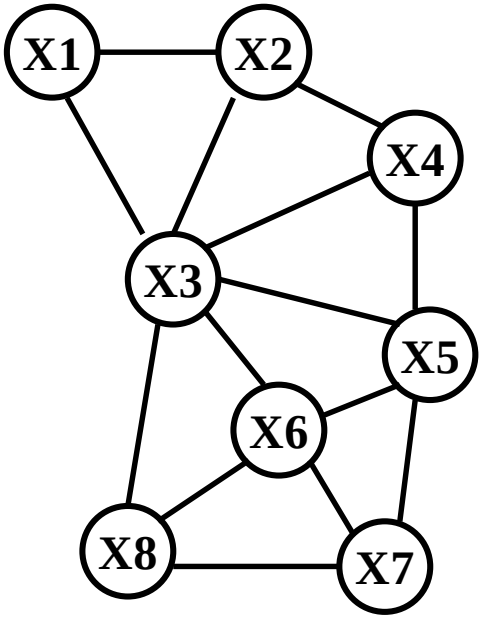


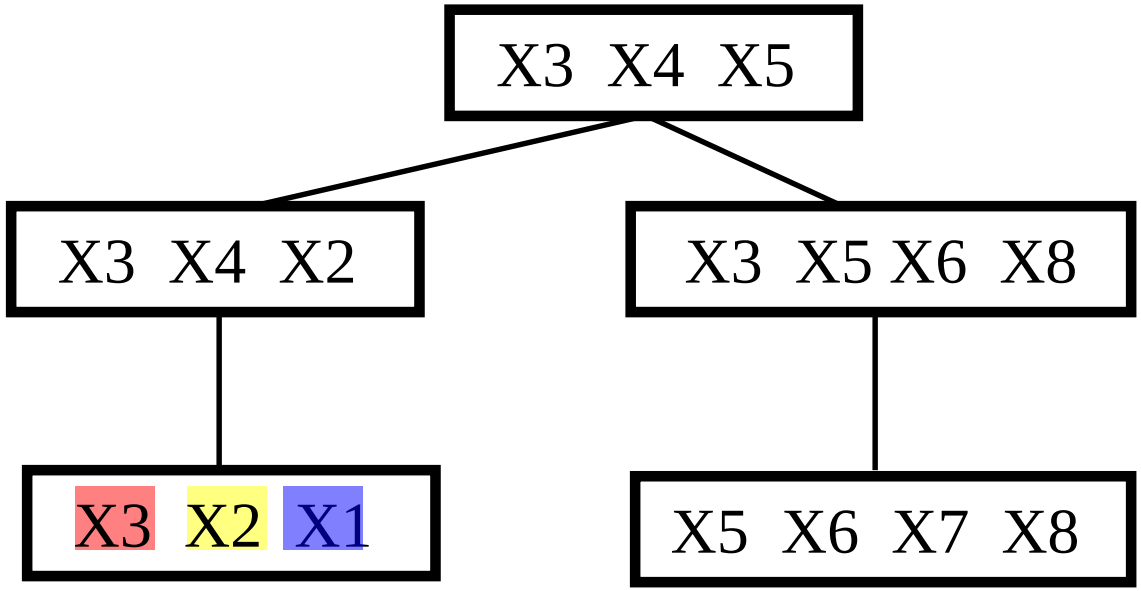


tree decomposition

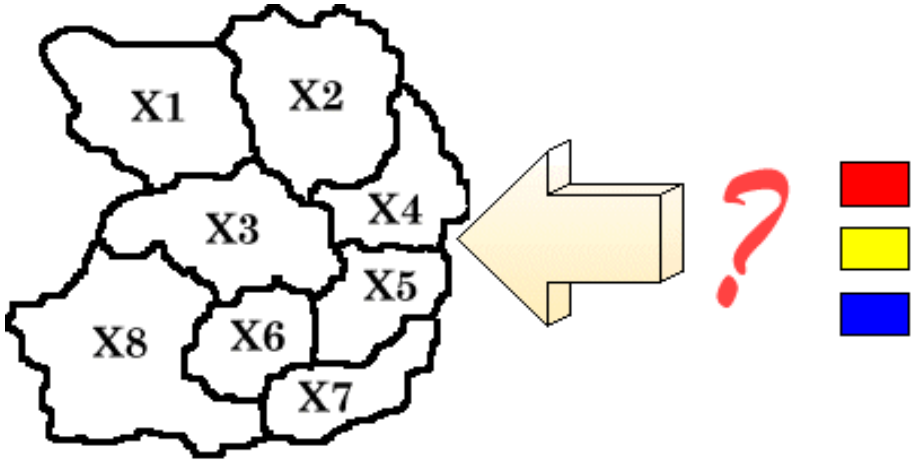
problem solving by insight

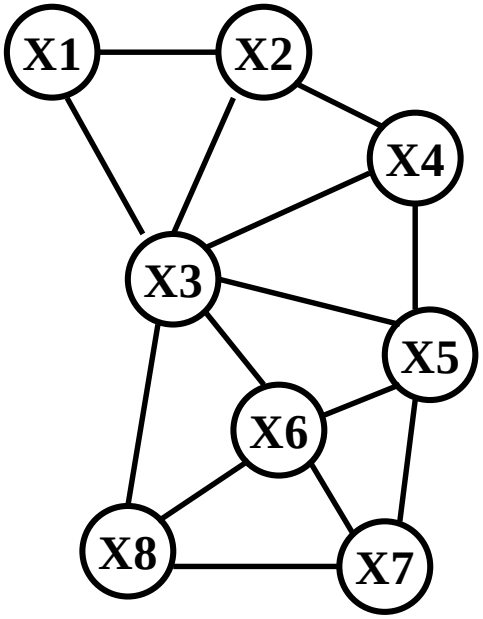


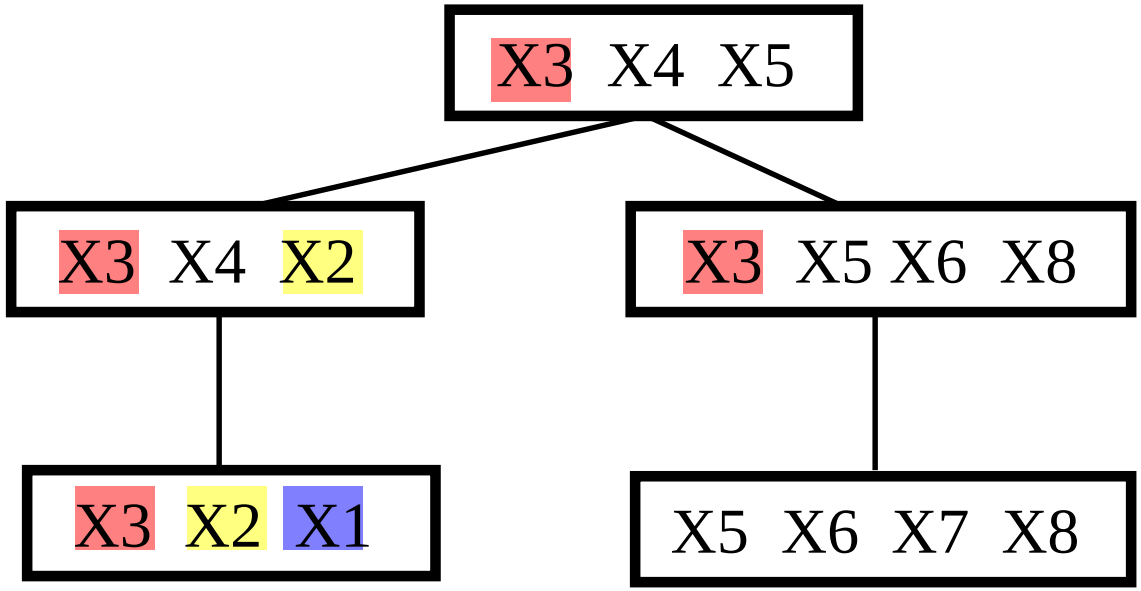




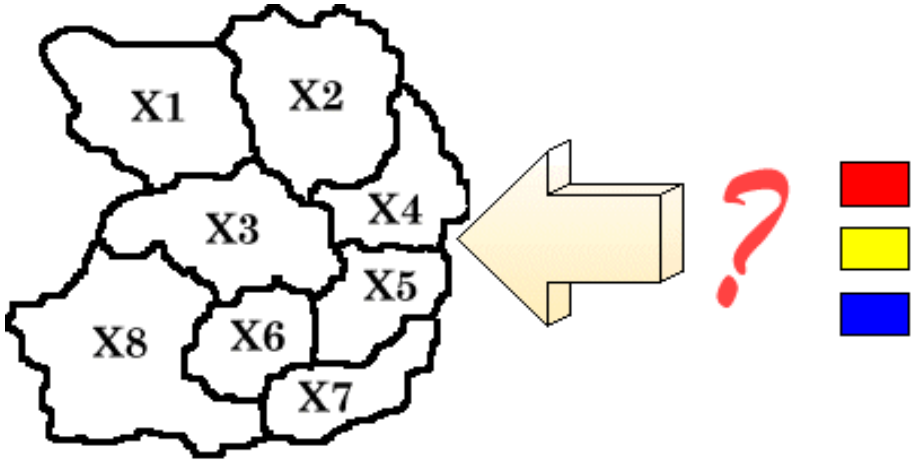
tree decomposition

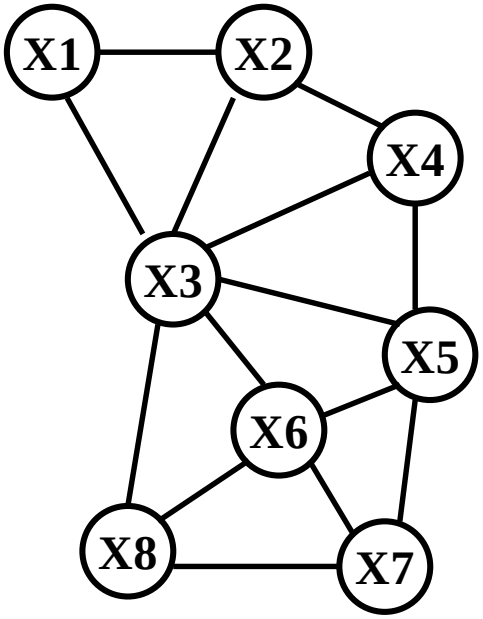


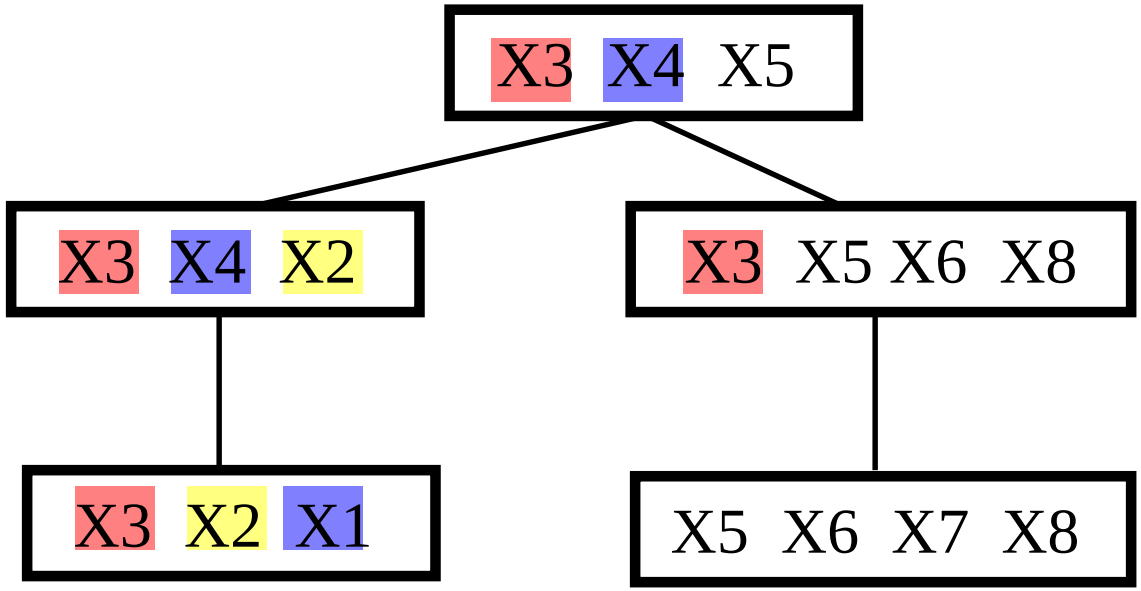




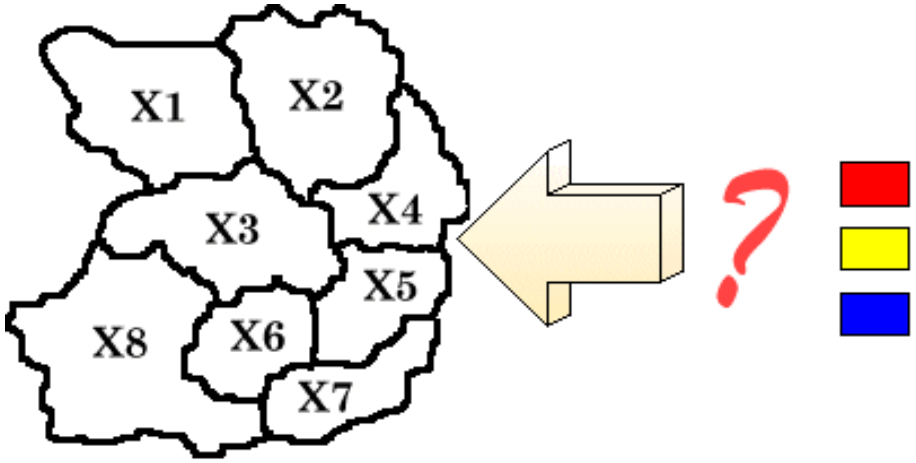
tree decomposition

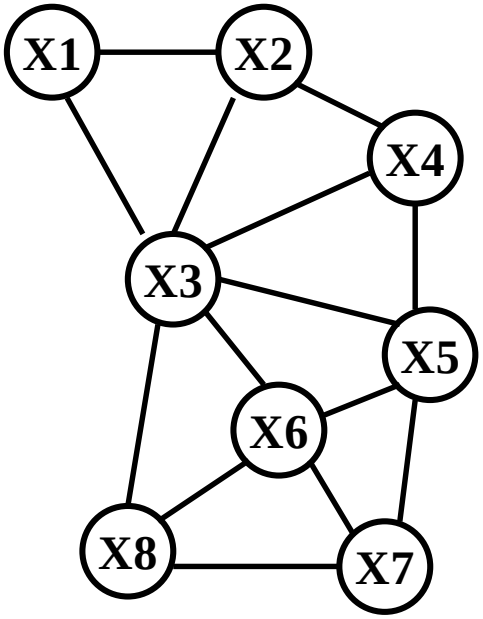


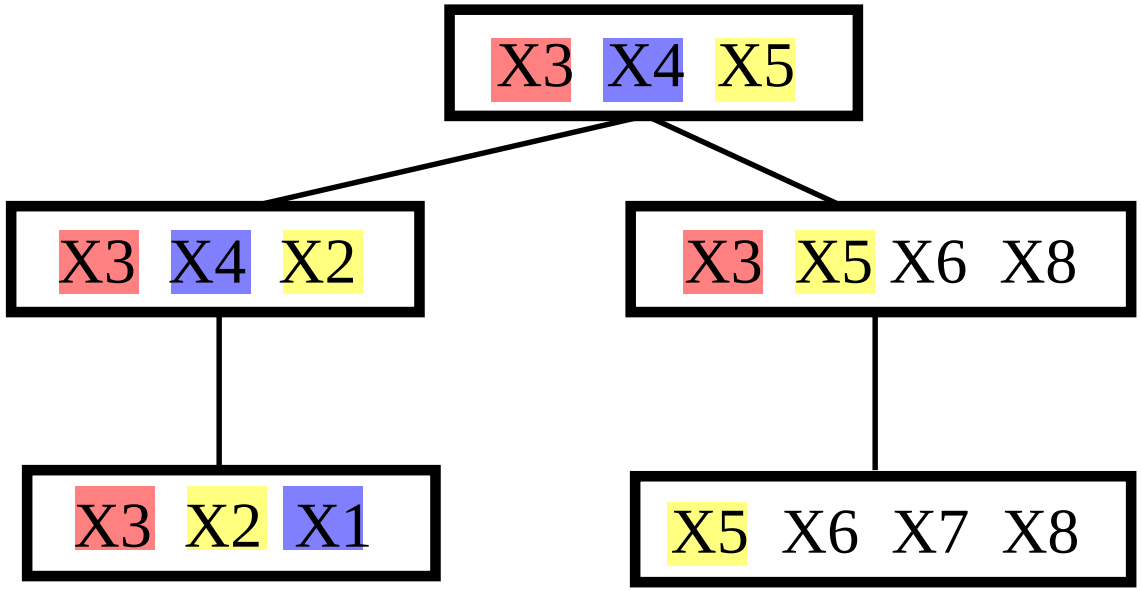




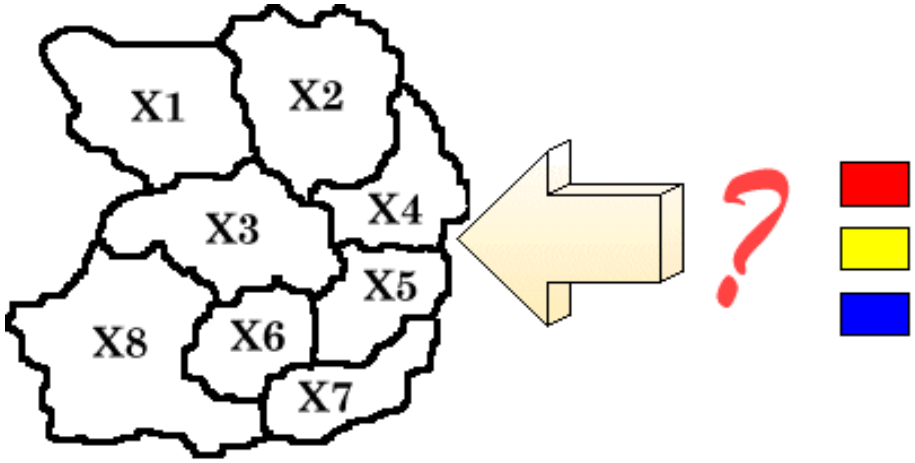
tree decomposition

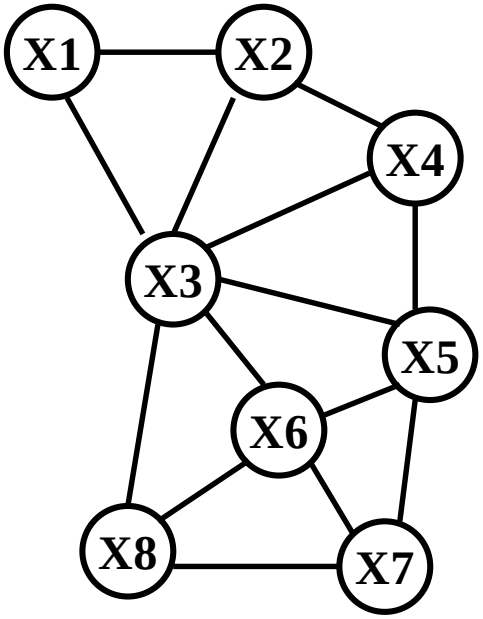


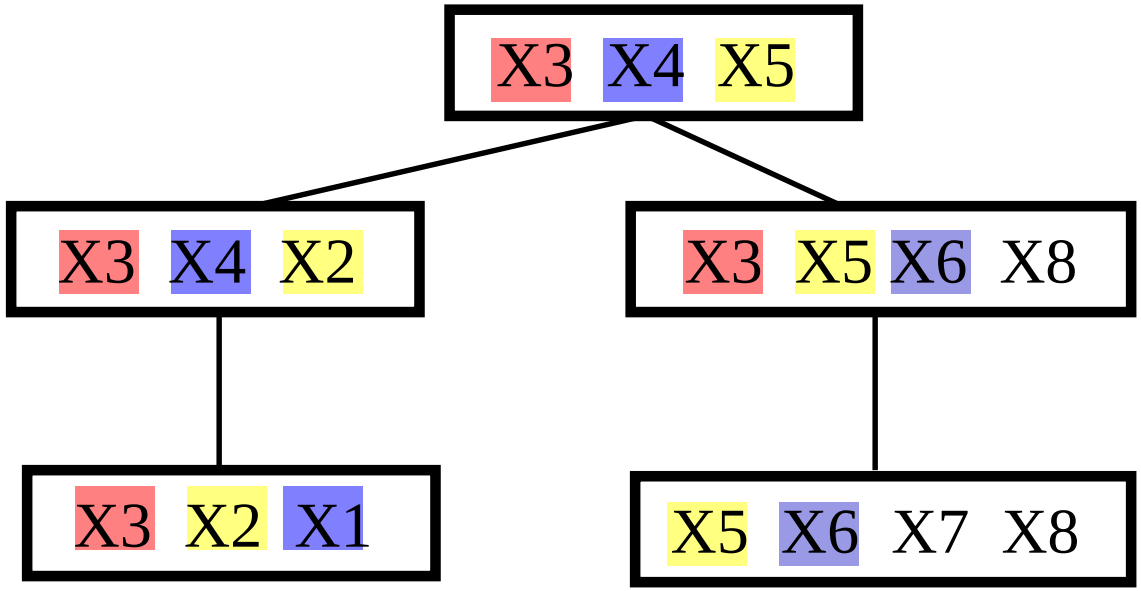




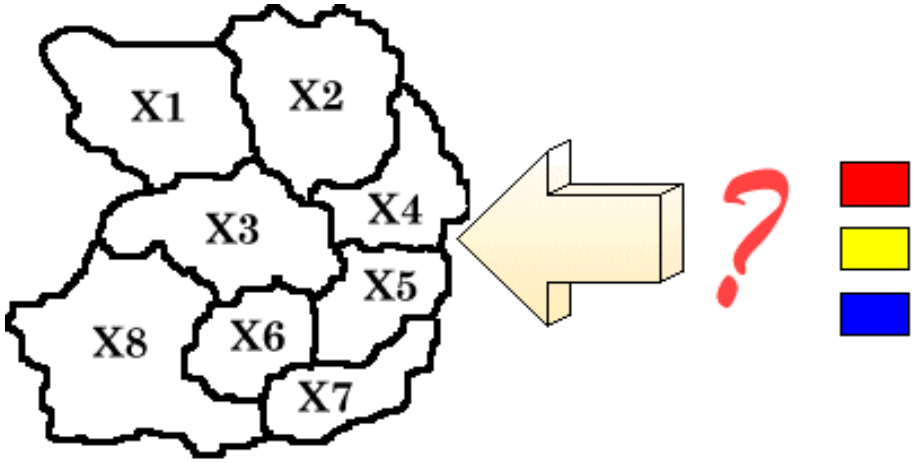
tree decomposition

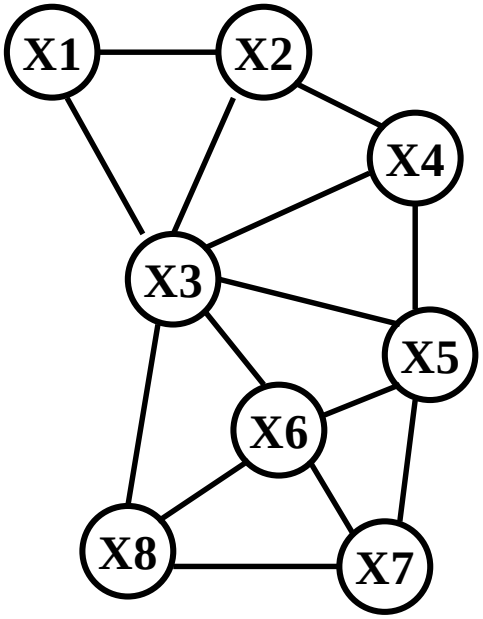


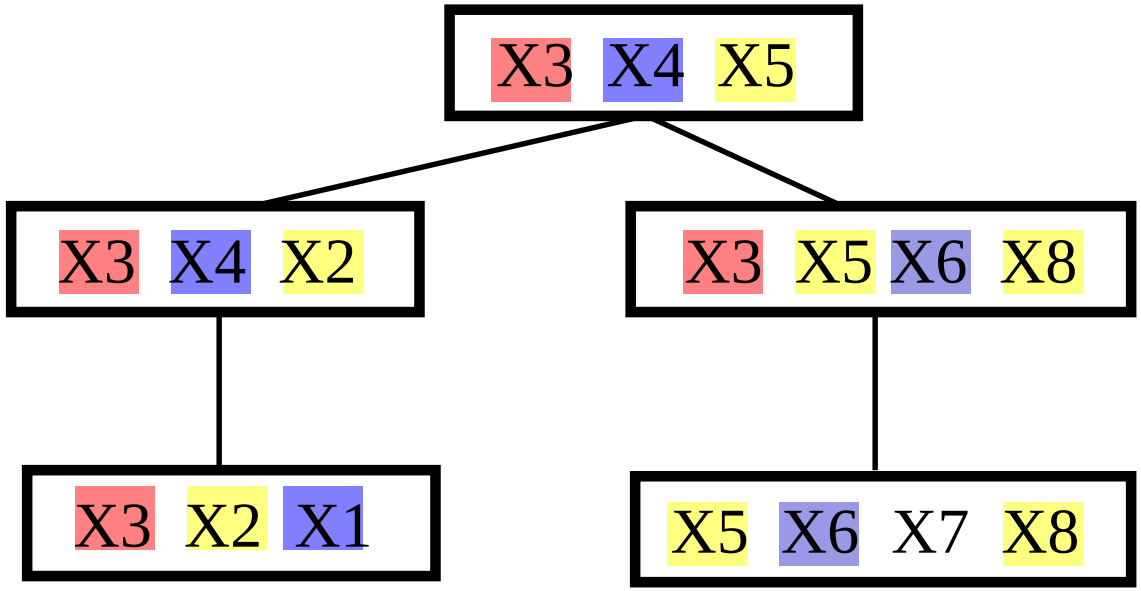




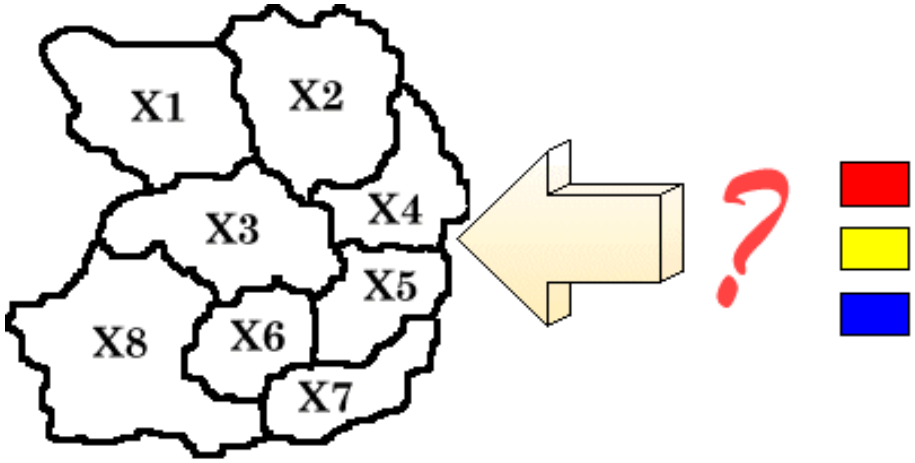
tree decomposition

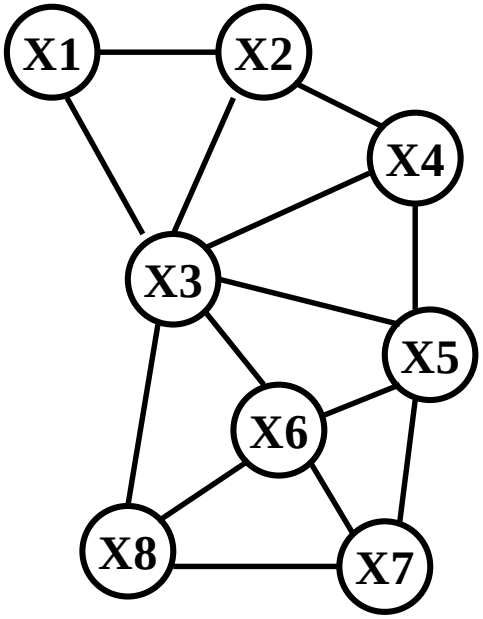


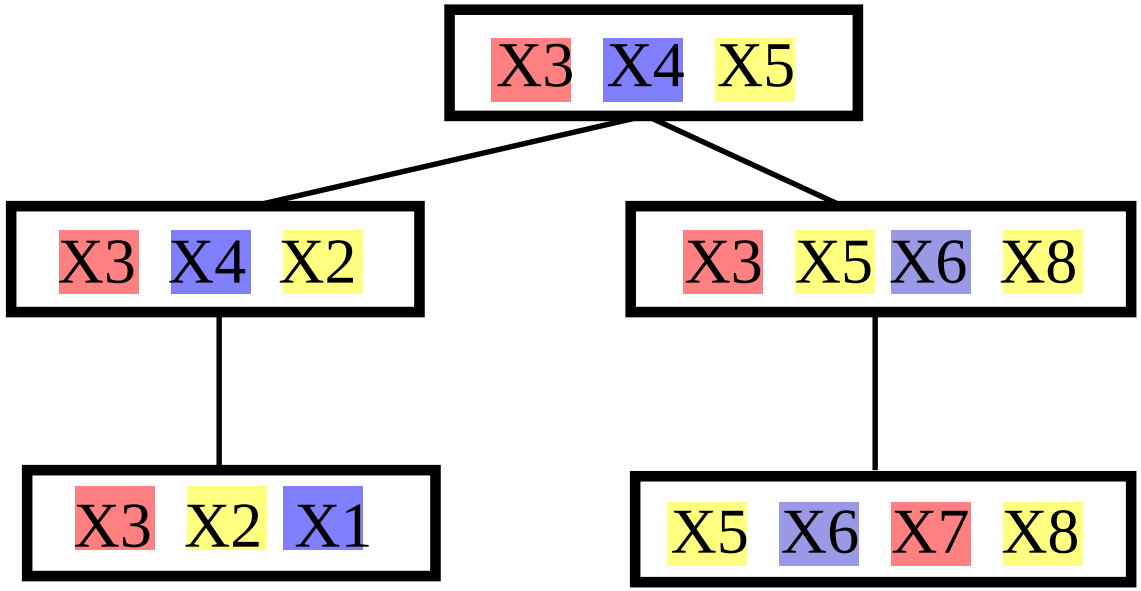




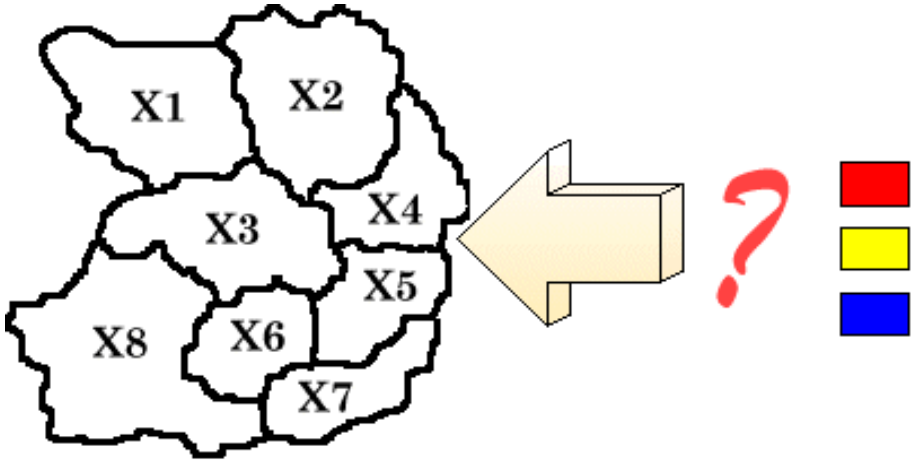
tree decomposition

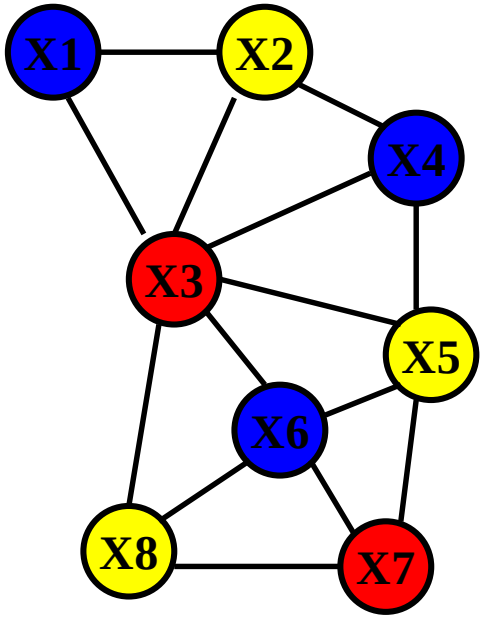


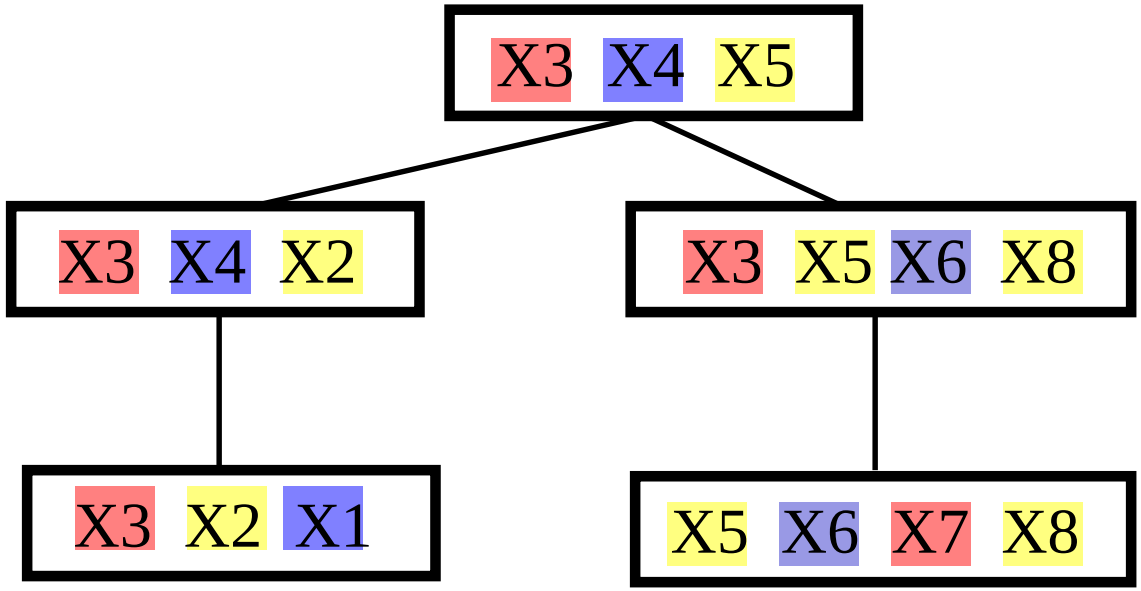




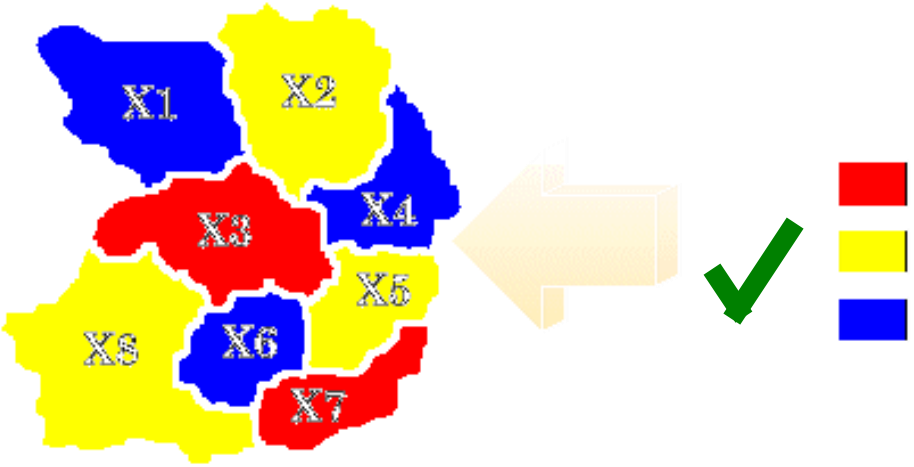
tree decomposition

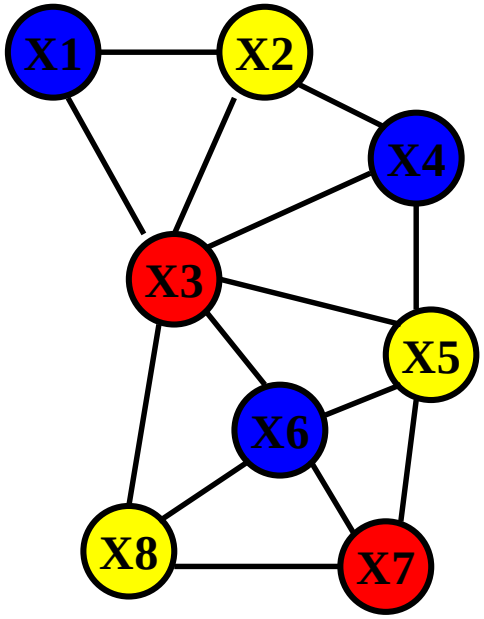


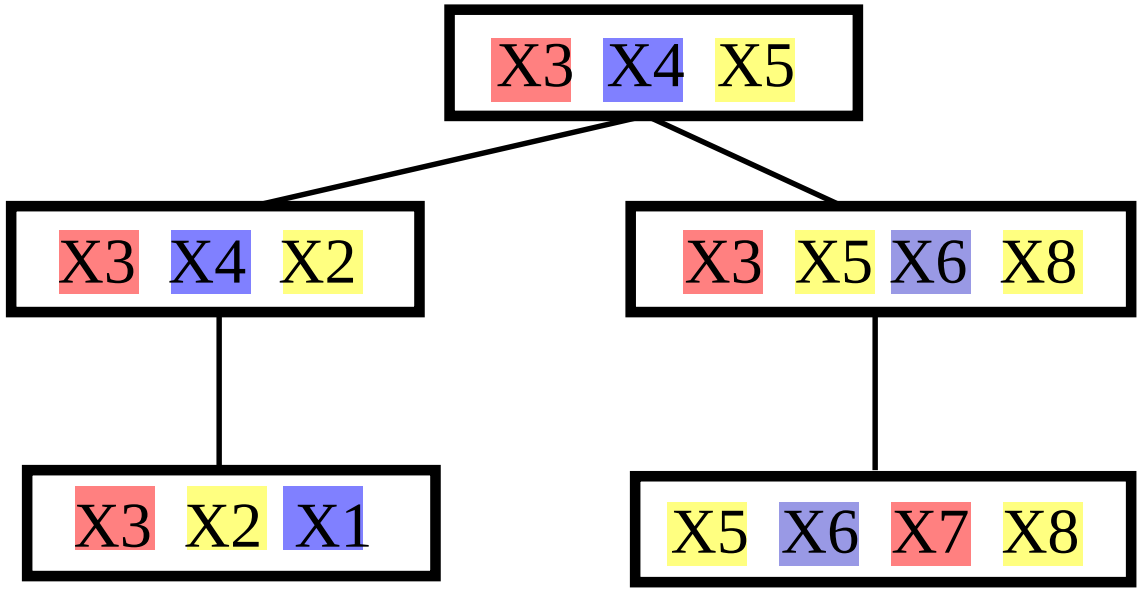




tree decomposition





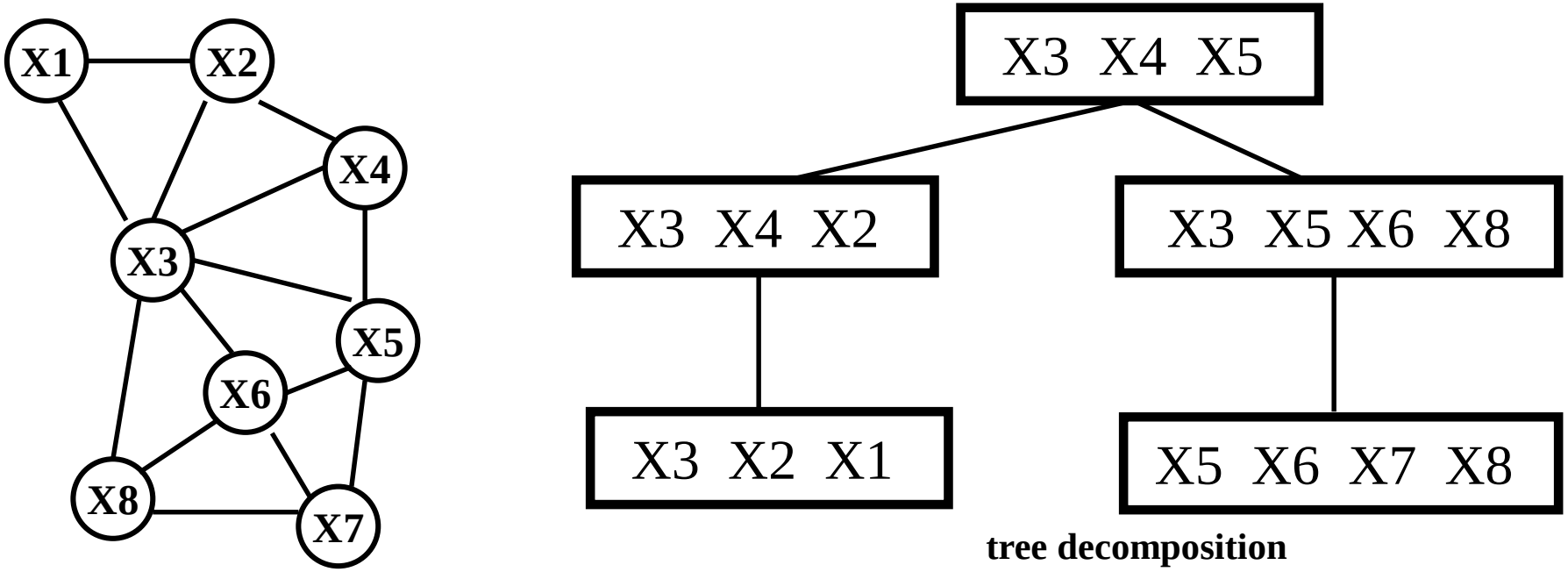


tree decomposition

Tree Decompositions: A systematically applicable method

1. Tree decompositions of bounded width can be found efficiently if they exist. [Bodlaender 1993]

2. If a tree decomposition exists, then the the colouring problem can be solved in polynomial time.



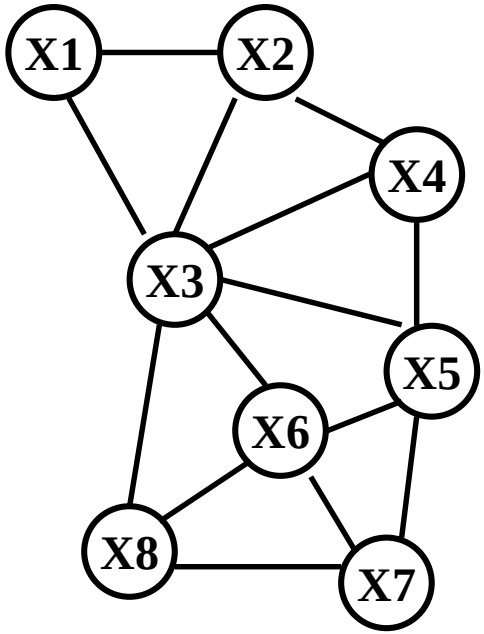
tree decomposition

Tree decompositions (1986) are optimal structural decompositions for a large class of problems:

Binary constraint satisfaction problems.

[Grohe, Schwentick, Segoufin 2001]

Such problems can be modelled by graphs whose edges are labelled by constraints.



Beyond Tree decompositions

The structure of many problems is not well-represented by a graph.

Examples:

Conjunctive database queries (SQL)

Puzzles

Constraint Satisfaction Problems (CSP)

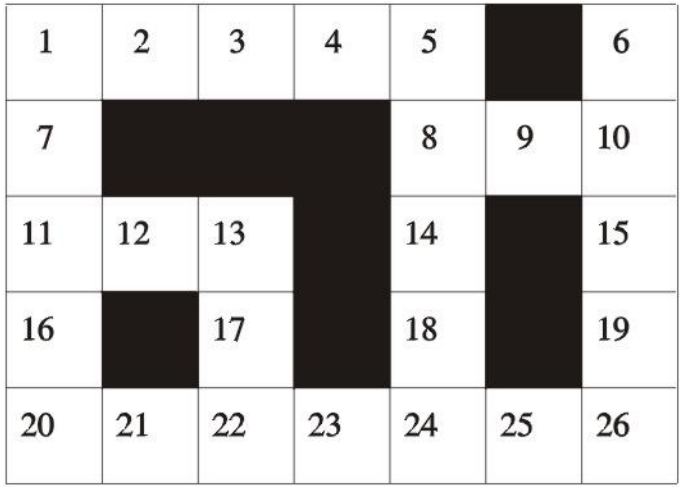
Integer programming (sparse case)

Combinatorial Auctions

……

These problems are better modeled by **hypergraphs**

Example of CSP: Crossword Puzzle



1h:

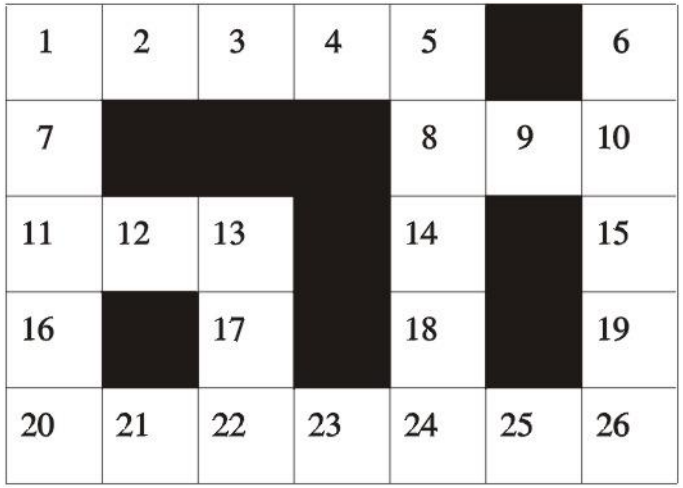
P	A	R	I	S
P	A	N	D	A
L	A	U	R	A
A	N	I	T	A

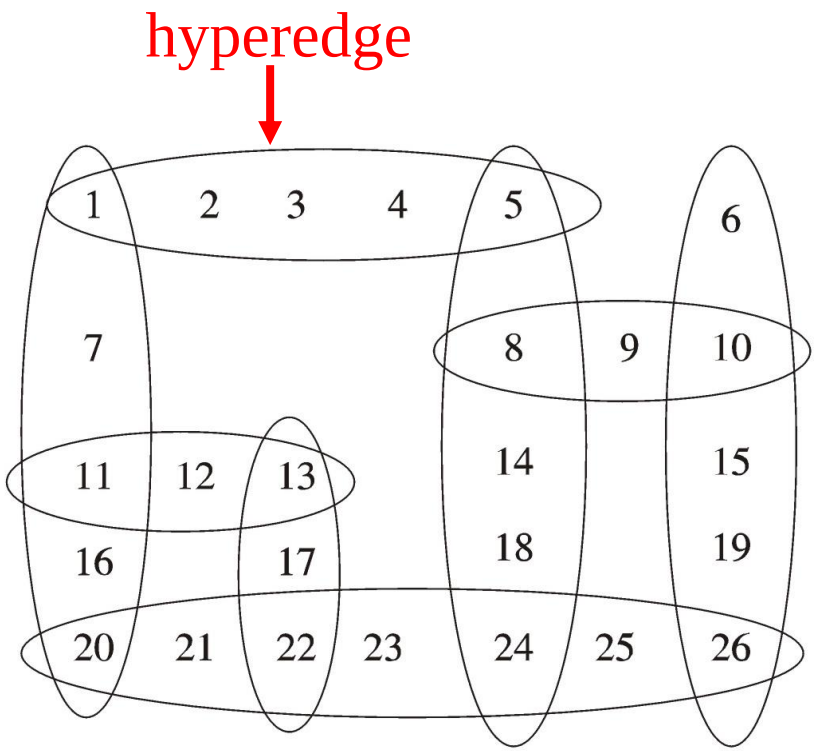
1v:

L	E	O	N	A
L	I	M	B	O
P	E	T	R	A
P	A	M	P	A
P	E	T	E	R

and so on

Example of CSP: Crossword Puzzle





1h:

P	A	R	I	S
P	A	N	D	A
L	A	U	R	A
A	N	I	T	A

1v:

L	I	M	B	O
L	I	N	G	O
P	E	T	R	A
P	A	M	P	A
P	E	T	E	R

and so on

hypergraph

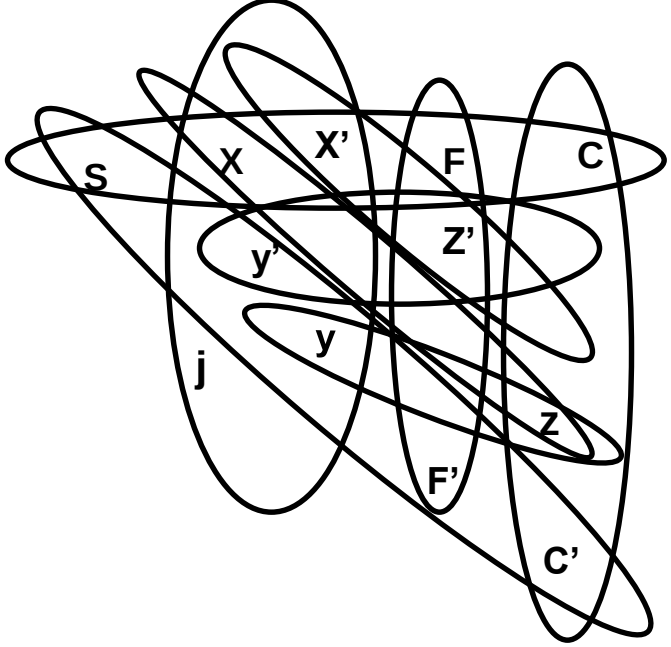
Decomposing Hypergraphs

We need a method for hypergraph decomposition such that:

1.) If the hypergraph is decomposable, then a bounded-width decomposition can be found efficiently.

2.) If there exists such a decomposition, then the problem can be solved efficiently

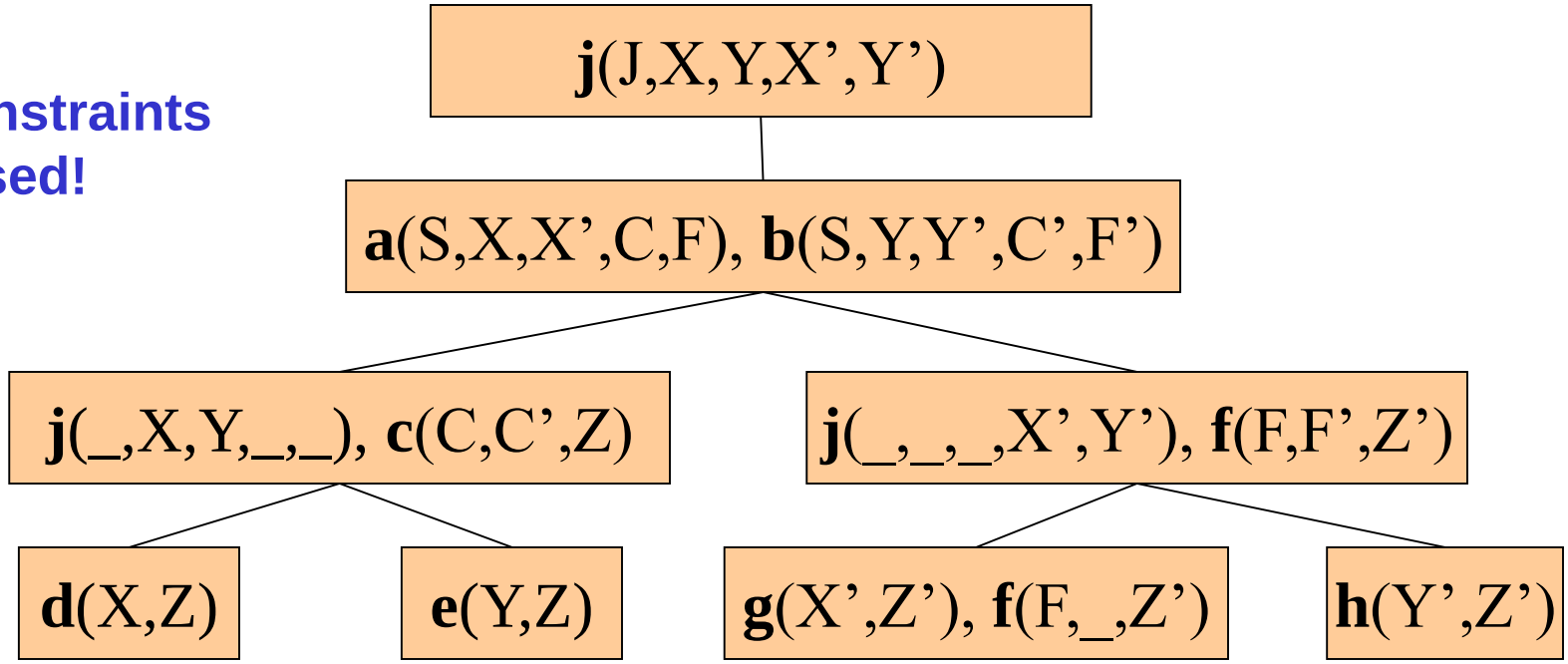
Constraints:
a(S,X,X’,C,F), b(S,Y,Y’,C’,F’), b(S,Y,Y’,C’,F’),
c(C,C’Z), d(X,Z), e(Y,Z), f(F,F’,Z), g(X’,Z’),
h(Y’,Z’), j(X,Y,X’,Y’)



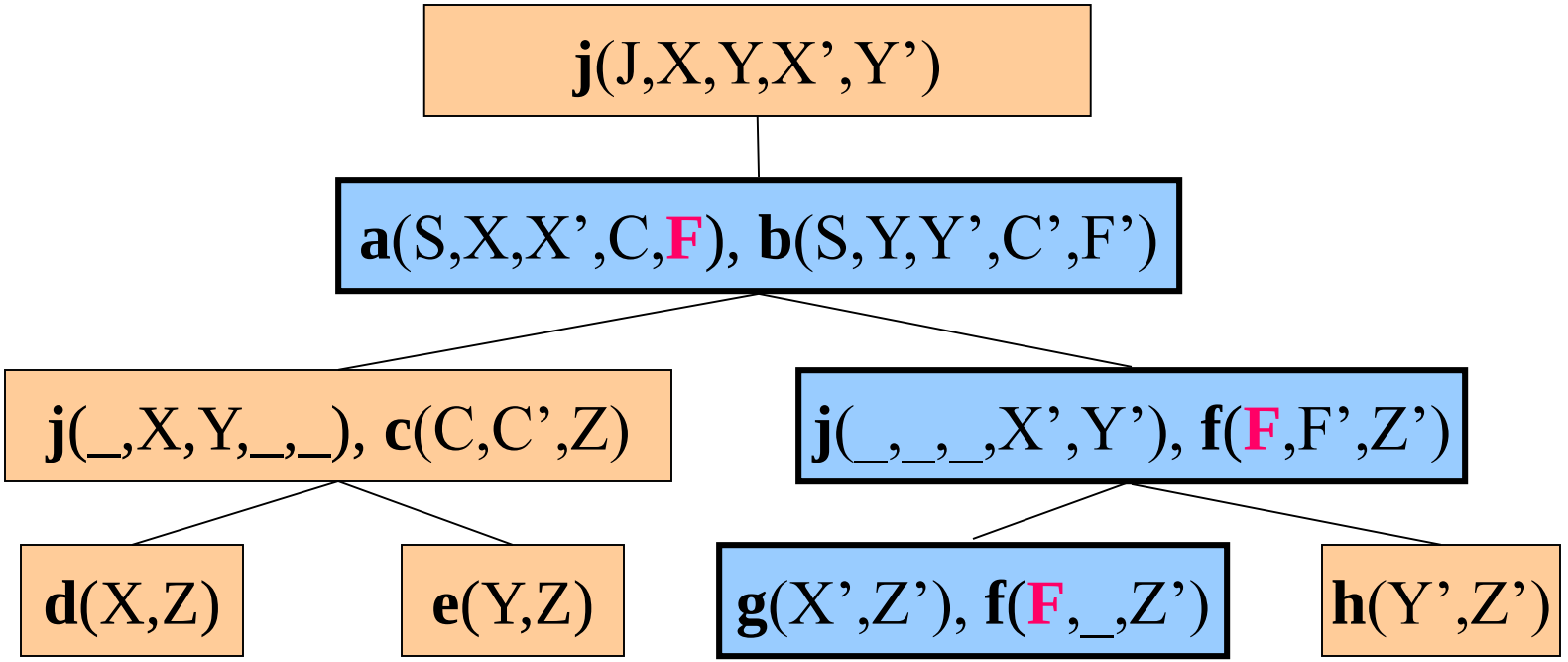
Hypertree decomposition:

[G., Leone, Scarcello 1989]

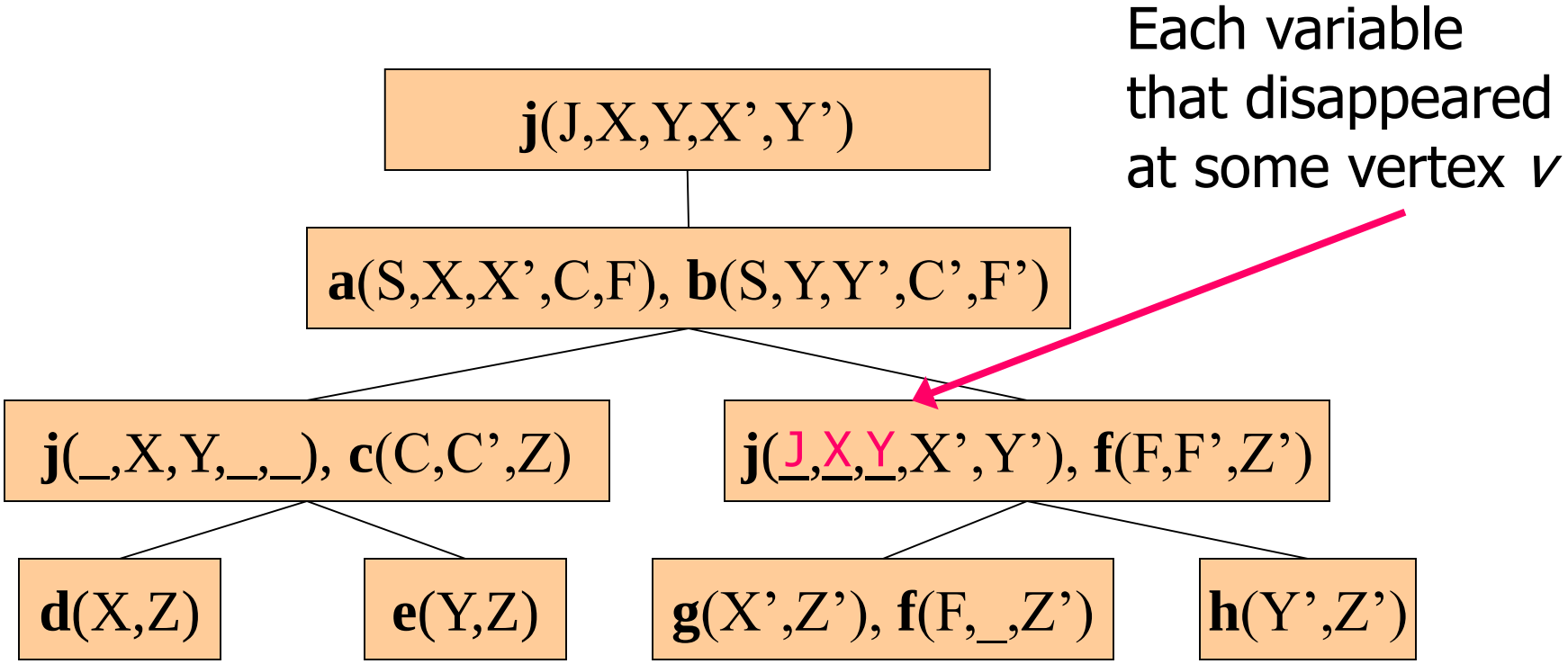
Partial constraints may be used!



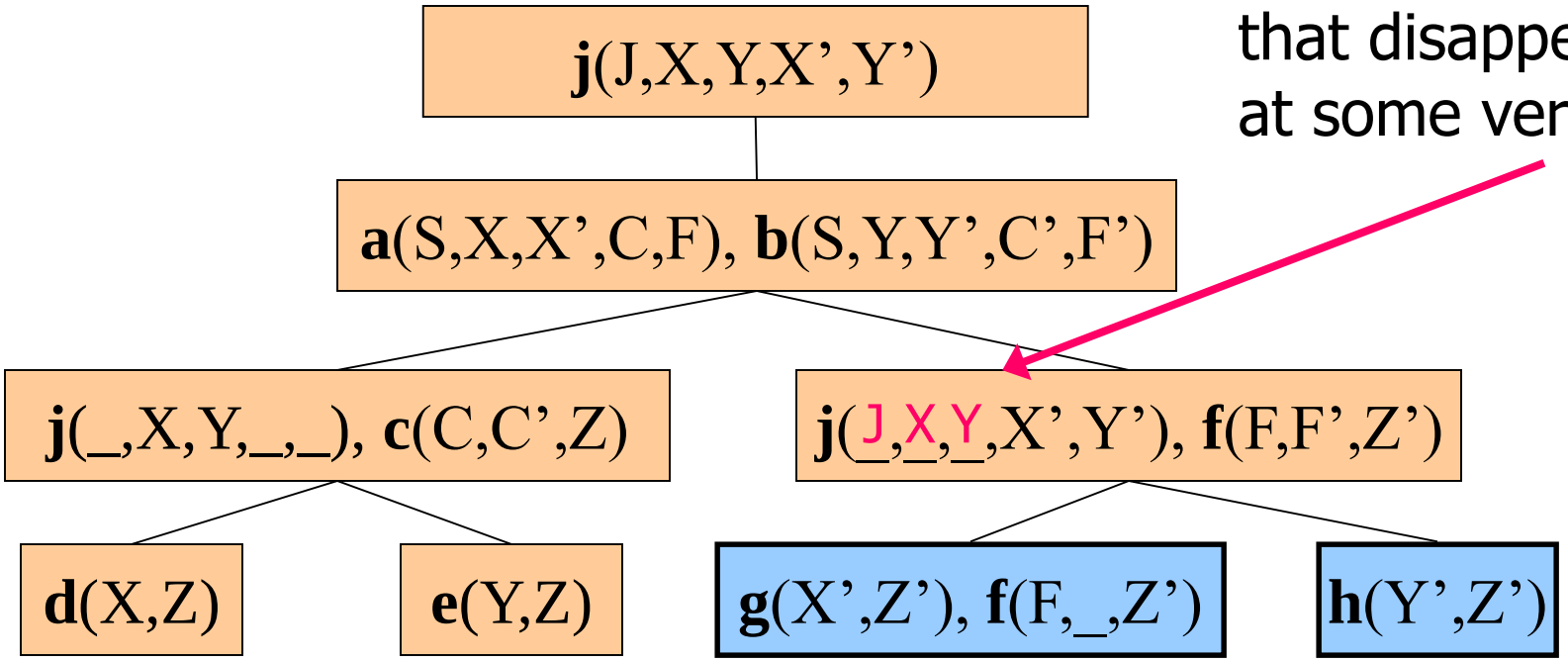
Connectedness Condition



Special Condition



Special Condition



Positive Results

1. Checking whether a hypertree decomposition of small width ($\leq$ constant k) feasible in polytime.

2. CSPs of small hypertree width can be solved efficiently.

3. For each hypergraph H $HW(H) \leq QW(H)$

4. For some hypergraph H $HW(H) < QW(H)$

Combinatorial Auctions

Bidders can place bids on Packages of items.

Winner determination: Choose a set of compatible bids of maximum revenue or minimum cost.

For classical auctions, winner determination is obviously tractable. Not so for CAs.

Combinatorial Auctions

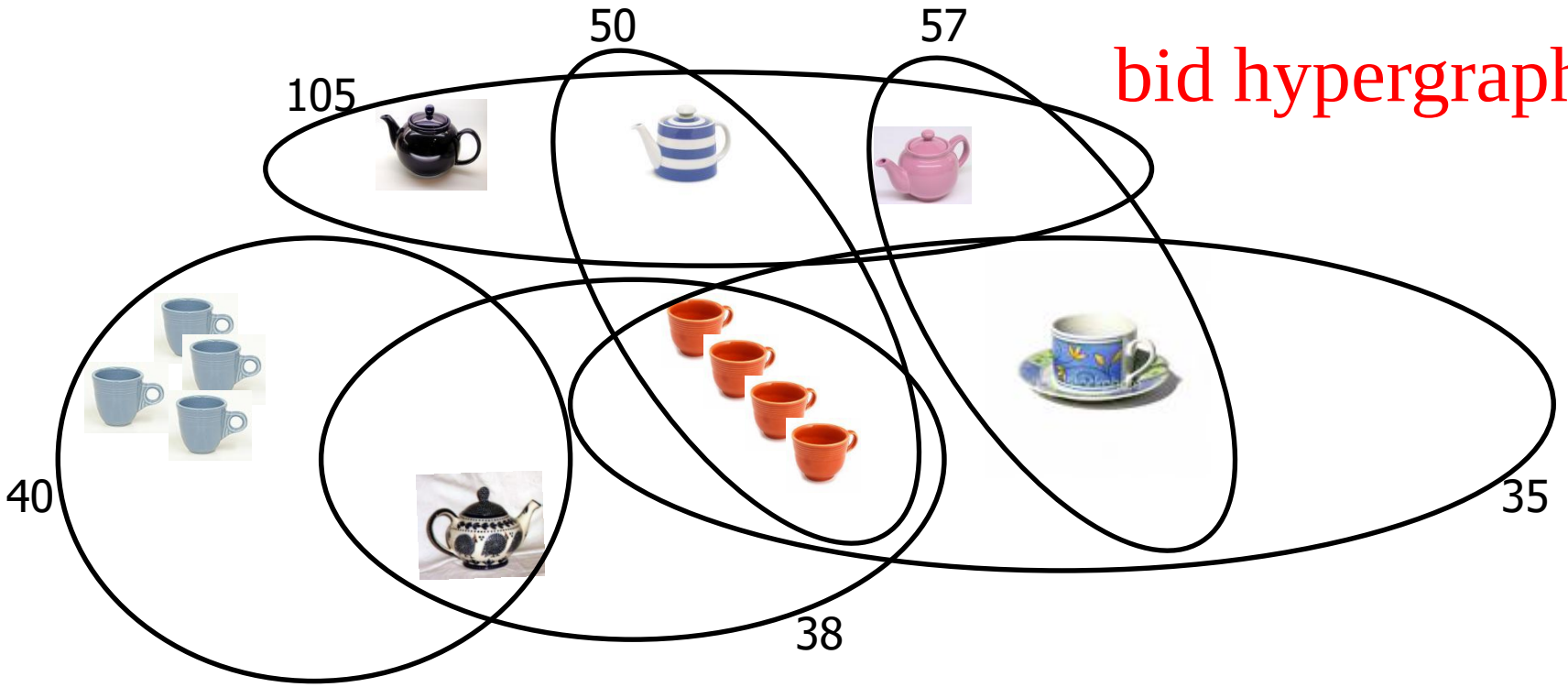






Combinatorial Auctions

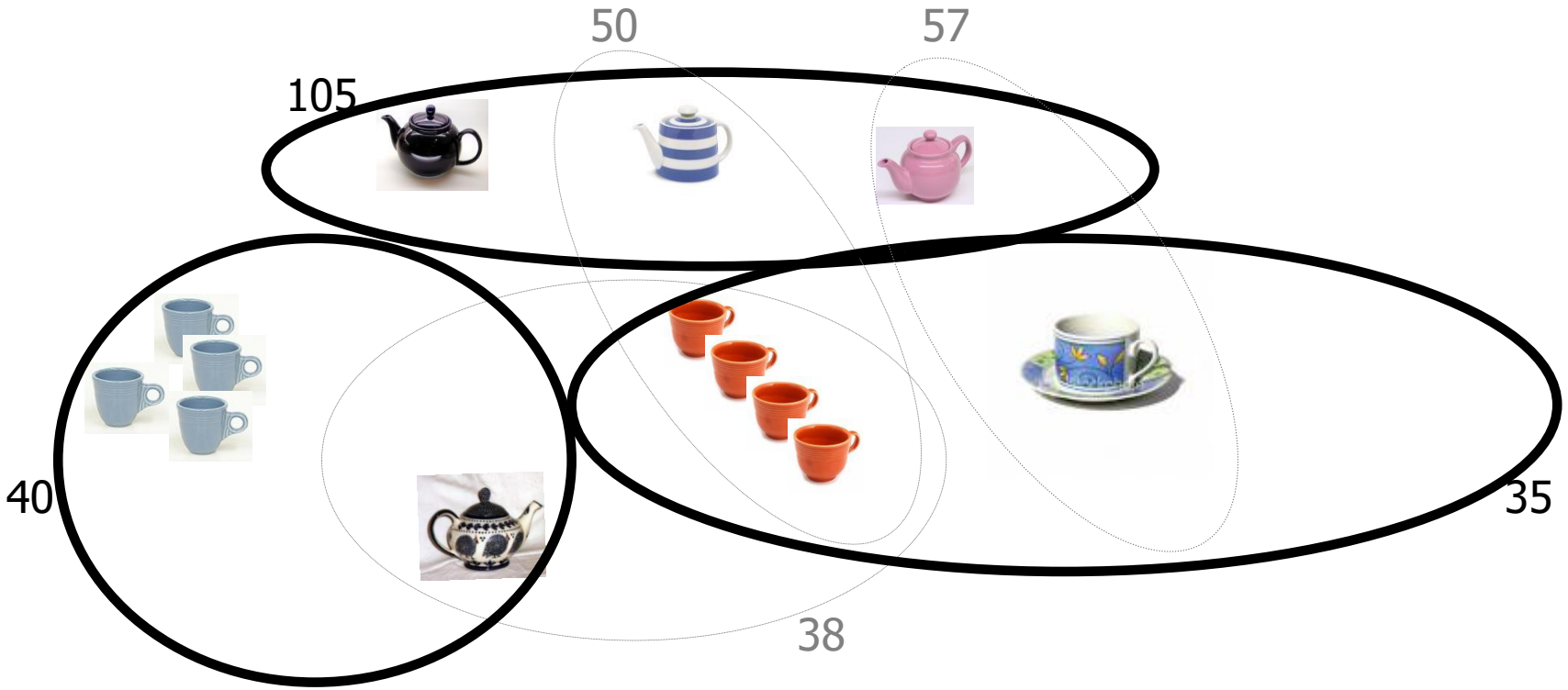
Combinatorial Auctions



Winner determination is intractable (NP-hard)

≡ weighted set packing problem

Combinatorial Auctions

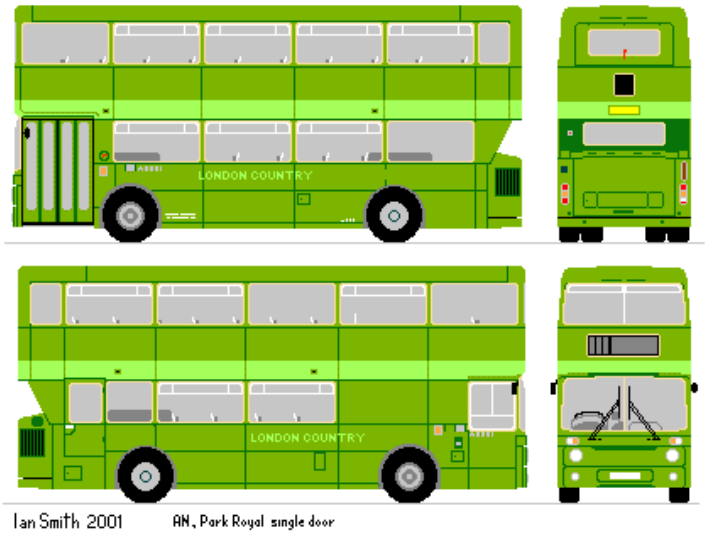


Winner determination is intractable (NP-hard)

≡ weighted set packing problem

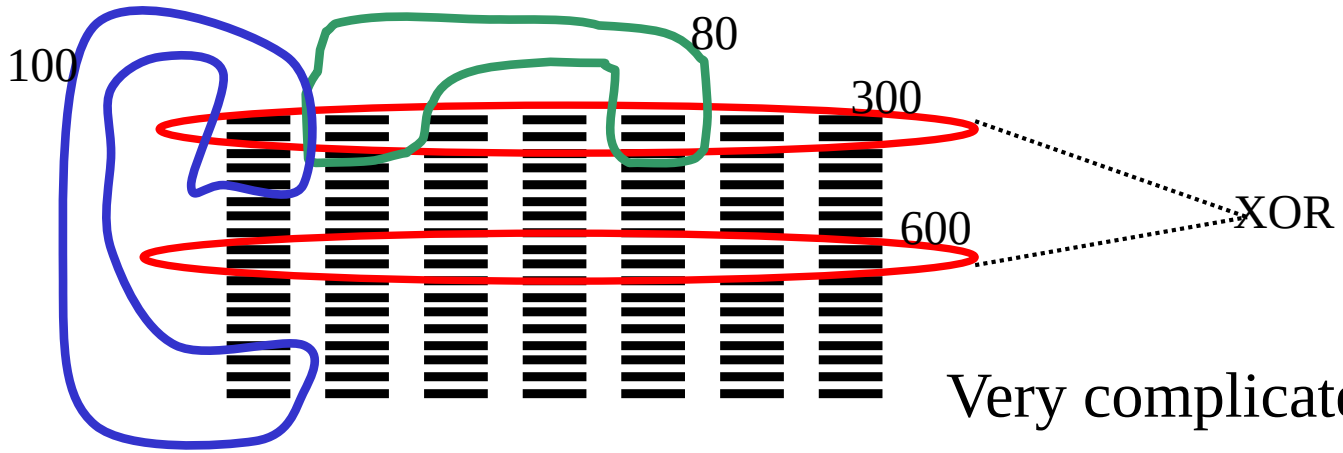
Total £ 180.--

Applications in different domains



London Regional Transports:
Combinatorial auctions of bus routes.
Private bus companies bid on bundles of routes.

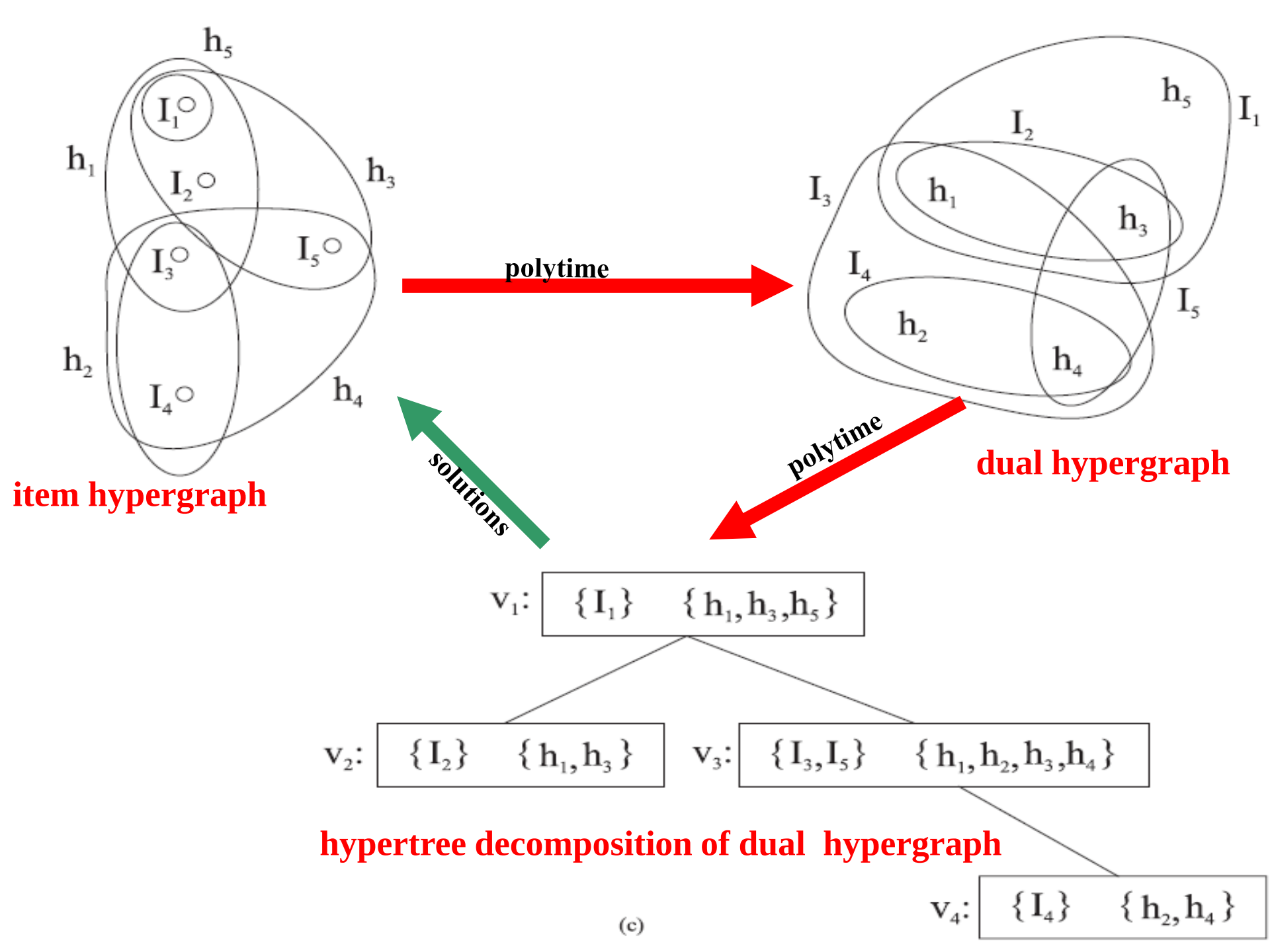
Airport slot auction



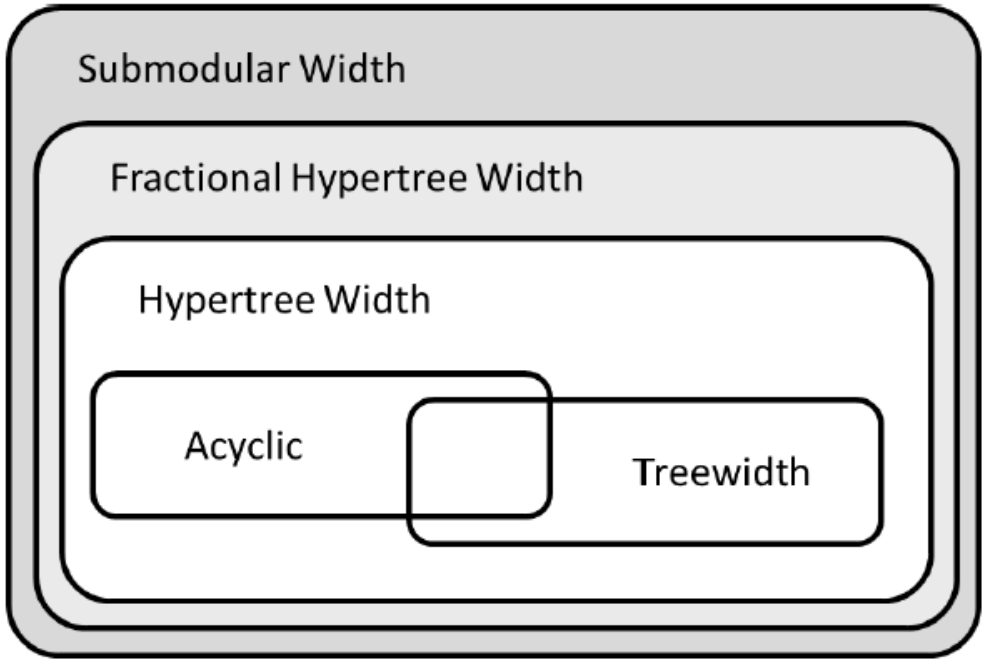
Very complicated hypergraphs!

Theorem: The winner determination problem can be solved efficiently if the dual hypergraph has bounded hypertree width. [G.&Greco, 2007]

In practice, this is often the case!


h5
I1
h1
I2
h3
I3
I5
h2
I4
h4
item hypergraph
polytime
I1
h5
I2
I3
h1
h3
I4
I5
h2
h4
dual hypergraph
polytime
solutions
v1: {I1} {h1,h3,h5}
v2: {I2} {h1,h3}
v3: {I3,I5} {h1,h2,h3,h4}
v4: {I4} {h2,h4}
hypertree decomposition of dual hypergraph
(c)

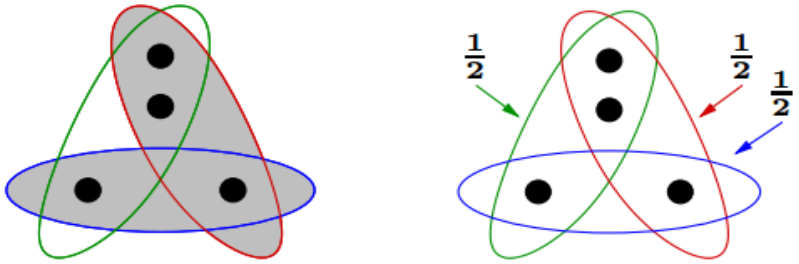
Can we go beyond hypertrees?



- *Tractable Recognizability, Tractable Query-Answering*
- *Unknown Complexity of Recognizability, Tractable Query-Answering*
- *Unknown Complexity of Recognizability, Exponential Algorithms for Query-Answering*

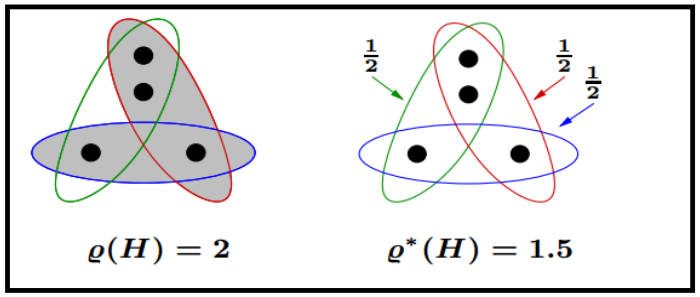
A **fractional edge cover** is a weight assignment to the edges such that every vertex is covered by total weight at least 1.

$\varrho^*(H)$: smallest total weight of a fractional edge cover.



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

For each query Q, an optimal fractional edge cover for Q minimizing $\rho^*(Q)$ can be computed in polynomial time via a linear program.

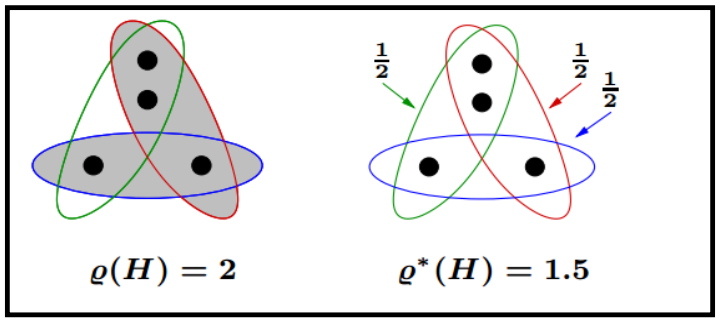


Theorem [Grohe & Marx 06] : The answer to a query of fractional cover of weight $\rho^*(Q)$ is at most of size $rmax^{\rho^*(Q)}$ and can be computed in time $|Q| \times rmax^{\rho^*(Q)+O(1)}$

Theorem [Atserias, Grohe & Marx 08] : The size-bound is essentially tight.

Theorem [Atserias, Grohe & Marx 08] : No join-program can simulate the GM-method efficiently.

A closer look at the triangle query



$r(X,Y) \bowtie s(Y,Z) \bowtie t(X,Z)$

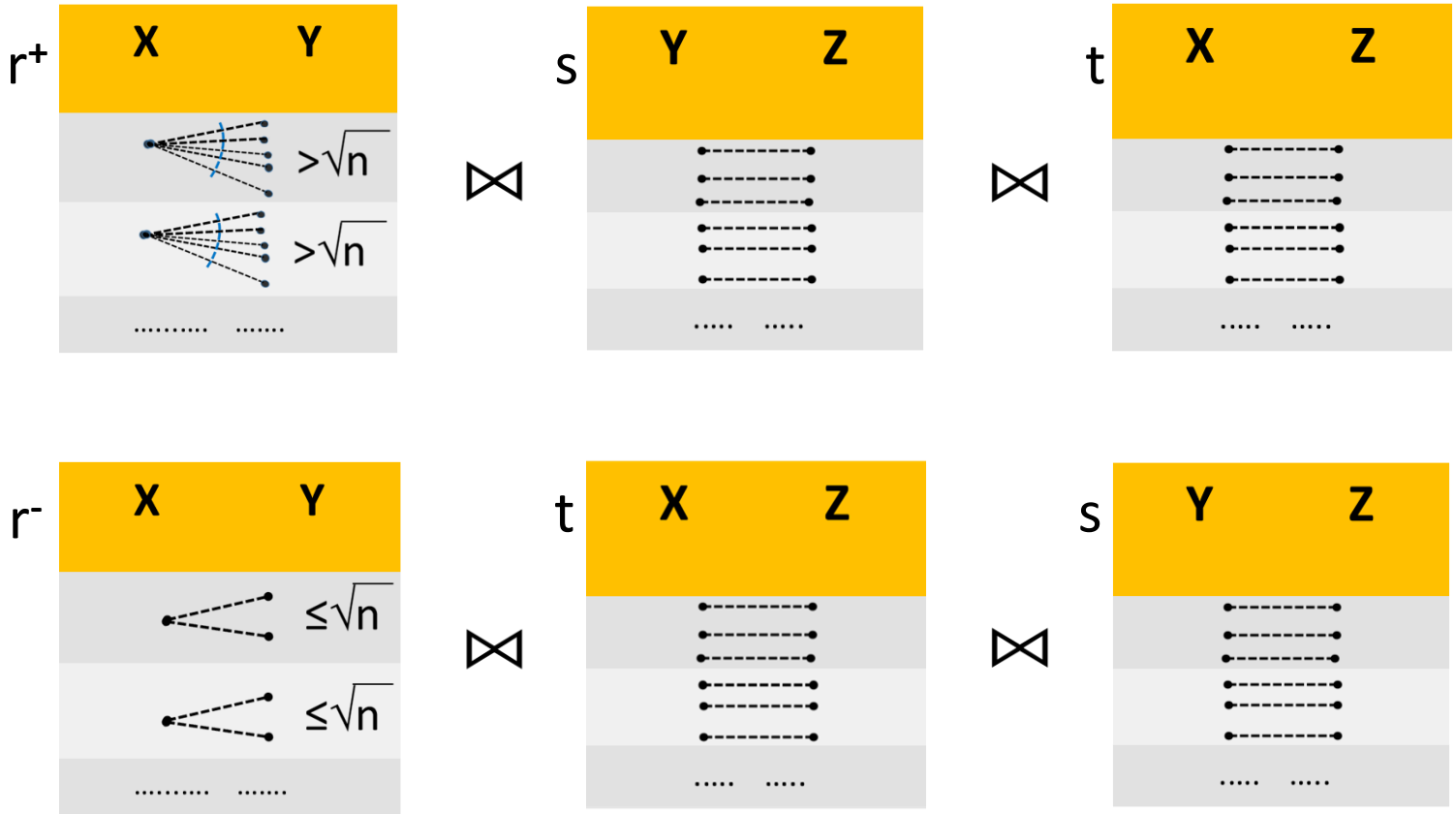
Let $n = r_{max}$ = size of largest relation.

Any of the joins $r \bowtie s$, $r \bowtie t$, $s \bowtie t$ may have size n^2 already.

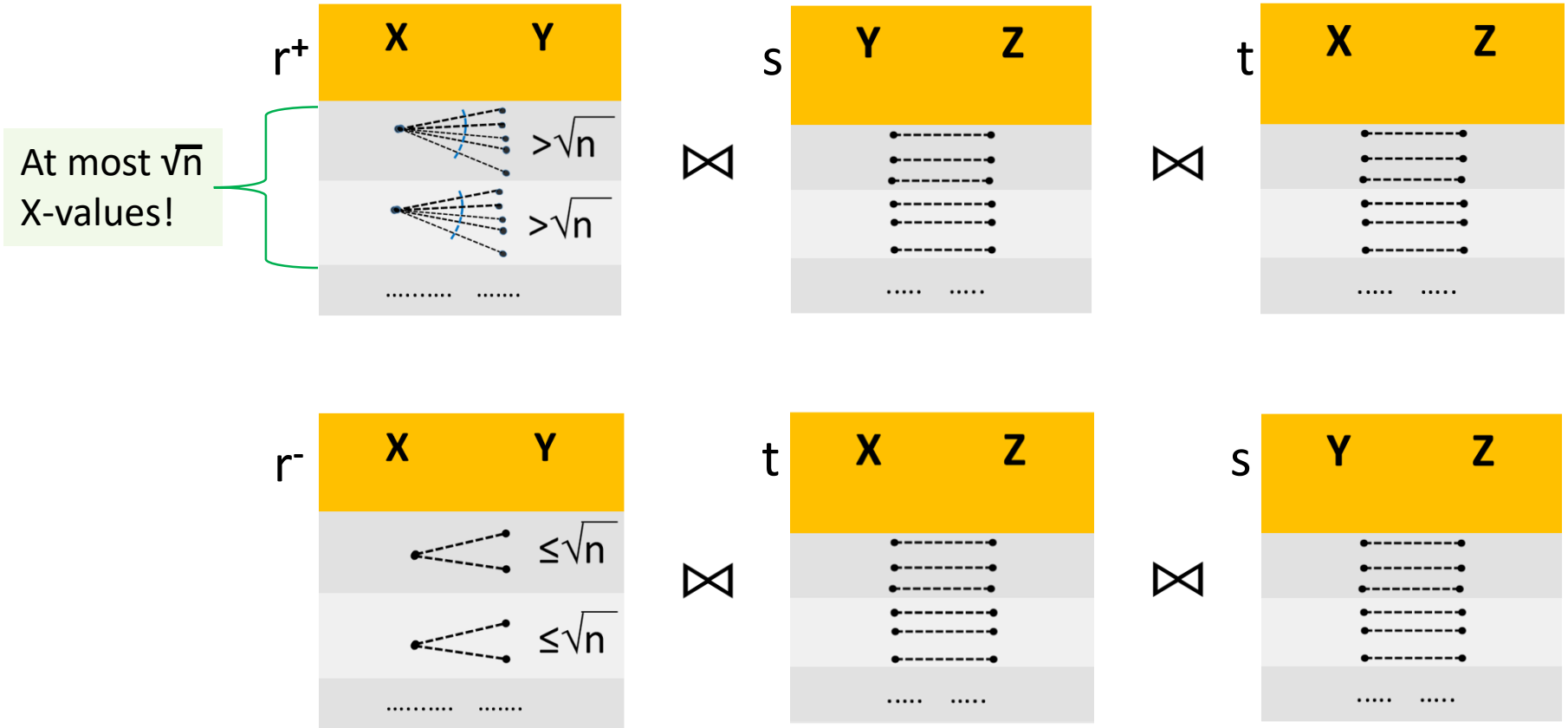
But GM say that $r \bowtie s \bowtie t$ has size $n^{\rho^*} = n^{3/2}$ only.

How that, and how is it possible to compute it w/o intermediate results of size n^2 ?

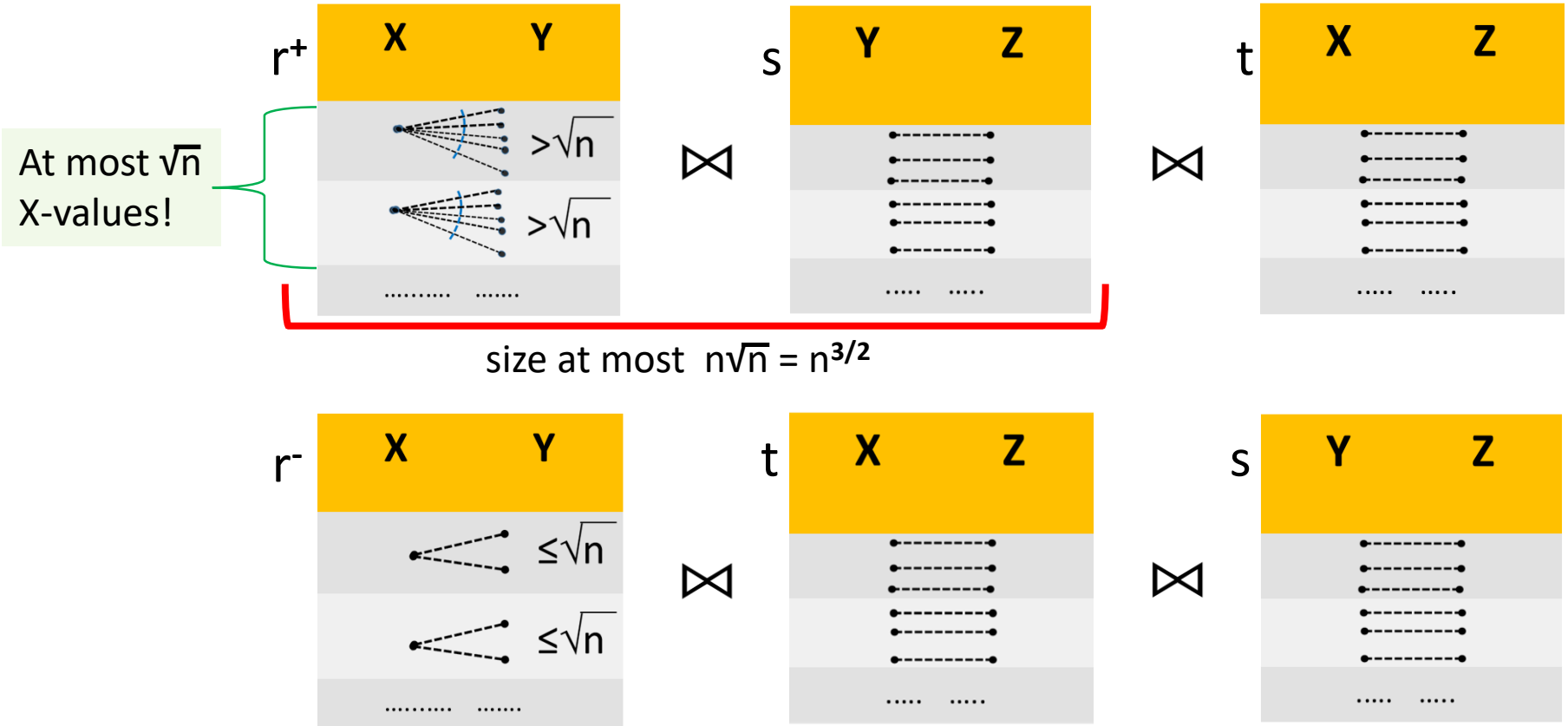
$$r(X,Y) \bowtie s(Y,Z) \bowtie t(X,Z)$$



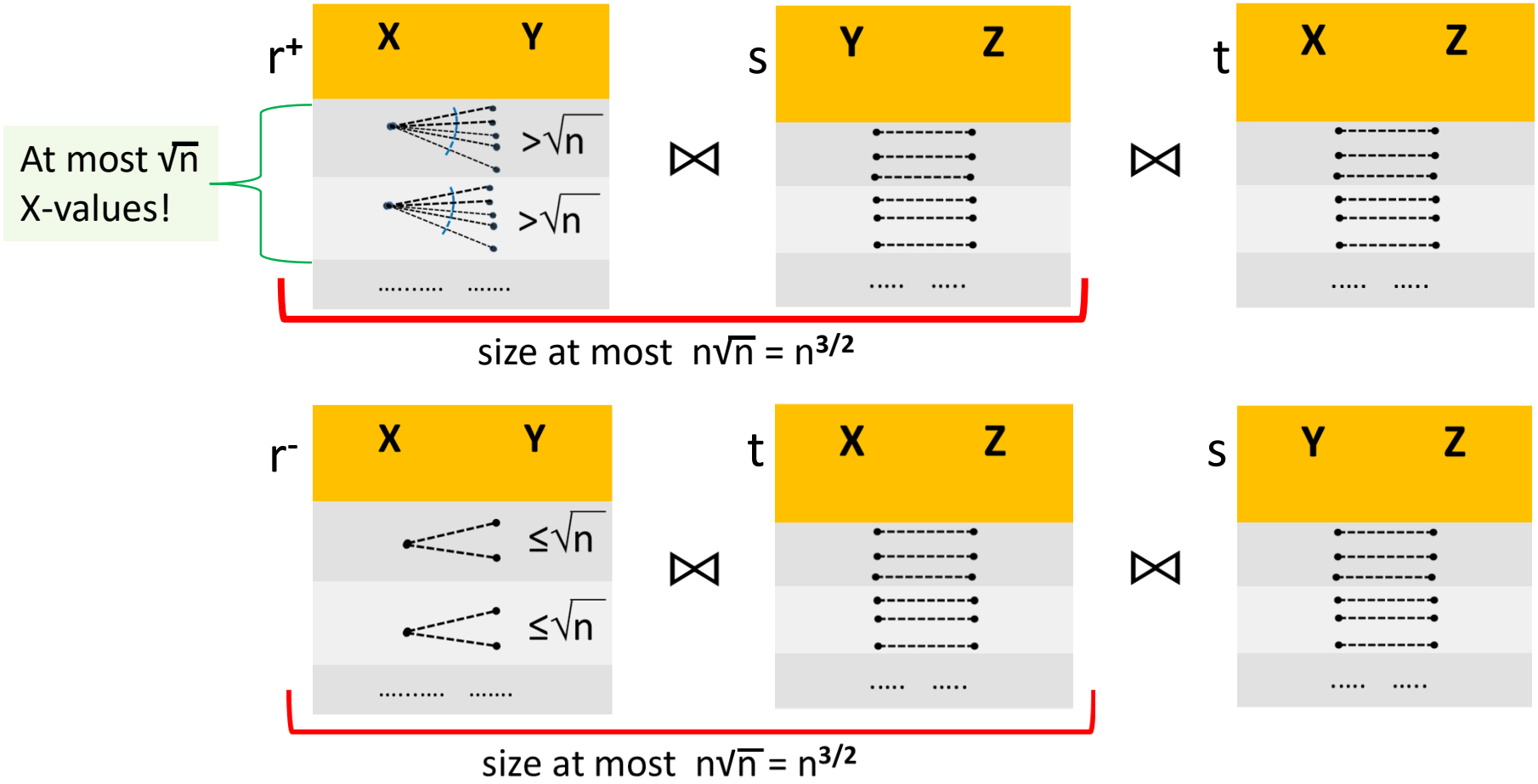
$$r(X,Y) \bowtie s(Y,Z) \bowtie t(X,Z)$$



$$r(X,Y) \bowtie s(Y,Z) \bowtie t(X,Z)$$



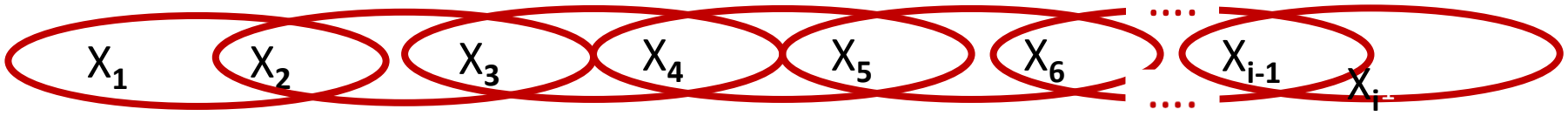
$$r(X,Y) \bowtie s(Y,Z) \bowtie t(X,Z)$$



GHW and Fractional Cover Width ρ^* are Incomparable

Consider the infinite classes C of hypergraphs:

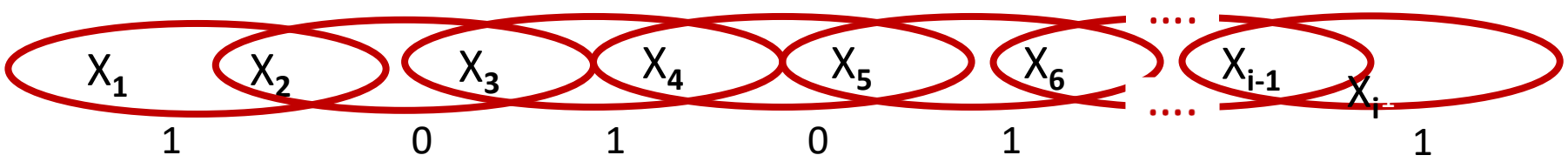
Chains H_i :



GHW and Fractional Cover Width ρ^* are Incomparable

Consider the infinite classes **C** of hypergraphs:

Chains H_i :

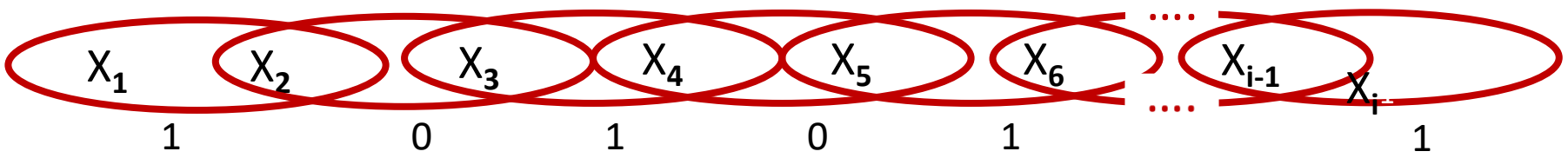


$\rho^*(Hi) \geq i/2$ and thus $\rho^*(C) = \infty$, however hw(C)=ghw(C)=1.

GHW and Fractional Cover Width ρ^* are Incomparable

Consider the infinite classes **C** of hypergraphs:

Chains H_i :

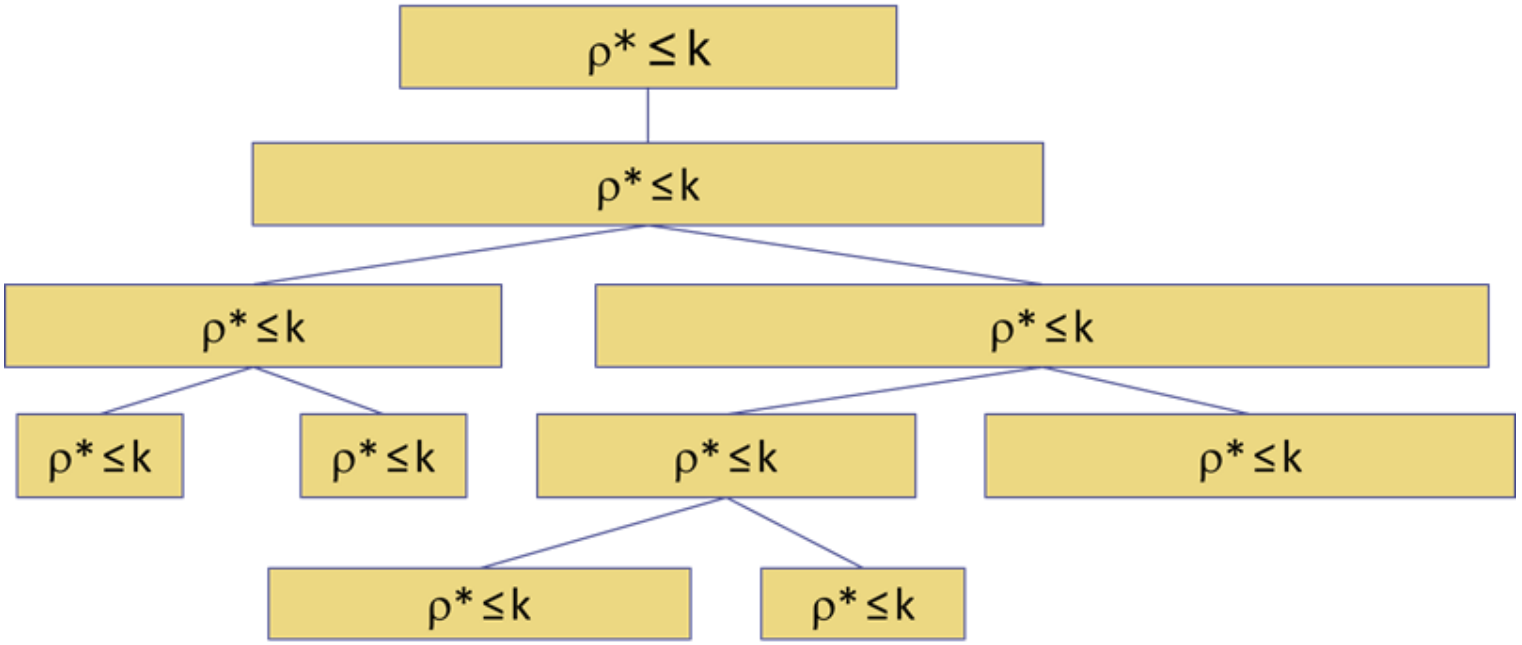


$\rho^*(Hi) \geq i/2$ and thus $\rho^*(C) = \infty$, however hw(C)=ghw(C)=1.

On the other hand, there are (more complicated) examples in the literature where GHW is infinite and ρ^* finite.

To combine the two notions profitably, Grohe and Marx defined **fractional hypertree decompositions (FHDs)** and correspondingly FHW

FHD of width ≤k:



Recent result [Fischl,G.,Pichler]:

Theorem : Deciding fhw(Q)=2 is NP-complete.

Recent result [Fischl,G.,Pichler]:

Theorem : Deciding fhw(Q)=2 is NP-complete.

Various hypergraph restrictions lead to:

- tractability (for GHD, FGD), and
- approximability (for FHD).

Restrictions for Tractability

A class C of hypergraphs enjoys:

BIP (bounded intersection property): $\exists i\ \forall H \in C,\ \forall e_1, e_2 \in E(H),\ |e_1 \cap e_2| \leq i.$

BMIP (bd. multi-intersection prop.): $\exists i\ \exists c\ \forall H \in C,\ \forall e_1 \ldots e_c \in E(H),\ |e_1 \cap \ldots \cap e_c| \leq i.$

BR (bounded rank): $\exists r\ \forall H \in C\ \forall e \in E(H),\ |e| \leq i.$

BD (bounded degree): $\exists d\ \forall H \in C\ \forall v \in V(H),\ |\{e \in E(H) \mid v \in e\}| \leq d.$

BVC (bounded VC-dimension): $\exists \delta\ \forall H \in C\ \ vc(H) \leq \delta$

Notes: BR→BIP→BMIP→BVC; BD→MIP; none of the implications reversible.

Results

Property	GHW=k	FHW=k
BIP	PTIME-checkable	PTIME Approx: $O(k \log k)$
MIP	PTIME-checkable	PTIME Approx: $O(k \log k)$
BR	PTIME-checkable	PTIME-checkable
BD	PTIME-checkable	PTIME Approx: $\delta(H)*k$
BVC	NP-complete	PTIME Approx: $O(k \log k)$

If no decomposition is possible…

- Polynomial-time approximations

  Beautiful theory, but:

  - Often an exact solution is desired
  - More negative than positive results

    E.g. Winner determination problem not PTA unless NP=ZPP

- Probabilistic and randomized algorithms

  Well-suited for a certain class of problems

  Not well suited for the NP-hard problems we are interested in

  Heuristics: randomized local search

If no decomposition is possible…

- Polynomial-time approximations

  Beautiful theory, but:

  - Often an exact solution is desired
  - More negative than positive results

    E.g. Winner determination problem not PTA unless NP=ZPP

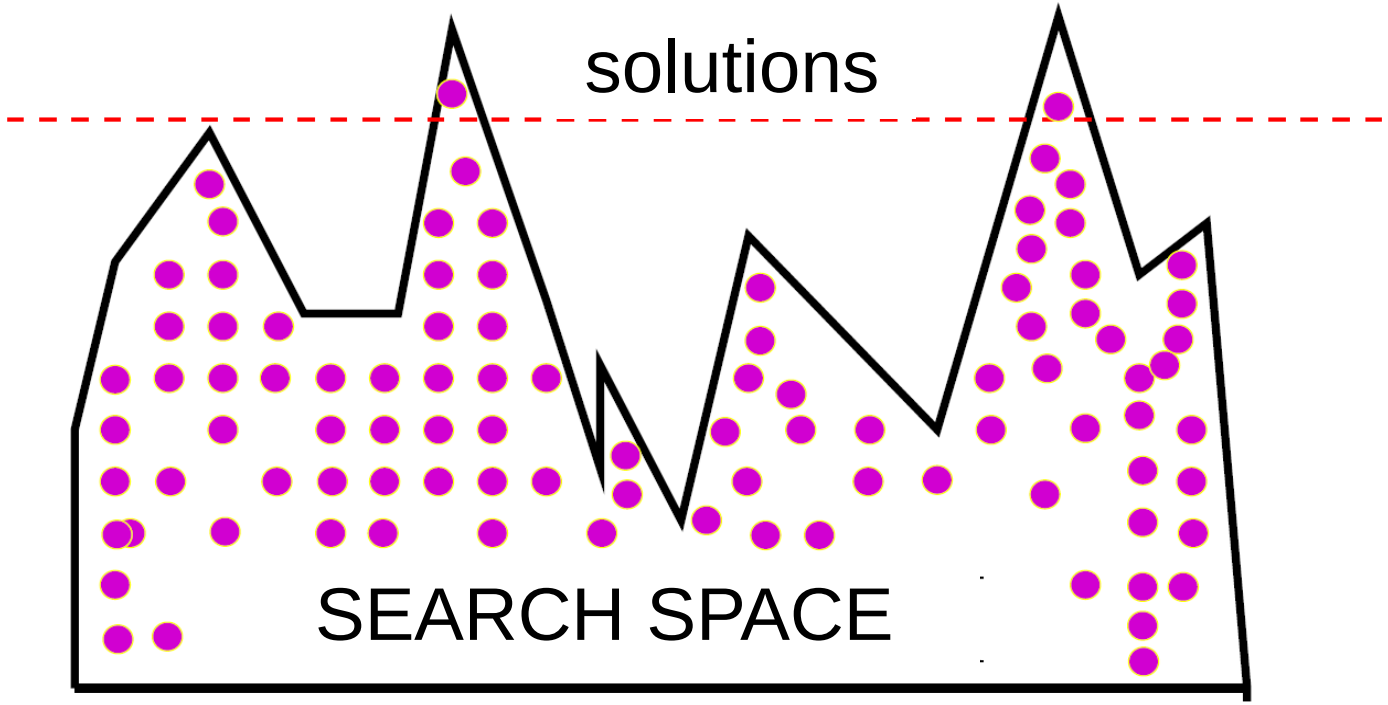
- Probabilistic and randomized algorithms

  Well-suited for a certain class of problems

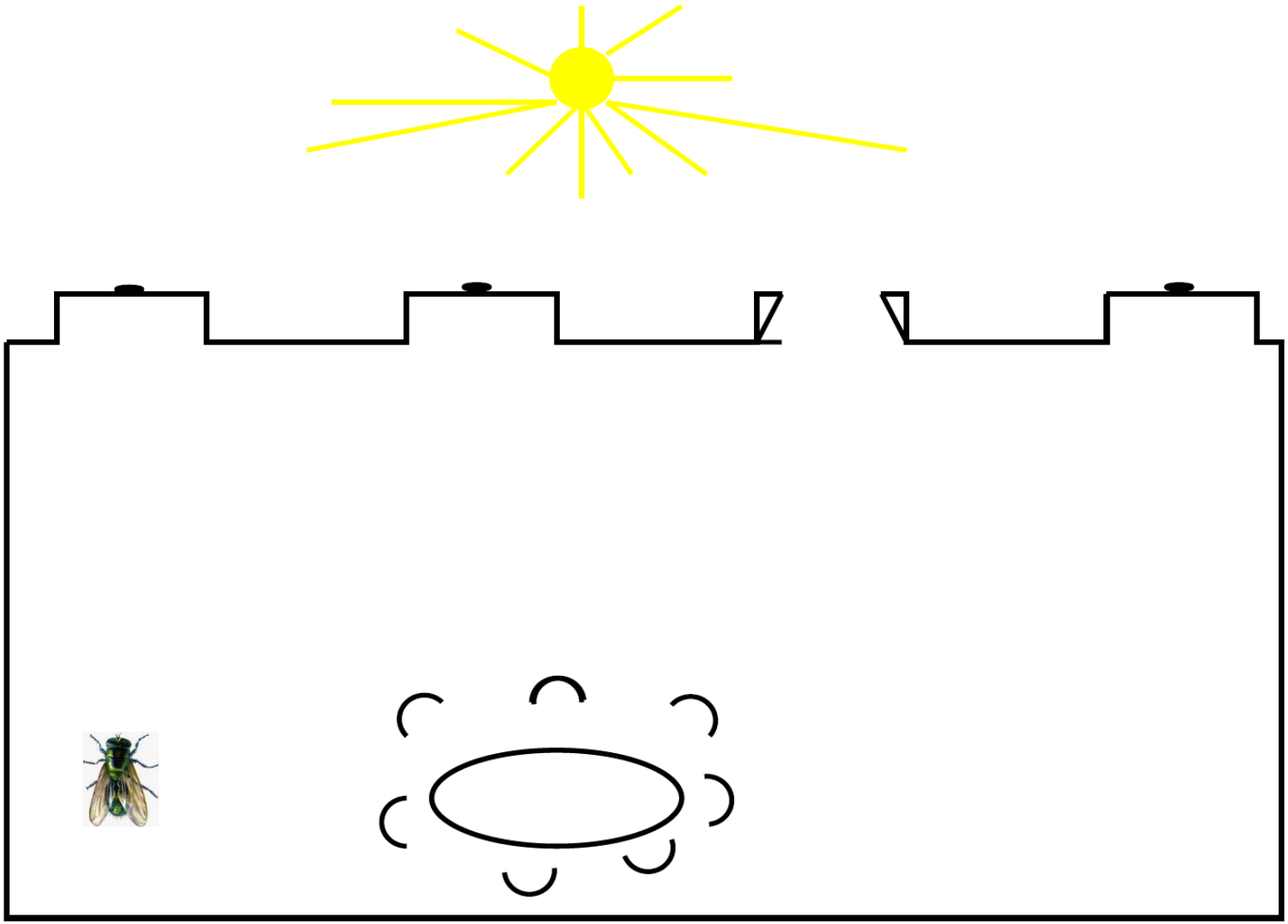
  Not well suited for the NP-hard problems we are interested in

Heuristics: randomized local search

- Randomized local search



Note: This method has been developed by evolution in insects and mammals



This randomized local strategy allows a fly to escape from a room even though it has an extremely limited cognition of the environment.

Randomized local search allows a computer to find with high probability a “good” solution even though it has a limited “cognition” of the exponentially large solution space.

DISCLAIMER: The next slides are speculative ideas and not part of the scientific talk.

Randomized local search:

The method of choice if search space is too complex to be "*cognitively*" explored

<u>Triggers of randomness:</u>

For flies probably:

adrenaline excess?

→ need of motion to lower adrenaline

→ random move

Triggers of randomness:

For humans:

1. Frustration-induced anger (makes us lose control)

Triggers of randomness:

For humans:

1. Frustration-induced anger (makes us lose control)

Triggers of randomness:

For humans:

2. Oracles, horoscopes, …
   deliberate sources of randomness

Intractability in Nature?

Intractability in Nature?

VOL. 176, NO. 6 THE AMERICAN NATURALIST DECEMBER 2010

Travel Optimization by Foraging Bumblebees through Readjustments of Traplines after Discovery of New Feeding Locations

Mathieu Lihoreau,[1] Lars Chittka,[1] and Nigel E. Raine[1,2,*]

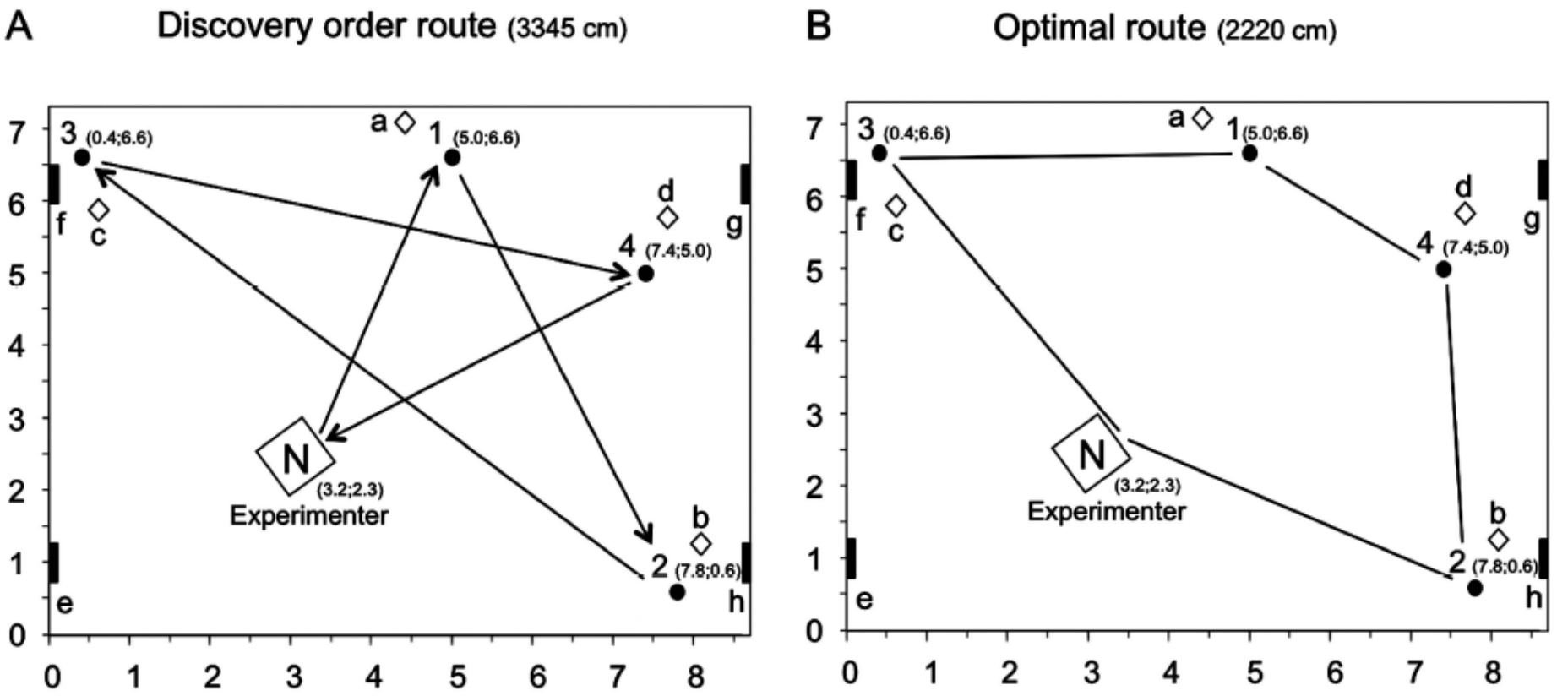
1. Research Centre for Psychology, School of Biological and Chemical Sciences, Queen Mary University of London, Mile End Road, London E1 4NS, United Kingdom; 2. School of Biological Sciences, Royal Holloway University of London, Egham, Surrey TW20 OEX, United Kingdom

Submitted March 24, 2010; Accepted August 10, 2010; Electronically published October 25, 2010

Online enhancements: appendix figures.

ABSTRACT: Animals collecting resources that replenish over time

Navigation can be achieved using different mechanisms—such as directional compasses (e.g., Muheim et al. 2006)


A
Discovery order route (3345 cm)
B
Optimal route (2220 cm)
3 (0.4;6.6)
a
1 (5.0;6.6)
d
f
c
g
4 (7.4;5.0)
N
(3.2;2.3)
Experimenter
b
2 (7.8;0.6)
e
h
0 1 2 3 4 5 6 7 8

Trapline Foraging and Dynamic Traveling Salesman Problems

Traplining animals are faced with complex routing problems analogous to the well-known traveling salesman problem (Applegate et al. 2006; Gutin and Punnen 2006). Rather than calculating and comparing all possible alternative paths, it is generally assumed that animals rely on simple heuristics coupled with some form of spatial memory that gives a reasonable approximation to the optimal solution with relatively little cognitive effort (Anderson 1983; Cramer and Gallistel 1997).

Evolution can be seen as a massively parallel randomized local optimization process. Goal = maximum fitness.

Nature proceeds in a fundamentally local manner.

Global phenomena are often of catastrophic.

Nature neither poses nor solves NP-hard problems.

Such problems arise from our capability of abstraction and of making (mental) models → **intelligence**

Using such models (e.g. maps) we can pose hard problems on non-local issues, and our computers can solve them

But – on the other hand – we are part of nature!