Cost Function Networks

PFIA – 4 July 2019

Martin Cooper
IRIT, Toulouse

Simon de Givry
INRA, Toulouse

Cost Function Networks

- What are they and why do we need them?
- Tractable classes
- Search using soft consistency
- Hybrid tree and local search
- Toulbar2 solver demonstration

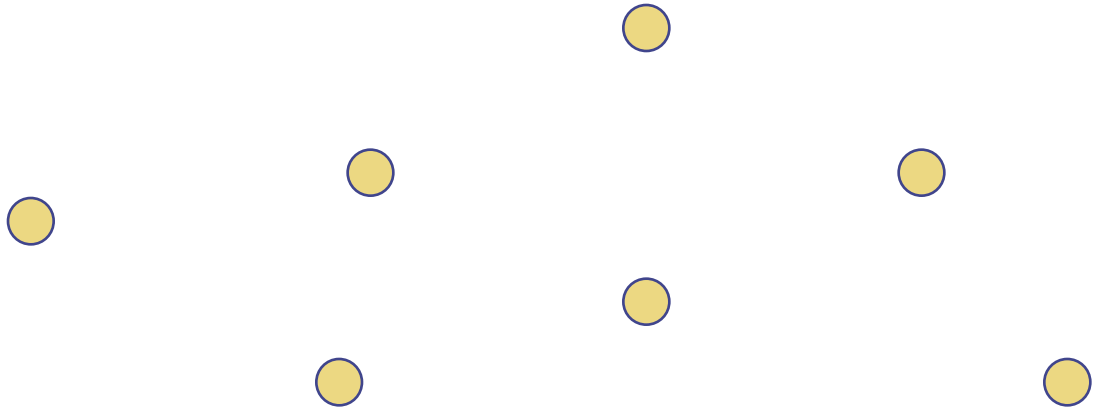
Chapter 1. What are CFNs?

Motivation,

Definitions,

Some general theorems

A unifying abstraction: CSP



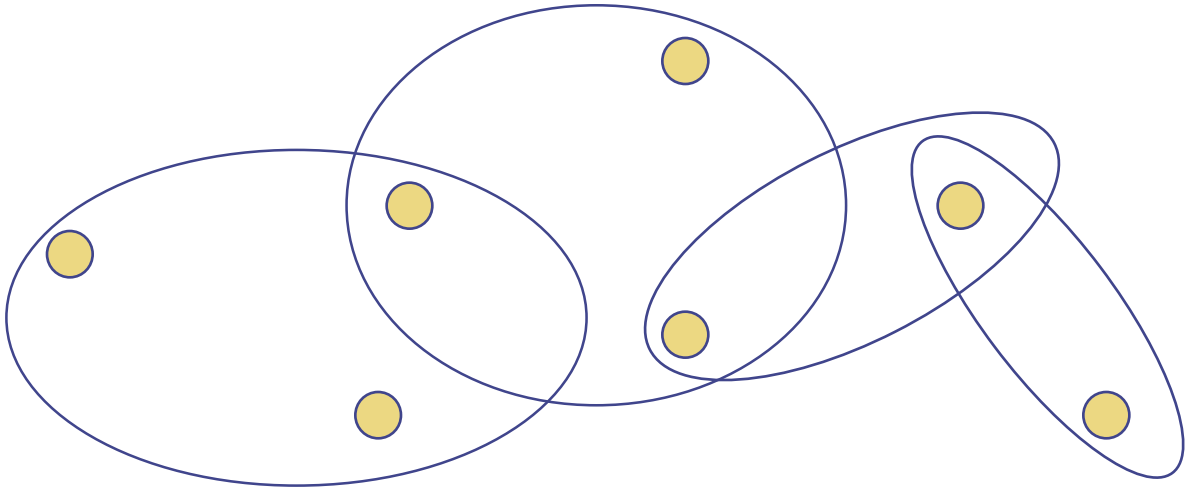
Variables = Talks to be scheduled at conference

Transmitters to be assigned frequencies

Amino acids to be located in space

Circuit components to be placed on a chip

A unifying abstraction: CSP



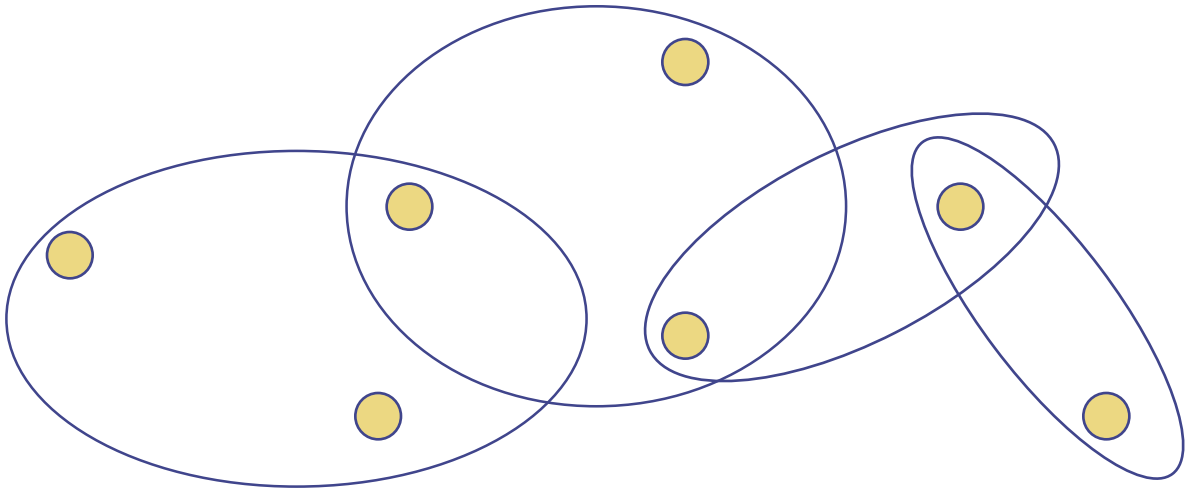
Constraints = All invited talks on different days

No interference between near transmitters

x + y + z > 0

Foundations dug before walls built

A unifying abstraction: CSP



A solution is an assignment of values to variables that satisfies all the constraints

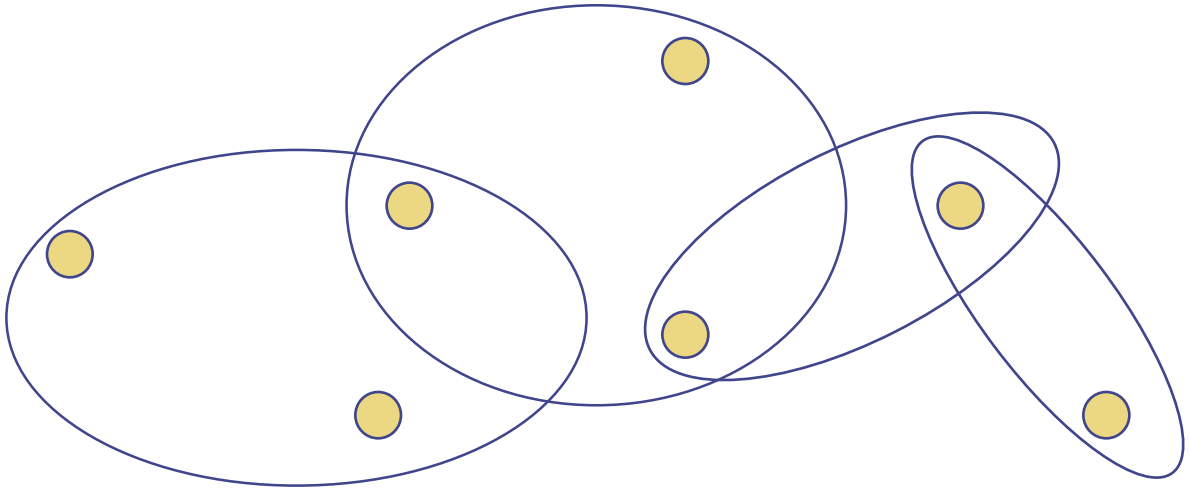
But what if…

- There are lots of solutions, but some are better than others?
- There are no solutions, but some assignments satisfy more constraints than others?
- We don’t know the exact constraints, only probabilities, or fuzzy membership functions?
- We’re willing to violate some constraints if we can get a better overall solution that way?

Fragmentation of research

- Constraint Optimisation Problem
- Max-CSP
- Max-SAT
- WCSP
- Fuzzy CSP
- Pseudo-Boolean Optimisation
- Bayesian Networks
- Markov Random Fields
- Integer Programming
- …

A unifying abstraction: CFN



Constraints ⬭ now associate costs with each assignment

A solution is an assignment of values to variables that minimises the combined costs

Definition of a CFN instance

- a set of n variables X_i with domains d_i
- a set of valued constraints, where each constraint has a
  - scope (list of variables)
  - cost function (function from assignments to costs)

It only remains to specify what the possible costs are, and how to combine them

Definition of a valuation structure

- a set S of costs
- a total order $<$
- minimum and maximum elements:
  we denote these by 0 and ∞
- an aggregation operator $\oplus$ which is commutative, associative, monotonic, and such that $\forall \alpha,\ \alpha \oplus 0 = \alpha$

Examples of valuation structures

- If S = {0, ∞}, then CFN ≡ CSP
- If S = non-negative reals ∪ {∞}, and ⊕ is addition, then CFN generalizes MAX-CSP.

  And we can model MPE in Bayesian networks or MAP in Markov random fields, by calculating the sum of -log(P(x | parents(x))
- If S = [0,1], and ⊕ is max, then CFN ≡ Fuzzy CSP
- If S = {0, 1, …., k}, and ⊕ is bounded addition $+_k$ where $\alpha +_k \beta = \min\{k, \alpha+\beta\}$, then CFN ≡ WCSP

Families of valuation structures

A valuation structure is idempotent if

$\forall \alpha,\ \alpha \oplus \alpha = \alpha$

All idempotent valuation structures are equivalent to Fuzzy CSP

Families of valuation structures

A valuation structure is strictly monotonic if

$\forall \alpha < \beta,\ \forall \gamma < \infty,\ \ \alpha \oplus \gamma < \beta \oplus \gamma$

A valuation structure is fair if

aggregation has a partial inverse, that is,

$\forall \alpha \geq \beta,\ \exists \gamma$ such that $\beta \oplus \gamma = \alpha$

All strictly monotonic valuation structures can be embedded in a fair valuation structure

Families of valuation structures

A valuation structure is discrete if between any pair of finite costs there are finitely many other costs

All discrete and fair valuation structures
can be decomposed into
a contiguous sequence of valuation structures
with aggregation operator $+_k$

Bibliography

- For general background on different formalisms for soft constraints, see the chapter on "*Soft Constraints*" by Meseguer, Rossi and Schiex, in the *Handbook of Constraint Programming*, Elsevier, 2006, or

  Cooper, de Givry & Schiex, "Réseaux de contraintes valuées", Chapter 7, *Panorama de l'Intelligence Artificielle*, Cépaduès, 2014.

- For classification results on valuation structures see "*Arc Consistency for Soft Constraints*", Cooper & Schiex, *AIJ*, 2004.

Chapter 2. Tractable classes

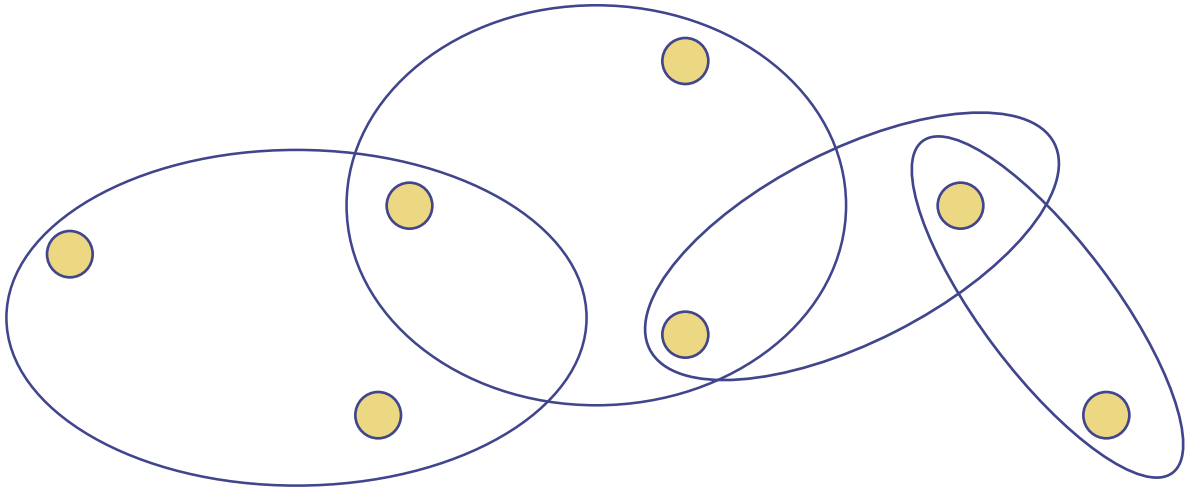
Structural restrictions,
Cost function languages,
Submodularity,
Weighted polymorphisms,
Hybrid tractability

General question

Having a unified formulation allows us to ask *general* questions about efficiency:

When is the CFN problem tractable?

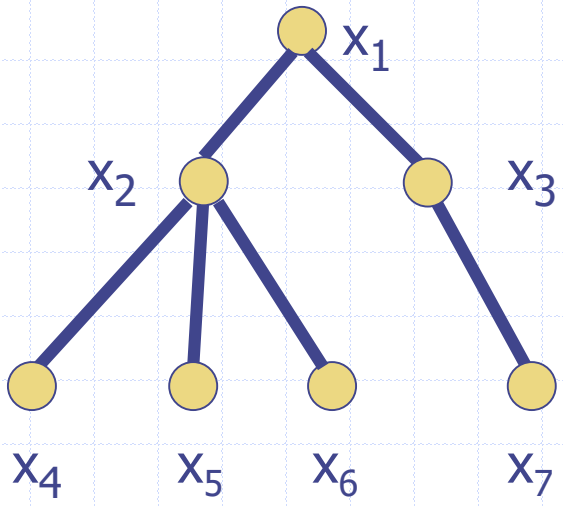
Problem features



- This picture illustrates the constraint scopes
- The set of scopes is sometimes called the constraint hypergraph, or the scheme
- Restricting the scheme can lead to tractability, as in the standard CSP

Structural tractability

Tree-structured binary CFNs are tractable



Time complexity $O(e\ d^2)$
Space complexity $O(n\ d)$

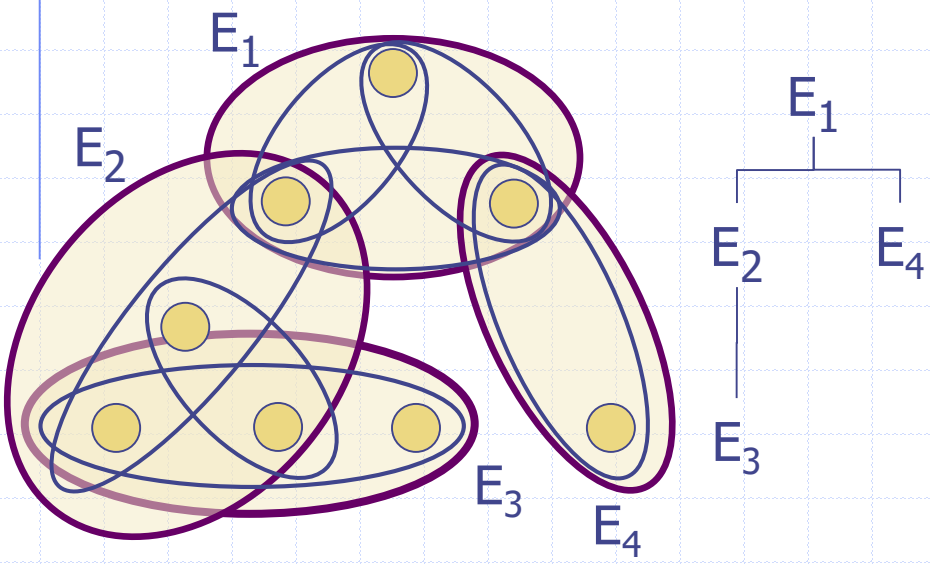
n: number of variables
d: maximum domain size
e: number of cost functions

Proceed from the leaf nodes to a chosen root node

Project out leaf nodes by minimising over possible assignments

Tree decomposition

Bounded treewidth CFNs are tractable



Time complexity $O(e\ d^{w+1})$
Space complexity $O(n\ d^{s})$

w: bounded treewidth
= max |Ei| - 1

s: max $\{|E_i \cap E_j|: i \neq j\}$

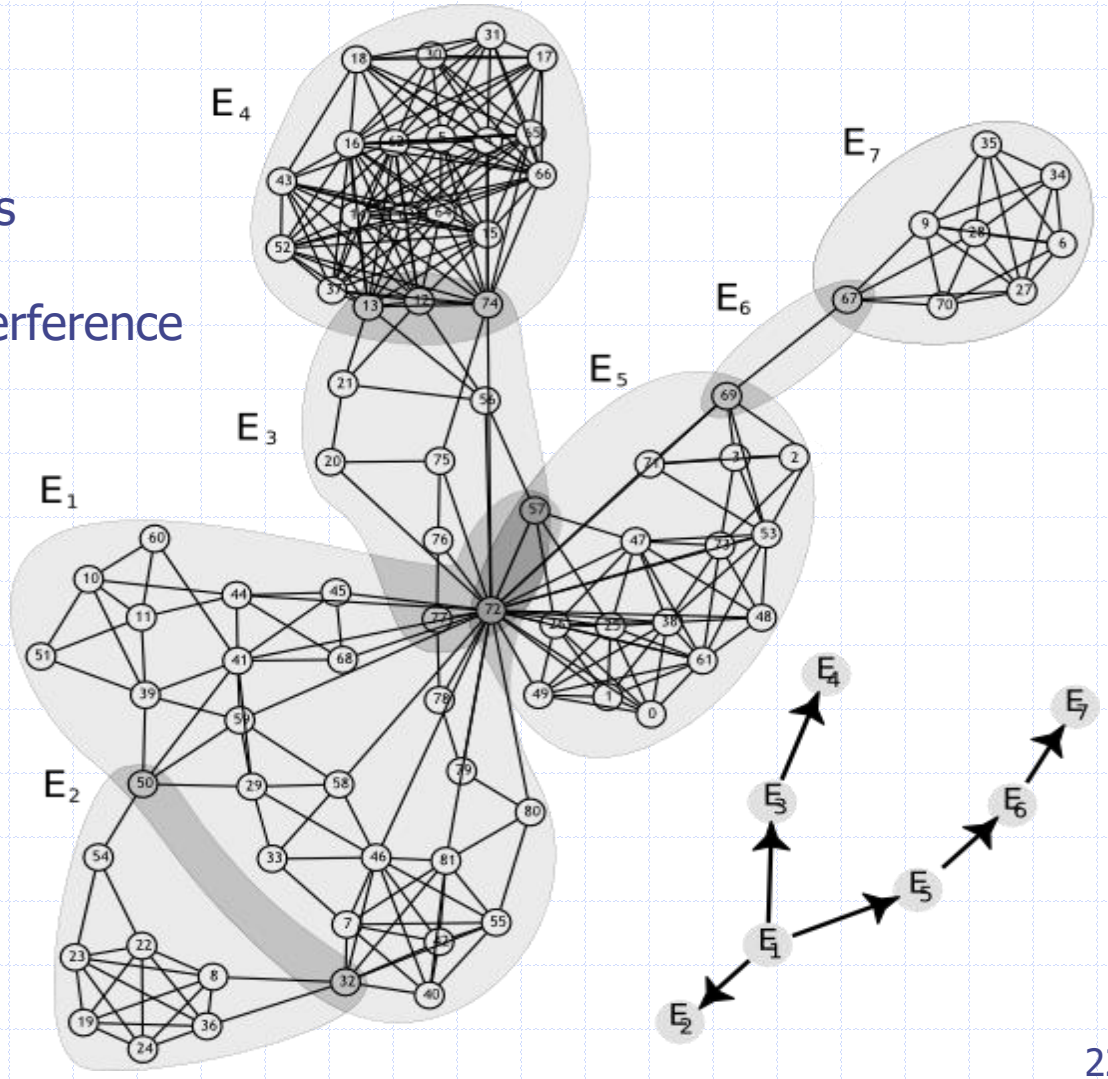
Finding a tree decomposition with minimum w* is NP-hard!

Tree decomposition example

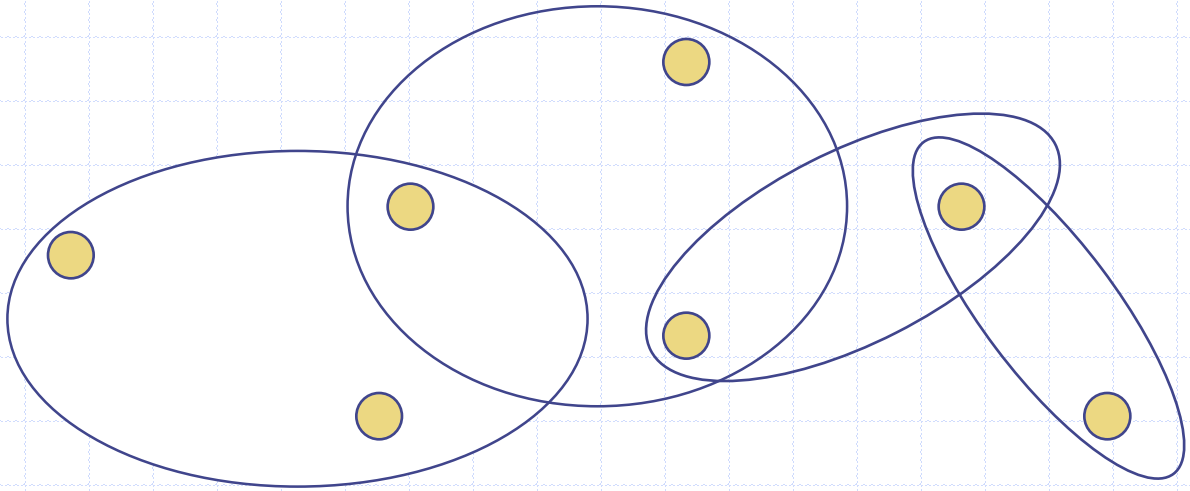
Benchmark problem
assigning frequencies
to transmitters
to minimise total interference

CELAR scen06r
n = 82
d = 44
e = 327

w = 26
s = 3

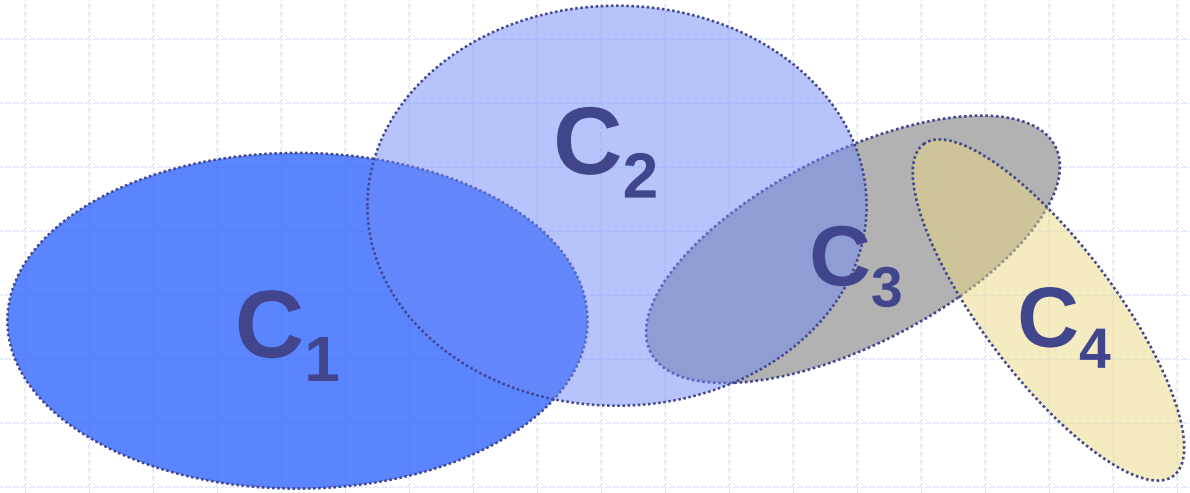


Problem features



- We have seen that structural features of a problem can lead to tractability
- This is very similar to the standard CSP
- What about other kinds of restrictions to the CFN?

More problem features



- The picture now emphasises the cost functions
- Restricting the cost functions we allow can also lead to tractability

Languages of cost functions

- A set of cost functions is called a language of cost functions
- CFN(Γ) represents the set of CFN instances whose cost functions belong to the language Γ
- For some choices of Γ, CFN(Γ) is tractable
- We consider examples where the valuation structure contains non-negative real values and infinity, and aggregation is standard addition

Submodular functions

A class of functions that has been widely studied in OR is the class of submodular functions…

A cost function c is submodular if $\forall \mathbf{s},\mathbf{t}$

$$c(\min(\mathbf{s},\mathbf{t})) + c(\max(\mathbf{s},\mathbf{t})) \leq c(\mathbf{s}) + c(\mathbf{t})$$

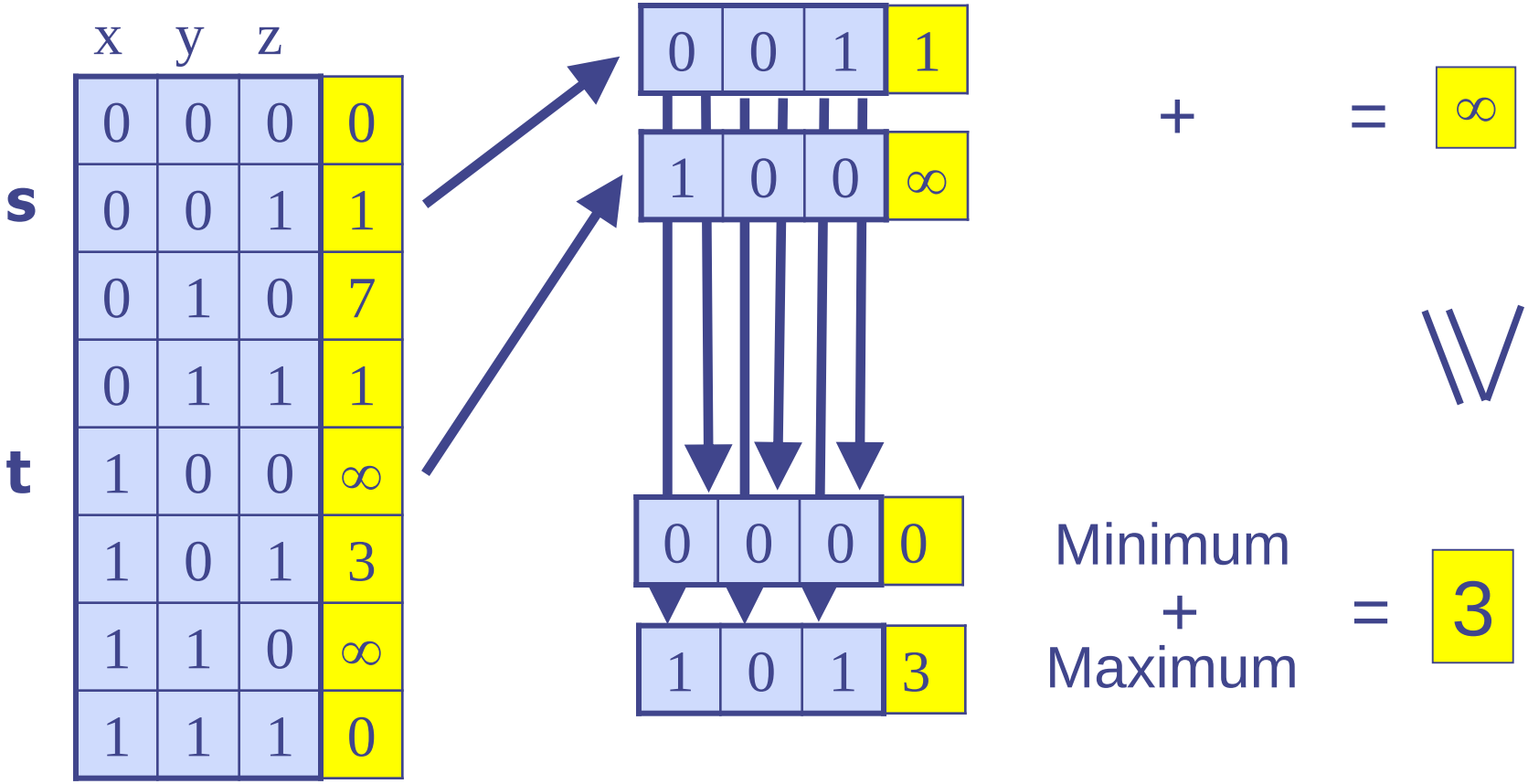
where min and max are applied component-wise, i.e.

$$\min(\langle s_1,\ldots,s_k\rangle,\langle t_1,\ldots,t_k\rangle) = \langle \min(s_1,t_1),\ldots,\min(s_k,t_k)\rangle$$

$CFN(\Gamma_{submodular})$ is tractable

Definition of submodularity

$\forall s,t \;\; c(\mathrm{Min}(s,t)) + c(\mathrm{Max}(s,t)) \leq c(s) + c(t)$



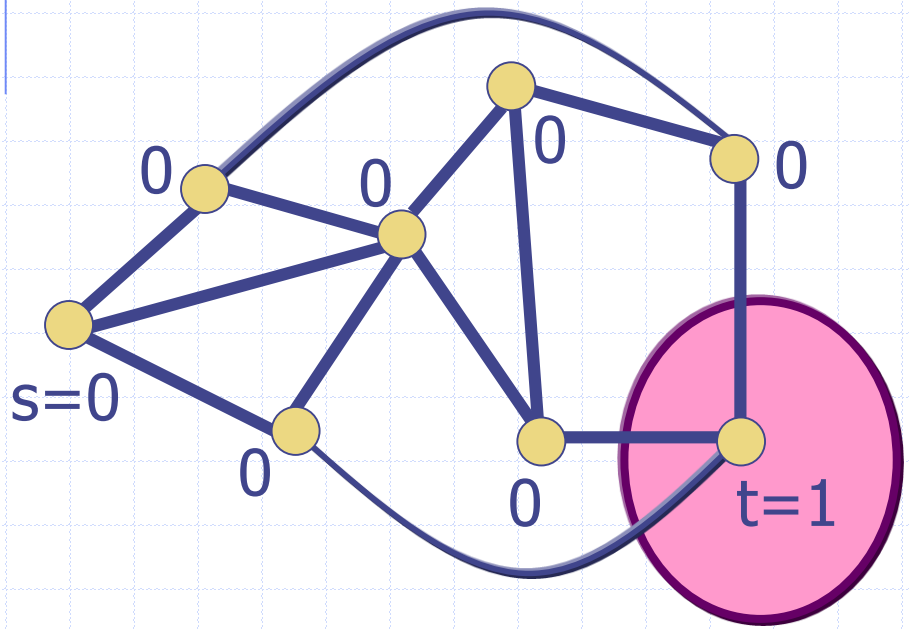
Examples of submodular functions

- all unary functions
- all linear functions (of any arity)
- the binary Boolean function ϕ_{cut}

  where $\phi_{cut}(a,b)=1$ if $(a,b)=(0,1)$ (0 otherwise)
- the rank function of a matroid
- the Euclidean distance function between two points (x_1, x_2), (x_3, x_4) in the plane
- $\phi(x,y)=(x-y)^r$ if $x \geq y$ (∞ otherwise) for $r \geq 1$

Example: Min-Cut

The s-t Min-Cut problem can be modelled using the single submodular binary cost function ϕ_{cut}



CFN with domain {0,1}

Valued constraints on all edges (both ways)

with cost function ϕ_{cut}

$\phi_{cut}(a,b)=1$ if $(a,b)=(0,1)$

Solution to CFN is a Min-Cut

Algorithms

- The best known *general* algorithm for Boolean submodular function minimisation is $O(e\, n^3 \log^{O(1)} n)$, where e=number of cost functions

  See Yin Tat Lee et al., "A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization", FOCS, 2015

- However, many special cases can be solved more efficiently by flow algorithms...

Boolean submodular functions

Many Boolean submodular functions can be expressed using the binary function ϕ_{cut}

(these include all {0,1}-valued Boolean submodular functions, all binary and all ternary Boolean submodular functions, and many others)

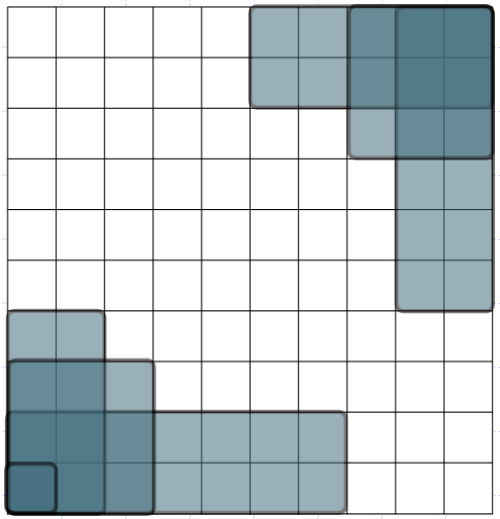
$CFN(\{\phi_{cut}\})$ is $O(n^3)$

See Živný & Jeavons, "Classes of submodular constraints expressible by graph cuts", *Constraints*, 2010

Binary submodular functions

Binary submodular functions
over any finite domain
can be expressed as a sum of
"Generalized Interval" functions

(they correspond to Monge matrices)



Binary CFN($\Gamma_{submodular}$) is $O(n^3d^3)$

See Cohen et al, "A maximal tractable class of soft constraints", *JAIR* 2004

Beyond submodularity

$\forall s,t \;\; c(\mathrm{Min}(s,t)) + c(\mathrm{Max}(s,t)) \leq c(s) + c(t)$

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	3
1	1	0	∞
1	1	1	0

We say that the cost function has the ***multimorphism*** (Min,Max)

By choosing *other* functions, we can obtain other tractable cost function languages…

Known tractable cases

If the cost functions all have one of these eight multimorphisms, then the problem is tractable:

1) (Min,Max)
2) (Max,Max)
3) (Min,Min)
4) (Majority,Majority,Majority)
5) (Minority,Minority,Minority)
6) (Majority,Majority,Minority)
7) (Constant 0)
8) (Constant 1)

See Cohen et al, "The complexity of soft constraint satisfaction", *AIJ* 2006

A dichotomy theorem

If the cost functions all have one of these eight multimorphisms, then the problem is tractable:

1) (Min,Max)
2) (Max,Max)
3) (Min,Min)
4) (Majority,Majority,Majority)
5) (Minority,Minority,Minority)
6) (Majority,Majority,Minority)
7) (Constant 0)
8) (Constant 1)

For Boolean cost functions...

In all other cases the cost functions have **no** significant common multimorphisms and the CFN problem is **NP-hard**.

See Cohen et al, "The complexity of soft constraint satisfaction", *AIJ* 2006

Benefits of a general approach

- This dichotomy theorem immediately implies earlier results for SAT, MAX-SAT, Weighted Min-Ones and Weighted Max-Ones
- Multimorphisms have also been used to show that not all submodular functions can be expressed using binary functions

  see Živný et al "The expressive power of binary submodular functions", *Discrete Applied Maths*, 2009.
- Multimorphisms (and its generalisations) allow submodularity to be generalised to larger classes of tractable languages.

Generalising multimorphisms

- Fractional/weighted polymorphism - each function has an associated rational weight >0 (in multimorphisms this weight is always 1)

  see Cohen et al, "An Algebraic Theory of Complexity for Discrete Optimization", SIAM J. Comput, 2013.

- This generalisation has led to ground-breaking results such as …

Recent theoretical results:

- Identification of all tractable languages of *finite-valued* cost functions.

  J. Thapper and S. Živný, J.ACM 2016.

- Necessary and sufficient condition for the tractability of any language (assuming the identification of all tractable crisp languages)

  V. Kolmogorov, A. Krokhin, M. Rolinek, SIAM J. Comput. 2017.

Recent theoretical results:

- Identification of all tractable languages of *finite-valued* cost functions.
  J. Thapper and S. Živný, J.ACM 2016.

- Necessary and sufficient condition for the tractability of *any* language (assuming the identification of all tractable crisp languages)
  V. Kolmogorov, A. Krokhin, M. Rolinek, SIAM J. Comput. 2017.

- Identification of all tractable crisp languages
  A. Bulatov, FOCS 2017 and D. Zhuk, FOCS 2017.

Hybrid classes

- Certain important properties of instances cannot be defined by purely structural or purely language restrictions.

- *For example:* absence of saddle points implies a unique local minimum (but this is neither a structural nor a language property).

Truly hybrid tractability

- Truly hybrid = independent restrictions on the language Γ and on the constraint graph.

- *For example:*

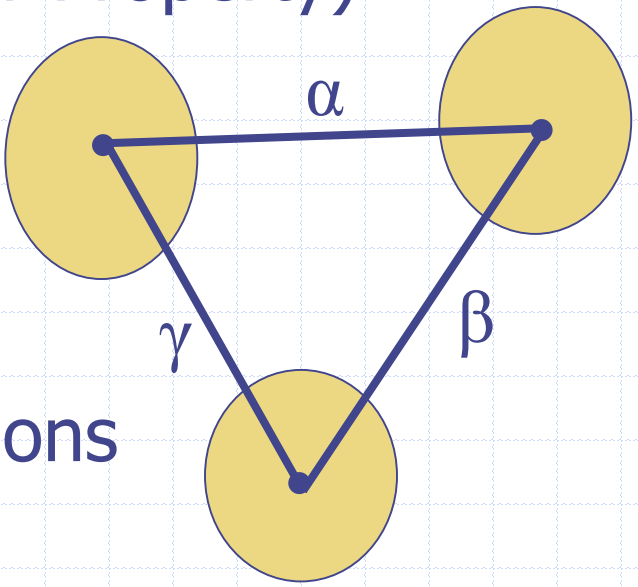
  Γ + planar $\Rightarrow$ no new tractable languages

  P. Fulla & S. Živný, MFCS 2016

Tractability from local patterns

- Example: JWP (Joint Winner Property)

$$\alpha \geq \min(\beta,\gamma)$$



+ arbitrary unary cost functions

- Generalisation:

  cross-free convex (CFC) ≈ convex cardinality cost functions on nested scopes

Bibliography

- For general background on tractable structures, see the chapter on "Tractable Structures" by R. Dechter, in the *Handbook of Constraint Programming*, Elsevier, 2006.

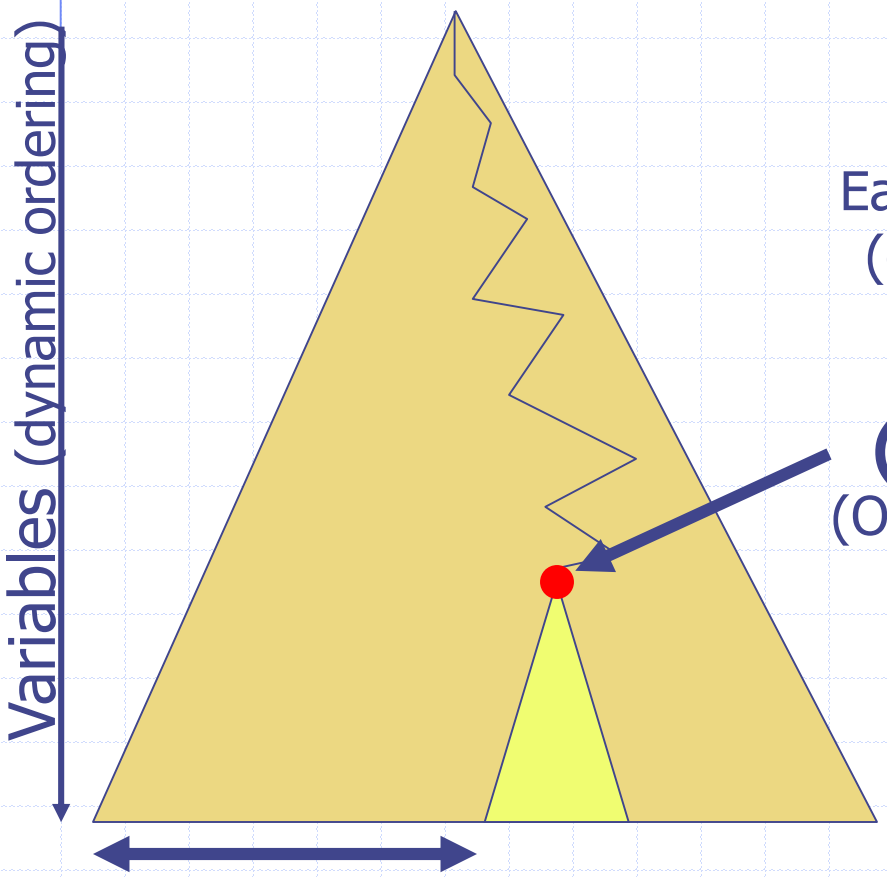
- For language and hybrid tractability see

  A. Krokhin & S. Živný, "The Constraint Satisfaction Problem: Complexity and Approximability". *Dagstuhl Follow-Ups*, 2017.

Chapter 3. Search using problem transformations

Branch and Bound,
Equivalence-preserving operations,
Soft local consistency (node, arc, existential, virtual, optimal),
Soft global constraints.

DepthFirst Branch and Bound (DFBB)



Each node is a CFN subproblem
(on the unassigned variables)

(LB) Lower Bound = c_0
(Obtained by enforcing local consistency)
= underestimate of the best solution in the sub-tree

If $c_0 \geq UB$ then prune

(UB) Upper Bound
= best solution found so far

Equivalence-preserving transformations (EPT)

- An EPT transforms CFN instance P_1 into another CFN instance P_2
- with the same objective function.
- Examples of EPTs:

  - Propagation of inconsistencies (∞ costs)
  - UnaryProject
  - Project/Extend

UnaryProject(i,α)

Precondition: $0 \leq \alpha \leq \min\{c_i(a) : a \in d_i\}$

$c_0 := c_0 + \alpha$;
for all $a \in d_i$ **do**
    $c_i(a) := c_i(a) - \alpha$;

Increases the lower bound c_0 if all unary costs $c_i(a)$ are non-zero.

Project(M,i,a,α)

Precondition: $i \in M$, $a \in d_i$, $-c_i(a) \leq \alpha \leq \min\{c_M(x): x[i]=a\}$

$c_i(a) := c_i(a) + \alpha$;

for all $x \in$ labelings(M) s.t. $x[i]=a$ **do**

$c_M(x) := c_M(x) - \alpha$;

If $\alpha>0$, this projects costs from c_M to c_i

If $\alpha<0$, this extends costs from c_i to c_M

Node and soft arc consistency

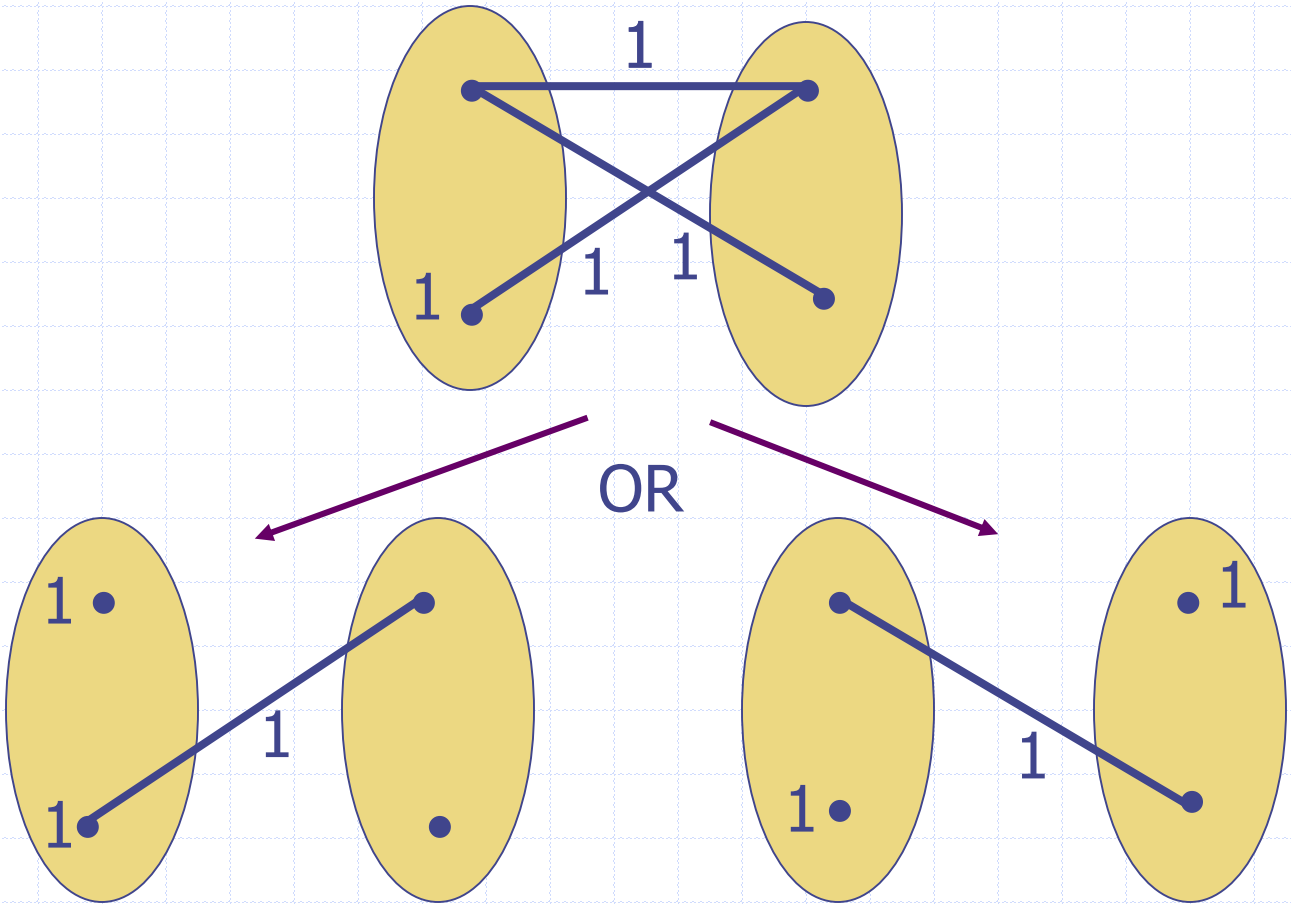
- Node consistent (NC) if $\forall i$

  no UnaryProject(i,α) is possible for $\alpha>0$ and no propagation of ∞ costs possible between c_i and c_0 (value $a \in d_i$ removed if $c_i(a)+c_0 \geq UB$)

- Soft arc consistent (SAC) if $\forall M,i,a$

  no Project(M,i,a,α) is possible for $\alpha>0$

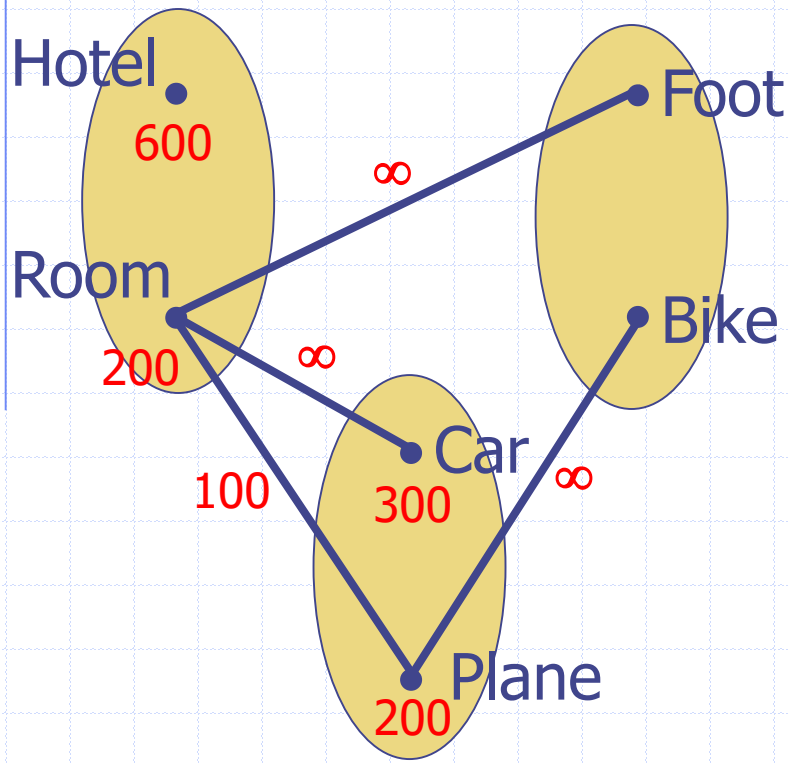
Soft AC closure is not unique



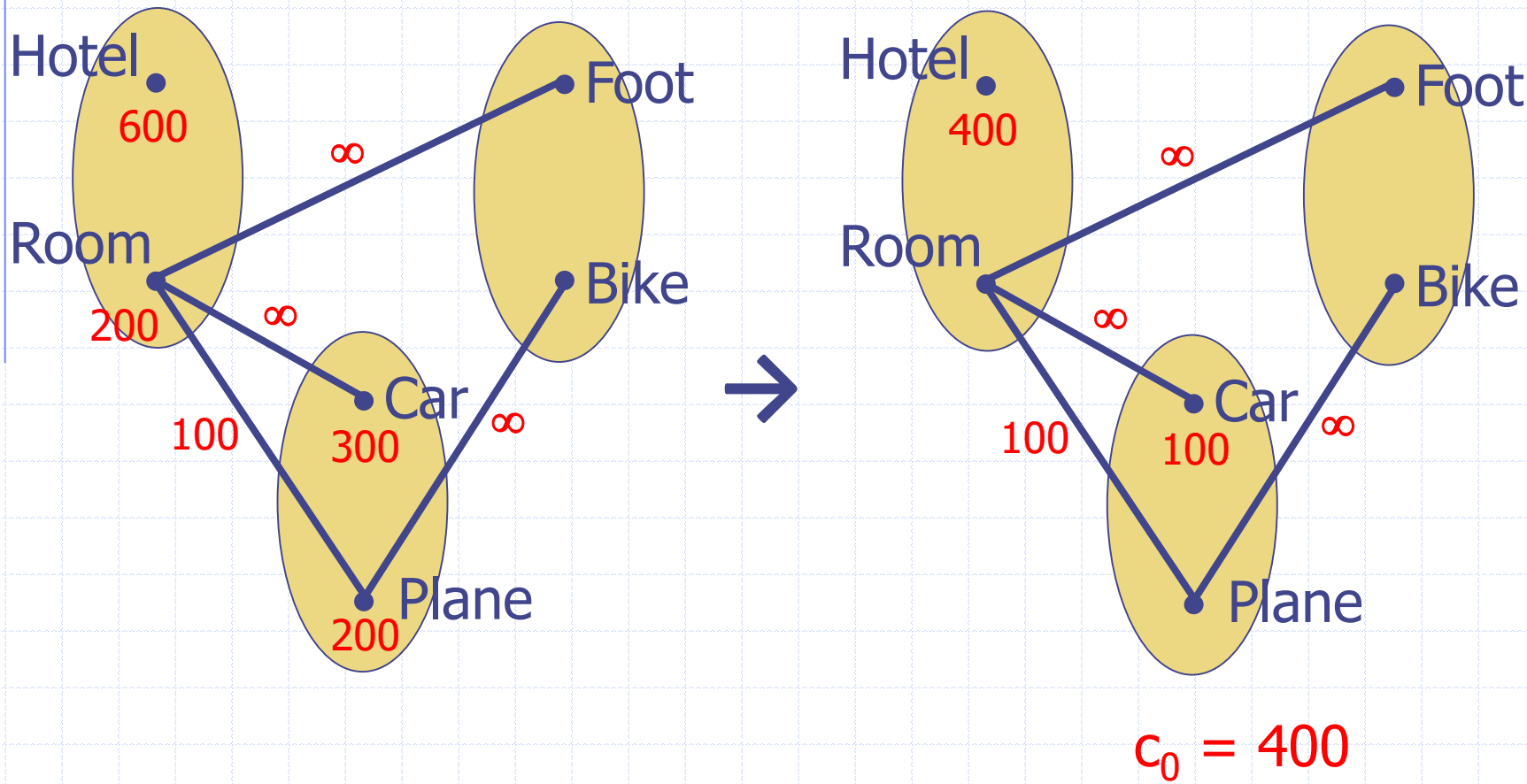
Different soft AC notions:

- Directional: send costs from X_j to X_i if $i<j$ (in the hope that this will increase c_0)
- Existential: $\forall i$, send costs to X_i simultaneously from its neighbor variables if this increases c_0
- Virtual: no *sequence* of Projects/Extends increases c_0
- Optimal: no *simultaneous set* of Projects/Extends increases c_0

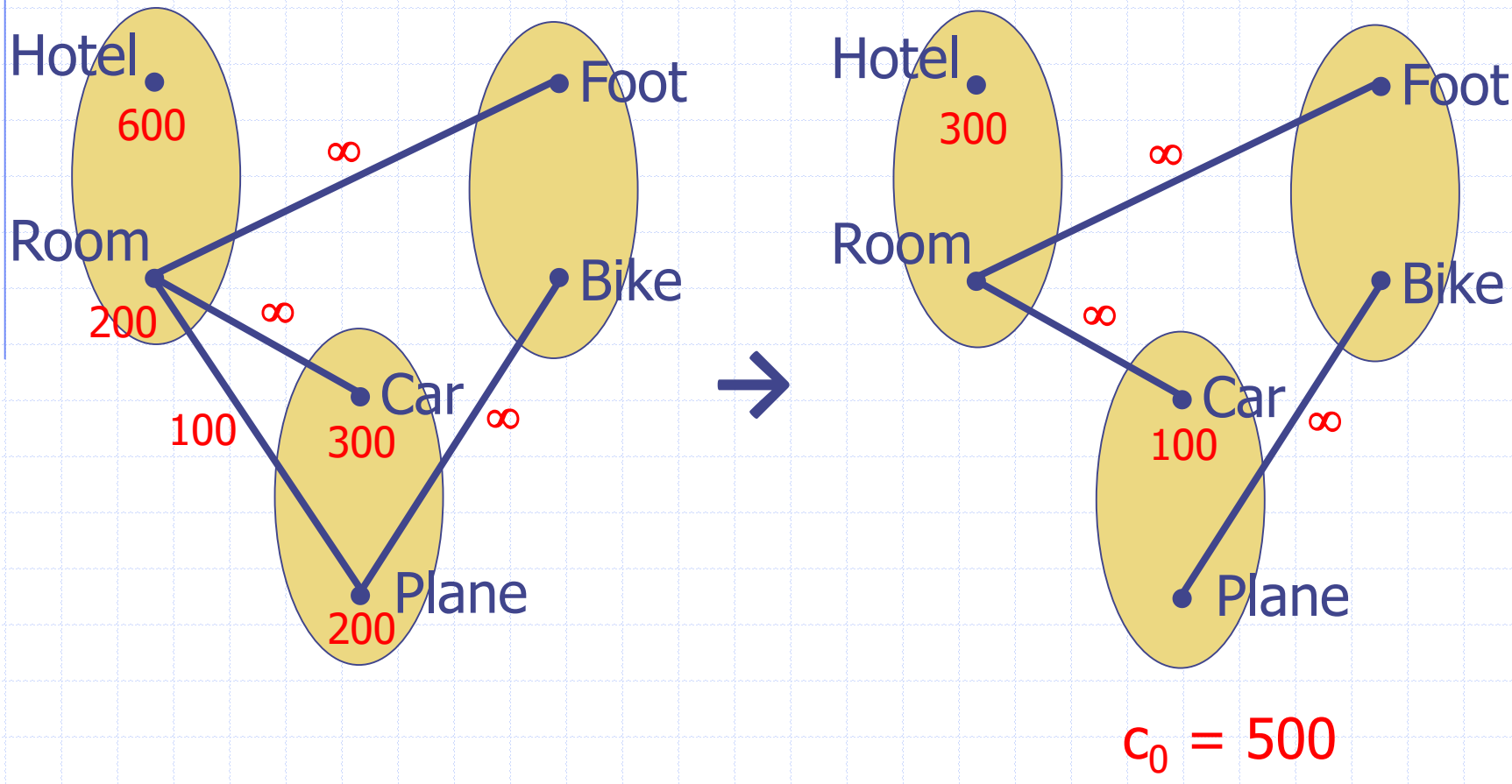
Example instance



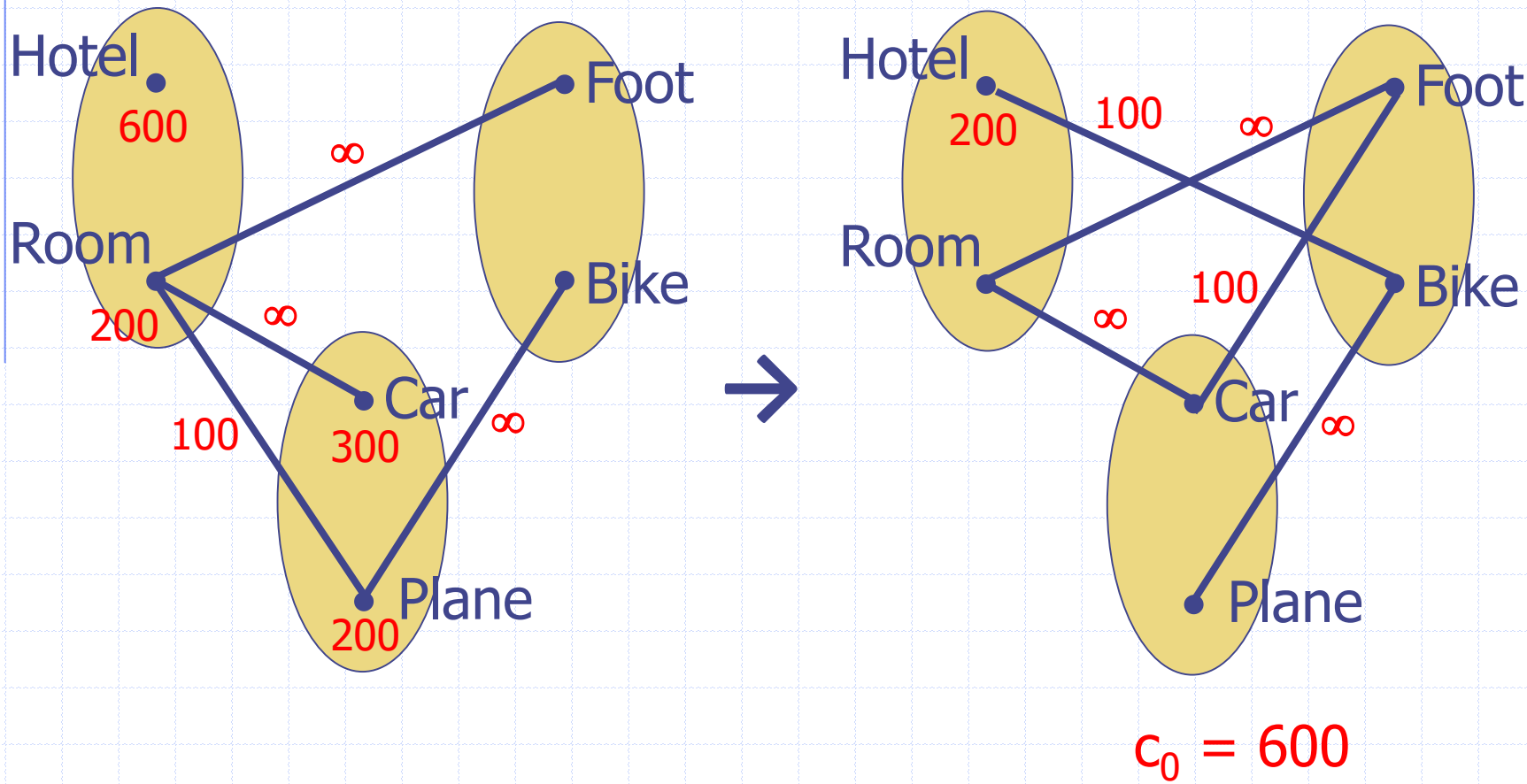
Example instance after NC



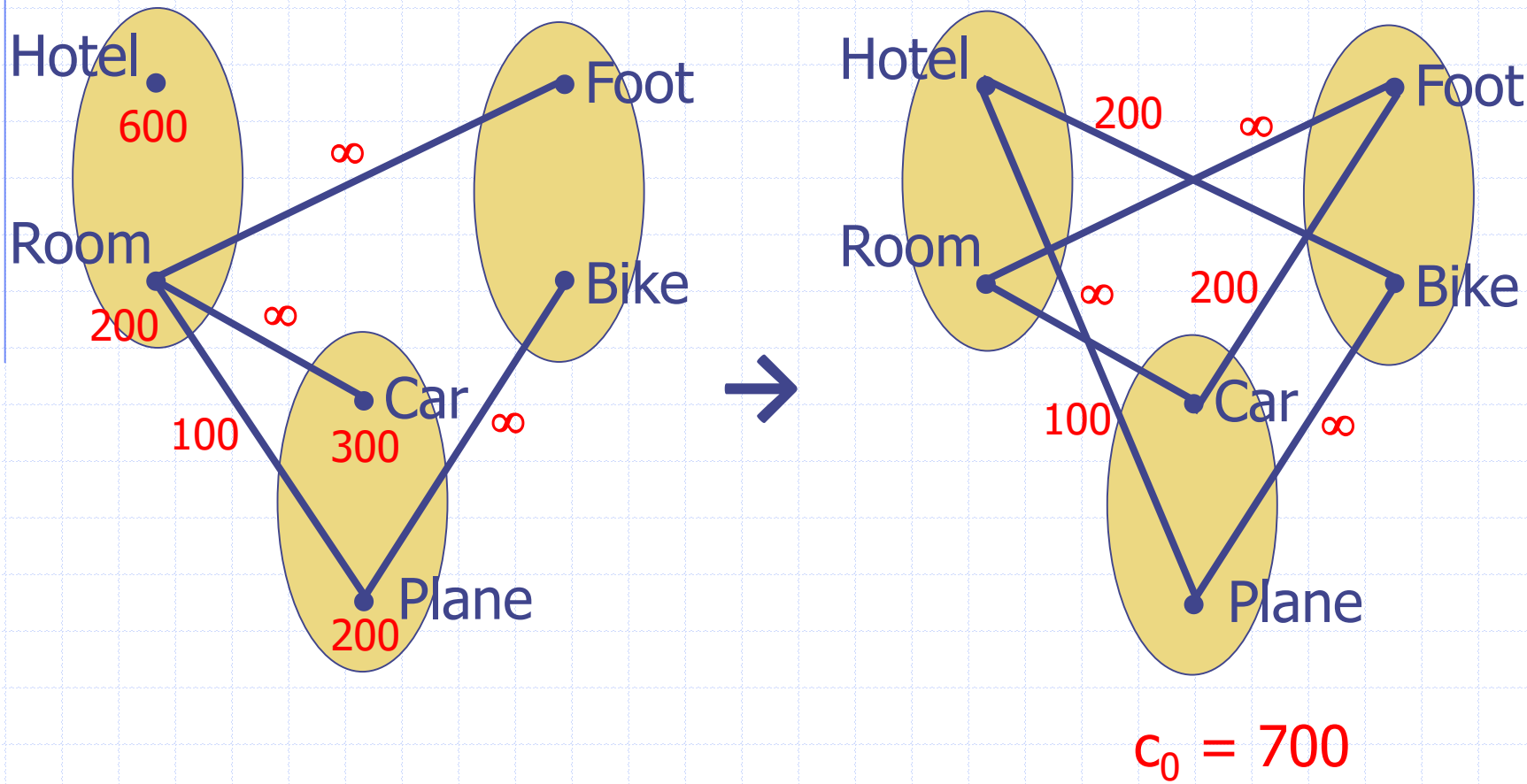
Example instance after SAC



Example instance after EDAC



Example instance after VAC



Directional Arc Consistency

- for all $i<j$, $\forall a \in d_i \; \exists b \in d_j$ such that $c_{ij}(a,b) = c_j(b) = 0$.
- Solves tree-structured CFNs
- DAC can be established in $O(ed^2)$ time

Existential Arc Consistency

- node consistent and $\forall i, \exists a \in d_i$ such that $c_i(a) = 0$ and for all cost functions c_{ij} , $\exists b \in d_j$ such that $c_{ij}(a, b) = c_j(b) = 0$
- EDAC = Existential AC + Directional AC + Soft AC
- EDAC can be established in $O(ed^2 \max\{nd, UB\})$ time

Virtual Arc Consistency (VAC)

- If P is a CFN instance then Bool(P) is the CSP instance whose allowed tuples are the zero-cost tuples in P-c_0
- If Bool(P) has a solution, then P has a solution of cost c_0 (but usually Bool(P) has no solution)
- Definition: P is VAC if Bool(P) is AC.

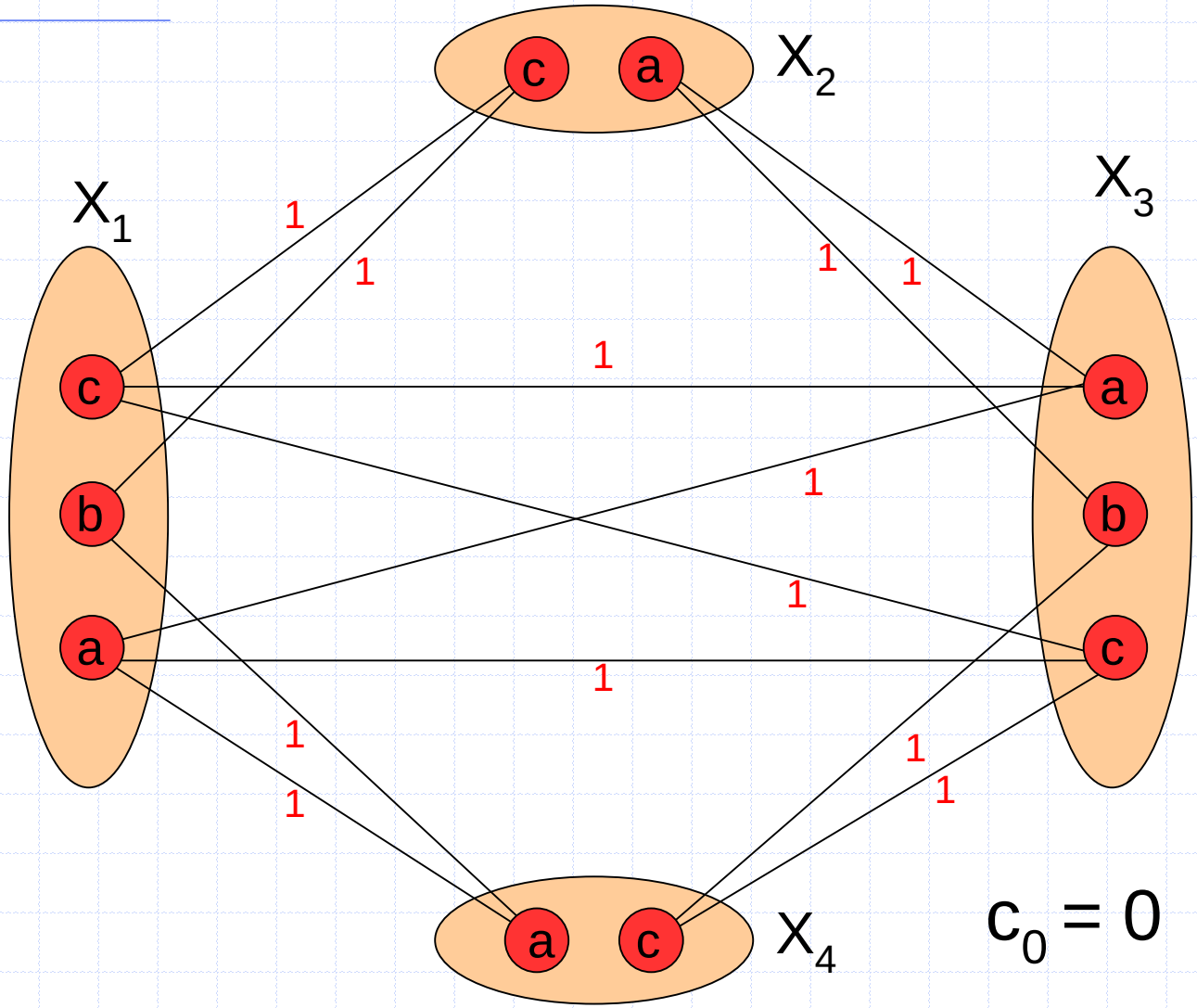
Approximating VAC

- If a sequence of AC operations in Bool(P) leads to a domain wipe-out, then a similar sequence of SAC operations in P increases c_0
- But, in this sequence, costs may need to be sent in more than one direction from the same $c_M \Rightarrow$ Introduction of *fractional* weights
- VAC algorithm may converge to a local minimum (an instance P' which is *not* VAC)

Optimal Soft Arc Consistency

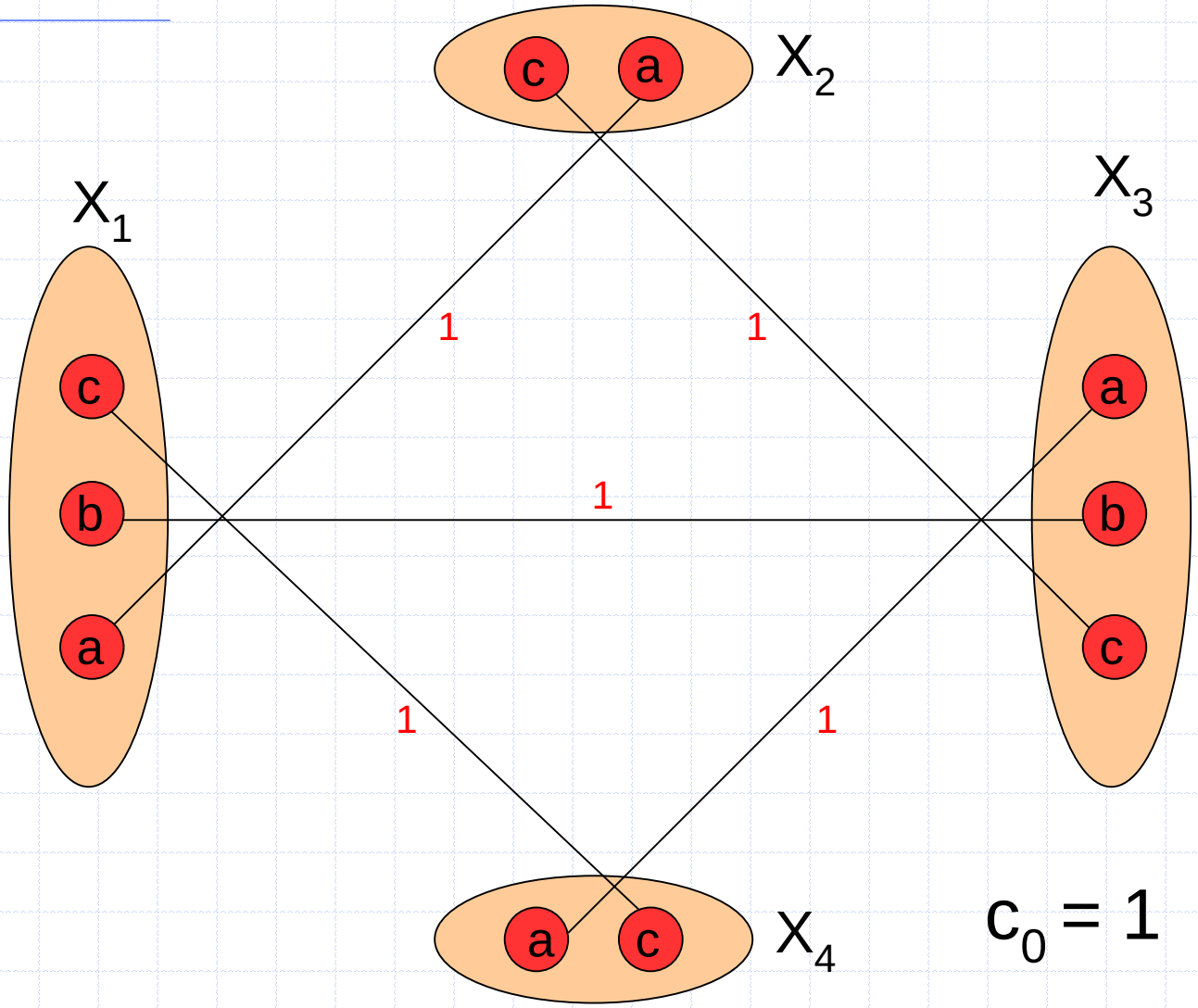
- We can overcome this problem of convergence by solving a LP to find the *set* of simultaneous UnaryProject and Project operations which maximises c_0.
- The resulting CFN instance is OSAC (Optimal Soft Arc Consistent).
- OSAC is strictly stronger than VAC.
- Unfortunately, the LP has O(edr+n) variables and O(ed^r+nd) constraints, and hence only useful for pre-processing.

Example of OSAC



$$p_{2c}^{23} = p_{3a}^{34} = p_{3b}^{31} = p_{1a}^{12} = p_{1c}^{14} = -1$$

$$p_{3a}^{23} = p_{3b}^{23} = p_{4c}^{34} = p_{1a}^{31} = p_{1c}^{31} = p_{2c}^{12} = p_{4a}^{14} = u_4 = 1$$



$c_0 = 1$

OSAC: Theoretical results

The value c_0 returned by OSAC is equal to the minimum expected cost of a locally (but not necessarily globally) coherent probabilistic assignment (= dual of LP)

E.g. 2-colouring on 3 variables: the probabilistic assignment

$$Pr_i(0) = Pr_i(1) = 0.5, \quad Pr_{ik}(01) = Pr_{ik}(10) = 0.5$$

is *locally* but not *globally* coherent and has expected cost 0.

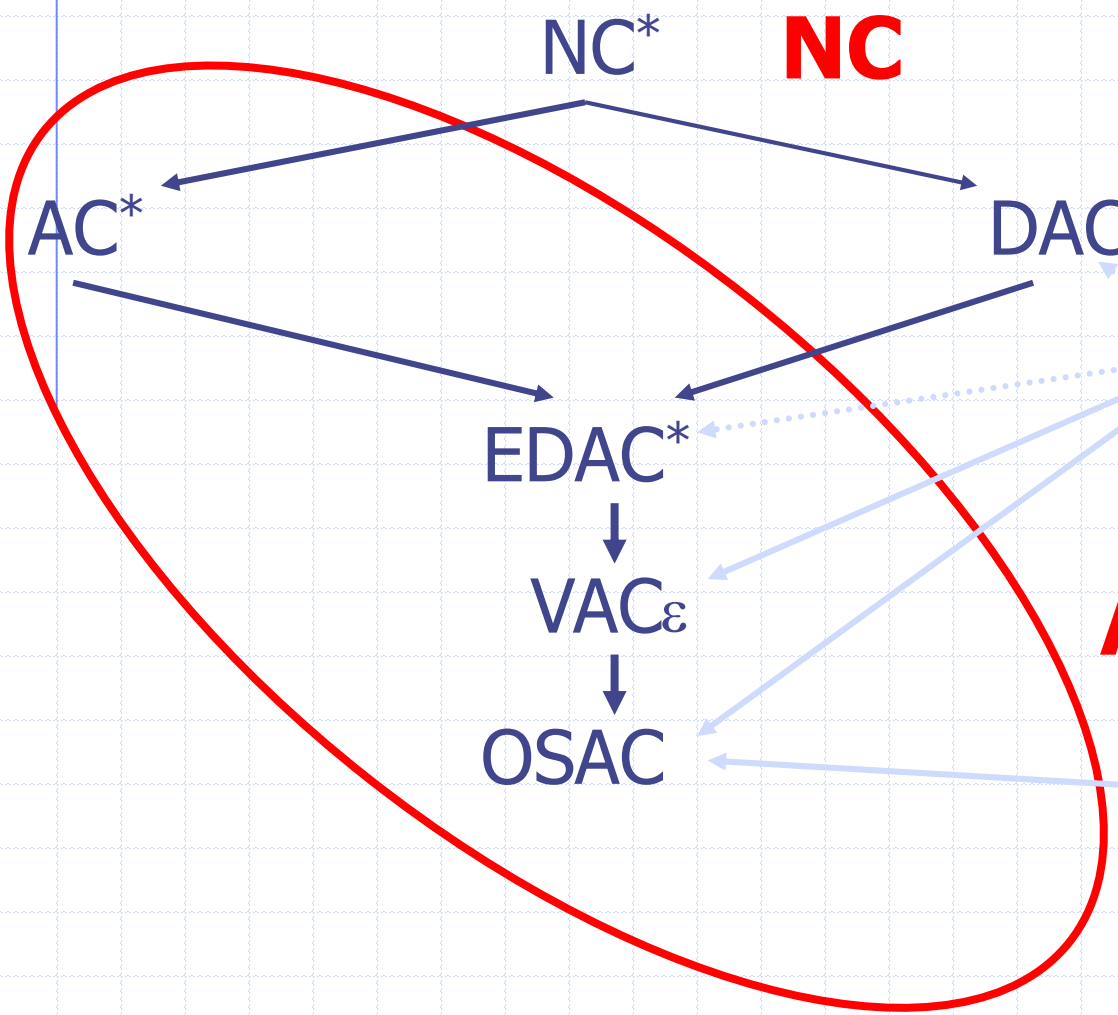
See M. Schlesinger, "Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh" (Syntactic analysis of two-dimensional visual signals in noisy conditions), *Kibernetika* (1976).

OSAC: Theoretical results

OSAC solves all tractable languages of finite-valued cost-functions (including submodular functions)

Hierarchy

Special case: CSP



NC

DAC

Solve tree-like primal graphs

AC

Solves all finite -valued tractable languages

Some practical observations

- For very hard-to-solve instances, maintaining VAC provides a significant speed-up, however for many problems, maintaining a simpler form of soft arc consistency (EDAC) is faster.
- Unary costs $c_i(a)$ useful for value and variable ordering heuristics.
- Consistencies can be generalised to abitrary arities but computational cost is high.

AC for soft global constraints

(See van Hoeve et al, *J. Heur.* 2006, Lee & Leung, *IJCAI* 2009, Allouche et al. AIJ 2016)

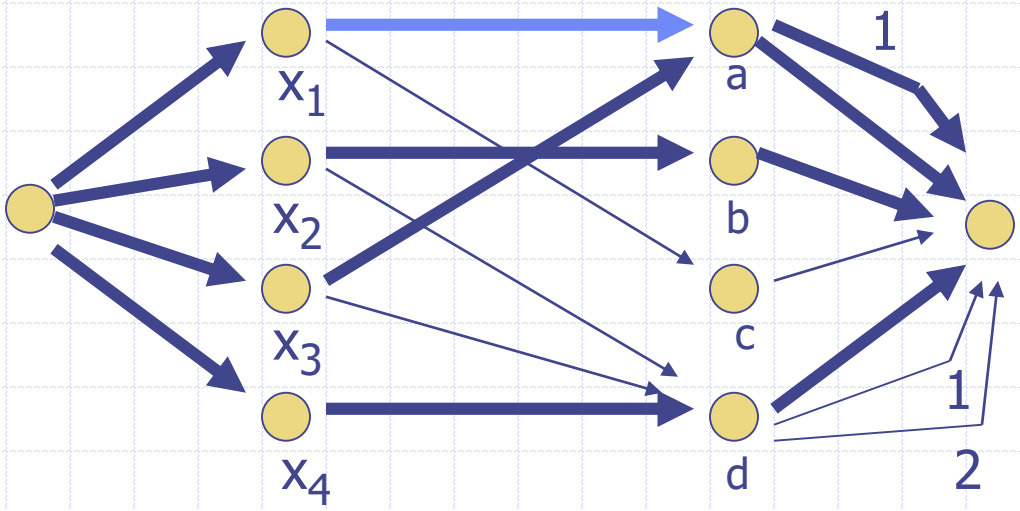
- Recent research has shown that many global cost functions allow projection/extension to/from unary cost functions in poly-time.
- Examples: cost-function variants of AMONG, ALLDIFFERENT, GCC, GRAMMAR, REGULAR, SAME, SUM...

Example: AC for flow-based soft global constraints

- Suppose that a global cost function c_M can be coded as the minimum cost of a maximum flow in a network in which (a) there is a one-to-one correspondence between max-flows and global labelings and (b) each assignment (x_i,a) has a corresponding edge e_{ia} such that the max-flow is 1 in e_{ia} if $x_i=a$ (0 if $x_i \neq a$).
- Then it is possible to project α from c_M to $c_i(a)$ by reducing cost(e_{ia}) by α.

Network representing soft Alldiff

- Min number of variables with same value
  variable-based costs (Beldiceanu & Petit, CPAIOR'04)



All edge capacities are equal to 1

All edge costs are 0 if not indicated

The flow shown is a min-cost max-flow with x_1=a. We can project 1 from c_M to c_1(a) by reducing the cost of the light blue edge from 0 to –1.

Decomposition of global cost functions

- Rewrite the global cost function as a sum of smaller bounded scope cost functions (a sub-network) *(Schiex et al, AAAI 2012)*
- Polynomial transformation solved by DAC for AMONG, REGULAR, *SUM*…
- DAC and VAC solves Berge-acyclic decomposed networks

softRegular

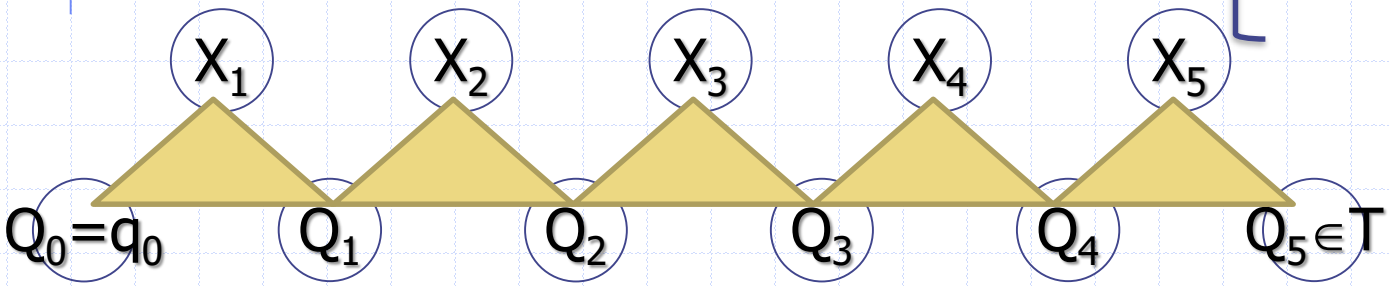
(*Hamming distance*)

softRegular(X_1, X_2, X_3, X_4, X_5, (Q, Σ, δ, q_0, T))

$$g(Q_{i-1}, X_i, Q_i) = \begin{cases} 0 & \text{if } (q_{i-1}, x_i, q_i) \in \delta \\ 1 & \text{if } (q_{i-1}, v, q_i) \in \delta, v \neq x_i \\ +\infty & \text{otherwise} \end{cases}$$

$D_{Xi} \in Q$

$D_{Qi} \in \Sigma$



➔ *Berge-acyclic decomposition*

1-softRegular

n	$\|\Sigma\|$	$\|Q\|$	Monolithic		Decomposed	
			filter	solve	filter	solve
25	5	10	0.12	0.51	0.00	0.00
		80	2.03	9.10	0.08	0.08
25	10	10	0.64	2.56	0.01	0.01
		80	10.64	43.52	0.54	0.56
25	20	10	3.60	13.06	0.03	0.03
		80	45.94	177.5	1.51	1.55
50	5	10	0.45	3.54	0.00	0.00
		80	11.85	101.2	0.17	0.17
50	10	10	3.22	20.97	0.02	0.02
		80	51.07	380.5	1.27	1.31
50	20	10	15.91	100.7	0.06	0.07
		80	186.2	1,339	3.38	3.47

toulbar2 version 0.9.5

Clique cuts

- Given a set S

$$x_i + x_j \leq 1 \quad \forall x_i, x_j \in S$$

$\Rightarrow$ Satisfied by $x_i = 0.5$

But we can get

$$\sum_{x_i \in S} x_i \leq 1$$

Clique cuts in CFN

(CP17)

Straightforward generalization
Given a set S of $\langle X_i, v_i \rangle$ with

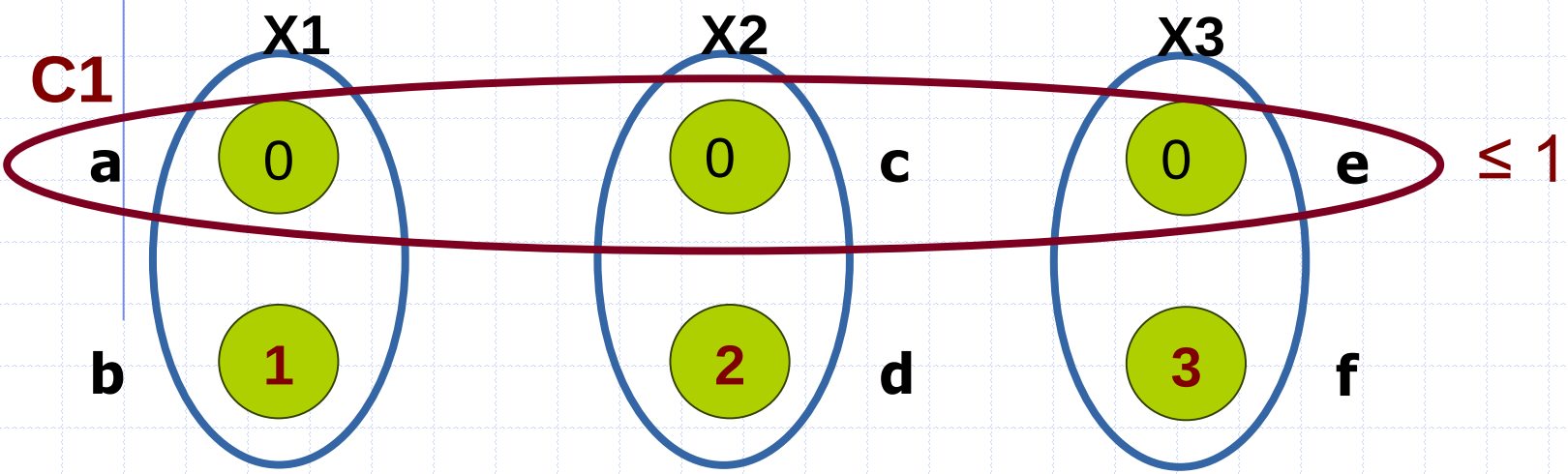
- $c_{ij}(v_i, v_j) = \infty$

Then derive

$$\sum_{ij \in S} x_{ij} \leq 1$$

Reparameterization for clique

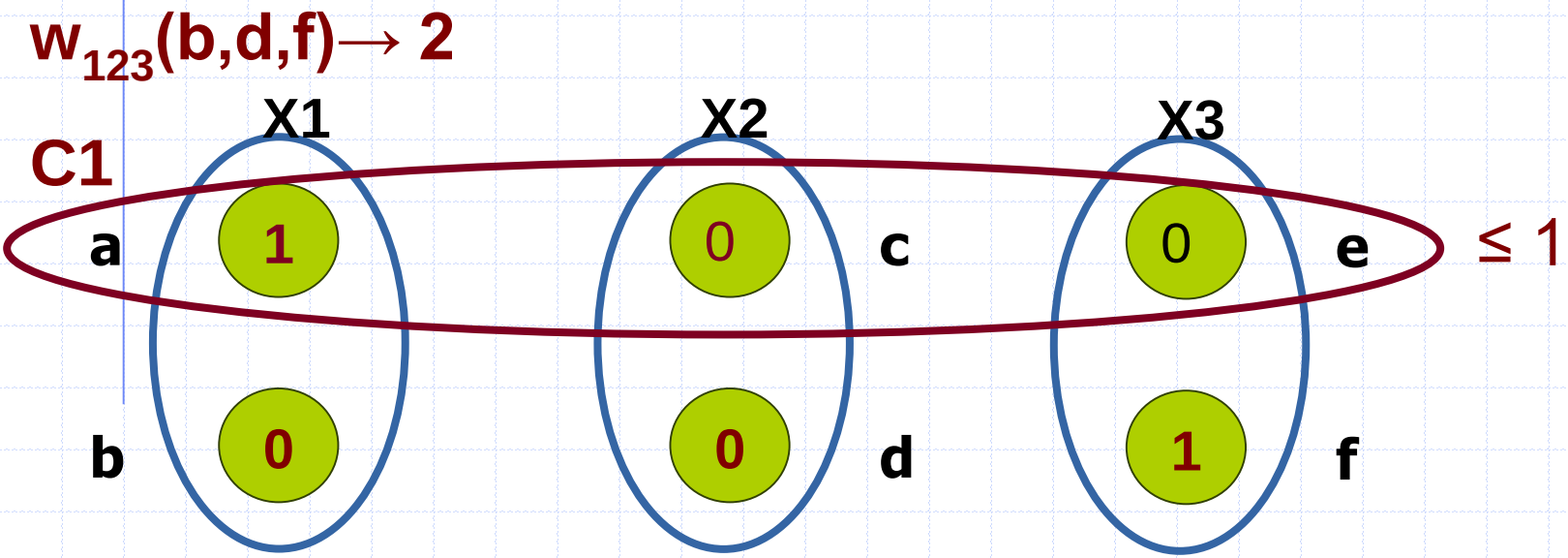
(CP17)



$w_{\varnothing}=0$

Reparameterization for clique

(CP17)



$w_{\emptyset}=3$

Experimental Results

(CP17)

problem	TOULBAR2		TOULBAR2^{clq}		CPLEX	
	solv.	time	solv.	time*	solv.	time
Auction/path	**86**	59	**86**	0.18	**86**	0.01
Auction/sched	**84**	110	**84**	0.23	**84**	0.04
MaxClique	31	1871	37	1508	**38**	**1533**
SPOT5	4	2884	6	2603	**16**	738

* Including bounded clique detection with Bron-Kerbosch algorithm in preprocessing

Bibliography

- For an overview of soft local consistencies, see "*Soft arc consistency revisited*", Cooper, de Givry, Sanchez, Schiex, Zytnicki & Werner, *AIJ* 2010.

- For soft global constraints (flow, dp, dec), see "*Tractability-preserving transformations of global cost functions*", Allouche *et al*, *AIJ* 2016.

- "*Clique Cuts in Weighted Constraint Satisfaction*", Katsirelos et al, *CP2017*.