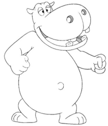
Improving the computation of the VAC lower bound in graphical models

Hiep Nguyen[1] Christian Bessiere[2] Thomas Schiex[1]

[1]INRA-BIA
UR875, Toulouse, France

[2]Université de Montpellier
Montpellier, France

INRA-BIA'2013

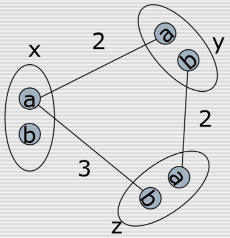


Overview

- Weighted Constraint Satisfaction Problems
- Branch and Bound search
- Virtual Arc Consistency
- Dynamic Virtual Arc Consistency
- Experiments
- Conclusion

Weighted Constraint Satisfaction Problem - WCSP

- set of variables — x, y, z
- set of values for each variable — a, b
- set of positive cost functions — $f_x, f_y, f_z, f_{xy}, f_{xz}, f_{yz}$ (or $w_x, w_y \ldots$)
  - $f_\varnothing$: 0-arity function, defines a LB on the cost of any solution



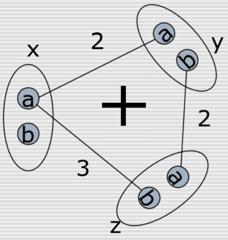
f_{yz}

tuple ⇒

y	z	
a	a	0
a	b	0
b	a	2
b	b	0

Weighted Constraint Satisfaction Problem - WCSP

- set of variables — x, y, z
- set of values for each variable — a, b
- set of positive cost functions — $f_x, f_y, f_z, f_{xy}, f_{xz}, f_{yz}$ (or $w_x, w_y \ldots$)
  - $f_\varnothing$: 0-arity function, defines a LB on the cost of any solution



Global cost distribution

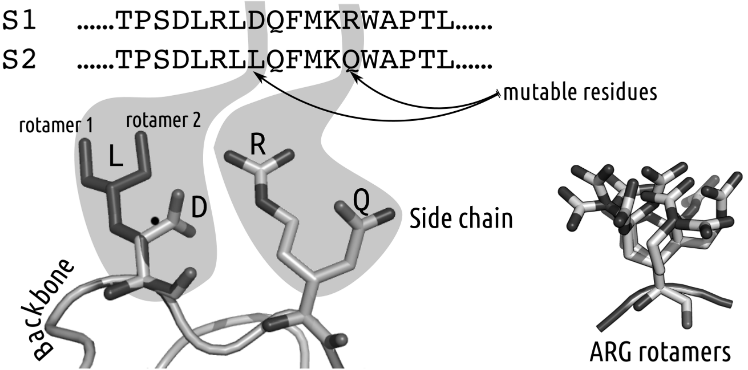
$F(X) = \oplus_{i=1}^{m} f_i(X)$

$f_y \oplus f_z \oplus f_{xy} \oplus f_{xz} \oplus f_{yz}$

x	y	z	
a	a	a	2
a	a	b	5
a	b	a	2
a	b	b	3
b	a	a	0
b	a	b	0
b	b	a	2
b	b	b	0

Solution: intantiation with minimum global cost $\Rightarrow$ **optimization problem**

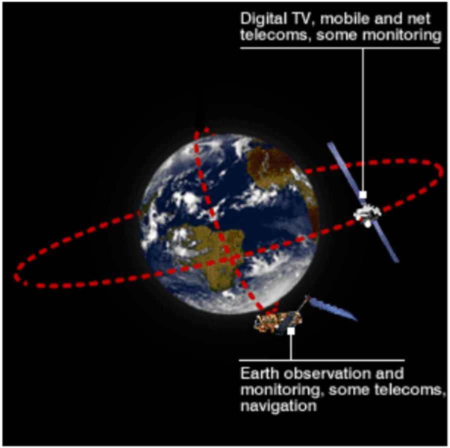
Combinatorial Optimization



Global Minimum Energy Conformation

Minimize $E = E_c + \sum_i E(i_r) + \sum_i \sum_j E(i_r, j_s)$

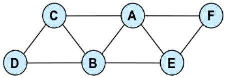
Combinatorial Optimization



Find an schedule for the satellite that maximizes the number of photographs taken, subject to the on-board recording capacity

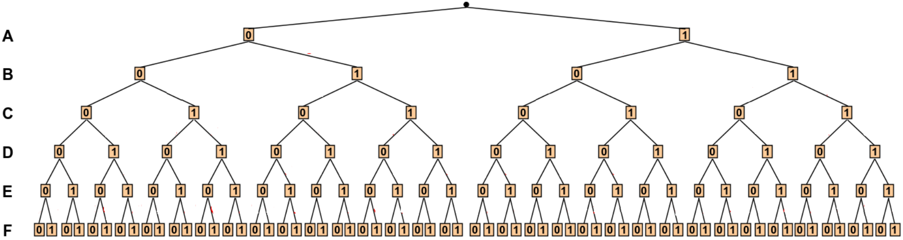
Assign frequencies to a set of radio links such that the
maximum frequency is minimized
sum of interferences is minimized

Resolution by tree search

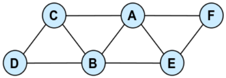


A	B	f1	A	C	f2	A	E	f3	A	F	f4	B	C	f5	B	D	f6	B	E	f7	C	D	f8	E	F	f9
0	0	**2**	0	0	**3**	0	0	**0**	0	0	**2**	0	0	**0**	0	0	**4**	0	0	**3**	0	0	**1**	0	0	**1**
0	1	**0**	0	1	**0**	0	1	**3**	0	1	**0**	0	1	**1**	0	1	**2**	0	1	**2**	0	1	**4**	0	1	**0**
1	0	**1**	1	0	**0**	1	0	**2**	1	0	**0**	1	0	**2**	1	0	**1**	1	0	**1**	1	0	**0**	1	0	**0**
1	1	**4**	1	1	**1**	1	1	**0**	1	1	**2**	1	1	**4**	1	1	**0**	1	1	**0**	1	1	**0**	1	1	**2**

Objective function: $f(X) = \min_X \sum_{i=1}^{9} f_i(X)$

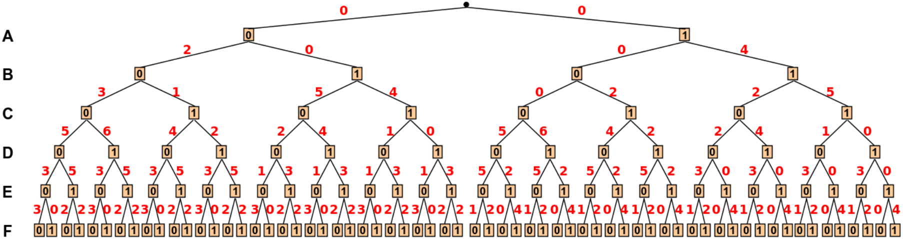


Resolution by tree search

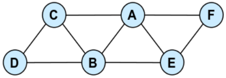


A	B	f1	A	C	f2	A	E	f3	A	F	f4	B	C	f5	B	D	f6	B	E	f7	C	D	f8	E	F	f9
0	0	**2**	0	0	**3**	0	0	**0**	0	0	**2**	0	0	**0**	0	0	**4**	0	0	**3**	0	0	**1**	0	0	**1**
0	1	**0**	0	1	**0**	0	1	**3**	0	1	**0**	0	1	**1**	0	1	**2**	0	1	**2**	0	1	**4**	0	1	**0**
1	0	**1**	1	0	**0**	1	0	**2**	1	0	**0**	1	0	**2**	1	0	**1**	1	0	**1**	1	0	**0**	1	0	**0**
1	1	**4**	1	1	**1**	1	1	**0**	1	1	**2**	1	1	**4**	1	1	**0**	1	1	**0**	1	1	**0**	1	1	**2**

$$f(X) = \min_X \sum_{i=1}^{9} f_i(X)$$

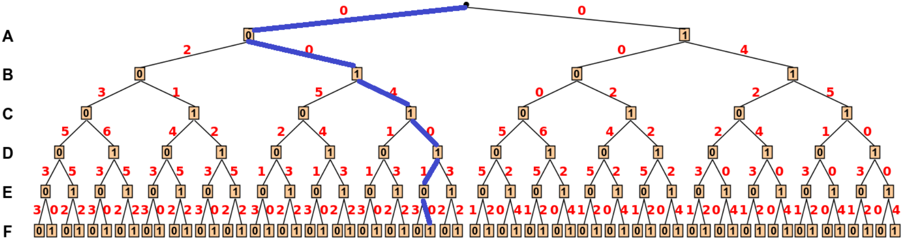


Resolution by tree search



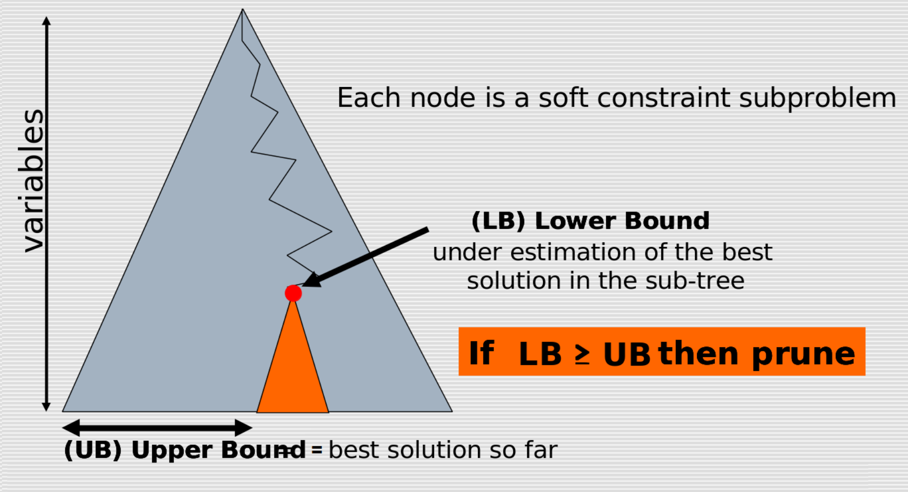
A	B	f1	A	C	f2	A	E	f3	A	F	f4	B	C	f5	B	D	f6	B	E	f7	C	D	f8	E	F	f9
0	0	**2**	0	0	**3**	0	0	**0**	0	0	**2**	0	0	**0**	0	0	**4**	0	0	**3**	0	0	**1**	0	0	**1**
0	1	**0**	0	1	**0**	0	1	**3**	0	1	**0**	0	1	**1**	0	1	**2**	0	1	**2**	0	1	**4**	0	1	**0**
1	0	**1**	1	0	**0**	1	0	**2**	1	0	**0**	1	0	**2**	1	0	**1**	1	0	**1**	1	0	**0**	1	0	**0**
1	1	**4**	1	1	**1**	1	1	**0**	1	1	**2**	1	1	**4**	1	1	**0**	1	1	**0**	1	1	**0**	1	1	**2**

$$f(X) = \min_X \sum_{i=1}^{9} f_i(X)$$



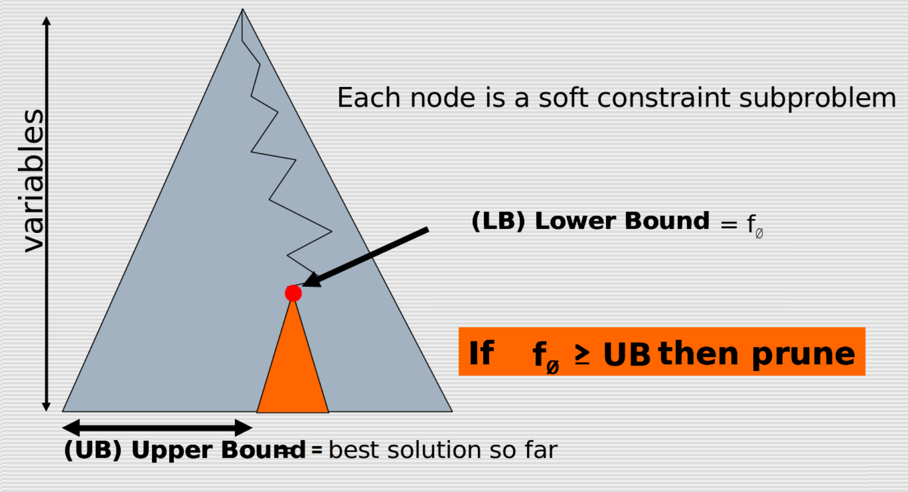
Resolution by Branch and Bound search

Use a lower bound (lb) on the cost of the best extension of partially assigned subproblem to prune the depth-first search tree



Resolution by Branch and Bound search

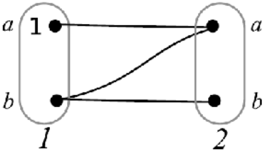
Use a lower bound (lb) on the cost of the best extension of partially assigned subproblem to prune the depth-first search tree



Overview

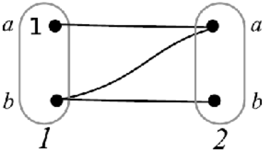
- Weighted Constraint Satisfaction Problems
- Branch and Bound search
- **Virtual Arc Consistency**
- Dynamic Virtual Arc Consistency
- Experiments
- onclusion

Equivalence Preserving Transformations (EPTs)

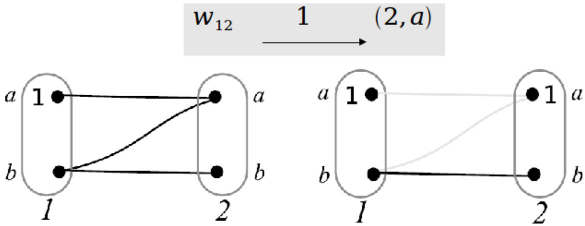


Equivalence Preserving Transformations (EPTs)

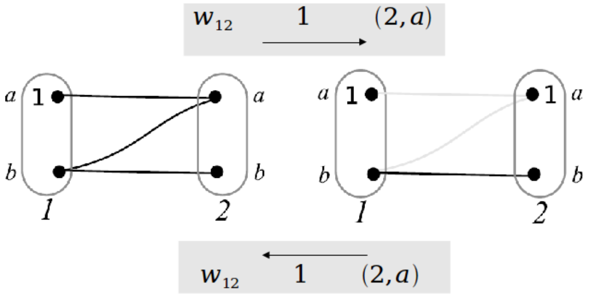
$$w_{12} \xrightarrow{\quad 1 \quad} (2, a)$$



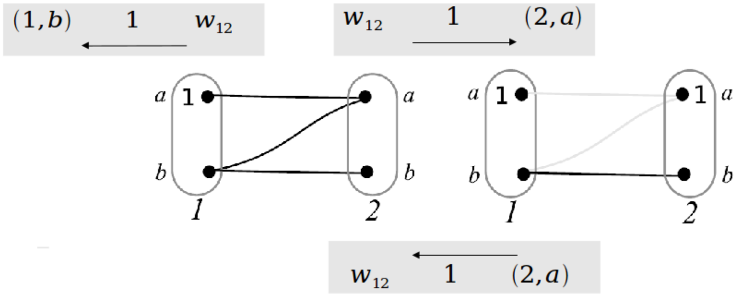
Equivalence Preserving Transformations (EPTs)



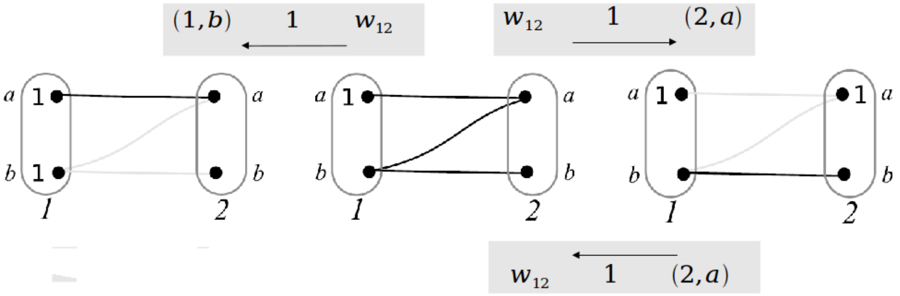
Equivalence Preserving Transformations (EPTs)



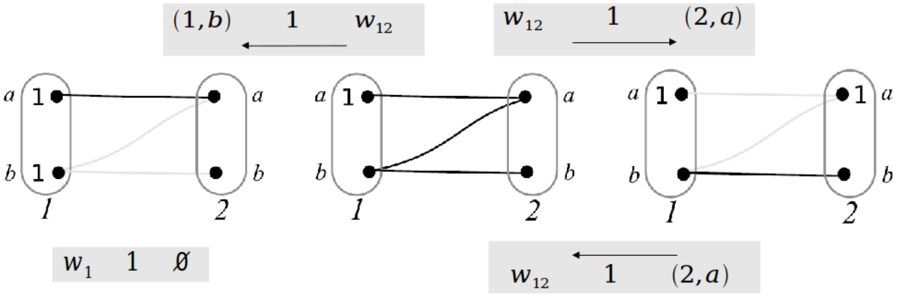
Equivalence Preserving Transformations (EPTs)



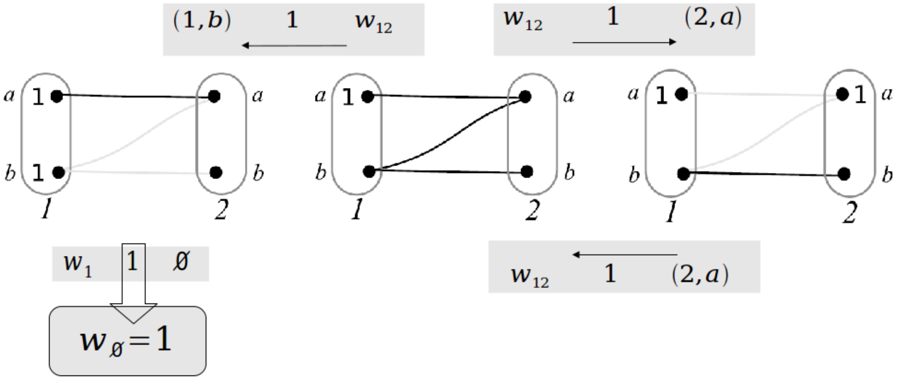
Equivalence Preserving Transformations (EPTs)



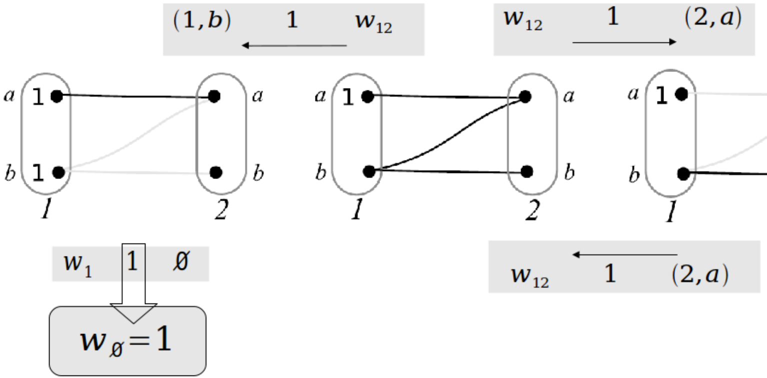
Equivalence Preserving Transformations (EPTs)



Equivalence Preserving Transformations (EPTs)

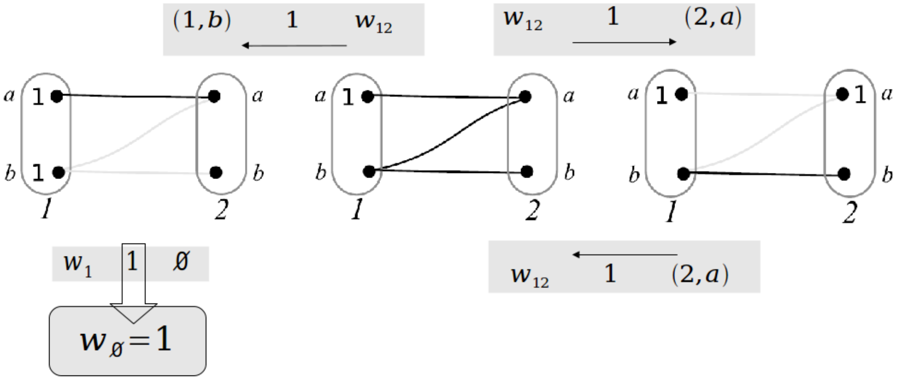


Equivalence Preserving Transformations (EPTs)



- EPTs may lead to different $w\varnothing$

Equivalence Preserving Transformations (EPTs)



- EPTs may lead to different $w\varnothing$
- VAC: defines EPTs to increase $w\varnothing$

Arc consistency (AC)

Classical CSP

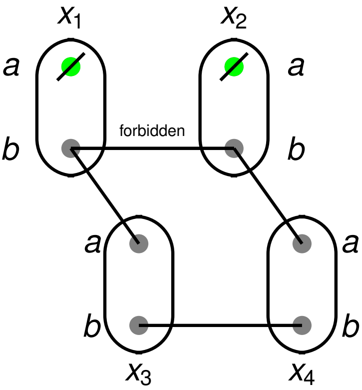
- WCSP without costs
- 2 possibilities for values and tuples: authorized or forbidden

Arc Consistency

for each value and each constraint, there is a tuple that allows this value

Filtering by AC

deleting all values that do not satisfy this property



Arc consistency (AC)

Classical CSP

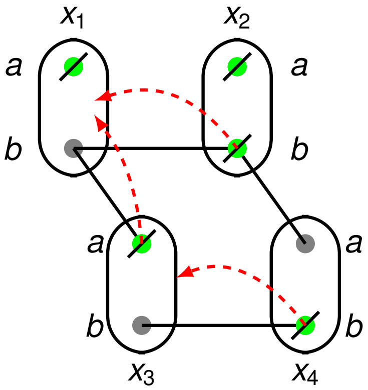
- WCSP without costs
- 2 possibilities for values and tuples: authorized or forbidden

Arc Consistency

for each value and each constraint, there is a tuple that allows this value

Filtering by AC

deleting all values that do not satisfy this property



Virtual Arc Consistency (VAC – [AIJ2010])

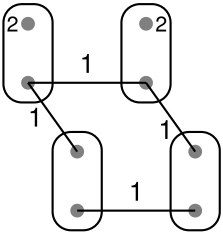
Bool(P)

classical CSP induced by a WCSP P that authorizes only zero cost values and tuples.

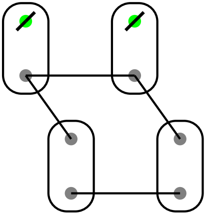
Virtual Arc Consistency (VAC – [AIJ2010])

Bool(P)

classical CSP induced by a WCSP P that authorizes only zero cost values and tuples.



P



Bool(P)

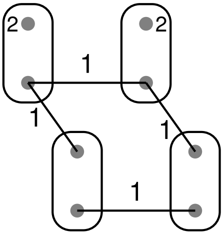
Virtual Arc Consistency (VAC – [AIJ2010])

Bool(P)

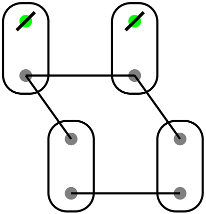
classical CSP induced by a WCSP P that authorizes only zero cost values and tuples.

VAC

P is VAC iff the AC closure of Bool(P) is non-empty



P



Bool(P)

Virtual Arc Consistency (VAC – [AIJ2010])

Bool(P)

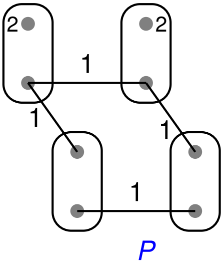
classical CSP induced by a WCSP *P* that authorizes only zero cost values and tuples.

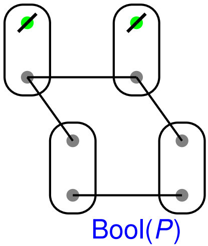
VAC

P is VAC iff the AC closure of Bool(*P*) is non-empty

If *P* is not VAC:

- enforcing AC in Bool(*P*) leads to a wipe out
- $\exists$ a way of shifting costs in *P* which leads to an increase of $f_{\varnothing}$.



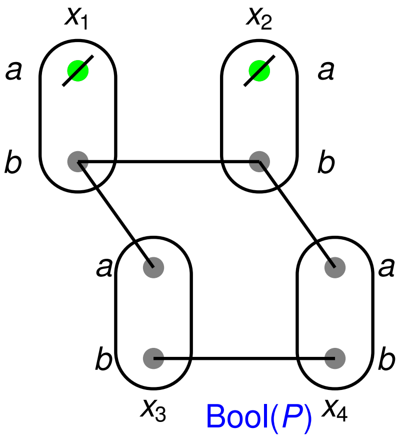


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

  Using EPTs to incrementally shift costs to the wipe-out variable.



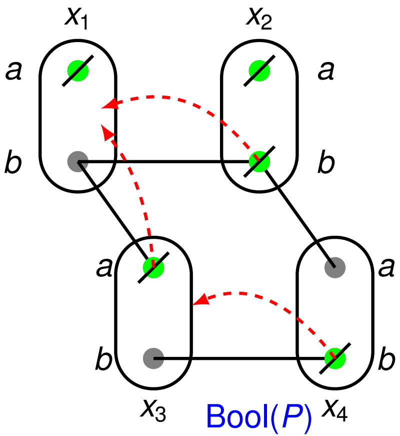
AC constraint order : f_{13}, f_{34}, f_{12}, f_{24}

Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

  Using EPTs to incrementally shift costs to the wipe-out variable.



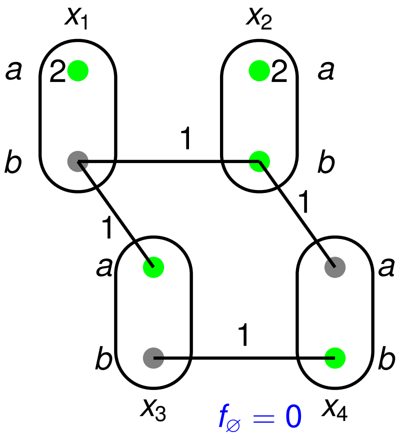
AC constraint order : f_{13}, f_{34}, f_{12}, f_{24}

Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

  Using EPTs to incrementally shift costs to the wipe-out variable.

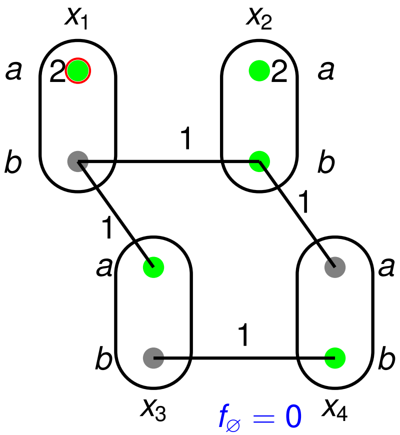


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

Using EPTs to incrementally shift costs to the wipe-out variable.

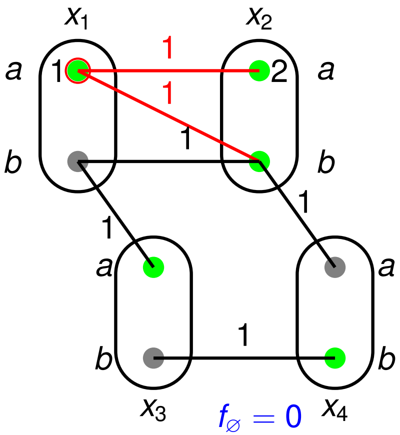


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

Using EPTs to incrementally shift costs to the wipe-out variable.

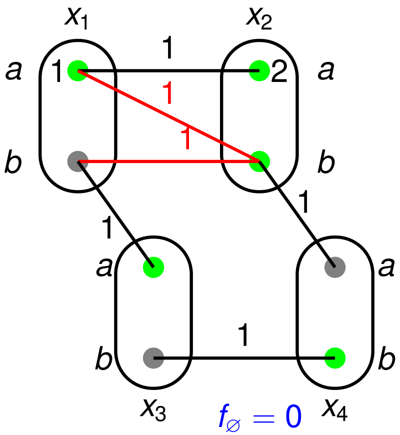


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

  Using EPTs to incrementally shift costs to the wipe-out variable.

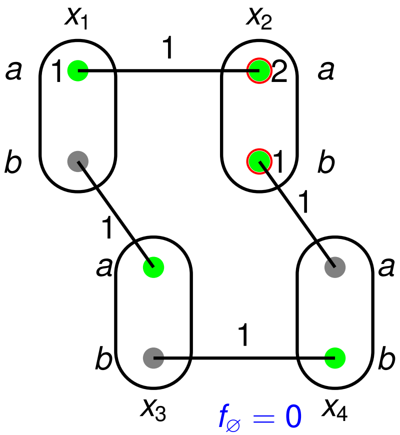


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

Using EPTs to incrementally shift costs to the wipe-out variable.

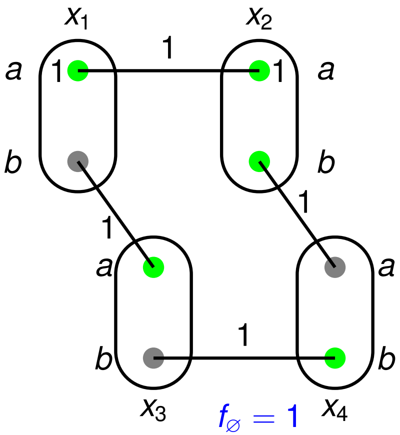


Enforcing VAC

Iterative process

- enforcing AC in Bool(P) until a wipe-out occurs
- transforming P into an equivalent problem with an increased $f_\varnothing$.

  Using EPTs to incrementally shift costs to the wipe-out variable.



Properties of VAC

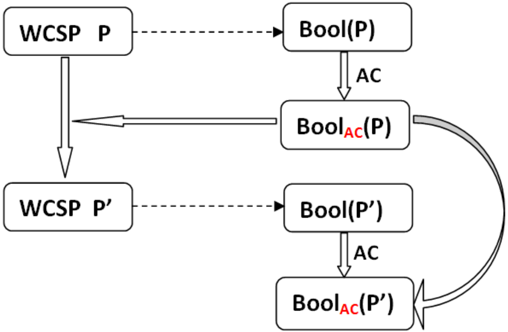
If Bool(P) is solved by AC, then P is solved by VAC

- trees
- problems with one value/variable
- submodular cost functions

Closed open hard problems

Radio link frequency assignment benchmarks

Motivation for Dynamic VAC



- VAC enforces AC on a sequence of incrementally modified CNs
- Maintaining Bool(P) by Dynamic AC $\Rightarrow$ Dynamic VAC

Dynamic VAC

Property

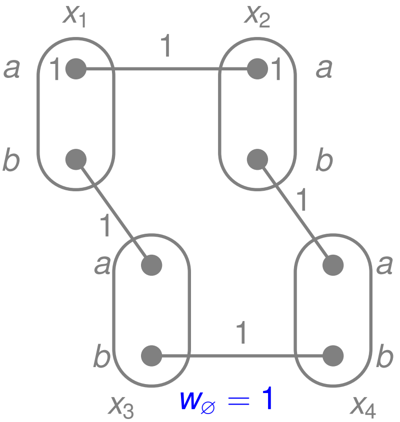
Each VAC iteration leads only to constraint relaxations in Bool(P)

Maintaining Bool(P) by dynamic AC (AC/DC2 – [FLAIRS 2005])

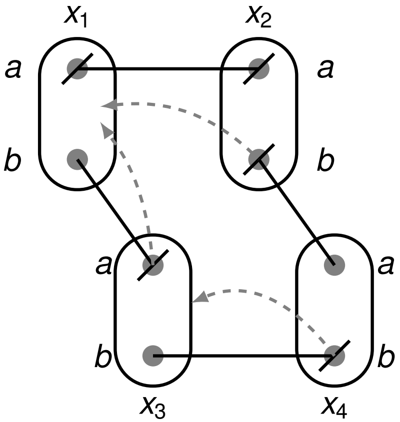
Relaxation procedure:

1. Restoring restorable values (see above)
2. Propagating restored values to neighborhood (new support)
3. Rechecking the restored values for AC

Avantages of DynVAC

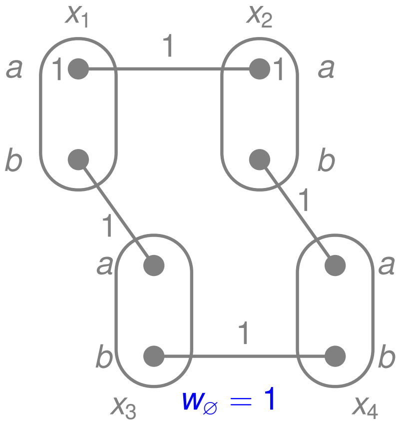


P'

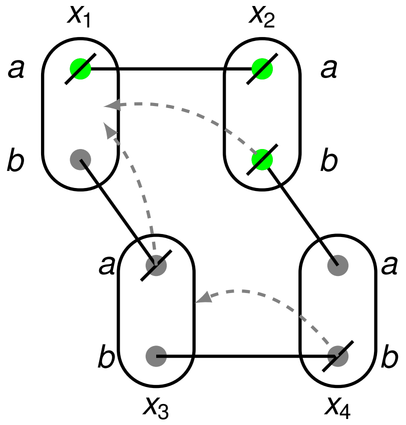


AC(Bool(*P*))

Avantages of DynVAC

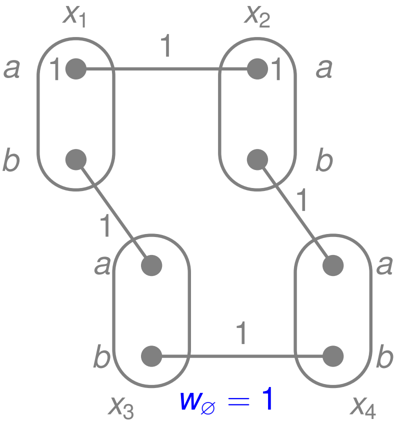


P'

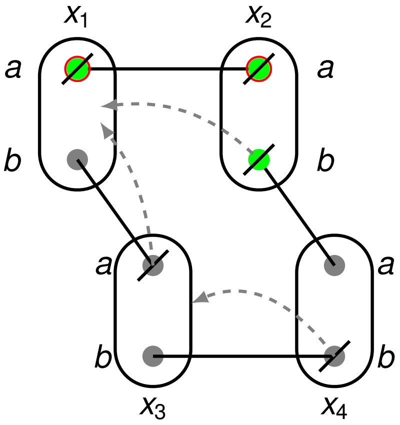


AC(Bool(P))

Avantages of DynVAC

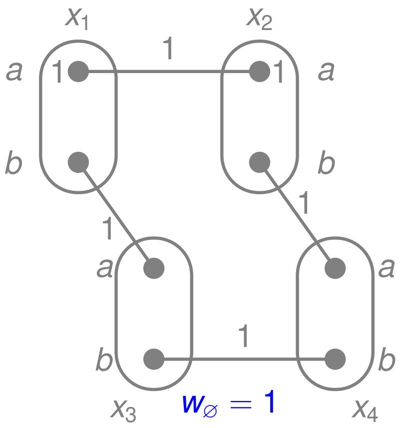


P'

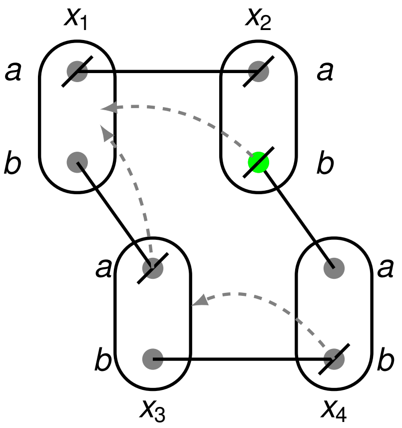


AC(Bool(*P*))

Avantages of DynVAC

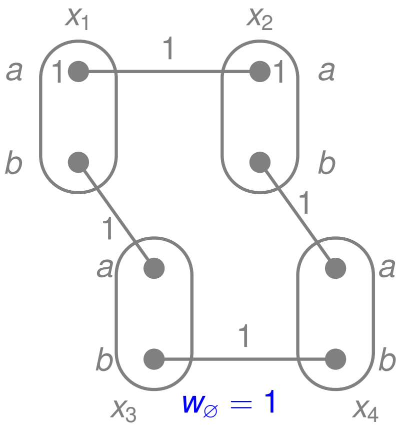


P'

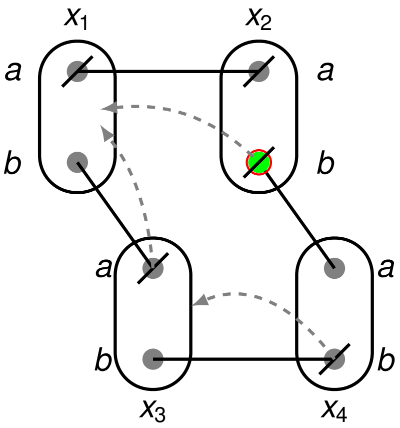


AC(Bool(*P*))

Avantages of DynVAC

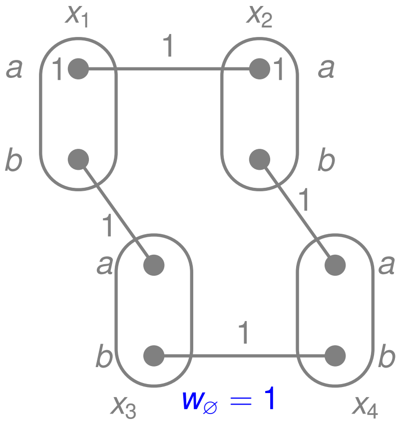


P'

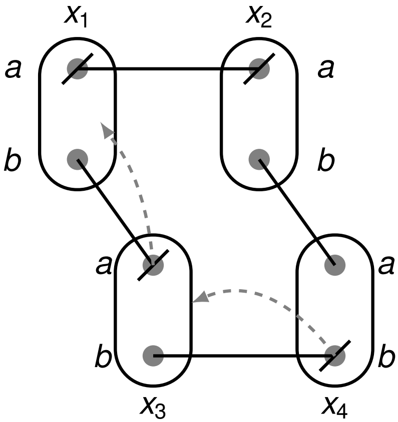


AC(Bool(P))

Avantages of DynVAC

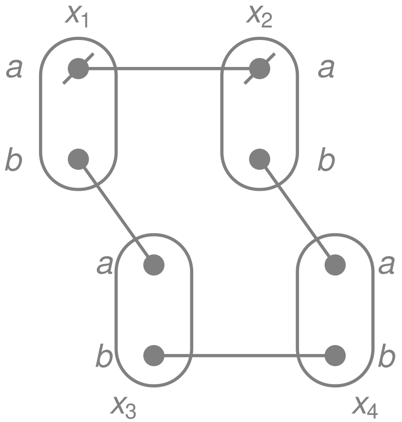


P'

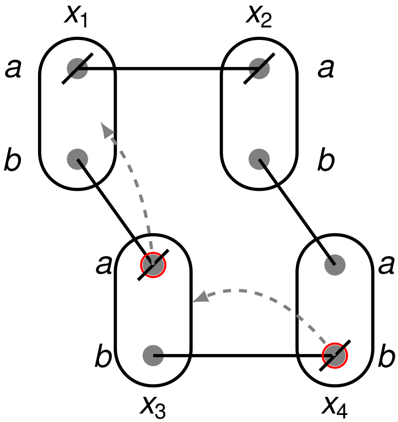


AC(Bool(*P*))

Avantages of DynVAC



Bool(P')



updated Bool(P)

- Two extra deleted values : (x_3, a), (x_4, b)

Experimentation: pre-processing

class	#prob	VAC	DynVAC	DynVAC + heuristic
celar	32	3.14	**1.92**	1.12
protein_maxclique	10	**51**	364	56.95
tagsnp_r0.5	25	364.31	**116.57**	81.46
tagsnp_r0.8	82	4.64	**1.53**	2.54
dimacs_maxclique	65	**0.78**	3.65	0.96
planning	68	0.25	**0.19**	0.23
warehouse	57	341	**66**	114.17

Cost Function Library

- DynVAC is faster than VAC for celar (1.6x), tagsnp (3x), warehouse (5x), but significantly slower for maxclique problems.
- A domain based heuristic handles those pathological cases.

			VAC	DynVAC	ratio	
			time(sec)	time(sec)	time	nodes
Results in search	CELAR	cl6-2	29	46	**1,59**	1.17
		gr11	470	366	0,78	0,65
		gr13	1.431	1.144	0,8	0,51
		sc06-18	1.511	736	**0,49**	0,77
		sc06-20	765	508	0,66	1,26
		sc06	5.227	2.454	0,47	0,79
		...				
	WAREHOUSE	capa	2.462	1.013	0,41	1,00
		capb	3.019	6.168	**2,04**	**1,35**
		capc	2.027	1.228	0,61	0,75
		capmo5	75	14	**0,19**	0.74
		capmq1	5.111	2.374	0,46	0,68
		capmq2	6.520	3.209	0,49	1,05
		...				
average (47 prob)			**1831**	**1117**	**0,61**	**0,83**

Only some worse and best cases are presented.

More experimentation

- Other CFLIB problems
- CPD (Computational Protein Design) problems
- ...

Conclusion

DynVAC

- incremental algorithm for enforcing VAC in WCSPs
- faster than VAC for large costs and large domains problems
- a heuristic that gets rid of pathological cases

Perspective

- extension to non-binary cost functions