Optimisation sous-contraintes :
Modèles, Algorithmes et Applications

Samir Loudni
Maître de Conférences
GREYC, CNRS UMR 6072, Université de Caen – France

Plan

1. Définition d'un problème d'optimisation combinatoire
2. Partie 1 - Contraintes molles : Modèles
3. Partie 2 - Contraintes molles : Algorithmes
4. Partie 3 - Contraintes molles : Applications
5. Partie 4 - Contraintes globales molles

Dans tout bon livre traitant de l'optimisation combinatoire

Une instance d'un problème d'optimisation combinatoire est définie par un triplet $(\mathcal{S}_p, \mathcal{S}_a, f)$ tel que :

- $\mathcal{S}_p$ est l'ensemble des solutions potentielles ;
- $\mathcal{S}_a \subseteq \mathcal{S}_p$ est l'ensemble des solutions admissibles ;
- f est une fonction de $\mathcal{S}_p$ dans $\mathbb{R}$.

Le problème est de trouver un élément $s^* \in \mathcal{S}_a$ qui minimise (ou maximise) la valeur de la fonction coût f.

$\rightarrow$ L'élément s^* s'appelle un optimum global.

Le formalisme des CSP (Montanari 1974)

Une instance CSP est un quadruplet $(\mathcal{X}, \mathcal{D}, \mathcal{C}, R)$, où:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ est un ensemble de n variables ;
- $\mathcal{D} = \{d_{x_1}, \ldots, d_{x_n}\}$ est un ensemble de domaines finis ;
- $\mathcal{C} = \{c_1, \ldots, c_e\}$ est un ensemble de e contraintes ;
- $R = \{r_{c_1}, \ldots, r_{c_e}\}$ est un ensemble de relations.

Une solution pour le CSP: une affectation complète des variables ne violant aucune contrainte ?

$\rightarrow$ C'est un problème NP-complet.

Représentation usuelle de l'ensemble des solutions potentielles

Implicite, sous la forme d'un ensemble $\mathcal{X}$ de variables et d'un ensemble $\mathcal{D}$ de domaines de valeur associés.

$$\mathcal{S}_p = \prod_{x \in \mathcal{X}} d_x$$

Représentation usuelle de l'ensemble des solutions admissibles

Implicite, sous la forme d'un ensemble $\mathcal{C}$ de contraintes.

Chaque contrainte $c \in \mathcal{C}$ lie un sous-ensemble $\mathcal{X}(c)$ des variables de $\mathcal{X}$ et définit l'ensemble $r_c (\in R)$ des combinaisons de valeur autorisées (ou interdites) pour les variables de $\mathcal{X}(c)$.

$$r_c \subseteq \prod_{x \in \mathcal{X}} d_x$$

$\mathcal{S}_a$ est défini comme étant l'ensemble des éléments de $\mathcal{S}_p$ qui satisfont toutes les contraintes de $\mathcal{C}$.

Exemple de CSP

Variables	x_1, x_2, x_3
Domaines	$D(x_1) = \{1, 2\}, D(x_2) = \{0, 1, 2, 3\}, D(x_3) = \{2, 3\}$
Contraintes	$x_1 > x_2$
	$x_1 + x_2 = x_3$

CSP après filtrage

Variables	x_1, x_2, x_3
Domaines	$D(x_1) = \{2\}, D(x_2) = \{0, 1\}, D(x_3) = \{2, 3\}$
Contraintes	$x_1 > x_2$
	$x_1 + x_2 = x_3$
Solution 1:	$x_1 = 2, x_2 = 0, x_3 = 2$
Solution 2:	$x_1 = 2, x_2 = 1, x_3 = 3$

Exemple de COP

Variables	x_1, x_2, x_3
Domaines	$D(x_1) = \{1, 2\}, D(x_2) = \{0, 1, 2, 3\}, D(x_3) = \{2, 3\}$
Contraintes	$x_1 > x_2$
	$x_1 + x_2 = x_3$
Fonction objective	maximiser $x_2 + x_3$
Solution	$x_1 = 2, x_2 = 1, x_3 = 3$

Et la fonction à optimiser ??

Pourquoi ne pas utiliser le le même cadre pour la représenter ?

Imposer qu'elle soit une fonction de l'insatisfaction des contraintes.

→ Avantages escomptés :

- avoir un seul cadre de représentation : variables, domaines de valeur, contraintes ;
- prendre en compte dans le même cadre des contraintes dures (à satisfaire de façon impérative) et des contraintes molles (à satisfaire au mieux) ;
- représenter de la même façon des problèmes cohérents, avec préférences sur les solutions admissibles, et des problèmes incohérents ou sur-contraints, avec préférences sur la relaxation des contraintes insatisfaites.

Exemple

Soit le CSP sur-contraint ci-dessous :

$x_1 \in \{1, 2\}, x_2 \in \{2, 3\}, x_3 \in \{2, 3\}$

(i) $x_1 > x_2$

(ii) $x_1 + x_2 = x_3$

$(x_1, x_2, x_3) = (1, 2, 2)$ les deux contraintes sont violées

$(x_1, x_2, x_3) = (1, 2, 3)$ la contrainte (i) est violée ← violation minimale

$(x_1, x_2, x_3) = (1, 3, 2)$ les deux contraintes sont violées

Partie 1 - Contraintes molles : Modèles

- Motivation
- Classes de CSP spécifiques
- Modèles de CSP générique
- Présentation du modèle VCSP (CSP Valués)

Pourquoi les contraintes molles ?

- La modélisation d'un problème réel fait naturellement apparaître :

  - des contraintes dures ;

  - des contraintes préférablement satisfaites ;

- La modélisation dans les CSP peut conduire à l'incohérence (sur-contraint) ou à une abondance de solutions médiocres (sous-contraint).

- Traiter les problèmes sur-contraints.

- Nécessité d'exprimer la notion de préférence sur les contraintes.

  $\rightarrow$ Transformation du problème de décision en un problème d'optimisation.

- Définir un cadre homogène pour l'optimisation sous contraintes.

Plusieurs classes spécifiques de CSP:

- **Partial** CSPs (Freuder & Wallace, 1992)

  maximiser le nombre de contraintes satisfaites

- **Weighted** CSPs (Larossa, 2002)

  un poids est associé à chaque contrainte

  maximiser la somme des poids des contraintes satisfaites

- **Possibilistic** CSPs (Schiex, 1992)

  un poids est associé à chaque contrainte représentant son importance

  minimiser le poids de la contrainte la plus importante parmi celles qui sont violées

- **Lexicographic** CSPs (Fargier et al., 1993)

  pas uniquement fonction du poids de la contrainte la plus importante

  tenir compte du nombre de contraintes non satisfaites à chaque niveau d'importance

Possibilistic CSPs

- Chaque contrainte c est annotée par une priorité k : nécessité que la contrainte c soit satisfaite est au moins k ;
- La valeur d'une instanciation complète est donnée par la valeur des contraintes non satisfaites de priorité maximale :

  - si la *priorité* a est plus forte que celle de b, on préférera alors violer n'importe quel nombre de contraintes de *priorité* b qu'une seule contrainte de *priorité* a.
- Objectif : minimiser la contrainte la plus importante parmi celles qui sont violées ;
- Exemple :

  - priorités des contraintes insatisfaites par A_1 : $\{0.6\}$;

  - priorités des contraintes insatisfaites par A_2 : $\{0.5, 0.4, 0.4, 0.4\}$;

  - A_2 préférée à A_1.

Weighted CSPs

- Chaque contrainte c est annotée par un poids ;
- La valeur d'une instanciation complète est définie par la somme des poids des contraintes insatisfaites ;
- L'objectif : minimiser la somme des poids des contraintes insatisfaites ;
- Cadre Max-CSP :

  - chaque contrainte est annotée par 0 si satisfaite ou 1 si violée ,

  - valeur d'une instanciation complète : nombre de contraintes insatisfaites ,

  - objectif : minimiser le nombre de contraintes insatisfaites.

Modèles de CSP génériques

- Un cadre générique permettant de couvrir les propositions existantes :

  - éviter la redondance.

  - pas de sémantique particulière.

- Pour les réseaux de contraintes molles, deux modèles existent :

  - les CSP valués (VCSP) ;

  - les Semiring-based CSPs (SCSP).

Cadre VCSP

Valued Constraint Satisfaction Problem

(T. Schiex, H. Fargier, G. Verfaillie, IJCAI95)

Informellement

À chaque contrainte, est associé une valuation reflétant son importance : préférence, poids, gain, priorité, probabilité d'existence ...

La valuation d'une affectation est l'agrégation des valuations des contraintes insatisfaites.

Le problème est de produire une affectation de valuation minimale.

Cadre VCSP : plus formellement

- VCSP = $(\mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{R}, \mathcal{S}, \varphi)$, $\mathcal{S} = (E, \succ, \bot, \top, \otimes)$
- E = ensemble de valuations, permettant de valuer les contraintes et les affectations ;
- $\top$ = élément maximum de E, permettant de valuer les contraintes dures et les affectations totalement incohérentes ;
- $\bot$ = élément minimum de E, permettant de valuer affectations totalement cohérentes ;
- $\succ$ = ordre total sur E, permettant de comparer deux valuations ;
- $\otimes$ = opérateur sur E, permettant d'agréger deux valuations ;
- $\varphi : \mathcal{C} \rightarrow E$ = fonction de valuation, permettant de valuer chaque contrainte.

Valuation d'une affectation complète

Soit $\mathcal{A}$ une affectation *complète* des variables, et soit $\mathcal{C}_{unsat}(\mathcal{A})$ l'ensemble de toutes les contraintes *insatisfaites* par $\mathcal{A}$;

$$\varphi(\mathcal{A}) = \bigotimes_{c_i \in \mathcal{C}_{unsat}(\mathcal{A})} \varphi(c_i)$$

$\varphi(c_i)$ représente la valuation associée à la violation de la contrainte c_i.

$\rightarrow$ L'objectif est de produire une *solution* (affectation complète) de valuation *minimale*.

Propriétés minimales de la structure de valuation

- $\otimes$ commutatif et associatif;

  la valuation d'une affectation ne dépend pas de la façon de réaliser l'agrégation ;

- $\bot$ élément neutre de $\otimes$: $\forall\, a \in E, a \otimes \bot = a$

  les contraintes satisfaites ne modifient pas le résultat de l'agrégation ;

- $\top$ élément absorbant de $\otimes$: $\forall\, a \in E, a \otimes \top = \top$

  les contraintes dures insatisfaites induisent une insatisfaction totale ;

- $\otimes$ monotone relativement à $\succ$: $\forall\, a, b, c \in E, a \succeq c \Rightarrow (a \otimes b) \succeq (c \otimes b)$;

  garantit que la valuation d'une instanciation A qui viole un sous-ensemble de contraintes violées par A', sera toujours meilleure que celle de A'.

Différents sous-cadres

CSP	E	$\succ$	$\bot$	$\top$	$\otimes$
Classical	$\{v, f\}$	$f \succ v$	v	f	$\wedge$
Possibilistic	$[0, 1]$	$>$	0	1	max
weighted	$\overline{\mathbb{N}}$	$>$	0	$+\infty$	$+$
Lexicographic	$\overline{\mathbb{N}}^*$	$>^*$	$\emptyset$	$\{+\infty\}$	$\cup$

Des variantes du cadre VCSP

- Valuation des tuples : une valuation est associée à chaque n-uplet interdit d'une contrainte c ;

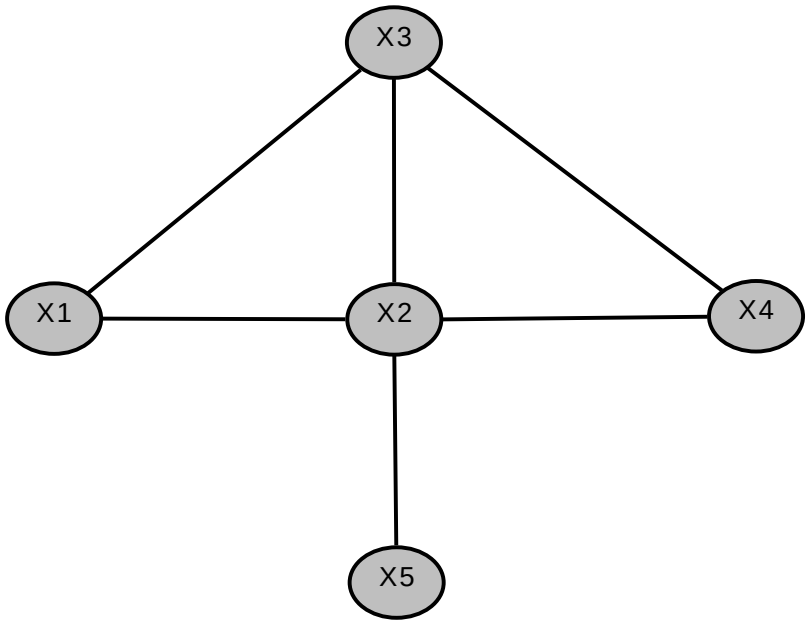
$$\varphi(\mathcal{A}) = \bigotimes_{c_{i,j} \in \mathcal{C}_{unsat}(\mathcal{A})} \varphi(\mathcal{A} \downarrow \{i,j\}) \bigotimes_{c_i \in \mathcal{C}_{unsat}(\mathcal{A})} \varphi(\mathcal{A} \downarrow \{i\})$$

- valuation des variables : la valuation d'une instanciation complète est fonction des variables non instanciées ;

$$\varphi(\mathcal{A}) = \bigotimes_{v \in \mathcal{V}/\text{v non instanciée dans A}} \varphi(v, \mathcal{A})$$

Weighted CSP example: 3-coloring graph

- Colors:Red (r), Grenn (g), Blue (b)



- Objectif: minimize the number of non blue vertices

Partie 2 - Contraintes molles : Algorithmes

- Weighted CSPs - Méthodes de résolution
- Weighted CSPs - Méthodes de résolution complètes
- Weighted CSPs - Méthodes de recherche locale

WCSP - Méthodes de résolution

- Les méthodes complètes
  - basées sur une énumération complète de l'arbre des solutions.
  - elles utilisent les algorithmes de type séparation et évaluation.
  - elles garantissent l'obtention de l'optimum.

- Les méthodes de recherche locale
  - une première solution est (rapidement) générée, puis itérativement améliorée par une succession de mouvements.
  - elles ne garantissent pas l'optimalité, mais fournissent de bonnes solutions très rapidement.

WCSP - Algorithme par séparation et évaluation

Schéma de base :

- recherche systématique en profondeur d'abord.
- majorant ub : la meilleure valuation connue jusqu'ici.
- minorant lb : de la meilleure valuation que l'on peut obtenir en étendant l'affectation courante.
- backtrack : quand $lb \geq ub$.

$\rightarrow$ Crucial: calcul du minorant.

Notations

- Variables : x_i, x_j, x_k, ...
- Valeurs :a, b, c, ...
- Contraintes : c_i, ...

- P : ensemble des variables déjà instanciées (variables passées).
- F : ensemble des variables non encore instanciées (variables passées).
- $\mathcal{A}_i$: instanciation partielle courante des i variables passées.

- $\mathcal{C}_{PFunsat}(\mathcal{A}_i, x, a)$: ensemble des contraintes insatisfaites par $\mathcal{A}_i \cup (x, a)$ et telles que toutes les variables qu'elles relient, sauf une (variable x), sont instanciées.
- $\mathcal{C}_{Funsat}(x_i, a)$: ensemble des contraintes binaires non-satisfaites portant sur des variables futures liés à x_i.

Algorithme par séparation et évaluation

procedure DFB&B$(\mathcal{A}_i, F)$

begin

  if $F = \emptyset$ **then**

    BESTS $\leftarrow \mathcal{A}_i$;

    UB $\leftarrow \varphi(\mathcal{A}_i)$;

  else

    $x \leftarrow$ select-variable(F);

    while $D_x \neq \oslash$ **do**

      $a \leftarrow$ select-value(D_x) ;

      if $(\text{LB}(\mathcal{A}_i \cup \{(x, a)\}) < \text{UB})$ **then** DFB&B$(\mathcal{A}_i \cup \{(x, a)\}, F \setminus \{x\})$;

      $D_x \leftarrow D_x \setminus \{a\}$;

end

function LB$(\mathcal{A}_p)$

begin

  return $\sum_{c_i \in \mathcal{C}_{unsat}(\mathcal{A}_p),\ \mathcal{X}(c_i) \subseteq P} \varphi(c_i)$;

end

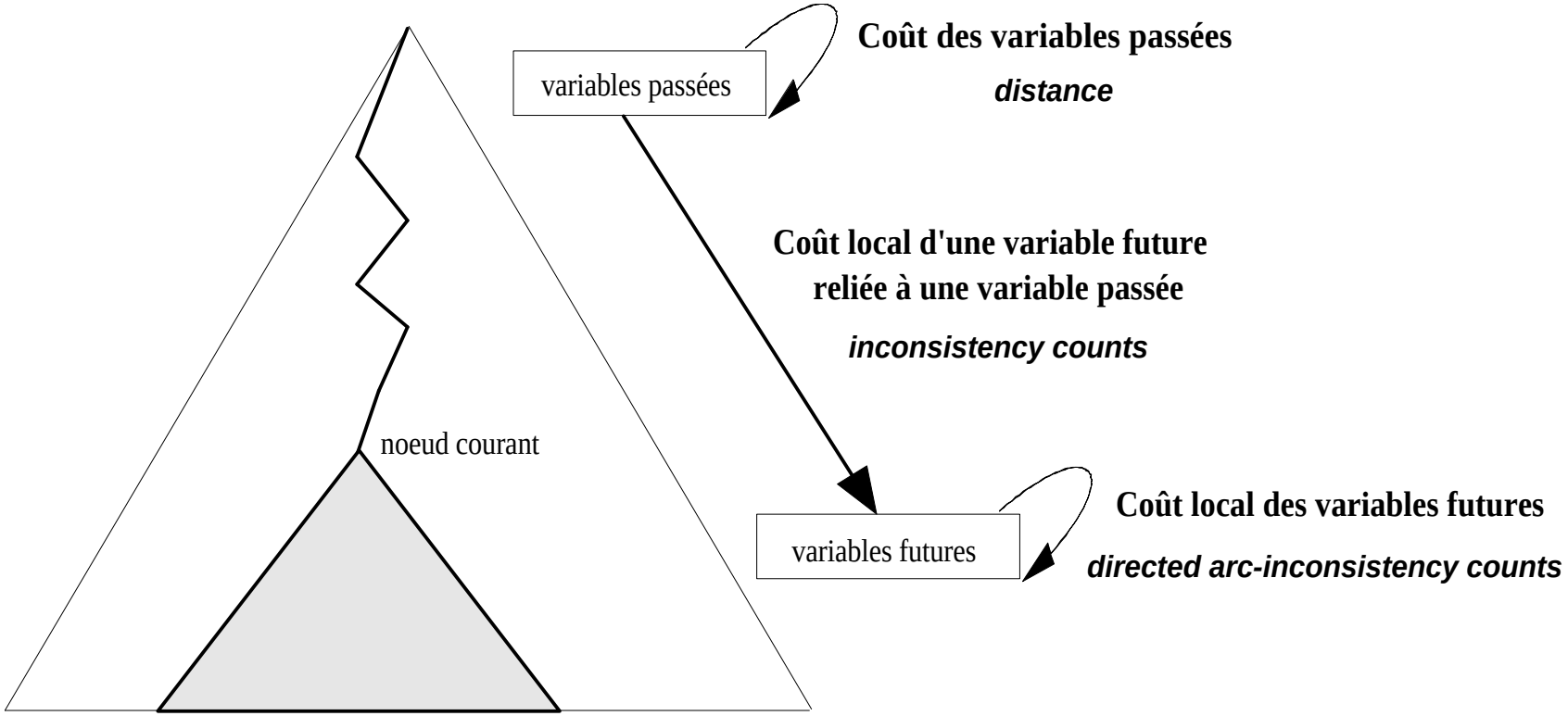
WCSP - calcul de minorants (LB)

- Backward-Checking (BC) : exploite les contraintes entre variables affectées.
- Forward-Checking (FC) : exploite les contraintes entre variables affectées et non affectées.
- Restent les contraintes entre variables non affectées:

  - Directed Arc-Inconsistency Counts:

  Idée principale: combiner des valuations dont les justifications, en terme de contraintes, sont disjointes dans des compteurs DAC externes aux CSP.

WCSP - calcul de minorants (LB)



WCSP - calcul de minorants (LB)

- Directed Arc-Consistency Counts (DAC)

  (R. Wallace, ECAI'94)

- Directed Constraint Graph

  (J. Larrosa, P. Meseguer et T. Schiex, AI'99)

  - Reversible DAC (RDAC)
  - Maintaining Reversible DAC (MRDAC)

- Weigthed Arc-Consistency Counts (WAC)

  (M. Affane, H. Bennaceur, ECAI'98)

Partial Forward-Checking

(Freuder & Wallace, 92)

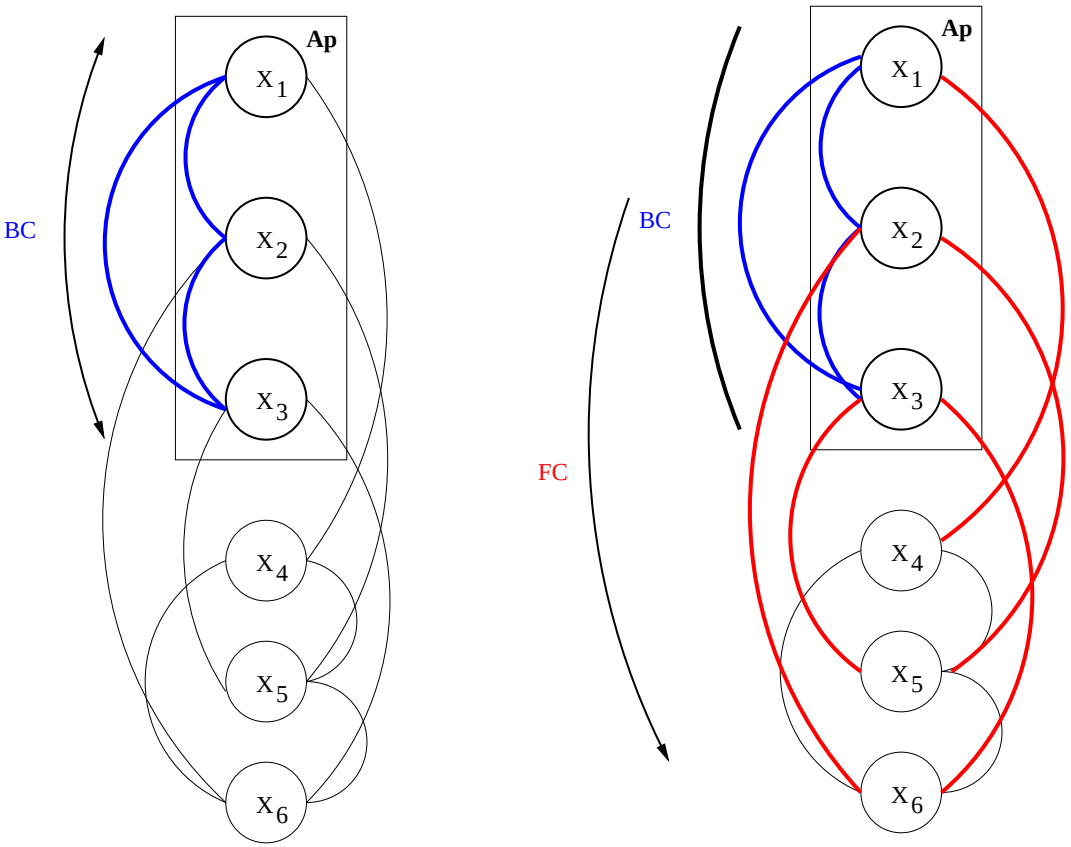
- Branch and Bound + Lookahead
- Lookahead : après instanciation d'une variable par une valeur :

  - pour chaque valeur a d'une variable future x est associé un compteur d'inconsistance ic_{ia} ;

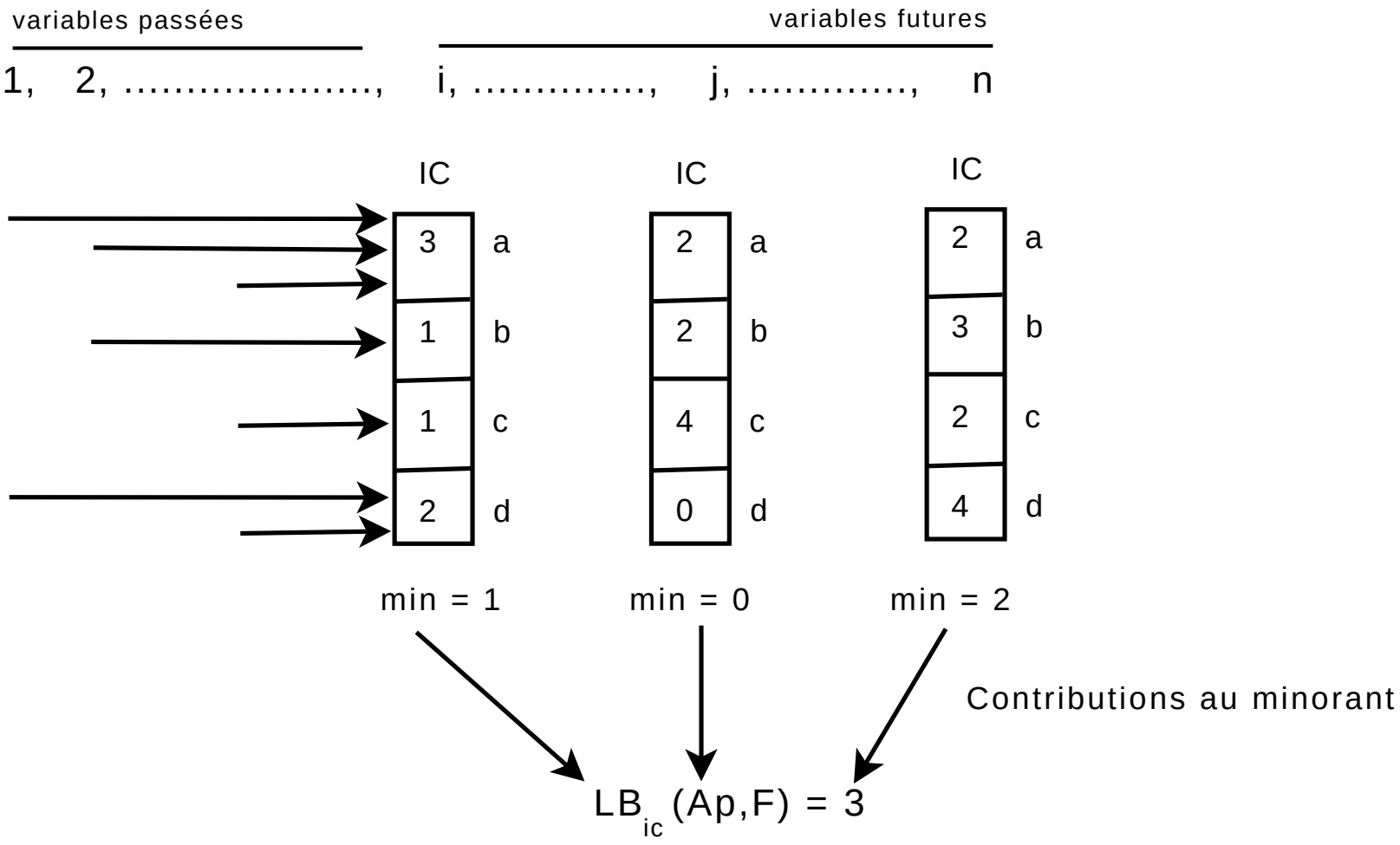
  - ic_{ia} mémorise la somme des coûts des contraintes appartenant à $\mathcal{C}_{PFunsat}(\mathcal{A}_p, x_i, a)$ (i.e., contraintes entre variables passées et x_i) qui seront violées si $x_i = a$.

$$ic_{ia} = \sum_{c \in \mathcal{C}_{PFunsat}(\mathcal{A}_p, x_i, a)} \varphi(c)$$

Compteurs d'Inconsistance



Compteurs d'Inconsistance



PFC Lower Bound

(Freuder & Wallace, 92)

- Nouveau minorant :

$$LB_{fc}(\mathcal{A}_p, F) = LB_{bc}(\mathcal{A}_p) + \sum_{x_j \in F} \min_{b \in D_j} (ic_{jb})$$

- Un minorant LB_{jb} associé à la valeur $b \in D_j$:

$$LB_{fc}(\mathcal{A}_p, F, x_j, b) = LB_{bc}(\mathcal{A}_p) + \sum_{x_k \in F, k \neq j} \min_{c \in D_k} (ic_{kc}) + ic_{jb}$$

- Élimination des valeurs futures : $LB_{fc}(\mathcal{A}_p, F, x_j, b) \succeq ub$

DAC statique

(R. Wallace, ECAI'94)

- DAC : (directed arc-inconsistency count) sur les valeurs futures :

  - soit un ordre statique sur les variables: $1, 2, \ldots, j, \ldots, n$

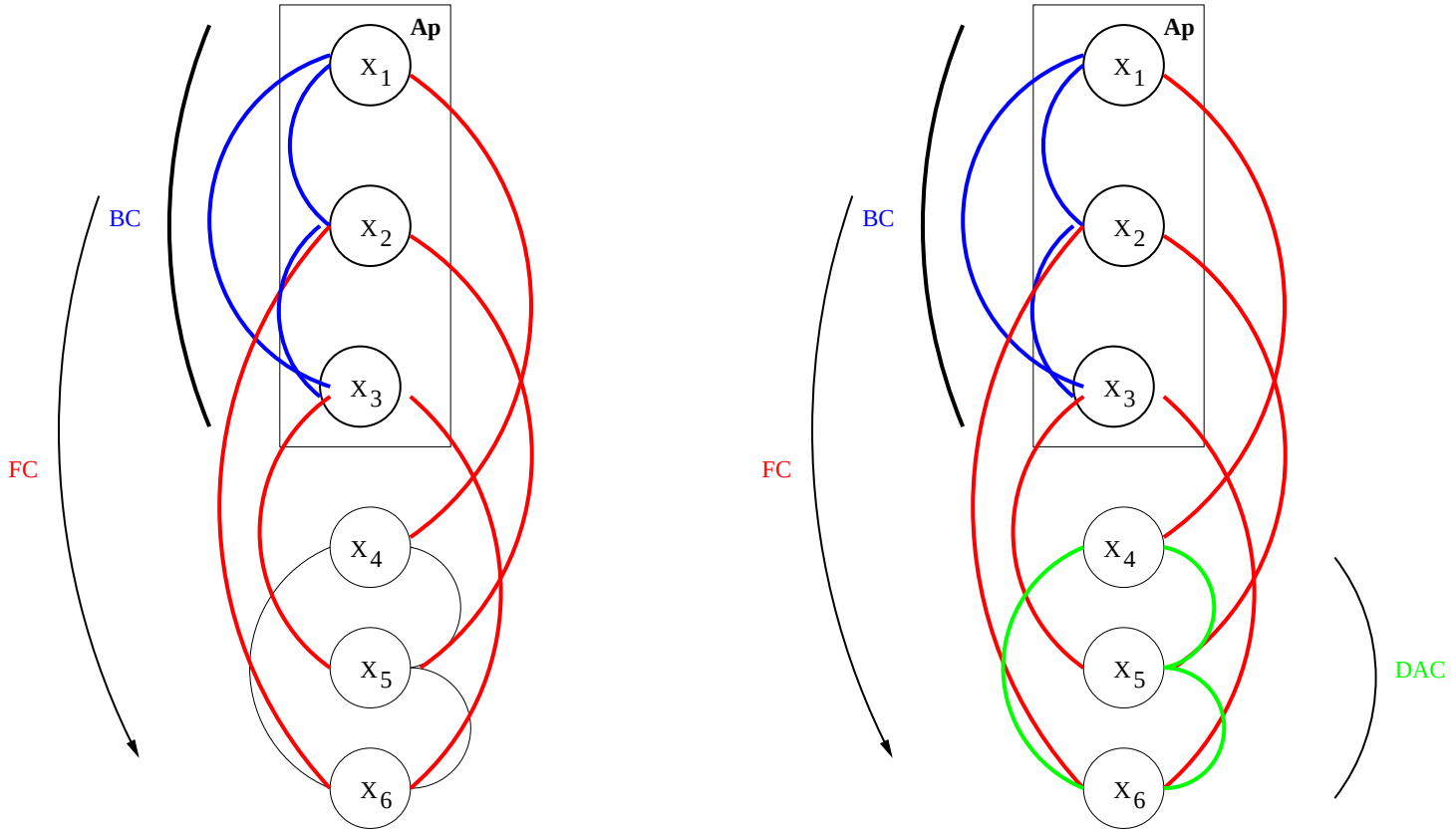
  - chaque contrainte est exploitée dans une direction et une seule.

  - pour chaque valeur a de chaque variable future x_i est associé un compteur d'arc-inconsistance orientée dac_{ia}

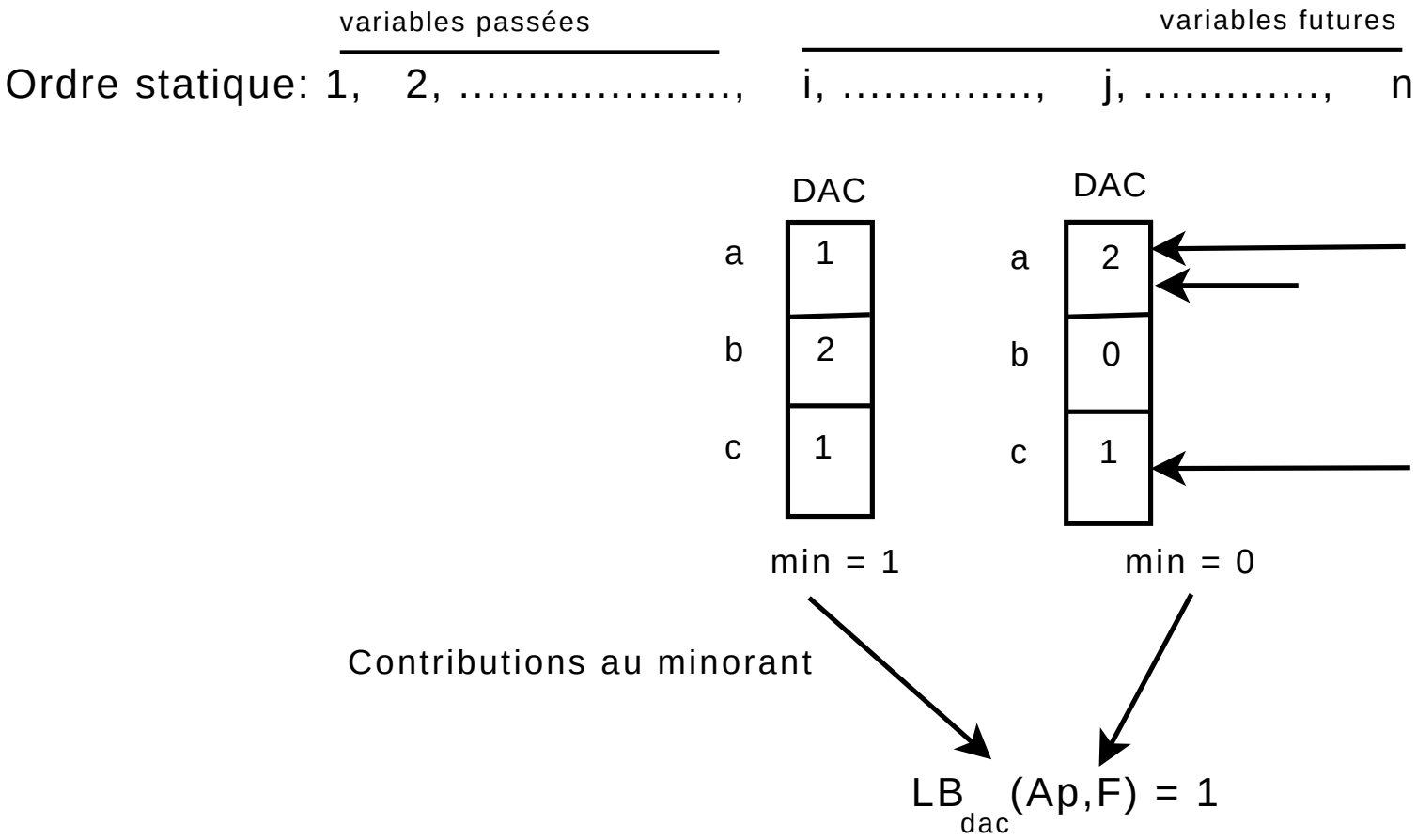
  - dac_{ia} mémorise la somme des valuations des contraintes c_{ij} portant sur des variables futures situées juste après x_i dans l'ordre utilisé ($i < j$) et appartenant à $\mathcal{C}_{Funsat}(x_i, a)$.

$$dac_{ia} = \sum_{c_{ij} \in \mathcal{C}_{Funsat}(x_i, a), \text{avec } i < j} \varphi(c_{ij})$$

Compteurs d'arc-inconsistance orientée



Compteurs d'arc-inconsistance orientée



Combinaison IC + DAC statique

(Larrosa & Meseguer, 96)

- L'ensemble des contraintes référencées par IC et DAC sont disjointes :

  - grâce à l'ordre statique sur les variables.

  - possibilité de combiner les deux compteurs.

- Nouveau minorant :

$$LB_{dac}(\mathcal{A}_p, F) \;=\; LB_{bc}(\mathcal{A}_p) \;+\; \sum_{x_i \in F} \min_{a \in D_i} (ic_{ia} + dac_{ia})$$

- Un minorant LB_{jb} associé à la valeur $b \in D_j$:

$$LB_{dac}(\mathcal{A}_p, F, x_j, b) \;=\; LB_{dac}(\mathcal{A}_p, F - \{x_j\}) \;+\; ic_{jb} \;+\; dac_{jb}$$

- Élimination des valeurs futures : $LB_{dac}(\mathcal{A}_p, F, x_j, b) \succeq ub$

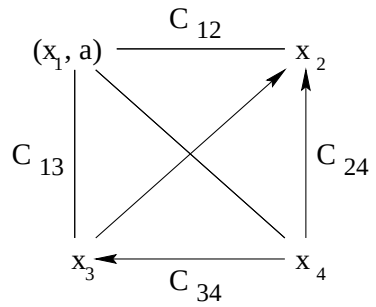
$\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ $\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}_3 = \mathcal{D}_4 = \{a, b, c\}$

$C_{12} = \{(a, c), (b, a), (b, c)\}$ $C_{13} = \{(a, b), (b, b)\}$ $C_{23} = \{(a, a), (a, c)\}$

$C_{14} = \{(a, b), (b, c), (c, a)\}$ $C_{24} = \{(b, a), (b, b)\}$ $C_{34} = \{(a, c), (c, c)\}$

$\mathcal{A}_p = \{(x_1, a)\}, P = \{x_1\}, F = \{x_2, x_3, x_4\}$

DAC calculé sous un ordre lexicographique : $(1 < 2 < 3 < 4)$



	ic	*dac*
a	1	1
b	1	1
c	0	2

	ic	*dac*
a	1	0
b	0	1
c	1	0

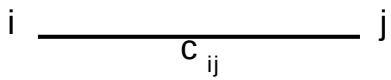
	ic	*dac*
a	1	0
b	0	0
c	1	0

$$LB_{bc}(\mathcal{A}_p) == 0$$

$$LB_{fc}(\mathcal{A}_p, F) = LB_{bc}(\mathcal{A}_p) + \sum_{i \in F} min_{a \in \mathcal{D}_i} (ic_{ia} = 0 + 0 = 0$$

$$LB_{dac}(\mathcal{A}_p, F) = LB_{bc}(\mathcal{A}_p) + \sum_{i \in F} min_{a \in \mathcal{D}_i} (ic_{ia} + dac_{ia}) = 0 + 3 = 3$$

DAC: Directed constraints

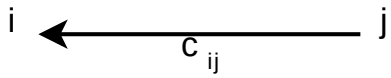


i	j	c_{ij}
a	a	1
a	b	2
b	a	3
b	b	4

dac	i	j
a	1	1
b	3	2

min dac(i) + min dac(j) = 1 + 1 = 2 mais C(i,j) = 1 !!

- **Pour utiliser les DAC, les contraintes doivent etre orientées**
- **Les DAC sont stockés dans la variable pointée par la contrainte**
- **implicite dans DAC statique par l'ordre statique des variables**



dac	i	j
a	1	0
b	3	0

min dac(i) + min dac(j) = 1 + 0 = 1 OK!

Graph-based DAC

(Larrosa & Meseguer, Schiex, AI'99)

- Motivation : faire disparaître la restriction à un ordre statique sur les variables : orienter toutes les contraintes binaires du graphe de contraintes.

- Le graphe de contraintes orienté G^F :

  - nœuds(G^F) = F,

  - edges(G^F) = $\{(j,i) \mid c_{ij}$, orientée de j vers $i\}$

- Nouveau DAC basé sur G^F :

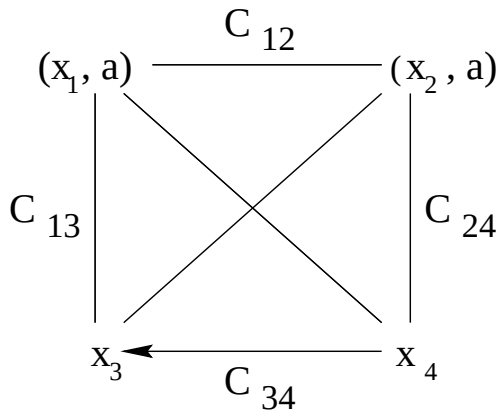
$$dac_{ia}(G^F) = \sum_{c \in \mathcal{C}_{Funsat}(x_i, a, G^F)} \varphi(c)$$

  avec $\mathcal{C}_{Funsat}(x_i, a, G^F)$ l'ensemble des contraintes binaires orientées entre x_i et ses prédécesseurs dans G^F qui sont arc-inconsistants avec (x_i, a).

Graph-based DAC et calcul de minorants

$\mathcal{A}_p = \{(x_1, a), (x_2, a)\}, P = \{x_1, x_2\}, F = \{x_3, x_4\}$, $\text{EDGES}(G_1^F) = \{(4, 3)\}$.

$LB_{dac}(\mathcal{A}_p, F, G_1^F) = LB_{bc}(\mathcal{A}_p) + \sum_{x_j \in F, j \neq i} \min_{b \in D_j} (ic_{jb} + dac_{jb}(G^F)) = 1 + 1 + 1 = 3$



	ic	*dac*
a	1	0
b	1	1
c	1	0

	ic	*dac*
a	2	0
b	1	0
c	2	0

Algorithme PFC-Graph-based DAC

```
procedure PFC-GraphDAC(A_i , F, G^F)

begin
    if F = ∅ then
        BestS ← A_i ;
        UB ← φ (A_i);
    else
        x ← select-variable (F) ;
        while D_x ≠ ⊘ do
            a ← select-value (D_x) ;
            if (LB(A_i , F, x, a, G^F) < UB) then
                look-ahead (A_i , F, x, a, UB, G^F) ;
                if (not empty-domain and LB (A_i , F, x, a, G^F) < UB) then
                    PFC-GraphDAC (A_i ∪ {(x, a)},F \ {x},G^F);
            D_x ← D_x \ {a} ;
end
```

Fonction Look-ahead

procedure look-ahead ($\mathcal{A}_p, F, x_i, a$,UB,G^F)

begin

  forall $x_j \in F - \{x_i\}$

    forall $b \in D_j$

      if $(\mathrm{LB}_{jb}(\mathcal{A}_p, F, x_i, a) \geq \mathrm{UB})$ **then** $D_j \leftarrow D_j \setminus \{b\}$;

      else if $(\mathit{inconsistent}(x_i, a, x_j, b))$ **then**

        $ic_{jb} \leftarrow ic_{jb} + \varphi(c_{ij})$;

        if $(\mathrm{LB}_{jb}(\mathcal{A}_p, F, x_i, a) \geq \mathrm{UB})$ **then** $D_j \leftarrow D_j \setminus \{b\}$;

end

Reversible DAC et Maintaining DAC

(Larrosa & Meseguer, Schiex, AI'99)

- Reversible DAC (RDAC) :

  - ordre dynamique ;

  - chaque contrainte est exploitée dans une direction qui change au cours de la recherche.

  - opération de base : réorientation des contraintes.

- Maintaining DAC (MRDAC) :

  - maintient des compteurs DAC durant la recherche, comme dans l'algorithme MAC ;

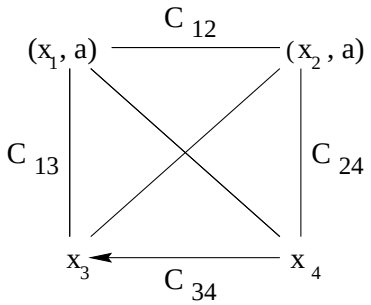
  - propagation des suppressions de valeurs des variables futures ;

  - utilisation des algorithmes à base d'AC.

Reversible DAC et calcul de minorants

$\mathcal{A}_p = \{(x_1, a), (x_2, a)\}, P = \{x_1, x_2\}, F = \{x_3, x_4\}$, EDGES$(G_1^F) = \{(4, 3)\}$.

$$LB_{dac}(\mathcal{A}_p, F, G_1^F) = 1 + 2 = 3$$

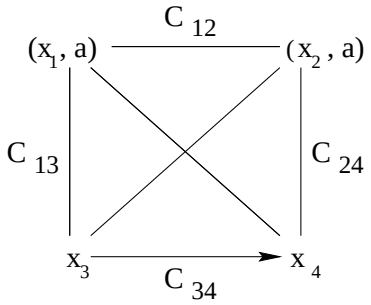


	ic	*dac*
a	1	0
b	1	1
c	1	0

	ic	*dac*
a	2	0
b	1	0
c	2	0

Réorientation de la contrainte C_{34} : l'arc $(4, 3)$ devient alors $(3, 4)$, EDGES$(G_2^F) = \{(3, 4)\}$.

$$LB_{dac}(\mathcal{A}_p, F, G_2^F) = 1 + 3 = 4$$



	ic	*dac*
a	1	0
b	1	0
c	1	0

	ic	*dac*
a	2	1
b	1	1
c	2	0

Partie 3 - Contraintes molles : Applications

Affectation de fréquences

Étant donné un ensemble de liens de communications radio, affecter à chaque lien de communication deux fréquences en :

- respectant une distance fixe entre les deux fréquences d'un lien

$$| f_i - f_j | = d_{ij} \quad (1)$$

- minimisant les interférences (poids des contraintes violées).

$$| f_i - f_j | > \alpha_{ij} \quad (2)$$

Plusieurs critères de minimisation :

- minimiser la fréquence maximum utilisée (CSP possobilistes) ;
- minimiser le nombre de fréquences différentes utilisées ;
- minimiser la somme des poids des contraintes violées (CSP Additifs).

Quelques instances

Instance	nb de variables	nb de contraintes	Satisfiable	Type	Meilleure Solution
scen01	916	5548	Oui	O1	16*
scen02	200	1235	Oui	O1	14*
scen06	200	1322	Non	O3	3389*
scen07	400	2865	Non	O3	342592
scen08	916	2744	Non	O3	262
scen09	680	4103	Non	O3	15571
scen10	680	4103	Non	O3	31516
scen11	680	4103	Oui	O1	22*
6-Sub1	28	314	Non	O3	2669*