Chapitre II – Problèmes de satisfaction de contraintes (CSP)

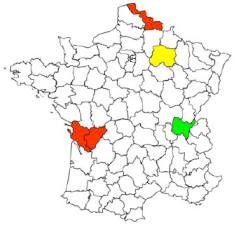
Master ANDROIDE – RP (M1) – Patrice Perny

LIP6 – Université Paris 6

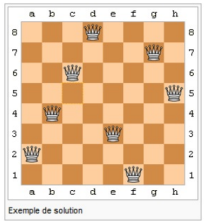
Plan

Introduction aux CSP

Emploi du temps



Coloration



N-reines

3		5	8		4			
	8	1		6	7		5	4
2							1	7
		3			9		8	1
		8				4		
1	4		3			7		
6	3							8
8	5		1	3		9	4	
			9		8	5		6

Sudoku

I) Formulation d'un CSP

Définition d'un CSP

Définition

Un pb de satisfaction de contraintes (CSP) est un problème $\mathcal{P}$ caractérisé par un triplet (X, C, D) où :

- $X = (x_1, \ldots, x_n)$, n variables de décision
- $D = (D_1, \ldots, D_n)$, n domaines (finis), D_i étant le domaine de la variable x_i, i.e. $(\forall i = 1, \ldots, n, x_i \in D_i)$
- $C = (c_1, \ldots, c_m)$, m contraintes, chaque contrainte c_i étant définie par un couple (v_i, r_i) où :
  - v_i est une liste de variable $x_{i_1}, \ldots, x_{i_{n_i}}$
  - r_i est un relation définie par un sous-ensemble du produit cartésien $D_{i_1} \times \ldots \times D_{i_{n_i}}$

Exemple : Choix d'un séjour de vacances

Choix d'un stage de sport avec un(e) ami(e), différents lieux possibles (Biarritz, Chamonix, Deauville), différents sports possibles (Escalalde, Golf, Polo, Surf), différents amis (C, J, P, N).

Ici on a :

- $X = \{v, s, a\}$
- $D_v = \{$ Biarritz, Chamonix, Deauville $\}$,
  $D_s = \{$ Escalade, Golf, Polo, Surf $\}$, $D_a = \{$ C, J, P, N $\}$,
- c_1 : contrainte ville/sport (v_1, r_1) avec $v_1 = \{v, s\}$ et r_1 définie par :

r_1	Escalade	Golf	Polo	Surf
Biarritz	0	1	0	1
Chamonix	1	1	0	0
Deauville	0	1	1	1

Exemple

- c_2 : contrainte ami/sport (v_2, r_2) avec $v_2 = \{a, s\}$ et r_2 définie par :

r_2	C	J	P	N
Escalade	1	0	1	0
Golf	0	1	0	1
Polo	0	1	1	0
Surf	0	0	0	1

- c_3 : contrainte ville/ami (v_3, r_3) avec $v_3 = \{a, v\}$ et r_3 définie par :

r_3	C	J	P	N
Biarritz	1	0	1	0
Chamonix	0	1	0	1
Deauville	0	0	1	1

CSP Binaires

Définition (Arité d'une contrainte)

Soit $c_i = (v_i, r_i)$ une contrainte d'un CSP. On appelle arité de c_i *la quantité $|v_i|$.*

Exemple :

- arité 2 : $x \neq y$, $x^2 + y^2 \leq 1$
- arité 3 : $x + y \leq z$, $\neg x \vee \neg y \vee z$
- arité n : $\sum_{i=1}^{n} x_i = 1$

Définition (CSP binaire)

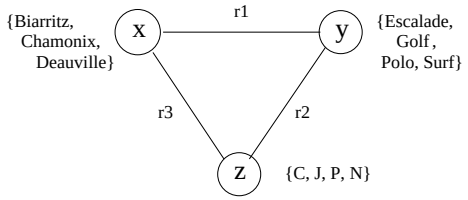
Un CSP $\mathcal{P} = (X, D, C)$ dont toutes les contraintes sont d'arité deux est appelé un CSP binaire.

Représentation graphique

CSP BINAIRES :

On représente le CSP par un graphe de contraintes $G = (X, C)$.

EXEMPLE :
Dans le pb de choix d'un stage de sport on a le graphe suivant :



La représentation graphique de CSP non-binaires est plus délicate.

Instanciation

Définition (Instanciation)

Etant donné un CSP $\mathcal{P} = (X, D, C)$, on appelle instanciation *d'un sous-ensemble de variables $Y \subseteq X$ une application i qui à tout élément $y \in Y$ associe un élément $i(y) \in D_y$.*

- Y s'appelle alors le *domaine de l'instanciation* i
- $i(y)$ s'appelle l'*instance* de la variable y
- on notera : $y \to i(y)$ le fait d'instancier y à la valeur $i(y)$
- Si $Y = X$, on dit que l'instanciation i est *complète* sinon on dit qu'elle est *partielle*.

EXEMPLE : $X = \{x, y, z\}$ et $D_x = D_y = D_z = \{1, 2, 3\}$, $C = (x \neq y, y \neq z, x \neq z)$. Si $Y = \{x, y\}$ alors $x \to 1, y \to 2$ fournit une instanciation partielle.

Consistance d'une instanciation

Définition (Satisfaction de contrainte)

Etant donné un CSP $\mathcal{P} = (X, D, C)$, une instanciation i de $Y \subseteq X$ satisfait la contrainte *$c_k = (v_k, r_k)$ de C (ce qui est noté $i \models c_k$) ssi $[v_k \subseteq Y$ et $i(v_k) \in r_k]$. Inversement, on dira que i* viole *c_k ssi $[v_i \subseteq Y$ et $i(v_k) \notin r_k]$.*

Définition (Instanciation localement consistante)

Etant donnée un CSP $\mathcal{P} = (X, D, C)$, une instanciation i de $Y \subseteq X$ est dite localement consistante *ssi :*
$\forall c_k = (v_k, r_k) \in C$ telle que $v_k \subseteq Y$ on a : $i \models c_k$

EXEMPLE : $X = \{x, y, z\}$ et $D_x = D_y = D_z = \{1, 2\}$, $C = (x \neq y, y \neq z, x \neq z)$. Si $Y = \{x, y\}$ alors $i(x) = 1, i(y) = 2$ est une instanciation localement consistante.

Solution d'un CSP

Définition (Solution d'un CSP)

On appelle solution *d'un CSP $\mathcal{P} = (X, D, C)$, une instanciation i consistante de X. L'ensemble des solutions de $\mathcal{P}$ sera noté $S_{\mathcal{P}}$.*

REMARQUE : Par abus de langage, on appelera également parfois *solution* le n-uplet $(i(x_1), \ldots, i(x_n))$.

Définition (Consistance d'un CSP)

Un CSP $\mathcal{P}$ est dit consistant ssi $S_{\mathcal{P}} \neq \emptyset$

EXEMPLE : Le problème du choix d'un stage de sport est consistant car il admet au moins une solution, par exemple : $(i(x) =$ Chamonix, $i(y) =$ Golf, $i(z) =$ J$)$.

Consistance globale

Définition (Instanciation globalement consistante)

Etant donné un CSP $\mathcal{P} = (X, D, C)$, une instanciation i de $Y \subset X$ est dite globalement consistante *ssi il existe une intanciation i' de $X \setminus Y$ telle que : $s = i' \circ i \in S_{\mathcal{P}}$.*

Autrement dit, une instanciation partielle des variables de X est globalement consistante si il est possible de la compléter en une solution du CSP.

Exemple : $X = \{x, y, z\}$ et $D_x = D_y = D_z = \{1, 2, 3\}$, $C = (x \neq y, y \neq z, x \neq z, y + z \leq 3)$. Si $Y = \{x, y\}$ on a :

- $i(x) = 3, i(y) = 1$ est globalement consistante ($i'(z) = 2$)
- $i(x) = 1, i(y) = 3$ n'est pas globalement consistante.

Equivalence de CSP

Définition (Equivalence de CSP)

Deux problèmes $\mathcal{P} = (X, D, C)$ et $\mathcal{P}' = (X, D', C')$ sont dits équivalents *(noté $\mathcal{P} \equiv \mathcal{P}'$) ssi $S_{\mathcal{P}} = S_{\mathcal{P}'}$.*

Définition (Contrainte induite)

Etant donné un CSP $\mathcal{P} = (X, D, C)$ et une contrainte $c = (v_c, r_c)$, on dira que c est induite par $\mathcal{P}$ ssi $\forall s \in S_{\mathcal{P}}, s \models c$

Exemple : Si $D_x = D_z \subset \mathbb{Z}$ et $D_y \subset \mathbb{N}$ alors $x < z$ est une contrainte induite par $x + y < z$.

Remarque : Etant donné un problème $\mathcal{P} = (X, D, C)$ et $c \notin C$ une contrainte induite par $\mathcal{P}$ alors $\mathcal{P}' = (X, D, C \cup \{c\})$ est équivalent à $\mathcal{P}$.

Consistance globale d'un CSP

Définition (Consitance globale d'un CSP)

Un CSP est globalement consistant *ssi toute instanciation localement consistante est globalement consistante.*

Exemple : $X = \{x, y, z\}$ et $D_x = D_y = D_z = \{1, 2, 3\}$, $C = (x \neq y, y \neq z, x \neq z, y + z \leq 4)$ n'est pas globalement consistant. Si $Y = \{x, y\}$ on a $i(x) = 1, i(y) = 3$ qui est localement consistant mais pas globalement consistant.

Remarque : Dans l'exemple précédent, si l'on avait $D_x = \{2, 3\}$ on aurait alors un CSP globalement consistant.

Problèmes de satisfaction de contraintes

Quelques problèmes classiques :

- prouver qu'un CSP est consistant
- exhiber toutes les solutions d'un CSP
- exhiber une solution qui maximise un ou plusieurs critères
- calculer ou estimer le nombre de solutions d'un CSP
- trouver, pour une ou plusieurs variables, un ensemble de valeurs qui figurent dans toutes les solutions.

II) Résolution d'un CSP

Engendrer et tester

NOTATIONS :
i instanciation courante ; X_i domaine d'instanciation de i

PROCEDURE EngTest(i, X, D, C)
si $X_i = X$ **alors**
  si i est consistante **alors** retourner *true* **sinon** *false* **fsi**
sinon
  choisir x_k dans $X \setminus X_i$
  faire pour tout $v \in D_{x_k}$
    si EngTest$(i \cup \{x_k \to v\}, X, D, C)$ **alors** retourner *true*
  fait
  retourner *false*
fsi

Appel : EngTest$(\emptyset, X, D, C)$

EngTest : Exemples et analyse

EXEMPLE : Simuler sur le problème du stage de sport

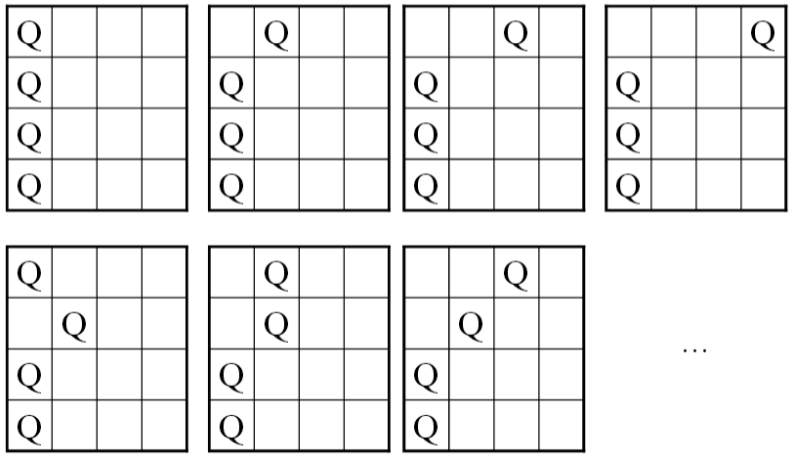
EXEMPLE : idem sur le problème des 4 dames

ANALYSE :

- on ne teste que la faisabilité de solutions complètement instanciées
- le nb de solutions complètes est très grand (pb du stage : 48 feuilles ; 4 dames : 3360 ; 8 dames $\sim 10^{16}$)
- pas de détection d'erreur quand une instanciation partielle viole une contrainte

  $\Rightarrow$ Très mauvaise procédure.

Engendrer et tester sur le pb 4-Queens



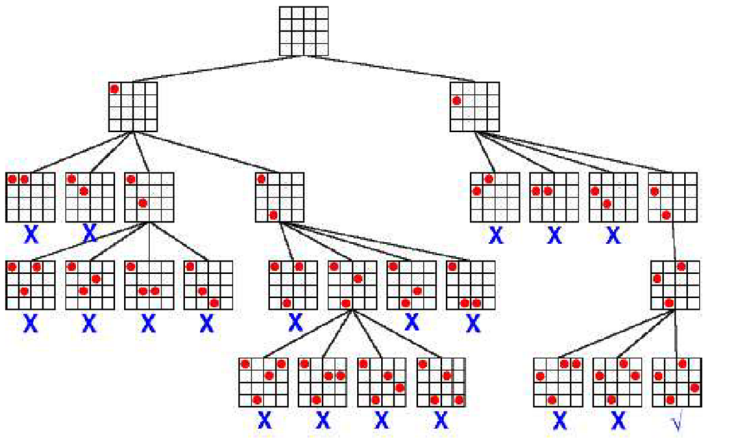
Retour Arrière Chronologique (Backtrack)

NOTATIONS :
i instanciation courante ; V liste de variables non-instanciées dans i

PROCEDURE RAC(i, V, D, C)
si $V = \emptyset$ **alors** retourner la solution i
sinon
choisir x_k dans V
faire pour tout $v \in D_{x_k}$
si $i \cup (x_k \to v)$ est localement consistante
alors RAC($i \cup \{x_k \to v\}, V \setminus \{x_k\}, D, C$)
fait
fsi

Appel : RAC($\emptyset, X, D, C$)

RAC sur le pb 4-Queens



Dessin tiré de *Online guide to Constraint Programming. R. Barták*

RAC : Exemples et analyse

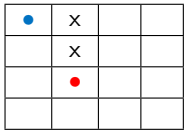
EXEMPLE : Simuler sur le problème du stage de sport

EXEMPLE : idem sur le problème des 4 dames

AVANTAGES :
- on backtrack souvent beaucoup plus tôt qu'avec EngTest
- permet d'éliminer des solutions potentielles par paquets

MAIS...
- on pourrait parfois s'apercevoir plus tôt qu'un sous-arbre ne contient plus de solution

4 dames :



Consistance locale

Idée : Consistance difficile à tester → consistances locales restreintes à des sous CSP (domaines réduits, sous-ensembles de variables).

Définition (k-consistance)
Un CSP est dit k-consistant si, pour toute instanciation i de $k-1$ variables $(x_1, \ldots, x_{k-1})$ il existe $v \in D_k$ tel que $i \cup \{x_k \to v\}$ est consistant.

EXEMPLE : 1-consistance (consistance aux sommets). $\mathcal{P}$ est 1-consistant si $\forall x_k \in X, \exists v \in D_k, \{x_k \to v\}$ est consistant.

Considérons par exemple $D_x = \{2, \ldots, 5\}$ et $x^2 - 7x + 10 > 0$ n'est pas 1-consistant.

k-consistance

EXEMPLE : 2-consistance. $\mathcal{P}$ est 2-consistant si $\forall x_i, x_j \in X$, $\forall v_i \in D_i, \exists v_j \in D_j$ telle que $\{x_i \to v_i, x_j \to v_j\}$ est consistant.

Considérons par exemple : $D_x = D_y = \{1, 2, 3, 4\}$ et $y > x + 1$.

N'est pas 2-consistant mais on peut rétablir la 2-consistance par réduction de domaines : $D'_x = \{1, 2\}$ et $D'_y = \{3, 4\}$.

Définition (k-consistance forte)

Un CSP $\mathcal{P}$ est dit fortement k-consistant si $\mathcal{P}$ est i-consistant pour tout $i = 1, \ldots, k$.

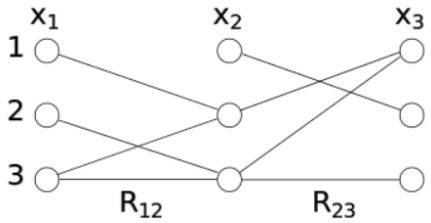
2-consistance forte = 1-consistance + 2-consistance

Arc-consistance d'une variable (cas de CSP binaires)

Définition (Arc-consistance d'une variable avec une autre variable)

Une variable x_k est arc-consistante avec une autre variable x_i pour une contrainte binaire $c_j = (v_j, r_j) \in C$ telle que $v_j = \{x_k, x_i\}$ si, pour tout $v \in D_k$ il existe $v' \in D_i$ telle que $(v, v') \in r_j$.

Exemple : CSP : variables x_1, x_2, x_3, domaines $D_i = \{1, 2, 3\}$, contraintes R_{12}, R_{23} (voir graphe ci-dessous)



Analyse de l'exemple

Dans l'exemple précédente

- x_2 est arc-consistant avec x_3 mais pas avec x_1 ($x_2 = 1$ ne laisse aucune valeur pour x_1).
- On peut rétablir l'arc-consistance en enlevant 1 du domaine de x_2 (supprime l'arc (1, 2) de R_{23}
- Mais alors x_3 n'est plus arc-consistant avec x_1

REMARQUE : cette définition s'étend à des contraintes n-aires (hyper-arc consistance).

Définition (Hyper-arc-consistance d'une var. avec une contr.)

Une variable x_k est hyper-arc-consistante *avec une contrainte $c_j = (v_j, r_j) \in C$ telle que $x_k \in v_j$ si $\forall v \in D_k$, il existe une instanciation i telle que $i(x_k) = v$ et $i \models r_j$.*

Hyper-Arc-consistance d'un CSP

Toute variable du CSP doit être (hyper)arc-consistante avec les contraintes qui portent sur elle. Plus précisément :

Définition ((Hyper)-arc-consistance d'un CSP)

Un CSP $\mathcal{P} = (X, D, C)$ est arc-consistant *si, pour toute variable $x_k \in X$ on a $D_k \neq \emptyset$ et $\forall v \in D_k, \forall c_j = (v_j, r_j) \in C$ tel que $x_k \in v_j$, $\exists i$ telle que $i(x_k) = v$ et $i \models r_j$.*

REMARQUE : Pour un CSP à contraintes binaires, l'hyper-arc consistance s'appelle l'arc-consistance (c'est alors équivalent à la 2 consistance introduite précédemment).

Comment obtenir un problème consistant (k-consistant ou arc consistant) à partir d'un problème qui ne l'est pas ?

Une solution : le filtrage

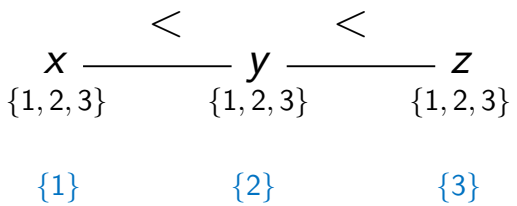
III) Filtrage

Filtrage

Définition (Filtrage)

Etant donné un CSP $\mathcal{P} = (X, D, C)$, constuire un problème $\mathcal{P}' = (X, D', C')$ avec $D \supseteq D'$ et $C \subseteq C'$ qui soit équivalent à $\mathcal{P}$ et soit consistant (k-consistant ou arc-consistant)

Exemple :



Procedure AC1

Procedure AC1

$C \leftarrow \{(x_i, x_j), i \neq j$ liées par une contrainte$\}$
répéter
  modification $\leftarrow$ false
  faire pour tout $(x_i, x_j) \in C$
    modification $\leftarrow$ REVISE$(x_i, x_j) \vee$modification
  fait
jusqu'à not(modification)

Procedure REVISE(x_i, x_j)

modification $\leftarrow$ false
faire pour chaque $v \in D_i$
  si $\nexists v' \in D_j \mid \{x_i \rightarrow v, x_j \rightarrow v'\}$ est consistante **alors**
  faire $D_i \leftarrow D_i \setminus \{v\}$
    modification $\leftarrow$ vrai
  fait
fait
retourner modification

AC3

Limite de AC1 : on répète l'application sur tous les couples de variables dans les deux sens mais à toute réduction de domaine on doit refaire une passe complète.

Idée de AC3 : utiliser une file L pour éviter l'application inutilement répétée de REVISE.

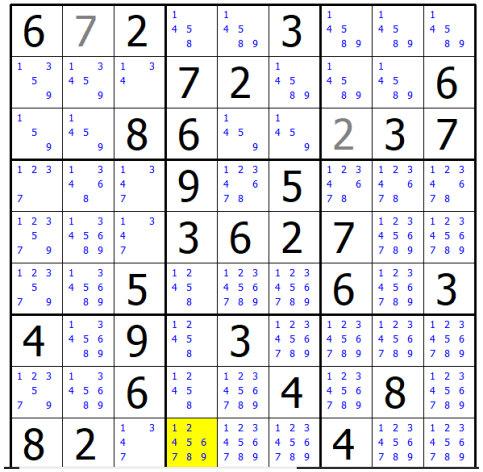
Procedure AC3

$L \leftarrow \{(x_i, x_j), i \neq j$ liées par une contrainte$\}$
tant que $L \neq \emptyset$
  choisir et supprimer dans L un couple (x_i, x_j)
  si REVISE(x_i, x_j) **alors**
    $L \leftarrow L \cup \{(x_k, x_i)$ / $\exists$ contrainte liant x_k et $x_i\}$
  fsi
ftq

Exemple : résolution de grilles de Sudoku

6		2			3			
			7	2				6
		8	6				3	7
			9		5			
			3	6	2	7		
		5				6		3
4		9		3				
		6			4		8	
8	2					4		

En cours d'application de AC3



Résolution après application de AC3

6	7	2	4	9	3	8	1	5
5	3	1	7	2	8	9	4	6
9	4	8	6	5	1	2	3	7
3	6	7	9	8	5	1	2	4
1	9	4	3	6	2	7	5	8
2	8	5	1	4	7	6	9	3
4	1	9	8	3	6	5	7	2
7	5	6	2	1	4	3	8	9
8	2	3	5	7	9	4	6	1

Limite du filtrage a priori

Procédure qui peut être couteuse en calculs et qui ne garantit pas a priori de pouvoir réduire les domaines et de simplifier la résolution.

EXEMPLE :
$X = (x_1, \ldots, x_n)$, $D_1 = D_2 = \ldots = D_n = \{1, 2, \ldots, n-1\}$,
$C =$ all-diff$(x_1, \ldots, x_n)$, i.e. $(\forall i, j,\ i \neq j \Rightarrow x_i \neq x_j)$

IDÉE : Utiliser le filtrage en cours de résolution plutôt qu'à priori.

Filtrage en cours de résolution

Forward Checking (FC)
Après chaque instanciation, on réduit par filtrage les domaines des variables directement liées par une contrainte à la variable que l'on vient d'instancier

EXEMPLE : simuler sur le problème des 4 dames

Procédure de Forward Checking

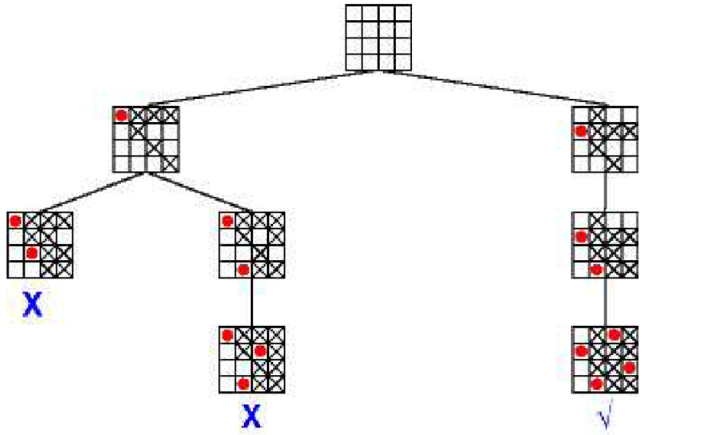
V : ens. des vars à instancier ; i : instanciation courante

PROCEDURE forward-checking(V, i)
si $V = \emptyset$ alors i est une solution
sinon
  choisir $x_k \in V$
  faire pour tout $v \in D_k$
    sauvegarde($V \setminus \{x_k\}$)
    si check-forward(x_k, v, V) alors
      forward-checking($V \setminus \{x_k\}, i \cup \{x_k \rightarrow v\}$)
    fsi
    restauration $V \setminus \{x_k\}$
  fait
fsi

La procédure check-forward

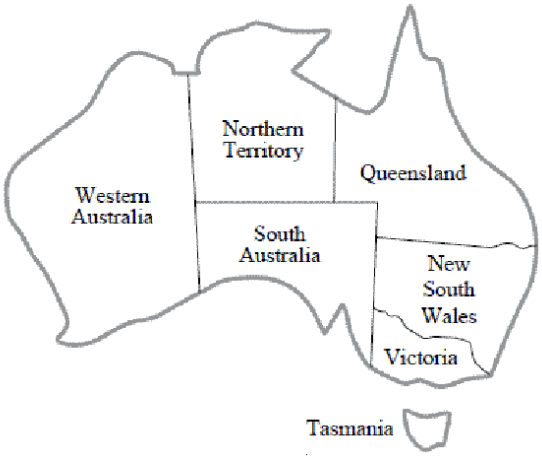
PROCEDURE check-forward(x_k, v, V)
consistant $\leftarrow$ true
pour chaque $x_j \in V \setminus \{x_k\}$ et tant que consistant
  faire pour chaque $v' \in D_j$
    si $\{x_k \rightarrow v, x_j \rightarrow v'\}$ non-consistant
      alors $D_j \leftarrow D_j \setminus \{v'\}$
    fsi
  fait
si $D_j = \emptyset$ alors consistant $\leftarrow$ false
retourner consistant

Forward Checking sur le pb des 4 dames



Dessin tiré de *Online guide to Constraint Programming. R. Barták*

Application du FC : coloriage de carte



Renforcer le filtrage

Full Look-Ahead (FLA)

Après chaque instanciation, réaliser une arc-cohérence complète sur les variables non-instanciées

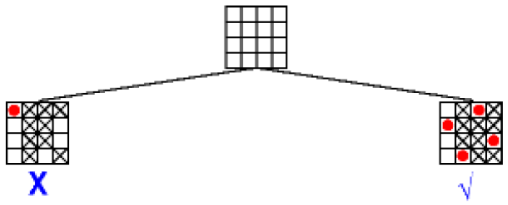
+ réduit les domaines encore mieux que le FC
- beaucoup plus cher en calculs

EXEMPLE : sur le problème des 4 dames

EXEMPLE : sur un problème de coloriage

EXEMPLE : comparer FC et FLA sur le problèmes suivant :
$X = \{x, y, z\}$ et $D_x = D_y = D_z = \{1, 2\}$,
$C = (x \neq y, y \neq z, x \neq z)$.

Full Look ahead sur le pb des 4 dames



Ordre et heuristiques (1)

- objectif : réduire la taille de l'espace exploré
- moyen : contrôle de l'ordre des opérations de choix de variables et d'instances
- outils : heuristiques visant à faire apparaître les echecs le plus tôt possible

Ordre d'instanciation des variables

EN STATIQUE :

- domaine min : variable de domaine de cardinalité minimale
- degré max : variable apparaissant dans le plus grand nombre de contraintes

EN DYNAMIQUE : domaine min après réduction des domaines pour garantir une condition de consistance.

Ordre et heuristique (2)

Ordre d'instanciation des valeurs

EN STATIQUE OU DYNAMIQUE :

- dépendant du domaine
- guidé par une fonction d'évaluation si le CSP est valué

Ordre de vérification des contraintes

- priorité aux contraintes les moins satisfiables

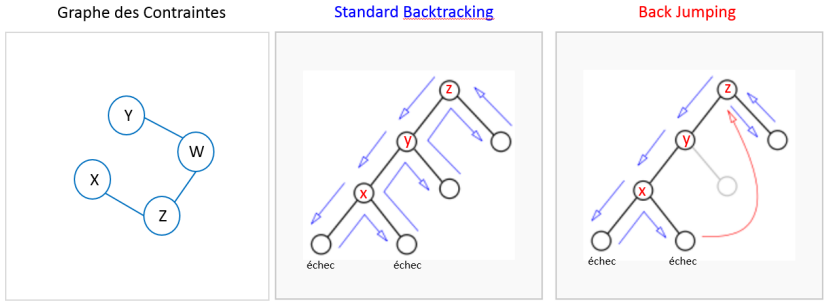
Améliorer les retours-arrière : le backjumping (BJ)

BJ evite un backtrack sur certaines variables dont l'instanciation n'est pas la cause de l'inconsistence locale pour aller directement remettre en question une instanciation susceptible de lever l'inconsistance.

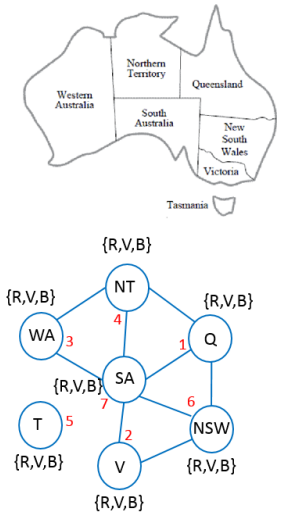
MISE EN OEUVRE

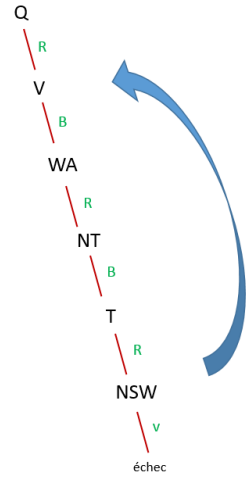
- on détermine un ensemble de variables en conflit avec la dernière variable instanciée. On en déduit un ensemble de conflit.
- BJ backtrack sur la variable la plus récemment instanciée dans l'ensemble de conflit de la variable qui vient de créer l'inconsistance locale.

Illustration du backjumping



Un exemple de backjumping





Conflict BackJumping (CBJ)

Calculer, lors de charque retour arrière, un ens. de variables dites "en conflit" dont les valeurs prises dans l'instanciation courante i suffisent à expliquer le retour arrière. Cela permet de reconsidérer immédiatement la valeur de la variable en conflit la plus basse dans l'ordre d'instanciation.

Mise en oeuvre

- Contrairement au Backjumping qui s'applique uniquement aux feuilles en échec, le CBJ s'applique aussi à des noeuds intermédiaires
- pour chaque variable x on définit un ensemble de conflits $C(x)$
- quand l'affectation $x \to a$ est localement inconsistante à cause d'une violation de contrainte impliquant une variable y on ajoute y dans $C(x)$
- quand on a plus de valeurs possible dans le domaine de la variable courante à instancier, CBJ backjump à la dernière variable w instanciée dans $C(x)$ (comme le BackJumping) mais en plus on révise $C(w)$ en faisant $C(w) \leftarrow C(w) \cup C(x)$. Ainsi d'autres backjump peuvent apparaître à partir de w.

CBJ

CBJ fait appel à la fonction consistante qui reçoit une instanciation et retourne l'ens des variables de la contrainte violée si i est inconsistante.

Procedure CBJ(V, i)
si $V = \emptyset$ **alors** i est solution, retourner $\emptyset$
sinon
  choisir $x_k \in V$, conflit $\leftarrow \emptyset$, nonBJ $\leftarrow$ true
  faire pour chaque $v \in D_k$ et tant que nonBJ
    conflit-local $\leftarrow$ consistante($i \cup \{x_k \to v\}$)
    si conflit-local $= \emptyset$ **alors**
      conflit-fils $\leftarrow$ CBJ($V \setminus \{x_k\}, i \cup \{x_k \to v\}$)
      si $x_k \in$ conflit-fils alors conflit $\leftarrow$ conflit $\cup$ conflit-fils
      sinon conflit $\leftarrow$ conflit-fils ; nonBJ $\leftarrow$ false **fsi**
    sinon conflit $\leftarrow$ conflit $\cup$ conflit-local
  fait
  retourner conflit
fsi

IV) traitement de contraintes spécifiques

Contraintes all-diff

se pose dans beaucoup de problèmes, e.g. sudoku, carrés magiques

une contrainte all-diff($x_1, \ldots, x_n$)
$\neq n(n-1)/2$ contraintes binaires d'inégalité

Exemple : all-diff(X, Y, Z) avec $D_X = \{1, 2\}, D_Y = \{1, 2\}, D_Z = \{1, 2\}$

On peut calculer :

- nv : nb de variables non-instanciées
- R : l'ensemble des valeurs restantes dans les domaine des variables non-instanciées

Si $nv > |R|$ alors il n'y a pas de solution
Dans l'exemple $nv = 3$ et $|R| = 2$

traitement de alldiff (1)

c : contraite all-diff

PROCEDURE all-diff-consist(c, D)
tant que $\exists v \in V$ tel que $D_v = \{d\}$ pour un d quelconque
    $V \leftarrow V \setminus \{v\}$
    pour tout $v' \in V$ si $d \in D_{v'}$ faire $D_{v'} \leftarrow D_{v'} \setminus \{d\}$
ftq
$nv \leftarrow |V|$
$R \leftarrow \emptyset$
faire $R \leftarrow R \cup D_v$ pour tout $v \in V$
si $nv > |R|$ **alors** retourner false **fsi**

traitement de alldiff (2)

Limite de l'algorithme précédent :
EXEMPLE : all-diff(X, Y, Z, T) avec
$D_X = \{1, 2\}, D_Y = \{1, 2\}, D_Z = \{1, 2\}, D_T = \{2, 3, 4, 5\}$
$nv = 4$ et $|R| = 5$ solution ?

non...

Pour faire mieux :

1. calculer le couplage maximal dans un graphe biparti (e.g. par un algorithme de flot)
2. si $cmax$ est la valeur du couplage maximal (flot max) alors pas de solution dès que $nv > cmax$

Contraintes arithmétiques non-binaires

EXEMPLE : $x = 3y + 5z$
avec $D_x = \{2, 3, 4, 5, 6, 7\}, D_y = \{0, 1, 2\}, D_z = \{-1, 0, 1, 2\}$

tester l'arc consitance revient à chercher à résoudre :
$3y + 5z = 2$, $3y + 5z = 3 \ldots x = 3y + 10$
et voir si toutes ces équations sont solubles...

Ici le filtrage donnerait :
$D'_x = \{3, 5, 6\}, D'_y = \{0, 1, 2\}, D'_z = \{0, 1\}$

les tests deviennent prohibitifs dans l'exemple suivant :
EXEMPLE : $12x + 107y - 17z + 5t = 2$ avec pour domaine des variables $\{0, 1, \ldots, 1000\}$

CSP arithmétique

Définition
Un CSP arithmétique est un CSP dont chaque variable a un domaine fini d'entiers et dont les contraintes sont des contraintes arithmétiques

EXEMPLE : sytemes d'équation linéaires en nb entiers
EXEMPLE : problème des n dames

Définition
Un intervalle $[l..u]$ *est l'ensemble d'entiers* $\{l, l + 1, \ldots, u\}$ *si* $l \leq u$ *et l'ensemble vide sinon. Pour tout domaine* D_x *d'une variable* x*, on note* $\min(D_x)$ *et* $\max(D_x)$ *ses éléments minimaux et maximaux respectivement.*

Consistance de bornes

Définition

Dans un CSP arithmétique avec des variables x_i de domaines $D_i = [l_i, u_i]$ une contrainte $c = (v, r)$ vérifie la consistante de bornes *par rapport à une variable $x_i \in v$ si et seulement si :*
$\exists (b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n) \in D_1 \times \ldots \times D_n,\ r(b_1, \ldots, b_{i-1}, l_i, b_{i+1}, \ldots, b_n)$
$\exists (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \in D_1 \times \ldots \times D_n,\ r(a_1, \ldots, a_{i-1}, u_i, a_{i+1}, \ldots, a_n)$

Définition

La contrainte $c = (v, r)$ est dite consistante de bornes *si et seulement si elle est consistante de bornes par rapport à toutes les variables de v.*

Consistance de bornes pour un CSP

Définition

Un CSP vérifie la consistance de bornes *si et seulement si toutes les contraintes qui le composent sont consistantes de borne.*

Exemple : $D_x = [2..7], D_y = [0..2], D_z = [-1..2]$ avec $x - 3y = 5z$ ne vérifie pas la consistance de bornes.

Exemple : $D_x = [2..7], D_y = [0..2], D_z = [0..1]$ avec $x - 3y = 5z$ vérifie la consistance de bornes.

on peut rétablir la consistance par réduction des domaines.

Cas de la contrainte $x = y + z$

$x = y + z \quad y = x - z \quad z = x - y$

$$x \geq \min(D_y) + \min(D_z) \qquad x \leq \max(D_y) + \max(D_z)$$
$$y \geq \min(D_x) - \max(D_z) \qquad y \leq \max(D_x) - \min(D_z)$$
$$z \geq \min(D_x) - \max(D_y) \qquad z \leq \max(D_x) - \min(D_y)$$

Algorithme

Procedure Consistance-bornes-add(D)

$x_{\min} \leftarrow \max\{\min(D_x), \min(D_y) + \min(D_z)\}$
$x_{\max} \leftarrow \min\{\max(D_x), \max(D_y) + \max(D_z)\}$
$D_x \leftarrow \{d \in D_x : x_{\min} \leq d \leq x_{\max}\}$
$y_{\min} \leftarrow \max\{\min(D_y), \min(D_x) - \max(D_z)\}$
$y_{\max} \leftarrow \min\{\max(D_y), \max(D_x) - \min(D_z)\}$
$D_y \leftarrow \{d \in D_y : y_{\min} \leq d \leq y_{\max}\}$
$z_{\min} \leftarrow \max\{\min(D_z), \min(D_x) - \max(D_y)\}$
$z_{\max} \leftarrow \min\{\max(D_z), \max(D_x) - \min(D_y)\}$
$D_z \leftarrow \{d \in D_z : z_{\min} \leq d \leq z_{\max}\}$

Exemple

EXEMPLE : $D_x = [4..8], D_y = [0..3], D_z = [2..2]$ avec $x = y + z$
vérification de la consistance de bornes.

$2 \leq x \leq 5 \quad 2 \leq y \leq 6 \quad 1 \leq z \leq 8$

$D_x = [4..5], D_y = [2..3], D_z = [2..2]$

$4 \leq x \leq 5 \quad 2 \leq y \leq 3 \quad 1 \leq z \leq 3$

$D_x = [4..5], D_y = [2..3], D_z = [2..2]$

Cas de contraintes linéaires plus complexes

$$4w + 3p + 2c \leq 9$$

ce qui induit :

$$w \leq \frac{9}{4} - \frac{3}{4}p - \frac{2}{4}c, \quad p \leq \frac{9}{3} - \frac{4}{3}w - \frac{2}{3}c, \quad c \leq \frac{9}{2} - 2w - \frac{3}{2}p$$

et donc les inégalités suivantes :

$$\begin{aligned} w &\leq \frac{9}{4} - \frac{3}{4}\min(D_p) - \frac{2}{4}\min(D_c) \\ p &\leq \frac{9}{3} - \frac{4}{3}\min(D_w) - \frac{2}{3}\min(D_c) \\ c &\leq \frac{9}{2} - 2\min(D_w) - \frac{3}{2}\min(D_p) \end{aligned}$$

Algorithme

PROCEDURE CONSISTANCE-BORNES-INEQ(D)

$w_{\max} \leftarrow \min\{\max(D_w), \lfloor \frac{9}{4} - \frac{3}{4}\min(D_p) - \frac{2}{4}\min(D_c) \rfloor\}$
$D_w \leftarrow \{d \in D_w : d \leq w_{\max}\}$
$p_{\max} \leftarrow \min\{\max(D_p), \lfloor \frac{9}{3} - \frac{4}{3}\min(D_w) - \frac{2}{3}\min(D_c) \rfloor\}$
$D_p \leftarrow \{d \in D_p : d \leq p_{\max}\}$
$c_{\max} \leftarrow \min\{\max(D_c), \lfloor \frac{9}{2} - \frac{2}{\min}(D_w) - \frac{3}{2}\min(D_p) \rfloor\}$
$D_c \leftarrow \{d \in D_c : d \leq c_{\max}\}$

EXEMPLE : Si $D_w = [0..9], D_p = [0..9], D_c = [0..9]$ alors on a :
$w \leq \frac{9}{4}, p \leq \frac{9}{3}, c \leq \frac{9}{2}$

et donc : $D_w = [0..2], D_p = [0..3], D_c = [0..4]$

Contrainte $x = \min(y, z)$

On a les inégalités suivantes :

$$\begin{aligned} y &\geq \min(D_x) \\ z &\geq \min(D_x) \\ x &\geq \min\{\min(D_y), \min(D_z)\} \\ x &\leq \min\{\max(D_y), \max(D_z)\} \end{aligned}$$

Contrainte $x = y \times z$

On a les inégalités suivantes :

$$\begin{aligned} x &\geq \min(D_y) \times \min(D_z) \\ x &\leq \max(D_y) \times \max(D_z) \\ y &\geq \min(D_x) / \max(D_z) \\ y &\leq \max(D_x) / \min(D_z) \\ z &\geq \min(D_x) / \max(D_y) \\ z &\leq \max(D_x) / \min(D_y) \end{aligned}$$

Exemple : Si $D_x = [1..4], D_y = [1..2], D_z = [1..4]$ alors on a :

$$x \geq 1, \quad x \leq 8, \quad y \geq 1, \quad y \leq 4, \quad z \geq 1, \quad z \leq 4$$

ce qui laisse les domaines invariants.