

Contenu cours 9

Contraintes réifiées

Optimisation

Machines de recherche en général

Contraintes réifiées

- ▶ Permettent d'exprimer des combinaisons entre contraintes au delà de la conjonction : négation, équivalence, implication, disjonction.
- ▶ Permettent d'exprimer des contraintes qui sont « souples » (angl. : soft constraints), en opposition aux contraintes dures vues jusqu'à maintenant qui doivent impérativement être satisfaites.
- ▶ Cela permet par exemple d'exprimer qu'on veut satisfaire un nombre maximal de contraintes souples.
- ▶ Les contraintes souples sont nécessaires pour modéliser des problèmes « sur-contraintes » (angl. : over-constrained).

Le principe

- ▶ La *réification* d'une contrainte C correspond à la formule logique suivante :

$$(C \leftrightarrow x = 1) \quad \text{avec} \quad D(x) = \{0, 1\}$$

où $x \notin \text{vars}(C)$.

- ▶ On appelle souvent une variable avec domaine $[0,1]$ une *variable 0/1* ou une *variable booléenne*.
- ▶ Contrainte *pseudo-boléenne* : les variables sont booléennes, mais on les utilise pour former une expression dont les valeurs peuvent être des entières non booléennes. Exemple :

$$X_1 + X_2 + X_3 + X_4 \leq 2$$

exprime qu'au plus deux des variables sont vraies (supposant que toutes ces variables sont booléennes)

Propagateurs pour contraintes réifiées

$$(C \leftrightarrow x = 1) \quad \text{où} \quad D(x) = \{0, 1\}$$

- ▶ Si la mémoire implique $x = 1$ alors le propagateur devient un propagateur pour C
- ▶ Si la mémoire implique $x = 0$ alors le propagateur devient un propagateur pour $\neg C$
- ▶ Si la mémoire implique C alors imposer $x = 1$, puis le propagateur disparaît.
- ▶ Si la mémoire n'est pas consistante avec C alors imposer $x = 0$, puis le propagateur disparaît.

Propagateurs pour des contraintes réifiées

- ▶ Voir le document « System Modules », section 5.10 « Reified Constraints »
- ▶ Quelques exemples :
 - ▶ $(X <: Y) = B$
 - ▶ $(X + Y + Z =: 0) = B$
 - ▶ $(X \setminus =: Y) = B$
 - ▶ $(X : : 0 \# 10) = B$
 - ▶ `{FD.reified.distance X Y '=' Z B}`

Exemples pour des connecteurs booléens

- ▶ `{FD.conj X Y Z}` pour $(X \wedge Y) = Z$
- ▶ `{FD.disj X Y Z}` pour $(X \vee Y) = Z$
- ▶ `{FD.impl X Y Z}` pour $(X \rightarrow Y) = Z$
- ▶ `{FD.equi X Y Z}` pour $(X \leftrightarrow Y) = Z$
- ▶ `{FD.nega X Y}` pour $(\neg X) = Y$

Exemple : la photo de groupe

- ▶ Prendre une photo du groupe Betty, Chris, Donald, Fred, Gary, Mary, and Paul (tous placés un à côté de l'autre)
- ▶ Betty veut être à côté de Gary et Mary
- ▶ Chris veut être à côté de Betty et Gary.
- ▶ Fred veut être à côté de Mary et Donald.
- ▶ Paul veut être à côté de Fred et Donald.

« Branch And Bound »

- ▶ On lance une machine de recherche spécialisée avec deux arguments :
 1. le script,
 2. une fonction de comparaison entre deux solutions du script.
- ▶ La machine va chercher une solution du script qui est optimale (ici : maximale)
- ▶ Fonction `SearchBest`
- ▶ Procédure `ExploreBest`

Implémentation de Branch And Bound

Autres techniques : Recherche locale

- ▶ Chaque fois qu'une solution X est trouvée, on ajoute dans les autres nœuds de l'arbre de recherche un propagateur pour le fait que la solution doit être meilleure que X .

- ▶ On part d'une solution qu'on cherche à améliorer.
- ▶ Pour toute affectation il y a une notion de voisinage, et une fonction de mesure.
- ▶ Tant que possible on passe d'un point au meilleur de ses voisins (Hill Climbing).
- ▶ Il est possible qu'on reste bloqué sur un optimum local.

Recherche locale : exemple

Recherche locale : exemple

- ▶ Problème de n dames ($n = 5$)
- ▶ Affectations : associer à chaque ligne une colonne.
- ▶ On cherche à minimiser le nombre de paires de dames qui se menacent.
- ▶ Les voisins d'une affectations sont obtenues par échangeant deux lignes.

1			♔		
2		♔			
3	♔				
4				♔	
5					♔

Coût : 6. Échanger lignes 2 et 3.

Recherche locale : exemple

1			👑		
2	👑				
3		👑			
4				👑	
5					👑

Coût : 2. Échanger lignes 2 et 5.

Recherche locale : exemple

1			👑		
2					👑
3		👑			
4				👑	
5	👑				

Coût : 0. Minimum atteint (local et global).

Recherche locale : exemple

1			👑		
2	👑				
3					👑
4				👑	
5		👑			

Coût : 1. Minimum local, mais pas minimum global.

Autres techniques : Simulated Annealing

- ▶ Annealing : recuit (métallurgie).
- ▶ Technique probabiliste.
- ▶ Recherche locale, mais peut aussi avec une certaine probabilité passer à des solutions moins bonnes.
- ▶ Cette probabilité dépend de la « température » qui décroît pendant le processus.

Machines de recherche générales

- ▶ Voir le document « System Modules », section 4.2 « General Search Engines »
- ▶ Procédures comme `{Search.one.depth +ScriptP +RcdI ?KillP ?Xs}`.
Les arguments sont :
 1. le script
 2. « recomputation distance » : des valeurs > 1 font économiser de la place mais prennent plus de temps dans la construction de l'arbre de recherche
 3. une procédure qu'on pourra appeler pour « tuer » la recherche
 4. le résultat de la recherche.

Exemple d'utilisation

```
declare R K
in
{Browse R}
{Search.one.depth {Triangle 8} 1 K R}

{K} % <- tuer la recherche
```