Chapitre 9

CSP : représentation et résolution

Il s'agit là-aussi d'une représentation à partir de graphes mais avec des graphes plus spécifiques car représentant exclusivement des contraintes. Du coup, les algorithmes de résolution seront des variantes spécialisées des algorithmes de parcours vus précédemment.

9.1 Definition

Rappelons que dans les problèmes de recherche "classiques", on a les éléments suivants :

- Un *état* est une "boite noire" et
- cette "boite noire" correspond à n'importe quelle structure de données contenant a minima un *test pour le but*, une fonction d'*évaluation* et une fonction *successeur*.

Par opposition, dans les CSP, on a :

- Un *état* est défini par un ensemble de variables V_i, dont les valeurs appartiennent au domaine D_i ;
- le *test pour le but*, la fonction d'*évaluation* et la fonction *successeur* sont définies à l'aide d'un ensemble de contraintes qui spécifient les combinaisons autorisées pour les valeurs sur des sous-ensembles de variables, sachant qu'on cherche au final à affecter une valeur à chaque variable en respectant les contraintes.

Il existe une grande variété de CSPs :

- suivant le type de variable : discrète ou continue,
- suivant le type de domaine : fini ou infini,
- suivant le type de contrainte : linéaire ou pas, unaire, binaire ou n-aire, représentant des contraintes strictes ou des préférences, ...

Dans ce cours, on va se contenter de CSP manipulant des variables discrètes à domaines finis et des contraintes strictes.

Définition 15 (CSP) *Un CSP est un triplet (V, D, C) avec :*

- *V l'ensemble des variables $\{V_1, V_2, V_3, \ldots\}$,*
- *D l'ensemble des domaines de valeurs, au plus un par variable, $\{D_1, D_2, D_3, \ldots\}$ (les valeurs de V_i étant prises dans le domaine D_i),*
- *C l'ensemble des contraintes entre variables qui définissent des tuples possibles de valeurs sur les domaines, chaque contrainte étant notée $C_{ij\ldots} = \{(x_i, y_j, ...) \in D_i \times D_j \times \ldots\}$*

Quand on modélise un problème sous la forme d'un CSP, la difficulté principale est de choisir les bonnes variables car cela influe fortement sur l'expression des contraintes et donc sur l'efficacité de résolution.

9.2 Algorithmes

L'algorithme de recherche standard correspond à une recherche incrémentale :

- Les états sont définis par les valeurs des variables déjà affectées.
- Etat initial : Un ensemble d'affectations vide $\varnothing$.
- Fonction successeur : attribuer une valeur à une variable non encore affectée, de façon cohérente (par rapport aux contraintes) à l'affectation actuelle.
- Test du but : l'affectation courante est complète.

Cet algorithme de recherche marche pour tous les CSPs. Chaque solution apparaît à une profondeur de n s'il y a n variables ; cela revient à utiliser le principe de la recherche en profondeur d'abord.

Il existe beaucoup de variantes de cet algorithme standard. Nous nous contenterons de voir ici la variante la plus simple, appelée *backtrack*, et une de ses améliorations (le *backtrack avec ordonnancement*).

De manière formelle, cela correspond à l'algorithme 6 qui utilise à son tour l'algorithme 7.

Algorithme 6 : Backtrack

Données :
C : ensemble des contraintes
D : matrice des domaines (le domaine de la variable i est la ligne i de D)
n : nombre de variables

Variables locales :
A : affectation courante de valeurs
p : numéro de la variable courante
D' : copie de D qui va évoluer au fur et à fur du déroulement de l'algo
ok : résultat de la tentative d'affectation courante

début
- $p = 1$ // numéro première variable traitée (et aussi indice de profondeur)
- $A = \varnothing$ // affectation de valeurs vide au départ
- $D'[1]$ = copie de $D[1]$ // copie du domaine de la variable courante
- **tant que** $1 \leq p \leq n$ **faire**
  - ok = SelectValeur(A,$D'[p]$,p,C) // essai d'affectation d'une valeur à la variable p
  - **si** *not ok* **alors**
    - $p = p - 1$ // cela n'a pas marché, on "backtrack"
    - on enlève de A l'affectation concernant p // nettoyage de A
  - **sinon**
    - $p = p + 1$ // cela a marché, on passe à la variable suivante
    - **si** $p \leq n$ **alors**
      - $D'[p]$ = copie de $D[p]$ // en faisant une copie de son domaine
- **si** $p == 0$ **alors**
  - print(ECHEC : CSP incohérent)
- **sinon**
  - **retourner** A

Algorithme 7 : SelectValeur

Données :

A : affectation courante de valeurs
Dp : domaine courant de la variable p
p : la variable pour laquelle on cherche une valeur
C : ensemble des contraintes

Variables locales :

v : valeur courante

// Attention : la donnée Dp est modifiée par l'appel de cet algorithme
// (toutes les valeurs choisies et incohérentes avec l'affectation courante ont été
// supprimées de Dp)
//
// Attention : la donnée A peut être modifiée par l'appel de cet algorithme
// si on trouve une valeur cohérente pour p

début

  tant que $Dp \neq \varnothing$ **faire**
    choisir $v \in Dp$ // choix d'une valeur pour p
    supprimer v de Dp // mise à jour de Dp
    si $A \cup (p, v)$ *est cohérente par rapport à* C **alors**
      $A = A \cup (p, v)$ // on rajoute l'affectation de p avec v à A
      retourner *VRAI* // on a trouvé une valeur pour p
  retourner *FAUX* // on n'a pas trouvé de valeur pour p, A reste inchangée

Une version améliorée de l'algorithme de backtrack consiste à ordonner au préalable les variables. Un critère d'ordonnancement classique est le degré de cette variable dans le graphe des contraintes. En effet, plus une variable est impliquée dans des contraintes et plus le choix de sa valeur risque d'impacter d'autres variables. Cela donne l'algorithme de backtrack avec ordonnancement (voir l'algorithme 8).

Algorithme 8 : Backtrack avec ordonnancement

Données :

C : ensemble des contraintes
D : matrice des domaines (le domaine de la variable i est la ligne i de D)
n : nombre de variables

début

  ré-ordonner D en fonction du degré décroissant des variables dans le graphe des contraintes
  retourner *Backtrack(C,D,n)*

Notons que d'autres choix d'ordonnancement peuvent être faits (prise en compte de la taille des domaines par exemple). On peut aussi avoir des critères de choix de la valeur à assigner (la moins contraignante par exemple).

Et enfin, on peut aussi utiliser la propagation de contraintes pour anticiper l'impact d'une affectation. Cela consiste à propager le choix d'une valeur sur les domaines des variables non encore affectées. Cela peut se faire à chaque étape du backtrack. Cette variante est appelée le *forward checking*.

9.3 Exemples

Exemple 1 page 4 (cont'd) *Reprenons le problème de la coloration et codons-le maintenant sous la forme d'un CSP.*

> *Les variables correspondent aux sommets du graphe.*
> *Les domaines des variables sont tous identiques et égaux à la liste des couleurs possibles.*
> *Les contraintes sont données ici par les arêtes entre deux sommets.*

Dans le cas du graphe $G = (X, E)$ avec $X = \{a, b, c\}$ et $E = \{(a, b), (b, c), (c, a)\}$, considérons d'abord le problème Pb_1 : Peut-on colorer G avec 3 couleurs ?

Le CSP correspondant est défini par :

- $V = X$
- $D = \{D_a, D_b, D_c\}$ *avec* $D_i = \{R, B, J\}$ *pour* $i = a, b, c$
- $C = \{C_{ab}, C_{bc}, C_{ca}\}$ *avec* $C_{ij} = \{(R, B), (R, J), (B, R), (B, J), (J, R), (J, B)\}$ *donnant la liste des doublons de valeurs autorisés pour les variables i et j quand $i \neq j$. Notons qu'on pourrait aussi choisir de ne mémoriser dans C_{ij} que les doublons interdits.*

Appliquons maintenant le backtrack pour résoudre le problème Pb_1.

$p = 1$ *(1 étant le numéro de la variable a)* $A = \varnothing$ *(initialisation de A)* $D'[1] = \{R, B, J\}$

p étant $\leq$ *à 3 (nb de variables) on rentre dans la boucle* *appel de SelectValeur avec* $p = 1$ *et* $D'[1] = \{R, B, J\}$ *on essaye la valeur R pour 1 et* $D'[1] = \{B, J\}$ *ce choix d'affectation est cohérent avec A (qui est vide pour l'instant)* *on met à jour A qui devient* $\{(1, R)\}$ *on sort du SelectValeur en renvoyant VRAI* *ok est VRAI donc* $p = 2$ *(2 étant le numéro de la variable b)* *et* $D'[2] = \{R, B, J\}$

p étant $\leq$ *à 3 (nb de variables) on rentre dans la boucle* *appel de SelectValeur avec* $p = 2$ *et* $D'[2] = \{R, B, J\}$ *on essaye la valeur R pour 2 et* $D'[2] = \{B, J\}$ *ce choix d'affectation est incohérent avec* $A = \{(1, R)\}$ *on essaye la valeur B pour 2 et* $D'[2] = \{J\}$ *ce choix d'affectation est cohérent avec* $A = \{(1, R)\}$ *on met à jour A qui devient* $\{(1, R), (2, B)\}$ *on sort du SelectValeur en renvoyant VRAI* *ok est VRAI donc* $p = 3$ *(3 étant le numéro de la variable c)* *et* $D'[3] = \{R, B, J\}$

p étant $\leq$ à 3 (nb de variables) on rentre dans la boucle
appel de SelectValeur avec $p = 3$ et $D'[3] = \{R, B, J\}$
on essaye la valeur R pour 3 et $D'[3] = \{B, J\}$
ce choix d'affectation est incohérent avec $A = \{(1, R), (2, B)\}$
on essaye la valeur B pour 3 et $D'[3] = \{J\}$
ce choix d'affectation est incohérent avec $A = \{(1, R), (2, B)\}$
on essaye la valeur J pour 3 et $D'[3] = \varnothing$
ce choix d'affectation est cohérent avec $A = \{(1, R), (2, B)\}$
on met à jour A qui devient $\{(1, R), (2, B), (3, J)\}$
on sort du SelectValeur en renvoyant VRAI
ok est VRAI donc $p = 4$

Arrêt de la boucle puisque $p = 4$ et sortie en renvoyant A.

Remarquons ici deux choses : nous n'avons pas fait de backtrack et l'utilisation du backtrack avec ordonnancement n'aurait rien apporté ici puisque tous les sommets du graphe G ont le même degré.

Si on veut traiter maintenant le problème Pb_2, "Peut-on colorer G avec 2 couleurs ?", les domaines seront différents ($D_i = \{R, B\}$ pour $i = a, b, c$) et les contraintes modifiées en conséquence ($C_{ij} = \{(R, B), (B, R)\}$).

Et l'application du backtrack va donner la chose suivante.

$p = 1$ (1 étant le numéro de la variable a)
$A = \varnothing$ (initialisation de A)
$D'[1] = \{R, B\}$

p étant $\leq$ à 3 (nb de variables) on rentre dans la boucle
appel de SelectValeur avec $p = 1$ et $D'[1] = \{R, B\}$
on essaye la valeur R pour 1 et $D'[1] = \{B\}$
ce choix d'affectation est cohérent avec A (qui est vide pour l'instant)
on met à jour A qui devient $\{(1, R)\}$
on sort du SelectValeur en renvoyant VRAI
ok est VRAI donc $p = 2$ (2 étant le numéro de la variable b)
et $D'[2] = \{R, B\}$

p étant $\leq$ à 3 (nb de variables) on rentre dans la boucle
appel de SelectValeur avec $p = 2$ et $D'[2] = \{R, B\}$
on essaye la valeur R pour 2 et $D'[2] = \{B\}$
ce choix d'affectation est incohérent avec $A = \{(1, R)\}$
on essaye la valeur B pour 2 et $D'[2] = \varnothing$
ce choix d'affectation est cohérent avec $A = \{(1, R)\}$
on met à jour A qui devient $\{(1, R), (2, B)\}$
on sort du SelectValeur en renvoyant VRAI
ok est VRAI donc $p = 3$ (3 étant le numéro de la variable c)
et $D'[3] = \{R, B\}$

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 3$ *et* $D'[3] = \{R, B\}$
on essaye la valeur R *pour* 3 *et* $D'[3] = \{B\}$
ce choix d'affectation est incohérent avec $A = \{(1, R), (2, B)\}$
on essaye la valeur B *pour* 3 *et* $D'[3] = \varnothing$
ce choix d'affectation est incohérent avec $A = \{(1, R), (2, B)\}$
$D'[3]$ *est vide on sort du SelectValeur en renvoyant FAUX*
ok est FAUX donc $p = 2$, $A = \{(1, R)\}$ $\Rightarrow$ ***backtrack***

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 2$ *et* $D'[2] = \varnothing$
on sort du SelectValeur en renvoyant FAUX
ok est FAUX donc $p = 1$, $A = \varnothing$ $\Rightarrow$ ***backtrack***

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 1$ *et* $D'[1] = \{B\}$
on essaye la valeur B *pour* 1 *et* $D'[1] = \varnothing$
ce choix d'affectation est cohérent avec A *(qui est vide pour l'instant)*
on met à jour A *qui devient* $\{(1, B)\}$
on sort du SelectValeur en renvoyant VRAI
ok est VRAI donc $p = 2$ *(2 étant le numéro de la variable b)*
et $D'[2] = \{R, B\}$

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 2$ *et* $D'[2] = \{R, B\}$
on essaye la valeur R *pour* 2 *et* $D'[2] = \{B\}$
ce choix d'affectation est cohérent avec $A = \{(1, B)\}$
on met à jour A *qui devient* $\{(1, B), (2, R)\}$
on sort du SelectValeur en renvoyant VRAI
ok est VRAI donc $p = 3$ *(3 étant le numéro de la variable c)*
et $D'[3] = \{R, B\}$

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 3$ *et* $D'[3] = \{R, B\}$
on essaye la valeur R *pour* 3 *et* $D'[3] = \{B\}$
ce choix d'affectation est incohérent avec $A = \{(1, B), (2, R)\}$
on essaye la valeur B *pour* 3 *et* $D'[3] = \varnothing$
ce choix d'affectation est incohérent avec $A = \{(1, B), (2, R)\}$
$D'[3]$ *est vide, on sort du SelectValeur en renvoyant FAUX*
ok est FAUX donc $p = 2$, $A = \{(1, B)\}$ $\Rightarrow$ ***backtrack***

p *étant* $\leq$ *à* 3 *(nb de variables) on rentre dans la boucle*
appel de SelectValeur avec $p = 2$ *et* $D'[2] = \{B\}$
on essaye la valeur B *pour* 2 *et* $D'[2] = \varnothing$
ce choix d'affectation est incohérent avec $A = \{(1, B)\}$
$D'[2]$ *est vide, on sort du SelectValeur en renvoyant FAUX*
ok est FAUX donc $p = 1$, $A = \varnothing$ $\Rightarrow$ ***backtrack***

p étant $\leq$ à 3 (nb de variables) on rentre dans la boucle
appel de SelectValeur avec $p = 1$ et $D'[1] = \varnothing$
on sort du SelectValeur en renvoyant FAUX
ok est FAUX donc $p = 0$, $A = \varnothing$ $\Rightarrow$ ***backtrack***

Arrêt de la boucle puisque $p < 1$
Sortie de l'algorithme avec le print ECHEC : CSP incohérent

9.4 Exercices

Énoncé 30 *Configuration de produits. Le but est de simuler la réalisation d'un produit complexe à partir de composants.*
Par exemple, pour un ordinateur il doit avoir un processeur (p), de la mémoire vive (m) et un disque dur (d) et on a le choix entre 3 types de p (p_1, p_2, p_3), 4 types de m (m_1, m_2, m_3, m_4) et 3 types de d (d_1, d_2, d_3). L'indice indique l'année de sortie (plus il est grand et plus c'est récent). Les contraintes sont les suivantes :

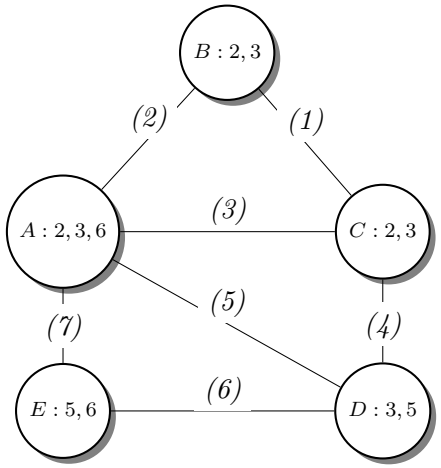
- *p_1 ne marche pas avec le composant d_3.*
- *Un processeur doit avoir une mémoire au moins aussi récente que lui (donc pas plus vieille).*
- *Le seul disque possible avec $p_2 + m_2$ est le disque d_2.*

Modélisez ce problème comme un CSP et dites :

1. *Quel choix faites-vous pour les variables ?*
2. *Quels sont les domaines des variables ?*
3. *Quelles sont les contraintes qui portent sur ces variables ?*

Énoncé 31 *Considérons le graphe suivant dans lequel on donne :*

- *pour chaque sommet son nom et son domaine,*
- *sur chaque arête, le numéro de la contrainte que cet arc représente sachant que cette contrainte signifie que les deux sommets adjacents ne peuvent pas avoir la même valeur.*



1. *Appliquez le backtrack en prenant les variables par ordre alphabétique et les valeurs par ordre croissant. Vous donnerez uniquement l'arbre d'affectation des variables en précisant les impasses (avec le numéro des contraintes violées) et en arrêtant à la première affectation valide.*
2. *Faire la même chose avec un ordonnancement préalable : ordre croissant sur la taille des domaines, puis si égalité ordre décroissant sur le nb de contraintes et enfin ordre alphabétique.*

3. *Pour chaque variable donnez la valeur la moins contraignante en expliquant.*

4. *Supposons qu'on assigne la valeur 2 à B. Que donnerait une propagation de contraintes jusqu'à ce qu'il n'y ait plus de changement ?*

Énoncé 32 *On considère 5 variables X_1, X_2, X_3, X_4, X_5 à valeurs entières, de domaines respectifs : $D(X_1) = \{2,3,4,5\}$, $D(X_2) = \{3,4,5\}$, $D(X_3) = \{2,4,5\}$, $D(X_4) = \{2,3,4\}$ et $D(X_5) = \{4,5,6\}$. Les valeurs possibles de ces variables sont restreintes par les 7 contraintes suivantes :*

— $C_1 : X_1 + 1 > X_2$
— $C_2 : X_1 \leq X_3$
— $C_3 : X_1 \leq X_4 + 1$
— $C_4 : 2 \times X_2 \geq X_3 + 3$
— $C_5 : X_2 + 1 \geq X_5$
— $C_6 : X_3 \geq X_4 + 1$
— $C_7 : X_5 > X_4 + 1$

Il s'agit de trouver une instanciation des 5 variables qui respecte toutes les contraintes.

Développer l'espace de recherche engendré par l'algorithme "Forward-Checking" jusqu'à l'obtention d'au moins **deux solutions**. *L'heuristique utilisée exploitera un ordre statique sur les variables (ordre X_1, X_2, X_3, X_4, X_5) et sur les valeurs (ordre croissant). Vous détaillerez l'évolution des valeurs affectées aux variables ainsi que l'évolution des domaines de ces variables.*

Énoncé 33 *[Lewis Caroll] 5 personnes de 5 métiers différents dans 5 maisons de couleurs différentes, avec un animal domestique et une boisson préférée distincts.*

On sait que :

— *L'anglais habite dans la maison rouge.*
— *L'espagnol a un chien.*
— *Le japonais est peintre.*
— *L'italien boit du thé.*
— *Le norvégien habite la première maison.*
— *L'habitant de la maison verte boit du café.*
— *La maison verte est après la blanche.*
— *Le sculpteur élève des escargots.*
— *Le diplomate habite la maison jaune.*
— *On boit du lait dans la 3e maison.*
— *La maison du norvégien est à côté de la bleue.*
— *Le violoniste boit du jus de fruit.*
— *Le renard est dans la maison après celle du docteur.*
— *Le cheval est la maison après celle du diplomate.*
— *Le zèbre est dans la maison blanche.*
— *Une des personnes boit de l'eau.*

Les questions auxquelles on doit répondre : Qui habite où ? Qui boit quoi ? Quel est le métier de qui ? Qui a tel animal de compagnie ? Quelle est la couleur de telle maison ?

Modélisez ce pb sous la forme d'un CSP et résolvez-le.

Bilan :
• *CSP* : "Constraint Satisfaction Problem"
• Problème NP-complet
• Méthode de représentation d'un pb à base de *variables* prenant leurs valeurs dans des *domaines* et devant respecter des *contraintes*
• But : attribuer une valeur à chaque variable en respectant les domaines et les contraintes
• Algorithme de base : le *backtrack*
• Amélioration de l'algorithme : ordonnancement des variables (parmi bq d'autres améliorations possibles)