Algorithme de Backtrack :

un outil de base pour résoudre de nombreux problèmes d'IA

Cours de HMIN107 (IA)

ML Mugnier

Une famille de problèmes fréquente en IA

- Problèmes définis sous la forme **(X,D,C)** :
  - **X** : ensemble de **variables**
  - **D** : ensemble de **domaines** (valeurs possibles)
  - **C** : ensemble de **conditions** sur la compatiblité des valeurs que peuvent prendre simultanément des variables
- Une **assignation** sur $Y \subseteq X$ est une application qui associe à chaque variable de Y une valeur de son domaine
- Elle est **totale** si $Y = X$
- Elle est **consistante** si elle satisfait les conditions qui portent sur les variables de Y (« **solution partielle** »)
- Une **solution** est une assignation **totale consistante**
- **Exemple** : **CSP** (« Constraint Satisfaction Problem »)

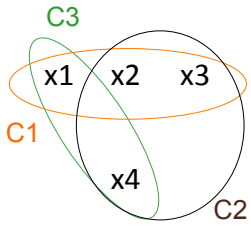
Nous allons en voir d'autres !

EXEMPLE DE CSP

Réseau de contraintes

- Ensemble de variables **X**={x1,x2,x3,x4}
- Ensemble de contraintes **C**={C1,C2,C3}

  ⟶ **hypergraphe**



- Domaines des variables **D1**=**D2**=**D3**=**D4**={a,b}
  (**D** : union des Di)
- Définitions des contraintes

C1

x1	x2	x3
a	a	b
a	b	a
b	a	a

C2

x2	x3	x4
a	b	a
b	a	b

C3

x1	x4
a	b
b	b

Une assignation *f* sur Y ⊆ X est **consistante** si :
pour tout Ci sur (y1,…,yk), (f(y1), …, f(yk)) appartient à la définition de Ci

ALGORITHME DE BACKTRACK (EXISTENCE D'UNE SOLUTION)

```
Fonction BacktrackingSearch() : Booléen  // accès aux données du problème
// retourne vrai ssi il existe une solution
   Début
      Prétraitements;
      retourner BT({});
   Fin
Fonction BT(Assignation a) : Booléen      // accès aux données du problème
   // retourne vrai ssi il existe une solution étendant a
Début
   si |a| = |X| alors retourner vrai; //solution trouvée
   x ← ChoixVariableNonAssignée(a);
   pour tout v ∈ Domaine(x) faire
      si Consistant(a∪{(x,v)}) alors
            si BT(a∪{(x,v)}) alors retourner vrai
   retourner faux;
Fin
```

ALGORITHME DE BACKTRACK (CALCUL D'UNE SOLUTION)

```
Fonction BacktrackingSearch() : Assignation (ou « échec », « null »)
   // retourne une solution s'il en existe une, sinon échec
   Début
      Prétraitements;
      retourner BT({});
   Fin
Fonction BT(Assignation a) : Assignation (ou échec)
// retourne une solution s'il en existe une étendant a
Début
   si |a| = |X| alors retourner a; //solution trouvée
   x ← ChoixVariableNonAssignée(a);
   pour tout v ∈ Domaine(x) faire
      si Consistant(a∪{(x,v)}) alors
            Assignation b = BT(a∪{(x,v)})
            Si b ≠ échec alors retourner b
   retourner échec;
Fin
```

ALGORITHME DE BACKTRACK (CALCUL DE TOUTES LES SOLUTIONS)

```
Fonction BacktrackingSearch() : Ensemble d'Assignation
// retourne l'ensemble des solutions
   Début
      Prétraitements;
      Ens → vide // ensemble de solutions
      BT(Ens, {}) // alimente Ens
      Retourner Ens
   Fin
Fonction BT(Ens d'Assignation Ens, Assignation a)
   // met dans Ens les solutions étendant a
Début
   si |a| = |X| alors ajouter a à Ens; //solution trouvée
   sinon
      x ← ChoixVariableNonAssignée(a);
      pour tout v ∈ Domaine(x) faire
         si Consistant(a∪{(x,v)}) alors BT(Ens, a∪{(x,v)})
Fin
```

UN PROBLÈME PROCHE : COLORATION DE GRAPHE

- Problème : étant donné un graphe G = (V,E)
  déterminer s'il existe une **k-coloration** de G

- k-coloration : assignation d'une couleur prise dans {1,..., k} à chaque sommet de V, telle que deux sommets adjacents n'aient pas la même couleur

→ Variables ?
→ Domaines ?
→ Conditions ?

- Variante : **colorer les arêtes**

BD RELATIONNELLE (→ VUE LOGIQUE)

- **Schéma** de BD : ensemble de relations (avec leurs attributs)

ex: Film [titre, directeur, acteur]
Pariscope [salle, titre, horaire]
Coordonnées [salle, adresse, téléphone]

On peut remplacer les attributs par une numérotation : 1,2,3
→ Vue logique : Film, Pariscope, Coordonnées
sont des relations (prédicats) ternaires

- **Instance d'une relation** : ensemble de k-uplets
  (k = **arité** de la relation)

→ Vue logique :
valeurs : constantes
instance de relation : ensemble d'atomes

- **Instance de BD** : ensemble des instances de relation

Une instance de la relation Film

titre	*directeur*	*acteur*
The trouble	Hitchcock	Green
The trouble	Hitchcock	Forsythe
The trouble	Hitchcock	MacLaine
The trouble	Hitchcock	Hitchcock
Cries and Whispers	Bergman	Anderson

Vue logique :

{ film(t,h,g), film(t,h,f), film(t,h,m), film(t,h,h), film(c,b,a) }

REQUÊTES CONJONCTIVES

En SQL: « SELECT ... FROM ... WHERE conditions de jointure »

Exemple :

« trouver les noms des films où Hitchcok joue »

```
SELECT Film.Titre
FROM Film
WHERE Film.Acteur = « Hitchcock »
```

Vue logique ?

trouver x *tel que l'on ait* Film(x,y,*h*)

Exemple :

« trouver les noms des salles dans lesquelles on joue un film de Bergman »

- Requête SQL ?
- Vue logique ?

Exemple :
« trouver les noms des salles dans lesquelles on joue un film de Bergman »

```
SELECT Pariscope.Salle
FROM Films, Pariscope
WHERE
    Films.Directeur = « Bergman »
    AND Films.Titre=Pariscope.Titre
```

Vue logique :
trouver z *tel que* Films(x,Bergman,y) ∧ Pariscope(z,x,t)

Vue logique d'une requête conjonctive :
ensemble d'atomes
+ une liste de « variables réponses » (à retourner comme réponse)

Si la liste des variables est vide, on a une requête **booléenne**

RÉPONDRE À UNE REQUÊTE CONJONCTIVE

- Requête **Q** et base de données **D** : deux ensembles d'atomes
- Un **homomorphisme h** de Q dans D est une substitution des variables de Q par des constantes de D
  telle que :
  Pour tout atome p(x1,...,xk) de Q
  p(h(x1),...,h(xk)) est dans D

  On considère que si *xi* est une constante, *h(xi) = xi*
- Les **réponses** à Q dans D sont obtenues en prenant les images des « variables réponses » par les homomorphismes de Q dans D

RECHERCHE D'HOMOMORPHISMES DE Q DANS D

Représentation sous la forme (X,D,C) ?

X = variables de Q

D = constantes de D

C = conditions à satisfaire par une assignation ?

Pour chaque atome p(x1,...,xk) de Q

p(h(x1),...,h(xK)) est dans D

Exemple : Q = { p(x,y,z), s(z,t) }
D = { p(a,b,c), s(b,a), s(c,b) }

- Dérouler l'algorithme de backtrack
  en prenant un ordre statique sur les variables : z t x y
- Quels sont les atomes à tester à chaque étape ?

EXEMPLE (RECHERCHE D'HOMOMORPHISME(S))

$\mathcal{A}1$ = { p(x,y), p(y,z), q(z,x) }, où x, y et z sont des variables

$\mathcal{A}2$ = { p(a,b), p(b,a), q(b,b), q(a,c), q(b,c) } où a, b et c sont des constantes

- Quels sont les **homomorphismes** de $\mathcal{A}1$ dans $\mathcal{A}2$?
- Dessiner l'**arbre de recherche** du backtrack obtenu :
  - en s'arrêtant dès qu'une solution est trouvée
  - en considérant les variables et les constantes selon l'ordre lexicographique

DE HOM À CSP

Problème de décision **HOM**
Données : $\mathcal{A}1$, $\mathcal{A}2$ « *instance* de HOM »
Question : existe-t-il un homomorphisme de $\mathcal{A}1$ dans $\mathcal{A}2$?

Problème de décision **CSP**
Données : réseau (X,D,C) « *instance* de CSP »
Question : ce réseau est-il (globalement) consistant ?

Peut-on résoudre HOM
en utilisant un algorithme qui résout CSP ?

(et on aimerait bien aussi trouver les homomorphismes)

EXEMPLE (RECHERCHE D'HOMOMORPHISME(S))

$\mathcal{A}1$ = { p(x,y), p(y,z), q(z,x) }, où x, y et z sont des variables
$\mathcal{A}2$ = { p(a,b), p(b,a), q(b,b), q(a,c), q(b,c) } où a, b et c sont des constantes

- Quels sont les **homomorphismes** de $\mathcal{A}1$ dans $\mathcal{A}2$?
- Dessiner l'**arbre de recherche** du backtrack obtenu :
  - en s'arrêtant dès qu'une solution est trouvée
  - en considérant les variables et les constantes selon l'ordre lexicographique

HOM → CSP

$\mathcal{A}1$ et $\mathcal{A}2$ → (X,D,C)

Supposons que $\mathcal{A}1$ n'ait pas de constantes et qu'il n'y ait pas deux fois la même variable dans un atome

- ensemble des termes (variables) de $\mathcal{A}1$ → X
- ensemble des termes (constantes) de $\mathcal{A}2$ → Di pour tout xi $\in \mathcal{A}1$

  toutes les variables de X ont le même domaine
- on note A1 ... An les atomes de $\mathcal{A}1$

  C = { Ci(x1, ..., xk) | Ai = p(x1, ..., xk) $\in \mathcal{A}1$ }

  La définition de Ci est l'ensemble des tuples (a1, ..., ak) tels que p(a1, ..., ak) $\in \mathcal{A}2$

HOM → CSP *(suite)*

- transformation de p(x,x) ?

  p(x,y) et x = y

- transformation de p(x,a) où a est une constante ?

  p(x,y) où y est une nouvelle variable

  et domaine(y) = {a}

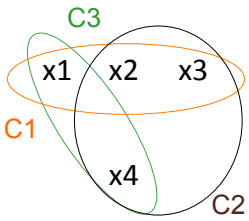
Cas général :

- on remplace les **multi-occurrences** de variables dans **un atome** en introduisant de nouvelles variables et en ajoutant des contraintes d'égalité
- on remplace chaque **constante** par une nouvelle variable dont le domaine est réduit à cette constante

CSP (Constraint Satisfaction Problem)
Données : un réseau de contraintes P = (X, D, C)
Question : P admet-il une solution? [trouver toutes les solutions]

Exemple de réseau de contraintes

- Ensemble de variables **X**={x1,x2,x3,x4}
- Ensemble de contraintes **C**={C1,C2,C3}
  → **hypergraphe**
- Domaines des variables **D1**=**D2**=**D3**=**D4**={a,b}
  (**D** : union des Di)
- Définitions des contraintes



C1

x1	x2	x3
a	a	b
a	b	a
b	a	a

C2

x2	x3	x4
a	b	a
b	a	b

C3

x1	x4
a	b
b	b

Réduction (polynomiale) de CSP à HOM (qui, de plus, préserve les solutions ?)
Illustrer sur l'exemple

CSP → HOM

(X,D,C) → $\mathcal{A}1$ et $\mathcal{A}2$

X → ensemble des termes (variables) de $\mathcal{A}1$
D → ensemble des termes (constantes) de $\mathcal{A}2$

$\mathcal{A}1$ = { Ci(x1, ..., xk) | Ci ∈ C
et porte sur (x1 ... xk) }

$\mathcal{A}2$ = { Ci(a1, ..., ak) | Ci ∈ C
et (a1, ..., ak) est dans sa définition }

RÉDUCTION DE PROBLÈMES

- Soient deux problèmes de décision P1 et P2.
  P1 se réduit à P2 s'il existe une transformation **t** qui,
  à toute instance **I1** de **P1** associe une instance **t(I1)** de **P2**,
  tel que la réponse à **t(I1)** est oui **ssi** la réponse à **I1** est oui
- La transformation **t** est dite **polynomiale** si elle est polynomiale en la taille de **I1**
- Elle **préserve les solutions** s'il existe une **bijection** entre les solutions à **I1** et les solutions à **t(I1)**

Nous avons construit

- une réduction polynomiale de HOM à CSP
- une réduction polynomiale de CSP à HOM

Ces réductions préservent les solutions

SAT (« PROBLÈME DE SATISFIABILITÉ D'UNE FORME CLAUSALE EN LOGIQUE DES PROPOSITIONS »)

- symbole propositionnel, *variable propositionnelle*, atome
- **Littéral** : variable propositionnelle ou sa négation
- **Clause** : disjonction de littéraux
- **Forme clausale** : conjonction de disjonctions
- **Problème SAT** : déterminer si une forme clausale est satisfiable

Exemple

$$F = (p \vee \neg q) \wedge (q \vee r) \wedge (\neg p \vee \neg r)$$

F est-elle satisfiable ?

Comment représenter SAT sous la forme

(variables, domaines, conditions de consistance) ?

- CSP
- SAT
- coloration de graphe
- recherche d'homomorphismes

[...]

On peut passer de n'importe lequel de ces problèmes à n'importe quel autre

- par une transformation polynomiale
- qui respecte les ensembles de solutions