# Représentations et résolution de problèmes en IA

Graphes d'états
Graphes de sous-problèmes
Problèmes de satisfaction de contraintes



## Plan

- Introduction
- Représentation et résolution de problèmes par graphes d'états
- Représentation et résolution de problèmes par graphes de sous-problèmes
- Problèmes de satisfaction de contraintes



## Quels problèmes ?

- Pas de méthode de résolution adaptée
- Description formelle du problème
- Procédure pour tester une solution proposée
- On peut engendrer et énumérer les solutions potentielles

Énumération + test =
procédure constructive de résolution



## Deux méthodes de résolution constructives

- Augmenter des solutions partielles
  Exemple : problème des 8 reines
- Décomposer le problème en sous-problèmes
  Exemple : tours de Hanoi

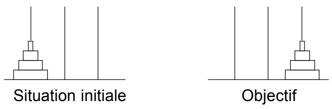


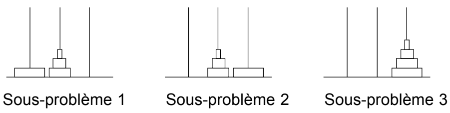
## Problème des huit reines

- Problème :
  placer 8 reines sans prise sur un échiquier
- Méthode de résolution :
  placer une $k^{ième}$ reine sur un échiquier où figurent déjà k-1 reines



## Les tours de Hanoi



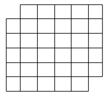
## Représenter le problème (1)

Il faut définir :

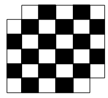
- Les états du problème
- L'objectif à atteindre
- Les opérateurs de transformation
  - Extension de solutions partielles
    □ représentation par graphe d'états
  - Décomposition en sous-problèmes
    □ représentation par graphe de sous-problèmes



## Représenter le problème (2)
## Problème de l'échiquier écorné



À recouvrir de dominos :



À recouvrir de dominos :



## Représentation par graphes d'états

- On représente l'ensemble des états du problème par un graphe orienté étiqueté
  - Les sommets sont les états du problème
  - Il existe un arc (u,v) si un opérateur transforme l'état u en état v, étiquette(u,v) = opérateur
- Recherche d 'une solution = recherche d'un chemin entre le nœud initial et un nœud terminal
- Solution = séquence des opérateurs étiquetant les arcs de ce chemin



## Choisir une bonne représentation des états
## Problème des huit reines

1. État : échiquier(i,j) □ {vide,reine}
   Opérateur : placer-reine(i,j) : si échiquier(i,j) = vide
   alors échiquier(i,j) □ reine
   Tests : échiquier-sans-prise, échiquier-complet
   □ $2^{64}$ états dont de nombreux inutiles
   □ intégrer une partie des contraintes du problème à la représentation
2. ligne(i) = j si la $i^{ème}$ ligne a une reine en colonne j
   ligne(i) = 0 si cette ligne est vide



## Stratégies d'organisation de la recherche

- Développement d'état : à chaque étape, l'état choisi sera
  - complètement développé
  - partiellement développé
- Organisation des alternatives : l'ensemble des états à développer est organisé dans
  - une pile □ recherche en profondeur
  - une file □ recherche en largeur
  - une liste selon un coût croissant



## Notion d'heuristique

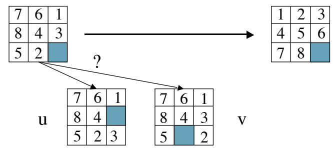
- Résolution de problèmes par développement et choix d'une alternative □ combinatoire
- Un algorithme de recherche doit guider la recherche d'une solution en faisant des choix et en gérant le retour sur ces choix tout en évitant l'explosion combinatoire
- Une heuristique est un moyen de guider les choix que doit faire l'algorithme pour réduire sa complexité, en ordonnant la liste des successeurs d'un état

## Définir une heuristique

- Une heuristique est spécifique à un problème, elle ne peut pas être généralisée
- Elle est souvent fondée sur une simplification du problème, grâce à une relaxation des contraintes difficiles
  - Simplification grossière → heuristique pauvre et peu efficace
  - Simplification élaborée → heuristique plus complexe à évaluer mais plus efficace



## Exemple d'heuristique : le jeu du taquin




Choisir = apprécier l'écart à l'objectif

h1 : nombre de pièces mal placées

h2 : somme des déplacements minimaux pour amener une pièce à sa place

h2 est plus efficace que h1



## Recherche heuristique dans les graphes d'états

- Pour tout nœud u, on définit :
  - g*(u) le minimum du coût des chemins de $u_0$ à u
  - h*(u) le minimum du coût des chemins de u à un état terminal quelconque
  - f*(u) = g*(u) + h*(u) le coût du chemin solution optimal passant par u
- Pour ordonner la recherche, on utilise :
  - une heuristique h(u) qui estime h*(u)
  - g(u) le coût du meilleur chemin connu pour aller jusqu'à u (estime g*(u))
  - f(u) = g(u) + h(u) la fonction d'évaluation



## L'algorithme A*

L'algorithme A* utilise f pour ordonner les nœuds à développer

- Si le graphe est fini, A* s'arrête et fournit un chemin solution s'il existe
- Si ∀u, h(u) ≤ h*(u), A* fournit la meilleure solution



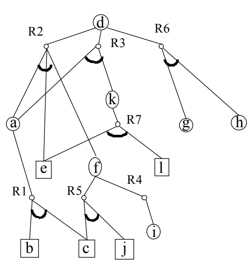
## Représentation par graphes de sous-problèmes

- État initial ou courant : problème à résoudre
  Opérateur : décomposition en plusieurs sous-problèmes
  États terminaux : problèmes triviaux
- Représentation par un graphe ET/OU sans circuit :
  - nœuds OU associés aux problèmes
  - nœuds ET de décomposition



## Exemple de graphe ET/OU

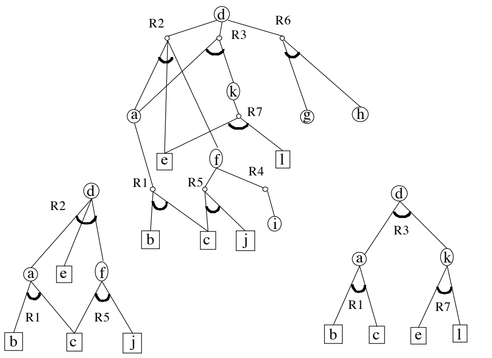
- Règles de décomposition :
  R1 : a → b, c
  R2 : d → a, e, f
  R3 : d → a, k
  R4 : f → i
  R5 : f → c, j
  R6 : d → g, h
  R7 : k → e, l
- Problèmes terminaux : b, c, e, j, l
- Problème à résoudre : d



## Recherche d'une solution dans un graphe ET/OU

Recherche d'une solution = recherche d'un sous-graphe contenant
- le nœud racine
- pour tout nœud non terminal : un seul connecteur et les nœuds auxquels il mène






Graphe solution 1

Graphe solution 2



## Principe de l'algorithme de recherche d'un sous-graphe ET/OU

- On définit
  - des états RÉSOLUS :
    - les états terminaux
    - les états dont un connecteur a tous ses fils résolus
  - des états INSOLUBLES :
    - les états non terminaux sans successeur
    - les états dont tous les connecteurs ont au moins un successeur insoluble
- On développe en profondeur un sous-graphe représentant une solution partielle avec retour arrière en cas d'échec



## Améliorer l'efficacité de la recherche

- On peut introduire un seuil sur le rang des nœuds pour réduire l'espace exploré
- On peut exploiter une information heuristique pour
  - ordonner l'examen des connecteurs d'un nœud
  - interrompre la récursion sur des états peu prometteurs en les rendant insolubles



## Problèmes de satisfaction de contraintes

- Définition :
  - X = { $X_1$, $X_2$, ..., $X_n$} est l'ensemble des variables du problème
  - On associe à chaque variable $X_i$ son domaine $D(X_i)$, l'ensemble des valeurs que peut prendre $X_i$
  - C = {$C_1$, $C_2$, ..., $C_k$} est l'ensemble des contraintes Chaque contrainte $C_j$ est une relation entre certaines variables de X, restreignant les valeurs que peuvent prendre simultanément ces variables
- Exemple : le problème des reines



## Résoudre un problème de satisfaction de contraintes (1)

- Résoudre = affecter une valeur à toutes les variables sans violer de contraintes
- Méthode « générer et tester »
  - On construit une solution puis on vérifie que les contraintes sont satisfaites
  - Dans l'exemple : placer une reine sur chaque ligne puis vérifier qu'elles ne sont pas en prise

## Résoudre un problème de satisfaction de contraintes (2)

- Méthode « retour arrière »
  - Ne pas développer une solution partielle qui viole déjà les contraintes
  - Dans l'exemple : à chaque fois qu'on place une reine, on vérifie qu'elle n'est en prise avec aucune autre, sinon on revient sur les derniers choix
- Méthode « filtrage »
  - réduire le domaine des variables à chaque affection
  - Dans l'exemple : à chaque fois qu'on place une reine, on élimine la colonne choisie des colonnes possibles pour les reines restant à placer
- Utiliser une heuristique : par exemple commencer par affecter la variable dont le domaine est le plus petit