Notes pour le cours BIA sur la résolution de problème

Alain Mille

6 septembre 2012

1 Exemples de "problèmes"

- Chercher un chemin dans une configuration avec obstacle (jeux vidéos, robotique, etc.)
- Trouver la succession des opérations à faire pour passer d'un état à un autre (jeux, production mécanique, etc.)
- Trouver un chemin dans un réseau (routier, télécommunications, ...
- Résoudre un casse-tête (jeux)
- Trouver une procédure qui fonctionne (administration, entreprises, ...)

2 Démarche "IA" retenue pour la résolution d'un problème

Le principe général de l'IA dans cette approche est de considérer qu'il existe des méthodes **générales** permettant de résoudre **n'importe quel type de problème**.

L'algorithme se doit donc d'être "neutre" sur le domaine concerné, et tout ce qui concerne les connaissances de description du problème et de sa résolution doit être clairement séparé de l'algorithme.

Malgré l'ambition unificatrice visée, il existe plusieurs démarches différentes pour aborder la question.

3 Démarches de construction de solution

La résolution de nombreux problèmes peut en effet être décrite comme la séquence d'opérateurs permettant de mener de l'état initial (la description de l'état du "monde" avant la résolution du problème) jusqu'à l'état final (ce qui caractérisera l'état du monde quand le problème sera résolu). Il peut y avoir plusieurs "états finaux" satisfaisants et permettant de considérer le problème résolu.

L'idée générale est de considérer les "opérateurs" permettant de changer l'état du monde, de tenter ces opérateurs de manière systématique et de tester si on arrive à un état "final" satisfaisant.

Dans ces conditions, à partir de l'état initial, il "suffit" donc d'explorer de manière plus ou moins astucieuse les états possibles à atteindre depuis cet état initial. Si l'état final est dans l'espace de ces états alors on a trouvé un chemin d'opérateurs à appliquer successivement pour passer de l'état initial à un état final.

Pour modéliser un problème de cette façon, il faut donc :

- décrire un état de départ
- définir les opérateurs permettant de passer d'un état à d'autres (fonction permettant de retourner l'ensemble des états atteignables par un opérateur particulier)
- construire l'espace des états (l'ensemble des états atteignables depuis l'état de départ)
- disposer d'un test permettant de savoir que la solution est trouvée (si on a trouvé l'état but final)
- construire un chemin de l'état de départ à l'état final (une séquence d'états dans l'espace des états)
- disposer d'une fonction de coût sur le chemin : cette fonction associe un coût au chemin, coût calculé comme la somme des coûts individuels des actions le long du chemin.

Dans ces conditions, qu'est-ce que faire une recherche de solution ?

Il s'agit alors de faire un examen systématique des états à trouver en partant de l'état de départ jusqu'à trouver l'état but.

L'ensemble des états possibles connectés par les opérateurs permettant de passer des uns aux autres représente **l'espace d'états.** L'ensemble des chemins possibles pour atteindre les états buts à partir de l'état initial représente **l'espace des solutions.**

Le résultat de l'algorithme de recherche est donc une solution (il peut y en avoir plusieurs), c'est-à-dire un chemin d'opérateurs conduisant de l'état initial à l'état qui satisfait les conditions du test d'atteinte du but.

En pratique, dans la vie courante, une telle recherche **systèmatique** est la conséquence d'un manque de connaissances. C'est vrai également en Intelligence Artificielle, pour laquelle l'algorithme est surtout un outil pour "attaquer" les problèmes qui ne peuvent pas être résolus d'une autre façon.

On peut classer les méthodes de résolution de problème en deux grandes familles (d'autres classifications sont possibles également naturellement) selon la manière de construire la solution à partir d'un état de départ.

3.1 Augmentation de solution partielle [Generate and Test]

Il s'agit de commencer à résoudre le problème en utilisant un opérateur valide (qui peut s'appliquer dans le contexte), et de continuer à le faire à partir de l'état précédent. S'il n'est plus possible de faire une nouvelle opération ou si on revient sur un état déjà parcouru, alors on revient en arrière (backtrack)

pour faire un nouvel essai avec un opérateur non encore essayé. Si plus aucun choix d'opérateur n'est possible, alors il n'y a pas de solution. Exemple sur le "problème des reines".

- Etats possibles : De 0 à N reines placées sur le damier sur une case non attaquée.
- Operateur : Placer une reine sur une case non attaquée du damier
- Etat but : N reines placées sur le damier sur des cases non attaquées.

Dans cette modélisation, une partie des connaissances du domaine sont dans la représentation. En effet, la notion de case attaquée suppose la connaissance des règles de prise de pièce aux échecs.

3.2 Application récursive [Divide and Solve]

Dans cette approche, il s'agit de décomposer le problème en sous-problèmes jusqu'à arriver à des sous-problèmes "triviaux" (que l'on sait résoudre par la méthode précédente à coup sûr). On résoud alors ces problèmes et on remonte l'arbre de décomposition jusqu'à obtenir la solution au problème complet. Exemple sur les "Les Tours de Hanoi".

- Le problème principal est découpé en sous-problème de même nature de complexité moindre.
- Ce sous-problème peut lui-même être découpé en sous-problème de la même façon, jusqu'à un problème dont la solution est immédiate (état but atteint immédiatement dans son espace d'état)

4 Stratégies d'organisation de la recherche

4.1 Rappels sur les stratégies de base

Un certain nombre de choix sont à faire pour organiser la recherche dans l'arbre des états. A partir de l'état de départ, on peut construire l'ensemble des états à développer :

- dans une pile -> exploration en profondeur
- dans une file -> exploration en largeur
- dans une liste selon un coût estimé croissant (best first)

4.2 Notion d'heuristique

Quelle que soit la stratégie de recherche utilisée, elle ne permet pas de prendre en compte des connaissances supplémentaires que l'on aurait sur la manière de trouver l'état but depuis un état quelconque de l'espace des états. L'explosion combinatoire est vite là !

Un algorithme de recherche efficace doit donc guider la recherche du chemin solution en faisant des choix et en gérant la révision de ces choix pour éviter tant que faire se peut l'explosion combinatoire.

Une HEURISTIQUE est donc un moyen de guider ces choix en permettant d'ordonner dynamiquement la liste des successeurs selon leur "promesse de

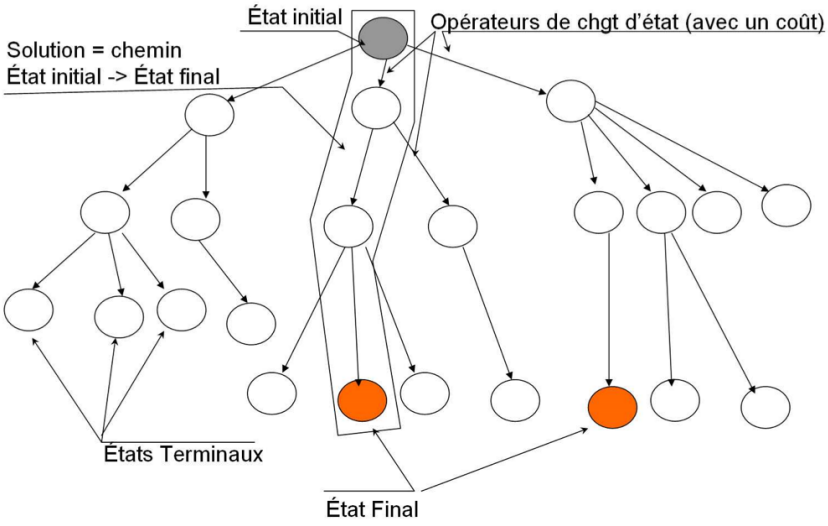

FIGURE 1 – Un espace d'états sous forme de graphe

rapprocher d'un but". Une heuristique est donc avant tout l'expression d'une connaissance spécifique au problème à résoudre (et donc indépendante de l'algorithme de recherche).

En conséquence, une heuristique pauvre ne révélant que quelques propriétés très simples du problème sera peu efficace, tandis qu'une heuristique riche révélant une connaissance approfondie du problème sera probablement plus difficile à évaluer mais beaucoup plus efficace.

4.3 Recherche heuristique dans un graphe d'état (principe de l'algorithme A*)

Pour tout noeud u du graphe, on définit $g^*(u)$ comme le coût minimum entre u_0 le noeud de départ et u le noeud courant.

On définit $h^*(u)$ le minimum du coût des chemins du noeud u à un noeud but terminal quelconque.

on définit $f^*(u) = g^*(u) + h^*(u)$ le coût du chemin de solution optimal passant par u (la solution optimale est celle dont la mise en oeuvre du "chemin d'opérateurs" est la moins "coûteuse").

En conséquence, pour ordonner la recherche on utilise :

- une heuristique $h(u)$ qui ESTIME $h^*(u)$
- $g(u)$ le coût effectif du meilleur chemin connu pour aller de u_0 jusqu'à u (estimateur de $g^*(u)$
- $f(u) = g(u) + h(u)$ la fonction d'évaluation

Exercice : *Appliquer ce principe à un déplacement d'un pion sur un échi-*

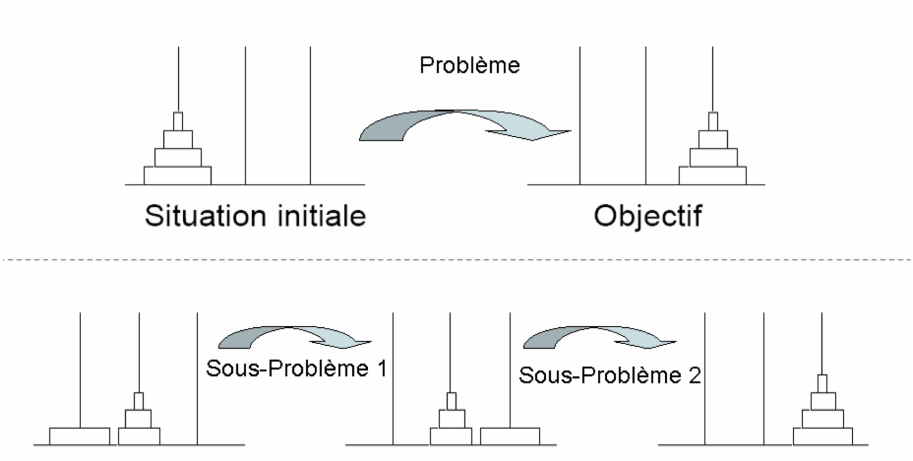

FIGURE 2 – Exemple de description du problème des Tours de Hanoi en sous-problèmes

*quier 5*5. Le pion part de la case du milieu de la colonne de gauche. Il doit atteindre la case du milieu de la ligne du bas. Proposer une heuristique. Faire sans obstacle et avec obstacle.*

5 Stratégie de résolution par décomposition en sous-problèmes

5.1 Représentation en graphe de sous-problèmes

Dans cette stratégie de résolution de problème :
- Un état est un problème (ou un sous-problème à résoudre).
- L'état initial est le problème non décomposé
- Les états terminaux sont des sous-problèmes triviaux (solution immédiate)
- L'opérateur est celui de décomposition de problème
- Il y a deux types de noeuds :
  - les noeuds "OU" associés aux problèmes
  - les noeuds "ET" associés aux règles de décomposition

5.2 Recherche de solution dans un graphe de sous-problèmes (graphe ET/OU)

Rechercher une solution consiste à rechercher un sous-graphe contenant :
- le noeud de départ,
- pour tout noeud non terminal : un seul connecteur et les noeuds auxquels il mène

On définit alors :

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

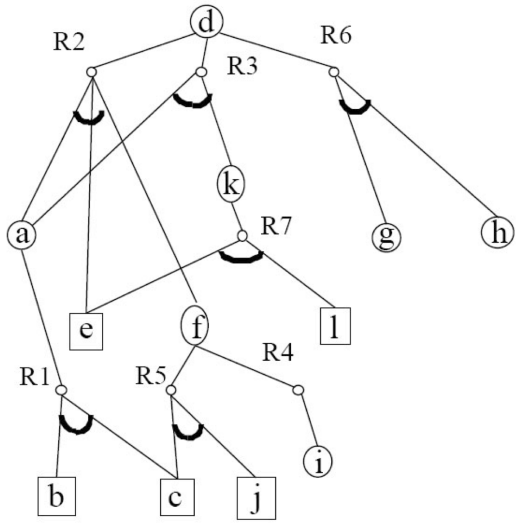

FIGURE 3 – Un espace de décomposition en sous-problèmes (arbre et-ou)

- des états RESOLUS
  - les états terminaux
  - les états dont un connecteur a tous ses fils RESOLUS
- 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.

On peut bien sûr améliorer l'efficacité de la recherche en introduisant un seuil sur le rang des noeuds pour réduire l'espace exploré.

On peut exploiter une heuristique pour ordonner l'examen des connecteurs d'un noeud ou encore interrompre la récursion sur des états peu prometteurs en les rendant insolubles arbitrairement.

6 Approche par la résolution de problèmes par satisfaction de contraintes

6.1 Représentation

Cette approche a un grand avantage : la représentation des connaissances est explicite (elle n'est pas "cachée" dans la fonction d'évaluation de l'heuristique) et elle peut être appliquée à de très nombreux problèmes. Toutefois, elle nécessite une bonne capacité de modélisation permettant de présenter le problème de

manière adaptée à la méthode.

De quoi s'agit-il ? Il s'agit d'identifier les "variables" du problème, c'est à dire ce qui peut varier dans la description d'un état. A chaque variable, on associe un ensemble de valeurs possibles théoriquement. Enfin, on va contraindre les possibilités de variations relatives des variables par des relations que leurs valeurs doivent respecter impérativement.

Plus formellement, on définit un problème par :

- $X = X_1, X_2, ..., X_n$ l'ensemble des variables caractérisant le problème,
- $D(X_i)$ le domaine de chaque variable X_i (l'ensemble des valeurs que X_i peut prendre),
- $C = C_1, C_2, ..., C_k$ 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.

Un excellent exemple est naturellement le problème "des reines".

6.2 Résolution

La résolution consiste à générer les solutions possibles théoriquement (celles conformes aux domaines des variables) puis à tester que les contraintes sont satisfaites (generate and test).

Bien entendu, la combinatoire est désastreuse, et il faut utiliser des connaissances supplémentaires pour rendre efficace la méthode.

Par exemple :

- Méthode du retour arrière
  - Ne pas développer une solution partielle qui viole déjà les contraintes : dans l'exemple des reines, à chaque fois que l'on place une reine, on vérifie qu'elle n'est pas attaquée, sinon on révise les choix précédents.
- Méthode du filtrage
  - réduire le domaine des variables à chaque affectation : dans l'exemple des reines, à chaque fois que l'on pose une reine on élimine du domaine toutes les positions qui sont maintenant attaquées.
- Utiliser une heuristique : par exemple, commencer par affecter la variable dont le domaine est le plus petit.

Les TD seront l'occasion de mettre en oeuvre ces trois approches et de préparer des représentations et des algorithmes qui pourront être codés en TP (Prolog).