Intelligence artificielle

resp.: Marie-Laure Mugnier

Algorithmes de recherche

Michel Leclère

leclere@lirmm.fr

IA

- Résolution de problèmes (algorithmique)
  - Espaces de recherche
  - Technique de « Backtrack »
  - CSP
  - Algos de jeux
  - Planification
- Représentation de connaissances (logique)
  - Bases de connaissances
  - Systèmes à règles
  - Représentation de l'incertain
  - Aide à la décision

Mais aussi…

- Agents intelligents
  - Modélisation/Programmation
  - Perception/Communication
    - Vision
    - TALN
  - Robotique
- Apprentissage

Support de cours

- "*Artificial Intelligence, a modern approach*" (3ème édition), S. Russel & P. Norvig, Prentice Hall, 2010
  - Existe en français, disponible à la BU

RÉSOLUTION DE PROBLEMES

Problème

- Un problème est une collection d'informations que l'agent utilise pour décider quelle(s) action(s) accomplir.
- Définir un problème c'est choisir une abstraction de la réalité en termes :
  - Identification d'un état initial (donc choix d' un langage de description d'états du problème)
  - Identification des actions possibles par définition d'opérateurs de changement d'état (donc définition de l'ensemble des états possibles du problème)

Espace de recherche

- On appelle **espace de recherche** (ou espace des états) d'un problème l'ensemble des états atteignables depuis l'état initial par n'importe quelle séquence d'actions
- Un espace de recherche peut être représenter par un graphe orienté
  - » Les sommets sont les états
  - » Les arcs sont les actions

Résoudre un problème

- Un problème est défini pour un objectif particulier. On doit donc également définir :
  - Une fonction de test de but atteint qui détermine si un état du problème correspond à un état but du problème
    - » Par liste d'états but
    - » Par la donnée d'une propriété pour un tel état
- Une solution est une séquence d'actions permettant de passer de l'état initial vers un état but
  - Donc un chemin dans l'espace des états de l'état initial vers un état but
- Résoudre un problème c'est trouver une/toutes les solutions
  - Pour certains problèmes, une fonction de coût de chemin permet de sélectionner une solution préférée parmi l'ensemble des solutions

Connaissance complète vs. incomplète

- Problèmes à état simple
  - On connaît l'état dans lequel on est
  - On connaît précisément l'effet des actions
  - On peut calculer à tout moment l'état dans lequel on se trouvera après une action
- Problèmes à états multiples
  - On ne sait pas exactement dans quel état on se trouve => seulement un ensemble d'états possibles
  - On ne connaît pas précisément l'effet des actions
  - On ne peut que caractériser par un ensemble d'états la situation où l'on est

Connaissances complètes vs. incomplètes

- Problèmes non complètement prévisibles
  - Le choix d'une action nécessite de tester l'environnement durant l'exécution
    - » La recherche d'une solution se fait durant la phase d'exécution de la solution en alternance (ex. les problèmes de jeux à 2 joueurs)
  - Le calcul des états atteints par une action est paramétré par les événements extérieurs pouvant modifier l'environnement

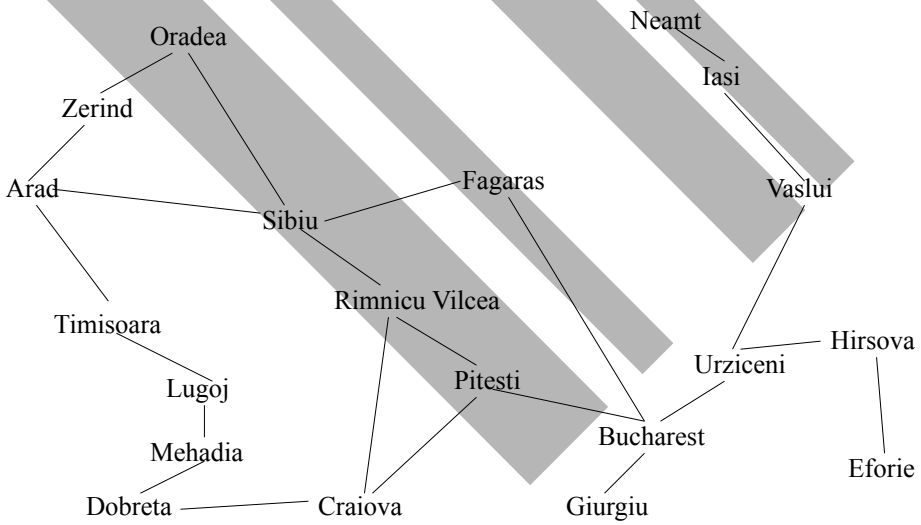
Nature du problème

- Problèmes jouets : concis et bien défini
  - » Le taquin, les reines…
  - » La modélisation est facile !
  - » Intéressant pour comparer les différentes stratégies de résolution
- Problèmes du monde réel : complexe et très ouvert
  - » Calcul de routes, voyageur de commerce, navigation de robots…
  - » Difficiles à résoudre dans le cas général (trop de paramètres) d´ où importance de la modélisation !

Formalisation d'un problème à état simple

Type de données	***Problem***
Composants	***InitialState, Operators***
InitialState	Etat(s) initial(aux) de l'agent
Operators	Actions possibles de l'agent
Opérations	***GoalTest, PathCost***
Bool←GoalTest(State)	Fonction qui teste si un état est un but
Real←PathCost(Sequence of Operator)	Fonction de coût d'un chemin (souvent définie comme la somme du coût des opérateurs utilisés)

Exemple 1 : Recherche d'une route de Arad à Bucharest



Exemple 1 : Modèle graphique des routes roumaines

- Les états et les opérateurs de l'agent sont respectivement représentés par les sommets S et les arêtes A du graphe G=(S,A) précédent.
- L'état initial est le sommet *Arad* et l'état but le sommet *Bucharest*.

Initial-State	{Arad}
Operators	A
GoalTest	{Bucharest}
PathCost	Somme(CostFunction(a)) pour tout a appartenant au chemin considéré

Pour tout a appartenant à A,

CostFunction(a) distance entre les deux villes liées par a

Algorithme de résolution

- Résoudre un problème consiste à trouver une séquence d'actions permettant de passer de l'état initial à un état but : une solution
- Il s'agit donc d'effectuer une recherche à travers l'espace des états
- L'idée est de maintenir et étendre un ensemble de solutions partielles : des séquences d'actions qui amènent à des états intermédiaires « plus proche » de l'état but.

Génération des solutions partielles

- Un cycle en 3 phases
  1. Tester si l'état actuel est un état but
  2. Générer un nouvel ensemble d'états à partir de l'état actuel et des actions possibles
  3. Sélectionner un des états générés (à cette étape ou précédemment) et recommencer…
- Le choix de l'état à considérer (la sélection) est déterminé par la stratégie de recherche

Processus de résolution

- Le processus de résolution de problème consiste donc à construire un arbre de recherche qui se superpose à l'espace des états du problème.
- Chaque nœud de l'arbre correspond soit à l'état initial du problème, soit à un développement du sommet parent par un des opérateurs du problème.

Arbres de recherche

- La racine de l'arbre correspond à l'état initial du problème.
- Les feuilles de l'arbre correspondent à des états sans successeur dans l'arbre ou des nœuds qui n'ont pas encore été développés.
- Un chemin est une séquence de sommets partant du sommet à une feuille.
  - Le coût d'un chemin est défini par la somme des coûts de chaque opérateur intervenant dans la construction du chemin.

Nœuds d'un arbre de recherche

Type de données ***Node***

Composants ***State, ParentNode, Operator, Depth, PathCost***

State	Etat dans l'espace des états auquel le nœud correspond
ParentNode	Le nœud ayant généré ce nœud
Operator	Opérateur utilisé pour générer ce nœud
Depth	Le nombre de nœud du chemin de la racine à ce nœud
PathCost	Le coût de ce chemin

Opérations ***MakeNode, Expand***

Node←MakeNode(State) Fabrique un nœud à partir d'un état (utilisé pour l'état initial)

Set of Node←Expand(Node,Set of Operator)

Calcule l'ensemble des nœuds générés par l'application des opérateurs au nœud spécifié

Nœuds d'un arbre de recherche

- On appelle frontière l'ensemble des nœuds non encore développés de l'arbre de recherche
- On choisit de gérer la frontière comme une liste
  - On sélectionne toujours le nœud en tête de liste
  - La stratégie de recherche est reportée sur la procédure d'insertion des nœuds dans la liste

Nœuds d'un arbre de recherche

Type de données ***Queue***

Opérations ***MakeQueue, Empty?, RemoveFront, QueuingFn***

Queue ←MakeQueue(Set of Node)	Construit une liste de nœuds
Bool ←Empty?(Queue)	Retourne vrai si la liste est vide
Node ←RemoveFront(Queue)	Extrait le nœud en tête
QueuingFn(Set of Node,Queue)	Insère des nœuds dans la liste selon une stratégie particulière

Fonction générale de Recherche

```
Node or nil ← GeneralSearch(Problem p, QueuingFn strategy)
// retourne une solution ou un échec
   Queue nodes←MakeQueue(MakeNode(p.InitialState));
   Loop do
        if Empty?(nodes) then return nil;
        Node n←RemoveFront(nodes);
        if GoalTest(n.state) then return n;
        nodes←strategy(Expand(n,p.operators),nodes)
   End
```

Performance d'une stratégie de résolution

- La performance d'une stratégie de résolution se mesure selon quatre points de vue :
  - Complétude : la technique de résolution marche t'elle dans tous les cas ?
  - Optimalité : la technique de résolution trouve t'elle une solution de coût minimal ?
  - Complexité : la technique est-elle coûteuse
    - » en temps ?
    - » en mémoire ?

Attention à ne pas confondre la performance de la résolution et celle de l'exécution de la solution