Intelligence Artificielle

Niveau 3ème SIL

Dorra BEN AYED MEZGHANI

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Plan

- Chapitre 1: **Introduction et Présentation Générale**
- Chapitre 2: **Résolution de problème en IA par recherche**
- Chapitre 3: **Résolution de problème en IA dans les arbres de jeux**
- Chapitre 4: **Représentation des connaissances et raisonnement**
- Chapitre 5: **Systèmes Experts et Moteurs d'Inférence**

Chapitre 1

Introduction et Présentation Générale

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Sommaire

- Introduction
- Historique
- Définitions de l'IA
- Domaines d'application
- Les approches de l 'IA
  - cognitive,
  - Pragmatique
  - connexionniste

Introduction

- IA : c'est le domaine de l'informatique qui a pour sujet l'étude et la simulation des activités intellectuelles de l'homme dans toute situation ou une solution classique de type algorithmique n'est pas disponible

- IA = représente un pb non déterministe
  NP (non polynomial)

Introduction

- **Les domaines privilégiés de l'IA :**
- **là où il n'y a pas d'algorithme à la portée des machines.**

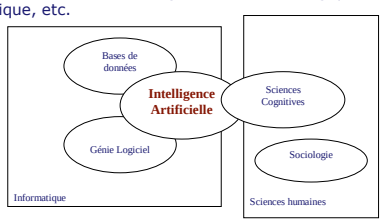
EXp:
Comme les problèmes qui ont une *combinatoire* trop importante ... qui nécessitent une démarche *heuristique*.
- exemple: le jeu d'échecs
- Les heuristiques relèvent de connaissances d'ordre pragmatique et traduisent un savoir-faire, une expérience plutôt qu'un calcul systématique.

- **L'intelligence artificielle a aussi la vocation à simuler le raisonnement humain.**
  - modéliser les modes de raisonnement d'un expert humain
  - les rendre accessibles à un non informaticien.

Introduction

- **L'IA et l'informatique**

  L'IA est une discipline de l'informatique comme les bases de données et le génie logiciel. Elle a beaucoup d'intersections avec les domaines des sciences cognitives[1], de la logique, de la linguistique, etc.



[1] Qui appartient au domaine de la psychologie et qui étudie l'intelligence humaine en analysant et en recréant différentes actions et divers comportements intelligents.

Introduction

- **L'IA et l'informatique**

Les produits de l'IA manipulent généralement les variables symboliques utilisant des opérateurs logiques

(de la même manière que les systèmes classiques qui manipulent des variables numériques utilisant des opérateurs algébriques).

Introduction

- **Qu'est ce que l'intelligence ?**

- Aptitude au calcul rapide ?
- Reconnaissance des formes ou des situations ?
- Aptitude à l'apprentissage ?
- Aptitude à régir à l'environnement ?

- **Consensus sur la définition de l'intelligence** :

- résolution de problèmes difficiles
- capacité d'apprendre, générer des généralisations ou des analogies, l'art de confronter le monde : communiquer, apercevoir, apprendre l'aperçu, etc.

Introduction

Qu'est ce que l'IA ?

1)Rechercher (analyser, résoudre des problèmes, trouver des méthodes de résolution)

2)Représenter des connaissances (logique, règles, mémoire, cas, langue naturelle, etc.)

3)Mettre en application les idées 1) et 2) (Systèmes Experts, pilotes automatiques, agents d'interfaces, robots, Data Mining, etc.)

Définition et objectif

Qu'est ce que l'IA ?

- Elle représente essentiellement la possibilité de concevoir une machine intelligente !
- L'IA est un ensemble de techniques qui tend à implanter, sur un ordinateur, des programmes qui ont des comportements intelligents (perception visuelle et auditive, compréhension, raisonnement, prise de décision, etc.).

Objectif de l'IA ?

- Les recherches en IA tendent à rendre la machine capable:
  - d'acquérir de l'information,
  - de raisonner sur une situation statique ou dynamique,
  - de résoudre des problèmes combinatoires,
  - de faire un diagnostic,
  - de proposer une décision, un plan d'action,
  - d'expliquer et de communiquer les conclusions qu'elle obtient,
  - de comprendre un texte ou un dialogue en langage naturel,
  - de résumer, d'apprendre, de découvrir, etc.

Qu'est ce qu'on attend d'un ordinateur intelligent ?

- comprendre l'esprit mental de l'homme et imiter au maximum son comportement représentent les tâches les plus difficiles que la science essaie de résoudre.

- Par exemple, pour l'interprétation de l'image

Réponse logique : « je vois un ensemble de traits sur l'écran »
Réponse intelligente : « je vois une maison »

⇒ La réponse intelligente est logiquement fausse !!!!

Intelligence et connaissance

Exemple: "Pourriez-vous me passer le sel?"

- L'intelligence ne se définit pas par un processus, mais par les *connaissances* qu'il implique:
  - vision: connaissances des objets physiques
  - langue naturelle: connaissances de grammaire et de vocabulaire
  - résolution de problèmes: connaissances des opérations admissibles

⇒l'ordinateur intelligent a besoin d'une grande quantité de connaissances pratiques: systèmes *basés sur les connaissances*

Historique

1/ La préhistoire

- L'un des premiers problèmes du domaine de l'IA et auquel se sont attaqués certains informaticiens était la traduction automatique.
- Ils pensaient réussir à mettre au point un traducteur automatique dans les cinq ans. Le problème s'avéra nettement plus complexe !
- Cet échec a mené à des questions sur :
  - la représentation à donner aux connaissances
  - la façon d' « extraire » ces connaissances d'un individu

on se rendit compte qu'on ne pouvait pas représenter toutes les connaissances.

Historique

2/ Les débuts (1955-1970)

- L'appellation IA est née en 1956 dans un congrès à Dartmouth par deux jeunes chercheurs :
  - John McCarthy : vision logique de la représentation des connaissances
  - Marvin Minsky : représentations structurées (frames) de stéréotypes de situation incluant différents types d'information
- **Logic Theorist** (Newell, Shaw et Simon, 1956) démonstration de théorèmes de la logique des propositions
- 1959 : Ces derniers développent aussi un système de résolution de problèmes généraux (General Problem Solver, GPS) basé sur l'évaluation de la différence entre la situation à laquelle le système est arrivé et le but qu'il doit atteindre.
- 1960 : McCarthy développe LISP qui va devenir le langage de l'IA pendant les 20 années à venir.
- Début des années 60 : Samuel et Bernstein développent le premier programme capable de jouer aux échecs. La recherche des années 70 dans ce domaine est marquée par l'idée de doter la machine de capacités de mise en œuvre de stratégies sophistiquées évoluant dynamiquement avec le jeu.
- 1965 : Apparition du système ELIZA de production automatique de parole en langage naturel
- 1969 : DENDRAL, un des premiers systèmes experts (analyse d'une spectrographie de masse en chimie)

Historique

3/ La spécialisation

- J.A. Robinson (1965) développe le principe de résolution qui sera à la base de la réalisation de PROLOG et des systèmes de résolution mathématiques
- L'IA va se scinder alors en plusieurs branches :
  - la compréhension du langage naturel
  - la démonstration automatique de théorèmes
  - les jeux
  - la résolution de problèmes généraux
  - la résolution de problèmes experts
  - la représentation des connaissances
  - la perception
  - l'apprentissage, etc.

Historique

4/ L'évolution

- L'évènement majeur des années 80 est l'arrivée en force des japonais dans le domaine de l'IA. Le japon lance le projet de 5ème génération dont le but est de développer à la fois sur le plan matériel et sur le plan logiciel des technologies et des techniques capables de faire de l'IA une discipline efficace.

Langages de l'IA

- **Les principaux langages de l'intelligence artificielle**

– Lisp (1960, J. MacCarthy)
– SmallTalk (1972, A. Kay)
– Prolog (1973, A. Colmerauer), Prolog avec contraintes
– JAVA (1994) , C++, Scheme, ...

IA aujourd'hui

- – on ne pense plus faire une IA à court terme
- **l'IA est partout**
  - objets, agents, méthodologies, représentation des connaissances
  - approches causales, qualitatives
  - fouille de données, fouille de texte
  - statistiques non linéaires (réseaux neuronaux)
  - programmation par contraintes
  - nouvelles méthodes d 'optimisation (évolution artificielle)
- L'IA est **utilisé au quotidien sans le savoir**
  - **Jeux de réflexion sur ordinateur**
  - **Téléphonie mobile**
  - **etc..**

Domaines de l'IA

- **Reconnaissance et synthèse de la parole(ex: réservation d'hôtel)**
- **Reconnaissance et synthèse d'images (ex. recherche d'info, Reconnaissance des visages : avec les réseaux de neurones**
- **Reconnaissance de l'écriture (ex: recon. cheques, codes postaux)**
- **Systèmes experts : MYCIN (diagnostic médical)**
- **Calcul formel : MAPLE, MATHEMATICA.**
- **Representation des connaissances**
- **Simulation du raisonnement humain: logique modale, logique floue.**
- **Traitement du langage naturel**
- **Résolution de problèmes (ordonnancement, planification, etc)**
- **Robotique**
- **Apprentissage**
- **Réseaux de neurones**
- **Systèmes complexes adaptatifs : les algorithmes génétiques.**

Les approches de l 'IA

1/ Approche cognitive

- Elle est pluridisciplinaire,
  - linguistes, informaticiens et psychologues essayent de travailler ensemble.
- Définition (W.J.Rapaport, 1983) : la science cognitive cherche à comprendre les fonctions cognitives humaines en terme d'états mentaux, c'est-à-dire en terme d'algorithmes qui réalisent la transformation de données d'entrée en données de sortie. L'approche calculatoire de la science cognitive dit que :
  - il existe des états mentaux et des processus qui interviennent entre les stimuli d'entrée et les réponses en sortie
  - ces états mentaux et ces processus sont des algorithmes et donc
  - ils sont susceptibles d'être étudiés scientifiquement.

Les approches de l 'IA

1/ Approche cognitive

- ⇒ L'IA est la réalisation de programmes imitant dans leur fonctionnement l'esprit humain.
- Ce type d'approche est généralement utilisé :face à des problèmes trop complexes:
  - pour tenter d'éclairer les problèmes de fiabilité des systèmes
  - reposant essentiellement sur des opérateurs humains
  - pour procurer des aides à la décision aussi compréhensibles que possible pour l'esprit humain.

Les approches de l 'IA

2/ Approche pragmatique

- Pour les pragmatistes:
  - les enseignements apportés à l'IA ne sont pas des fins en soi,
  - mais des moyens pour développer des théories permettant d'améliorer notre capacité à programmer efficacement un ordinateur.
- But :
  - dégager de l'étude du problème des algorithmes, en tenant compte des contraintes imposées par la structure de l'ordinateur
  - on cherche donc à utiliser la machine au mieux de ses capacités.

L'IA pour un pragmatiste est une boite noire :
Données ⇒ Boite noire ⇒ Résultats

Les approches de l 'IA

3/ Approche Connexionniste

Les systèmes symboliques n'ont doné de résultats que sur les domaines où l'être humain a besoin d'apprendre, alors que sur des domaines apparaissant comme innés (langage, la marche), ils se sont montrés inefficaces.

➔Réseaux de neurones

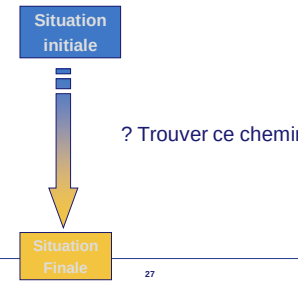
Chapitre 2

Résolution de problème en IA
Par recherche

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Introduction

- Résoudre un pb c'est chercher un chemin qui permet d'aller d'une situation initiale à une situation finale (but)



Introduction

Pour résoudre un problème il arrive qu'on puisse le décomposer en sous-problèmes puis décomposer ceux-ci, etc.,

→jusqu'à n'avoir plus que des problèmes dont la solution est considérée comme immédiatement accessible sans qu'il soit nécessaire de les décomposer à leur tour.

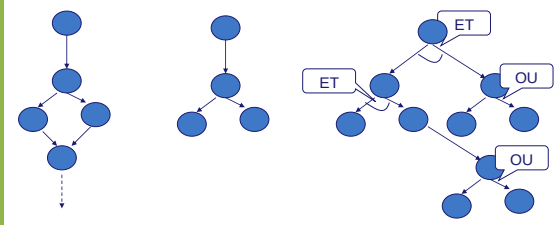
- L'ensemble des décompositions possibles peut être représenté par un " graphe des sous-problèmes ".

→La résolution d'un problème est alors ramenée à la recherche d'un certain sous-graphe du graphe des sous-problèmes.

Introduction

Un graphe sans cycle → arbre

Arbre ET/OU



Certains sommets marquent une conjonction de sous problèmes dont la résolution implique celle du problème décomposé.
D'autres sommets marquent une disjonction de décompositions possibles.

Exemple

- un état est une configuration du tableau 4x4

Etat initial

1	3	14	5
9	11	4	12
6	10	2	8
7		13	15

On distingue:

- **L'état initial**
- Un ou plusieurs **états finaux**

13	3	5	15
9		4	12
6	10	2	8
7	11	1	14

6	3	14	5
9	12	4	
1	10	2	8
7	11	13	15

	3	14	15
9	11	4	12
6	10	2	1
5	8	13	7

Représenter le problème

Il faut définir:

- Les états du problème (abstrait) = ensemble d'états réels
- L'objectif à atteindre: solution (abstraite) = ensemble de chemins-solutions dans le monde réel
- Les opérateurs de transformations (abstrait) = combinaison d'actions réelles (représentation par graphe)

Le monde réel est excessivement complexe

l'espace d'états doit être une abstraction de la réalité

Types de problèmes

- déterministe, accessible → problème à état unique
  - état exact connu
  - effets des actions connus
- déterministe, inaccessible → problème à états multiples
  - un ensemble parmi plusieurs ensembles d'états
  - effets des actions connus
- non déterministe, inaccessible → problème contingent
  (perception limitée, algorithmes plus complexes)
  - besoin de percevoir durant l'exécution
  - solution a une structure d'arbre
  - souvent mélange entre recherche et exécution
  - effet conditionnel des actions
- espace d'états inconnu → problème d'exploration ("online")
  - exécution révèle les états
  - besoin d'expérimenter pour trouver la solution
  - le plus difficile

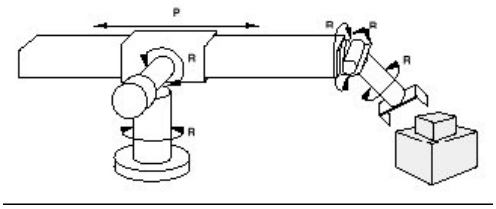
Formulation d'un problème à état unique

Un problème est défini par 4 éléments:

- **état initial**
- **opérateurs** (ou fonction successeur S(x))
- **Test-but**: fonction applicable à un état qui détermine si c'est l'état solution.
- **Coût-chemin**: permet de déterminer quel est le meilleur chemin menant à la solution si plusieurs chemins existent.

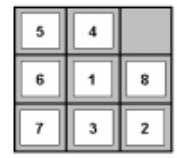
une **solution** est une séquence d'opérateurs menant de l'état initial à l'état final (solution)

Exemple: assemblage automatique

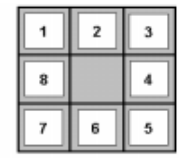


- état initial: coordonnées des articulations du robot et pièces à assembler
- opérateurs: mouvements continus du bras robotique
- test-but: assemblage terminé, robot en position de repos
- coût-chemin: temps d'exécution

Exemple : jeu de taquin (puzzle)



Configuration initiale
Etat initial



Configuration finale
Etat final

- état initial: positions des 8 plaquettes dans une des 9 cases
- opérateurs: déplacer la case vide
- test-but: état courant = état final
- coût-chemin: chaque déplacement coûte 1,
  coût total = nombre de déplacements

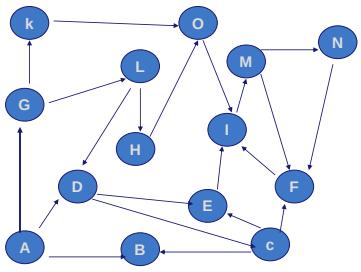
Opérateurs du jeu taquin

- Un opérateur transforme un état en un autre état.
- Il existe quatre opérateurs pour le taquin:
  - déplacer la case vide en haut
  - déplacer la case vide en bas
  - déplacer la case vide à gauche
  - déplacer la case vide à droite

Représentation par graphes d'états

L'application des opérateurs sur les états en démarrant de l'état initial conduit à la construction d'une **arborescence**

Problème général de recherche



Graphe d'état général

?

Représentation par un arbre

États vs. noeuds

• Un état est une représentation d'une configuration physique

• un noeud est un élément d'une structure de donnée constitutive d'un arbre de recherche; il possède les informations de:
– parent, enfants, profondeur, coût de chemin g(x)

Méthodes de recherche

Stratégies d'exploration

• Méthodes de recherche « aveugles » = Explorations non informées

– recherche en largeur
– recherche en profondeur
– recherche en profondeur limitée
– recherche par approfondissement itératif

• Méthodes de recherche heuristiques = Explorations informées

Critères d'évaluation

Les différentes **méthodes de recherche** sont évaluées selon les critères suivants:

• **Complétude**: est-ce que la méthode garantit de trouver une solution si elle existe?
• **Complexité en temps**: combien de temps faut-il pour trouver la solution?
• **Complexité en espace**: quel espace mémoire faut-il pour effectuer la recherche?
• **Optimalité**: est-ce que la méthode trouve la meilleure solution s'il en existe plusieurs?

• Les complexités en temps et en espace sont mesurée en fonction de:
– **b** = facteur de branchement maximum de l'arbre de recherche
– **d** = profondeur à laquelle se trouve le (meilleur) noeud-solution
– **m** = profondeur maximum de l'espace de recherche (espace d'états ou arbre de recherche) - peut être infini

Exercice 1: problème du fermier

- quartes acteurs le fermier(f), le loup(l), la chèvre (c) et le chou (C) se trouvent sur la rive gauche d'une rivière.
- On considère:
  - Un bateau qui peut transporter le fermier seul ou avec un des trois acteurs restants de gauche à droite
  - Un bateau qui peut transporter le fermier seul ou avec un des trois acteurs restants de droite à gauche
  - Le loup peut manger la chèvre sans présence du fermier
  - La chèvre peut manger le chou sans présence du fermier

Pb?
Comment faire pour passer les 4 acteurs à l'autre rive

Algorithme de recherche

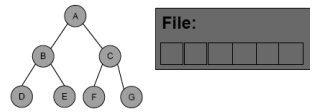
Largeur d'abord
(breadth-first-search)

Principe de la recherche en largeur

- L'expansion des noeuds les moins récemment engendrés s'effectue en premier
- L'arborescence est construite niveau après niveau et donc de manière horizontale

- **Stratégie**: étend le noeud le moins profond
- **Implémentation**:
- insertion des successeurs à la fin de la file d'attente

Exemple largeur d'abord



Ordre de visite: A – B – C – D – E – F - G

Algorithme

1-Insérer le noeud initial s dans une liste appelée OPEN

2-Si OPEN est vide alors échec sinon continuer

3-Retirer le premier noeud de OPEN et l'insérer dans une liste appelée CLOSED. Soit n ce noeud.

4-S'il n'existe pas de successeur alors aller à 2. Engendrer les successeurs de n et les insérer à la queue de OPEN. Créer un chaînage de ces noeuds vers n

5-Si parmi les successeurs, il existe un état final alors succès: la solution est obtenue en suivant le chaînage arrière de ce noeud vers la racine, sinon aller à 2

Propriétés de la recherche en largeur

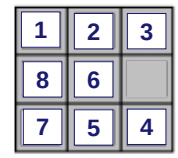
- **Complétude** Oui (si b est fini)
- **Complexité en temps**

$1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$ (exponentiel en d)

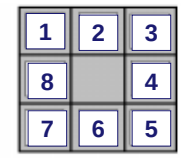
- **Complexité en espace** $O(b^d)$ (il faut garder chaque noeud en mémoire)
- **Optimalité** Oui (si coût = 1 par étape)
  en général pas optimal

Exercice 2 : taquin 3x3

- Appliquer la recherche en largeur d'abord sur la donnée suivante:

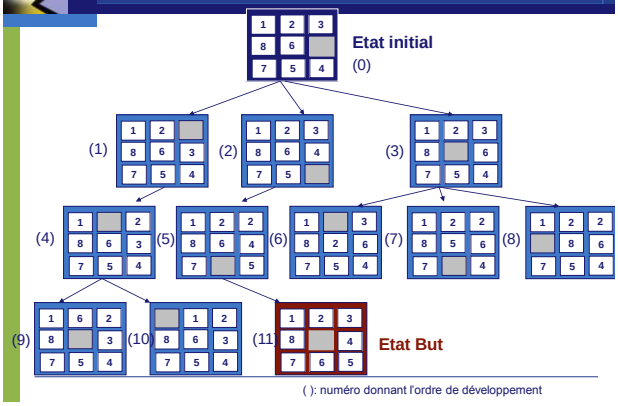


Configuration initiale
Etat initial



Configuration finale
Etat final

Solution Exercice 2



( ): numéro donnant l'ordre de développement

Algorithme de recherche

profondeur d'abord
(depth-first-search)

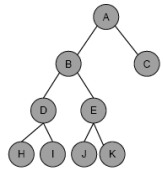
Principe de recherche en profondeur

- **Stratégie**: étend le noeud le plus profond
- **Implémentation**: insertion des successeurs en tête de la file d'attente

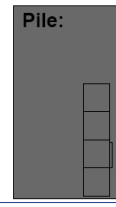
Exemple profondeur d'abord

Attention aux cycles infinis !
Il faut un espace de recherche fini et sans cycle,

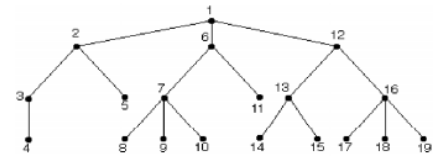
nécessité d'éliminer les nœuds déjà rencontrés.



Ordre de visite: A – B – D – H – I – E – J – K - C



Algorithme de recherche en profondeur



Les nœuds sont numérotés dans l'ordre de leur exploration

Propriétés de la recherche en profondeur

- **Complétude** : Non
  échoue dans les espaces infinis ou avec cycle
  → complet dans les espaces finis acycliques
- **Complexité en temps** :
  $O(b^m)$ = terrible si m est beaucoup plus grand que d
- **Complexité en espace:** O(b * m) linéaire!
- **Optimalité** : Non
- **discussion**: besoins modestes en espace

– pour b = 10, d = 12 et 100 octets/noeud:

- recherche en profondeur a besoin de 12 Koctets
- recherche en largeur a besoin de 111 Tera-octets

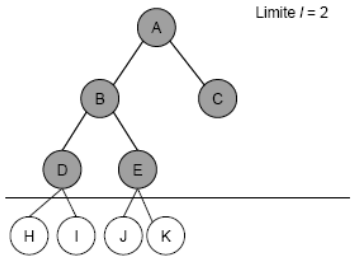
* 10^{10} !!!

Algorithme de recherche

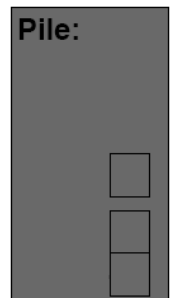
profondeur limitée

Principe de la recherche en profondeur limitée

Limite *l* = 2



Ordre de visite: A – B – D – E - C



Principe de la recherche en profondeur limitée

- algorithme de recherche en profondeur avec une limite de profondeur d'exploration L

Implémentation

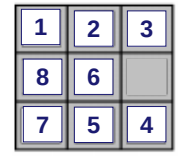
- les noeuds de profondeur L n'ont pas de successeurs
- exemple avec L = 2

Propriétés de la recherche en profondeur limitée

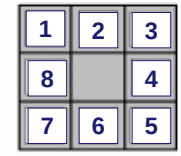
- **Complétude:** Oui si L ≥ d
- **Complexité en temps:** $O(b^L)$
- **Complexité en espace:** O(b * L)
- **Optimalité:** Non

Exercice 3 : taquin 3x3

- Reprendre l'exercice 2 en appliquant la recherche en profondeur limitée à 3 sur la donnée suivante:

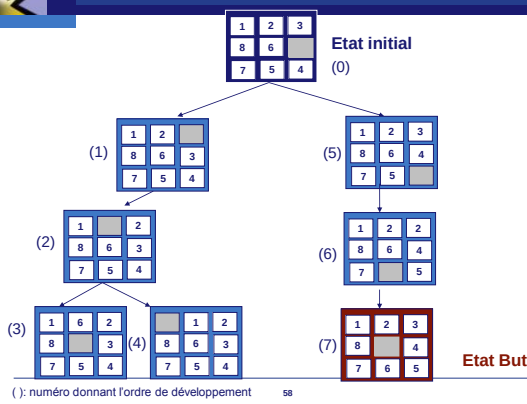


Configuration initiale
Etat initial



Configuration finale
Etat final

Solution Exercice 3



( ): numéro donnant l'ordre de développement

Algorithme de recherche

approfondissement itératif
=
Itérative en profondeur

Principe de la recherche itérative en profondeur

→ **Le problème avec la recherche en profondeur limitée est de fixer la bonne valeur de L**

• approfondissement itératif = essayer toutes les valeurs possibles de L à partir de L = 0 (en incrémentant la limite)

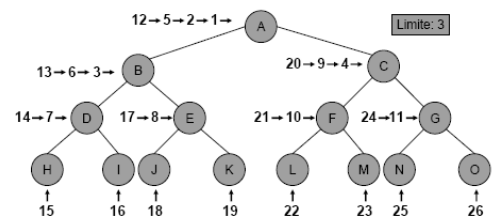
• combine les avantages de la recherche en largeur et en profondeur

– optimal et complet comme la recherche en largeur
– économe en espace comme la recherche en profondeur

• c'est l'algorithme de choix si l'espace de recherche est grand et si la profondeur de la solution est inconnue

Exemple de la recherche itérative en profondeur

Exemple - itérative en profondeur



Ordre de visite: A – A – B – C – A – B – D – E – C – F – G – A – B – D – H – I – E – J – K – C – F – L – M – G – N – O

Propriétés de la recherche itérative en profondeur

- **Complétude** Oui
- **Complexité en temps**
  $(d+1)b^0 + db^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- **Complexité en espace** O(b * d)
- **Optimalité** Oui si coût = 1 par étape

Les méthodes heuristiques (exploration informée)

Introduction aux heuristiques

- Les méthodes aveugles (non informées) sont des méthodes systématiques peu efficaces
- Il existe des limites pratiques sur le temps d'exécution et l'espace mémoire pour appliquer ces méthodes sur une grande catégorie de problèmes
- Toute technique visant à accélérer la recherche est basée sur une information appelée **heuristique**
- Une heuristique signifie '**aider à découvrir**'
- Ils utilisent des sources d'information supplémentaires. Ces algorithmes parviennent ainsi à des performances meilleures
- Les méthodes utilisant des heuristiques sont dites méthodes de recherche heuristiques

Implémentation des méthodes heuristiques

- Utiliser un critère pour réordonner tous les nœuds qui sont explorés (au lieu d'être mis dans une pile ou file)
- Une certaine mesure doit être établie pour évaluer « la promesse » d'un nœud.

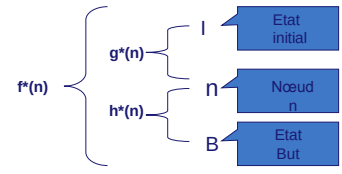
→Cette mesure est appelée **fonction d'évaluation** ou **d'adéquation** ou **objective**

Fonction d'évaluation

- La recherche ordonnée revient à choisir à développer le meilleur nœud au sens d'un certain critère
  ➔ centrée sur le nœud ayant les meilleurs chances de mener au but
- L'utilisation d'une **heuristique** est basée sur une **fonction d'évaluation** pour **ordonner la recherche appelée fonction heuristique** nommée ***h(u)***
- **h est la fonction heuristique qui estime le coût du passage de l'état u à l'état final.**

Méthode d'évaluation

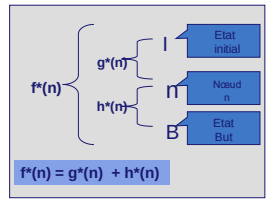
- Soit **f** une **fonction d'évaluation**, **f(n)** exprime la valeur de cette fonction pour le noeud n
- **$f^*(n)$ représente le coût idéal du chemin** passant par un nœud n pour arriver au but



$f^*(n) = g^*(n) + h^*(n)$

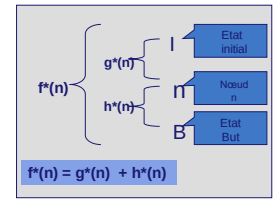
Méthode d'évaluation

- Estimation de **g***
  - g* peut être le coût du meilleur Chemin déjà rencontré de I à n
  - Soit g cette fonction du coût on a g(n) ≥ g* (n)
    - →Le choix de g est très dépendant du domaine traité
    - →Exemple pour le jeu de taquin g(n) le nombre de jeton déplacé (la longueur de la chaîne entre la racine et n)



Méthode d'évaluation

- Estimation de **h***
- **plus difficile car on connaît pas de chemin de n → But**
- Il faut se référer à des informations heuristiques sur le domaine
- → l'algorithme se fondant sur **f(n)** pour ordonner les nœud est nommé **l'Algorithme A**



•**L'algorithme A*** est une stratégie **du meilleur en premier** dans le cadre des graphes OU et des problèmes de minimisation des coûts additifs.
•L'algorithme A* est chargé de calculer le plus court chemin menant de l'état initial à l'état final.

Exercice 4: Recherche heuristique avec A*

- f(n)=g(n) +h(n)

avec

- g : nombre de jetons déplacés
- h: nombre de jetons mal placés

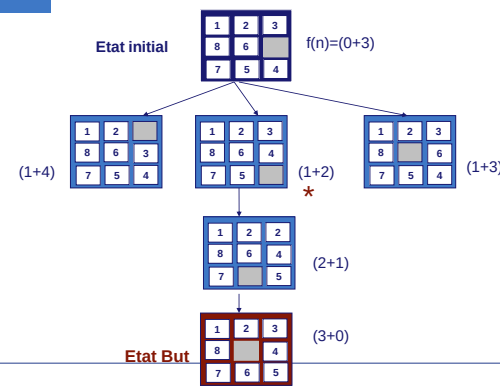
1	2	3
8	6	
7	5	4

Configuration initiale
Etat initial

1	2	3
8		4
7	6	5

Configuration finale
Etat final

Correction Exercice 4



Exercice 5: recherche avec algorithme A*

2	8	3
1	6	4
7		5

Configuration initiale
Etat initial

1	2	3
8		4
7	6	5

Configuration finale
Etat final

Rappel
Mouvements légaux: Déplace le <blanc> vers:
- le haut - la droite
- le bas - la gauche

Contraintes: Les mouvements en diagonal sont interdits

puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés jetons

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)

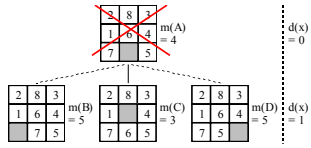
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



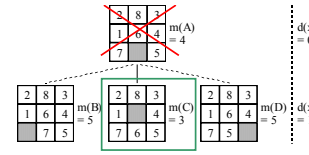
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



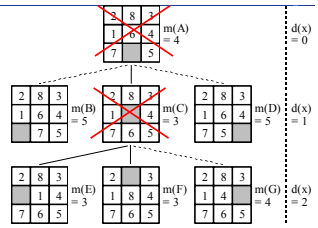
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



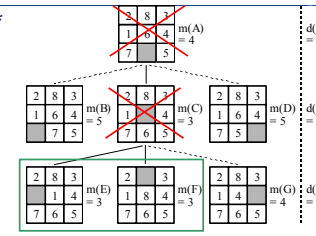
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



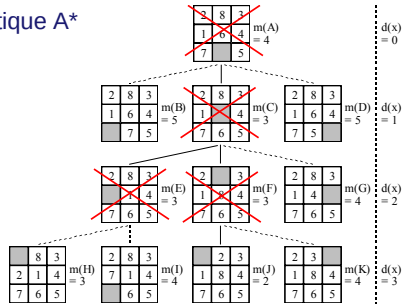
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



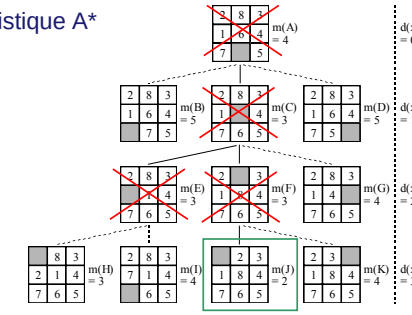
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



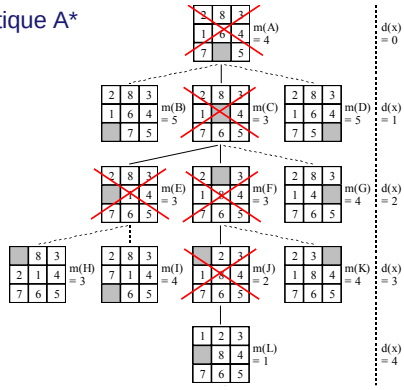
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



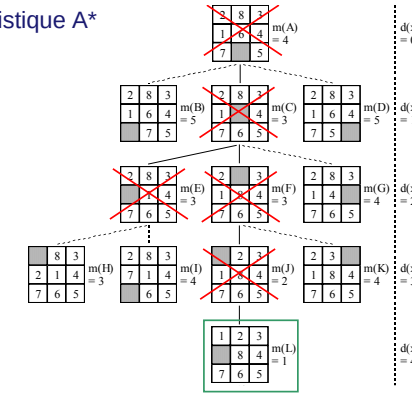
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



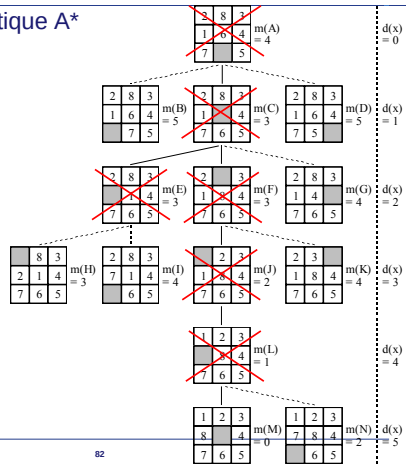
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



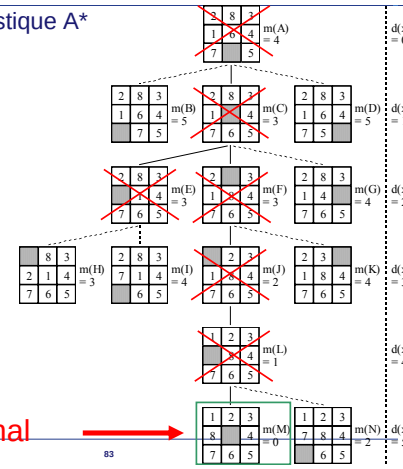
puzzle avec heuristique A*

A, ..., N: mouvements

m = mal placés carreaux

d = niveau dans l'arbre (profondeur)

f(x) = m(x) + d(x)



État final

Conclusion

- Les algorithmes de recherche sont une technologie de base importante en Intelligence Artificielle.
- Recherche d'arbres en profondeur d'abord: utilise peu de mémoire, mais peut rater la solution optimale.
- Recherche d'arbres en largeur d'abord: gourmande en
- mémoire, mais trouve toujours la solution optimale.
- Recherche heuristique: A* peut trouver la solution optimale de manière efficace.
- Votre tour …! Allez: jouez et visualisez
- http://www.dc.fi.udc.es/lidia/mariano/demos/8puzzle/EightPuzzle.htm

Chapitre 3

Résolution de problème en IA

Les stratégies de recherche dans les arbres de jeux : MinMax et Alpha-Beta

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Sommaire

- **Introduction**
- **Arbre de jeu et arbre de recherche**
- **Notion de fonction d'évaluation**
- **Algorithme MinMax**
- **Algorithme Alpha-Beta**
- **Etat de l'art de quelques jeux : dames, échecs, backgammon,Othello et Go**

Introduction

***Johan Huizinga*, historien néerlandais :**

- « Jouer est le propre de l'homme. »

Depuis le Moyen Age (tournois de gladiateurs)

- Jeux : activités comportant des règles précises.
- Vainqueur : celui qui a fait preuve d'intelligence et d'habileté.

But ultime de l'IA

- Réaliser un ordinateur qui joue aussi bien, voire mieux, qu'un humain.

Introduction à la théorie des jeux

La théorie des jeux :

- une branche des mathématiques
- technique de recherche opérationnelle
- s'intéresse aux situations dans lesquelles plusieurs personnes ont à prendre des **décisions** dont dépend un **résultat qui les concerne**.

Problème de jeu = présence de plusieurs **centres de décisions**.

Domaines

- économique
- politique
- diplomatique
- militaire

…

Introduction à la théorie des jeux

Dans les problèmes de jeux, deux facteurs essentiels :

- Coopération
- Compétition

Trois classes de problèmes :

- Jeux de coopération à l'état pur :
  - Étude de décisions concordantes
  - Étude des conditions amenant à un intérêt général
- Jeux de compétition à l'état pur :
  - duels = jeux à deux joueurs dont les intérêts sont strictement opposés
  - Partie la plus achevée de la théorie des jeux
- Jeux de compétition et de coopération :
  - Plus proche des situations réelles
  - Étude systématique plus difficile

Introduction à la théorie des jeux

Jeux à information complète :

Chaque joueur connaît lors de la prise de décision :

- ses possibilités d'action
- les possibilités d'action des autres joueurs
- les gains résultants de ces actions
- les motivations des autres joueurs

Jeux à information parfaite :

Jeux à mécanisme séquentiel, où chaque joueur a connaissance en détail de toutes les actions effectuées avant son choix.

Introduction à la théorie des jeux

Jeux à somme nulle :

Jeux où la somme « algébrique » des gains des joueurs est **constante**. Ce que gagne l'un est nécessairement perdu par un autre.

Jeux à somme non nulle :

Jeux dans lesquels certaines issues sont globalement plus profitables pour tous, ou plus dommageables pour tous.

Jeux synchrones :

Les joueurs décident de leur coup simultanément, sans savoir ce que les autres jouent.

Jeux asynchrones (alternatifs) :

Les joueurs se décident les uns après les autres, en disposant à chaque fois de l'information sur le coup des adversaires.

Classification de certains jeux

Jeux asynchrones, à somme nulle:
→ Echecs, dames, dominos,
→ Jeux de cartes (Poker, rami, belotte, tarot...),

Jeux synchrones, à somme nulle:
→ Feuille-Papier-Ciseaux...

Tous les jeux ne sont pas adaptés à une étude par des techniques d'IA.

Les jeux les plus étudiés : jeux à **somme nulle**, à **information complète** et **parfaite**.

Exploration en situation d'adversaire

Soient 2 joueurs : *MAX* et *MIN*, jouant à tour de rôle.
MAX joue en 1er.

- Construction de l'espace d'états (arbre de jeu), avec **connaissance parfaite** des états :
  - Etat initial
  - Etats finaux
  - Opérateur **déterministe** de génération des états successeurs : aucune incertitude sur les effets de l'opérateur ***succ***.

- Fonction ***eval***, qui indique si un état terminal est gagnant, perdant ou de résultat nul pour le joueur MAX.

- Choix du meilleur coup : algorithme MinMax ou Alpha-Beta

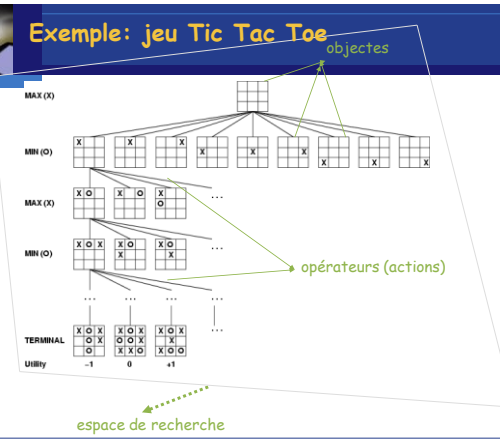
Définition d'un arbre de jeu MinMax

- Un arbre de jeu répertorie l'ensemble des coups réalisables au cours d'une partie, en partant du noeud racine, qui symbolise la configuration initiale, jusqu'aux noeuds terminaux : les positions finales.

- Chaque noeud de l'arbre représente une position de jeu et chaque arc un coup possible d'un joueur permettant de passer d'une position à une autre.

- A chaque noeud terminal est associé un **score ternaire** :
  - **+1 si MAX gagne**
  - **-1 si MIN gagne**
  - **0 s'il y a nulle**

Exemple: jeu Tic Tac Toe

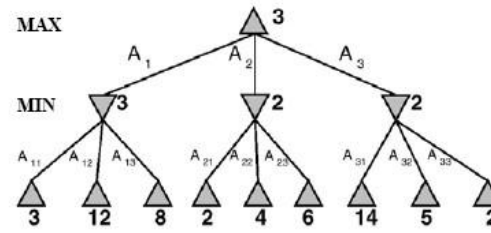


Algorithme MinMax

Principe :
Maximiser la valeur d'utilité pour MAX avec l'hypothèse que MIN joue systématiquement pour la minimiser.

Description :
- étendre l'arbre de jeu
- évaluer chaque noeud terminal
- propager ces valeurs aux noeuds non-terminaux :
  - la valeur minimum aux noeuds du joueur MIN
  - la valeur maximum aux noeuds du joueur MAX

Exemple d'Arbre de jeu MinMax



Propriété de l'algorithme MinMax

Complet : si l'arbre de jeu est fini.
Complexité en temps : O(b^m)
b = facteur de branchement
m = profondeur de l'arbre total

Complexité en espace : O(b^m)
Pour les échecs : b ≈ 35, m ≈ 100...

NB: L'algorithme MinMax est totalement intraitable pour certains problèmes.

MinMax avec profondeur limitée

Principe :

- Etendre l'arbre de jeu jusqu'à une profondeur N à partir du nœud courant
- Calculer la valeur de la fonction d'évaluation pour chaque nœud feuille, pas forcément terminal
- Propager ces valeurs jusqu'aux noeuds non-terminaux

MinMax avec profondeur limitée

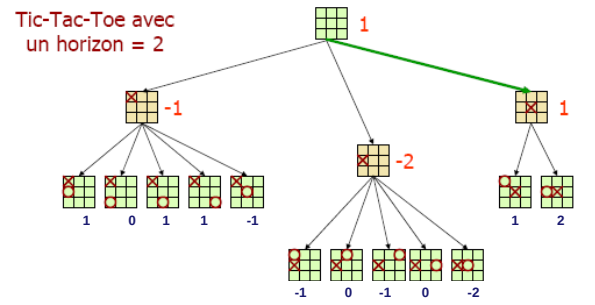
1) Etendre l'arbre de jeu à partir de l'état courant, où c'est à MAX de jouer, jusqu'à une **profondeur h**

2) Calculer la fonction d'évaluation pour chacune des feuilles

3) Rétro-propager les valeurs des noeuds feuilles vers la racine de l'arbre, de la manière suivante :
   - Un noeud MAX recoit la valeur maximum de l'évaluation de ses successeurs
   - Un noeud MIN recoit la valeur minimum de l'évaluation de ses successeurs

4) Choisir le mouvement vers le noeud MIN qui possède la valeur rétro-propagée la plus élevée.

Exemple MinMax avec profondeur limitée



Elagage Alpha-Beta

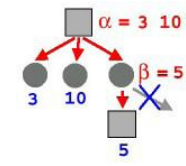
Principe :

- Etendre l'arbre de jeu jusqu'à une profondeur h par une recherche en profondeur
- Ne plus générer les successeurs d'un noeud dès qu'il est évident que ce noeud ne sera pas choisi, compte tenu des noeuds déjà examinés
- Chaque **noeud MAX** conserve la trace d'une alpha-**valeur**, qui est la valeur de son **meilleur successeur** trouvé jusqu'à présent
- Chaque **noeud MIN** conserve la trace d'une **ß-valeur**, qui est la valeur de son **pire successeur** trouvé jusqu'à présent
- Valeurs initiales : **alpha = -∞ et ß = +∞**

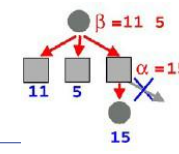
Elagage Alpha-Beta

Deux règles :

- Interrompre la recherche d'un noeud MAX si son alpha-**valeur** ≥ **ß-valeur** de son nœud parent.



- Interrompre la recherche d'un noeud MIN si sa **ß-valeur** ≤ alpha-**valeur** de son nœud parent.



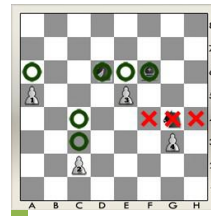
Etat de l'art :jeu d'échecs

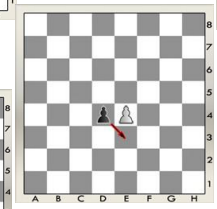
Position initiale

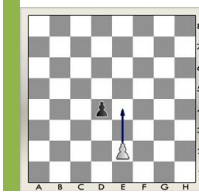
Quelles sont les règles du jeu ?

Règles du jeu

Déplacement du pion

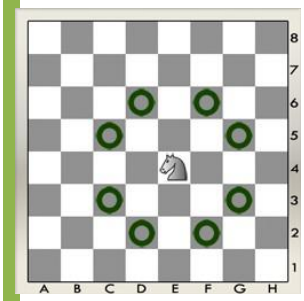




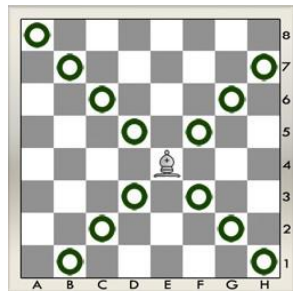


Règles du jeu

déplacement du cavalier

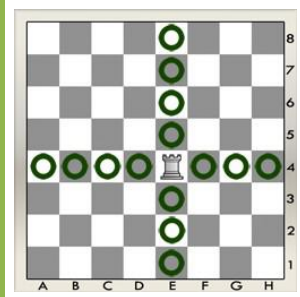


déplacement du fou

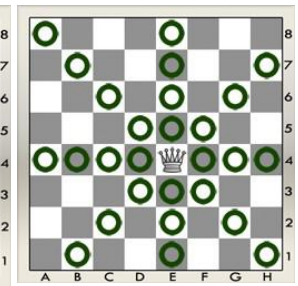


Règles du jeu

déplacement de la tour

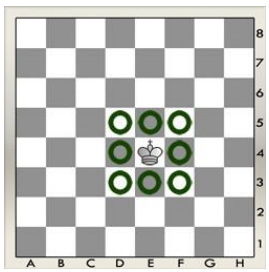


déplacement de la reine



Règles du jeu

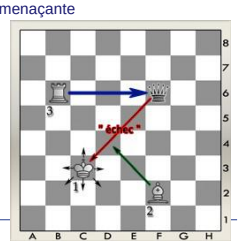
Déplacement du Roi



Echec au roi lorsqu'une pièce attaque le roi ennemi.

Trois manières de parer un échec :

- Bouger le roi attaqué
- Prendre la pièce attaquant le roi
- Intercaler une pièce entre le roi et la pièce menaçante



Règles du jeu

... encore d'autres règles et contraintes

- Jeu qui a suscité le plus d'attention et d'efforts.
- Au début les progrès ont été très lents.
- 1970 : 1er programme à gagner le championnat ACM d'échecs informatiques. Il utilisait un élagage a-b, un répertoire d'ouvertures classiques et un répertoire de fins de partie.
- 1985 : Hitech (*Berliner*) se classe parmi les 800 meilleurs joueurs du monde. Il est capable d'analyser plus de 10 Millions de positions par coup.

Techniques utilisées pour les échecs

Le milieu de partie

- Royaume de l'Alpha-Beta et de la fonction d'évaluation
- Fonction d'évaluation extrêmement complexe. Prend en compte :
  - Valeurs des pièces :
    - *Dame : 9*
    - *Tour : 5*
    - *Cavalier, fou : 3*
    - *Pion : 1*
  - Sécurité du roi
  - Paire de fous
  - Domination du centre
  - Mobilité
  - Cavaliers centralisés
  - Tours placées sur des colonnes ouvertes
  - Fous placés sur des diagonales ouvertes
  - Qualité de la structure de pions
  - ...

Techniques utilisées pour les échecs

L'algorithme **Alpha-Beta** utilisé dans un programme d'échecs incorpore quelques améliorations :

- Élagage de futilité
- Test prioritaire des coups meurtriers
- Retour à l'équilibre
- Heuristique du coup nul (*Null Move Heuristic*)

Fonction d'évaluation généralement de la forme :

eval(s) = w1 f1(s) + w2 f2(s)+ ... + Wn fn(s) avec :

- fi(s) : caractéristique de la position
- wi : poids associé à une caractéristique

Inconvénients de Alpha-Beta

Premier défaut : manque total de stratégie

- Programmes utilisant ∞-ß ne sont pas guidés par un plan
- Jouent des positions totalement indépendantes les unes des autres
- Pas de vision à long terme de la partie

Deuxième défaut : l'effet d'horizon (*Berliner*, 1974)

- Dû à la nécessité de fixer a priori une profondeur
- Programme ne peut percevoir les effets d'un coup au delà de la profondeur choisie
- Tendance à repousser au delà de son horizon une position défavorable

Chapitre 4

Représentation des connaissances et raisonnement

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Sommaire

- Introduction
- Définitions
- Les types de connaissances
- Représentation des connaissances
- Logique des propositions : ordre 0
- Logique des prédicats : ordre 1
- Réseaux sémantiques

Introduction

- ***L'homme***
- ***a des connaissances***
- ***a une représentation de ces connaissances***

} Pour résoudre un pb

La qualité d'un système intelligent est celle de sa base de connaissance BC

Introduction

- **Exemple 1:**
- « Robert est allé à Paris »
- ⇒ Pourrait être représentée telle quelle : x1, x2, .., xn ⇒ ensemble de listes
- Inconvénient : une représentation des données mais pas des connaissances.
- On ne peut pas répondre à : qui est allé à Paris ?
- Autre représentation : Moyen formel de représentation des connaissances.
  - Action : Aller
  - Agent : Robert
  - Source : ?
  - Destination : Paris
  - Temps : Passé
  - Moyen : ?

Introduction

- **Exemple 2:**
- « Sami entra dans un restaurant. Il commanda de la viande. Il n'a pas laissé de pourboire. »
- ⇒ Sami a mangé
- ⇒ Sami s'est assis
- ⇒ Sami n'est pas végétarien
- ⇒ Sami est radin
- ⇒ Sami a passé un certain temps dans le restaurant
- ⇒ ...

Introduction

- Comment réaliser toutes ces inférences ?
  - Un programme ne peut réaliser toutes les inférences possibles : explosion combinatoire.
  - On ne peut pas réaliser des inférences à la demande.
- ⇒ Il faut contrôler le raisonnement.

Dans un système intelligent on a 3 composants

- ***Une BC***
- ***Une partie pour faire les inférences (raisonnement) appelé moteur d'inférence ou interpréteur : I***
- ***Une structure de contrôle pour orienter le raisonnement: C***

SI = BC + I + C

Définitions

- **Définition de la connaissance** :
- Faculté de connaître, manière de comprendre, de percevoir.
- Connaître : avoir une idée plus ou moins juste, savoir de façon plus ou moins précise.

- **Définition de la représentation** :
- Action de rendre sensible quelque chose au moyen d'une figure, d'un symbole, d'un signe.
- Ex : l'écriture est la représentation de la langue parlée.

Les types de connaissances

- On entend par connaissances toutes les formes de savoir de l'homme
  - des faits: des définitions (la terre est ronde)
  - des événements : aspects temporels (x a rencontré y en 1988)
  - des inférences : (s'il tousse ….)
  - des règles de savoir faire (pour sortir du parking il faut …)
  - des méta connaissances: connaissances sur les connaissances

Représentation des connaissances

- Le problème de la représentation des connaissances
  → est celui de leur transcription sous une forme symbolique qui puisse être exploitée par un système de raisonnement (moteur d'inférence).

- Un mode de représentation associe ainsi deux aspects imbriqués :
  - la structure de données pour représenter l'information
  - la méthode associée d'exploitation de cette information ou de raisonnement.

Idée de base

- Avoir la connaissance stockée par morceaux dont l'ensemble fournit la connaissance entière et permet de raisonner sur cette connaissance.
- Il y un ensemble de modes de représentation :
  - Logique (logique d'ordre 0, logique d'ordre 1, temporelle, floue, …)
  - Réseaux sémantiques
  - Règles de production
  - Objets structurés (frames)
  - Approche orientée objet

Mode de représentation logique
Logique des propositions (d'ordre 0)

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Logique des propositions

Exemple :

- Toto vole

Vole (Toto)

prédicat

argument

- Toto est un oiseau

Oiseau (Toto)

prédicat

argument

Proposition vraie

Logique des propositions

- Un prédicat peut avoir plusieurs arguments

Exemple

argument1

argument2

Robe(Fatma, rouge)

prédicat

- La robe de Fatma est rouge
- Fatma a une robe rouge

Logique des propositions

- Déduction mathématique : déduire de nouvelles connaissances des anciennes.
- Le calcul des propositions se définit :
  - d'une part par sa syntaxe régissant l'ensemble des assertions exprimables dans le langage
  - et d'autre part par ses règles d'inférence décrivant comment on peut créer de nouvelles assertions à partir des anciennes.

Logique des propositions

Syntaxe

- Une **proposition** : vrai, faux
- **Variables propositionnelles** ou **atome** (affirmation) : P, Q, R, A1, A2, etc.
- Connecteurs (pour représenter des propositions plus complexes) :
  - ET : $\wedge$
  - OU : $\vee$
  - NON : $\neg$
  - Implique : $\rightarrow$
  - Equivalent : $\leftrightarrow$

Exemples :
$(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$
$(\neg P \vee (Q \wedge R)) \rightarrow S$

Logique des propositions

- P : tous les grecs sont mortels
- Q : Socrate est grec
- donc Socrate est mortel :R

$(P \wedge Q) \rightarrow R$ FBF(Forme Bien Formée)

Alphabet du langage :

- Ensemble des atomes {A, …, An } + ensemble les connecteurs {∧ ,…, →} + {(,)}

Définition récursive d'une **FBF**

- FBF
  - 1. Un atome est une FBF
  - 2. Si G est une FBF alors ¬G est une FBF
  - 3. Si G et H sont des FBF alors (G∨H), (G∧H), (G→H) et (G↔H) sont des FBF
  - Toutes les FBF sont construites à partir de 1, 2 et 3.
- Une interprétation I est une certaine combinaison des atomes A1, A2, …, An d'une formule G .
  Une formule est vraie ou fausse dans une interprétation I.
- Une formule est **valide** : si pour toute I, une FBF est vraie (sinon, elle est invalide)

Insuffisance de la logique (d'ordre 0)

- Si on veut déduire des propositions pour des ensembles d'éléments ???
- Exemple :

? exprimer que tous les oiseaux volent

- Vole(oiseau 1), Vole(oiseau 2), …, Vole(oiseau n)

? exprimer que certains oiseaux ne volent pas

…. (on ne peut pas)

Mode de représentation logique
Logique des prédicats
(d'ordre 1)

INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للاعلامية

la logique (d'ordre 1)

- C'est la logique des propositions d'ordre 0 à laquelle on ajoute:
  - Le quantificateur universel (∀)
  - Le quantificateur existentiel (∃)

Exemple :

? exprimer que tous les oiseaux volent

- ∀ x oiseau (x) → Vole(x)

? exprimer que certains oiseaux ne volent pas

- ∃ x oiseau (x) ∧ ¬ Vole(x)

Syntaxe

- Pour écrire des formules de logique des prédicats, on commence par se donner un vocabulaire:

- variables (x, y, z, x1, y1, z1, …)
- constantes (a, b, c, a1, b1, c1, …)
- fonctions (f, g, h, f1, g1, h1, …)
- prédicats (P, Q, R, P1, Q1, R1, …)
- parenthèses
- connecteurs logiques: ¬ , ^, v, →, ↔
- quantificateurs: ∀, ∃

Exercice

Mettre sous forme de formules les propositions suivantes :

S1. Pour tout crime, il ya quelqu'un qui l'a commis
S2. Seul les gens malhonnêtes commettent des crimes
S3. Ne sont arrêtés que les gens malhonnêtes
S4. Les gens malhonnêtes arrêtés ne commettent pas de crime
S5. Il y a que des crimes
S6. Il y a des gens malhonnêtes non arrêtés

Correction

Mettre sous forme de formules les propositions suivantes :

- *S1. Pour tout crime, il ya quelqu'un qui l'a commis*

C(X) : X est un crime
Commettre(Y, X) : Y a Commis X
(∀X), C(X) --> (∃Y) Commettre(Y,X)

- *S2. Seul les gens malhonnêtes commettent des crimes*

M(Y) : Y est malhonnête
(∀X) (∀Y) C(X) ^ Commettre(Y,X) --> M(Y)

- *S3. Ne sont arrêtés que les gens malhonnêtes*

(tout objet/ si l'objet est arrêté alors cet objet est malhonnête)

A(X) : X est arrêté
(∀X) A(X) --> M(X)

Correction suite

- *S4. Les gens malhonnêtes arrêtés ne commettent pas de crime*

(pour toute personne malhonnête et arrêté, il n'existe pas de crime commis par elle)

C(X) : X est un crime
Commettre(Y, X) : Y a Commis X
M(Y) : Y est malhonnête
A(X) : X est arrêté

(∀X) M(X) ^ A(X) → ¬ (∃Y) (C(Y) ^ Commettre(X,Y))

- *S5. Il y a que des crimes*

(∀X) C(X)

- *S6. Il y a des gens malhonnêtes non arrêtés*

(∃X) M(X) ^ ¬ A(X)

Raisonnement en logique des prédicats

- Raisonner en logique → démontrer des nouvelles formules à partir d'un ensemble de formules existantes.
- Règles ??

Règles d'inférence

Règles d'inférence (règles de dérivation) : **opérateur d'inférence** |=

alors

- ***Modus ponens*** *(implication - élimination)*
  - Si A et (A ⇒ B) alors on déduit B
  - On note {A, A ⇒ B} |= B
- ***Modus tollens***
  - Si ¬B et (A ⇒ B) alors on déduit ¬A
  - On note {¬B, A ⇒ B} |= ¬A
- ***Enchaînement***
  - Si A ⇒ B et B ⇒ C alors on déduit A ⇒ C
  - On note {A ⇒ B,B ⇒ C} |= A ⇒ C

Règles d'inférence

- ***Unification et filtrage***

Exemple

- Si Français(jean) et Français(y) → Région(y, Europe)

d'après

Modus Ponens

- ***Modus ponens*** *(implication - élimination)*
  - Si A et (A ⇒ B) alors on déduit B
  - On note {A, A ⇒ B} |= B

Région(jean, Europe)
Avec substitution de jean à la variable y

- ***Spécification universelle***
  - ∀(x) P(x) alors P(a)

Inconvénients

- Logique des prédicats du 1er ordre (V ou F)
  - Ne permet pas d'exprimer des nuances
  - Ne permet pas de décider avec des informations manquantes
  - → Contrairement à l'homme qui fait des raisonnements par défauts

-Logiques multivaluées
-Logique modales
-Logique non monotones
-Logique temporelle
-Logique floue
-…

Mode de représentation Réseaux Sémantiques


INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Les Réseaux sémantiques

- Les réseaux sémantiques sont une manière de représenter des relations entre des objets (nœuds). C'est un graphe étiqueté
- Les nœuds (objets) sont reliés entre eux et les liens ont une signification.
- Les liens sont orientés car la relation n'est pas symétrique.

Exemple :
- la pomme a une couleur rouge
- un canari est une sorte d'oiseau
- une pompe centrifuge est une sorte de pompe
- la pompe P1 est une pompe
- la pompe a un moteur

NB: Certains liens reviennent très souvent dans les réseaux

Représentation

- Type de lien :
  - est un (IS a)

    (est un exemple de, est une instance, est un élément de)
  - est une sorte de (kind of)

    (est une sous-classe de, est un sous-ensemble de)
  - a un (attribut)
- Symbolique :

Exercice

Représenter par un réseau sémantique les connaissances suivantes:

- Karim est secrétaire et travaille pour Ahmed;
- karim et Ahmed sont des humains;
- karim et Ahmed travaillent au département RH;
- karim a 30 ans et a les yeux bleus;
- Ahmed est directeur;
- Les directeurs ont des voitures de service;
- Les employés ont un permis de stationnement

Exercice

Représenter par un réseau sémantique chaque connaissance indépendamment
(vous pouvez enrichir les connaissances)

1. Ali a frappé salah
2. Ali a frappé salah et mohamed a frappé ramy
3. Ali a frappé salah avec un bâton dans le parc la nuit dernière
4. Sonia a donné à alia un livre

Chapitre 5

Systèmes Experts et Moteurs d'Inférence


INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Sommaire

- Introduction
- Structure d'un Système Expert
- Cycle d'un moteur d'inférence
- Caractéristique d'un moteur d'inférence
- 1/ Mode d'invocation des règles
  - Chainage Avant
  - Chainage Arrière
  - Chainage mixte
- 2/ La stratégie de recherche
- 3/ Régime de contrôle
- 4/ Critère de monotonie
- Exercices d'application

Introduction

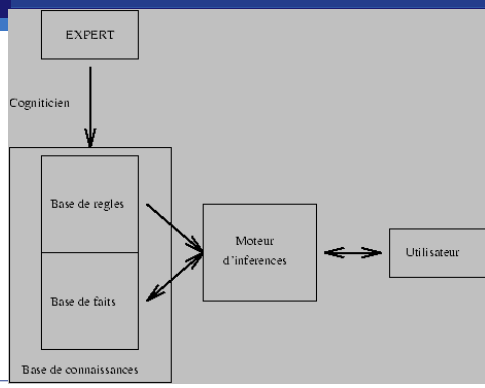
- Un **système expert** est un logiciel qui reproduit le comportement d'un expert humain accomplissant une tâche intellectuelle dans un domaine précis.

Structure d'un SE

Un système expert (SE) est composé de deux parties indépendantes :

- une **base de connaissances** (mémoire d'un SE)
  - **base de faits (BDF)** qui contient les séquences de faits établis et ayant une valeur de vérité vraie (constitue la partie statique)
  - **base de règles (BDR)** qui contient l'ensemble des règles de production pouvant être appliqués aux faits. (il s'agit de la partie dynamique)
- un **moteur d'inférences (MI)** Il s'agit du cerveau du SE. C'est un programme qui simule le raisonnement humain

Structure d'un SE



Cycle d'un Moteur d'inférence

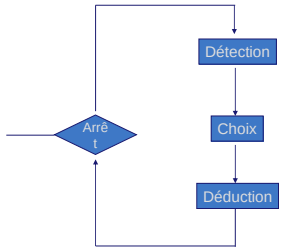
Le moteur d'inférence fonctionne en deux phases:

- Phase d'évaluation
  - **1/ Etape de Filtrage ou détection :** Elle consiste à définir pour l'ensemble des règles de BC, les règles potentiellement applicables → résultat ensemble de règles
  - **2/ Etape de Sélection avec réduction de conflits:** Elle consiste à choisir parmi l'ensemble des règles applicables, la règle à appliquer effectivement → résultat une règle
- Phase d'exécution
  - Elle consiste à appliquer la règle choisie et mettre à jour la base de faits BDF

Le moteur d'inférence exécute ces différentes phases de façon cyclique jusqu'à une condition d'arrêt soit vérifiée:

- Un objectif atteint
- Epuisement de toutes les connaissances
  - Etape de filtrage ne fournit aucune règles potentiellement applicable
  - Epuisement normale des règles

Cycle d'un Moteur d'inférence



Caractéristiques d'un Moteur d'Inférence

La programmation d'un MI nécessite la sélection d'un ensemble de critères:

- 1/ Mode d'invocation des règles
- 2/ La stratégie de recherche
- 3/ Régime de contrôle
- 4/ Critère de monotonie

1/Mode d'invocation des règles

La programmation d'un MI nécessite la sélection d'un ensemble de critères:

- 1/Mode d'invocation des règles
- 2/ La strat
- 3/ Régime
- 4/ Critère

•MI à approche de chainage avant
•MI à approche de chainage arrière
•MI à approche de chainage mixte

MI à chainage avant

1) *Les MI à chaînage avant* : un MI à chaînage avant (forward chaining) détermine le résultat (ou but) à partir de la BDF. Les règles à *déclencher* à chaque cycle sont celles dont les prémisses appartiennent à la BDF. L'exécution de ces règles modifie la BDF et d'autres règles peuvent alors être déclenchées au cycle suivant. Ce principe sera répété jusqu'à ce que le but soit dans la BDF.

Exemple :

soit BDF = {A, B, C, D}

et soit la règle : A,C → G

cette règle est déclenchable et on obtient BDF = { A, B, C, D, G }

L'étape de génération de conflit correspond à rassembler toutes les règles dont les prémisses sont dans la BDF.

MI à chainage arrière

2) *Les MI à chaînage arrière* : un MI à chaînage arrière (backward chaining) détermine l'ensemble des règles qu'il faut invoquer pour aboutir au but (fait à établir). Le but sera alors remplacé par les prémisses qui vont constituer les nouveaux faits à établir. Les règles à tirer à chaque cycle sont alors celles dont les conclusions sont égales au (ou contiennent le) fait à établir. L'exécution de ces règles ne modifie pas la BDF mais elle remplace le problème initial par d'autres. Ce type de raisonnement correspond à une décomposition du problème en un ensemble de sous problèmes.

Exemple :

Soit le but à établir P, et soit la règle :

A, B, C → P

Pour résoudre le problème P, il suffit de résoudre les sous problèmes A, B et C.

Ce mécanisme sera répété jusqu'à ce que la dernière liste de faits à établir soit entièrement dans la BDF.

L'étape de génération de conflit consiste à rassembler toutes les règles dont les conclusions contiennent le fait à établir.

MI à Chainage Mixte

3) *Les MI à chaînage mixte* : Ce type de moteur d'inférence peut être choisi lorsqu'une partie des faits du problème est à établir, l'autre est considérée déjà établie. Les conditions de déclenchement des règles dans ce cas peuvent porter sur les deux types de faits. Donc, pour résoudre le problème on sera amené à prendre en considération les faits déjà établis et de remplacer le problème à résoudre par une liste de sous problèmes. Ceci, correspond à un mode de raisonnement mixte faisant intervenir simultanément le chaînage avant et le chaînage arrière.

2/ La stratégie de recherche

La programmation d'un MI nécessite la sélection d'un ensemble de critères:

- 1/Mode d'invocation des règles
- 2/ La stratégie de recherche
- 3/ Ré…
- 4/ …

Au cours des cycles de recherche d'un MI, on développe un **arbre de recherche** dans lequel chaque niveau correspond à l'ensemble des règles applicables (ensemble de conflits) Chaque règle déclenchée crée une nouvelle situation et de nouvelles règles à invoquer.

Deux principales stratégies de recherche se présentent:

- soit on développe toutes les règles d'un même niveau l'un après l'autre avant de passer au niveau suivant **(stratégie en largeur d'abord)**
- soit d'un niveau à un autre à chaque fois qu'on déclenche une règle et on ne revient aux règles restantes que si on épuise toutes les règles en **profondeur (stratégie en profondeur d'abord)**

→ Le retour arrière dans le cas où la recherche en profondeur échoue sera appelé: **backtracking**

3/ régime de contrôle

La programmation d'un MI nécessite la sélection d'un ensemble de critères:

- 1/Mode d'invocation des règles
- 2/ La stratégie de recherche
- 3/ Régime de contrôle
- 4/ Critère de monotonie

Le régime de contrôle d'un MI peut être:

- **Irrévocable:**
  - L'application d'une règle dans un cycle du MI n'est jamais remise en cause et on n'opère pas de backtracking. S'il n'y a plus de règles à appliquer. Le MI s'arrête et signale un échec sans faire retour en arrière
- ou **par tentative:**
  - Ce régime peut remettre en cause des règles déjà appliquées si elles n'ont pas abouti, et faire un backtracking en retirant aussi les faits qui en étaient déduits

4/ Critère de monotonie

La programmation d'un MI nécessite la sélection d'un ensemble de critères:

- 1/Mode d'invocation des règles
- 2/ La stratégie de recherche
- 3/ Régime de contrôle
- 4/ Critère de monotonie

Un moteur d'inférence MI peut être :

- monotone:
  - En régime monotone: le MI ne fait qu'ajouter des faits à la BDF et n'élimine jamais une règle de BDR
- non monotone
  - En régime non monotone: le MI peut en cas de retour arrière par exemple retrancher de la BDF un fait précédemment ajouté

Exercice d'application

- Base de Fait (BDF): { B, C }
- But : H
- Base des règles (BDR):
  - R1: B^D^E →F
  - R2: D^G →A
  - R3: C^F →A
  - R4: B→X
  - R5: D→E
  - R6: A^X→H
  - R7: C→D
  - R8: X^C→A
  - R9: X^B→D

Résoudre ce problème en appliquant l'approche de **chainage avant** selon une stratégie en **profondeur d'abord monotone** avec régime irrévocable :

- La 1ère règle qui apparaît dans l'ensemble des règles détectées est celle à appliquer
- Une règle appliquée est définitivement écartée de BDR

Solution

Solution 1

Sous forme d'un tableau

Numéro d'inférence	Filtrage	sélection	déduction

Solution 2

Sous forme d'un arbre de recherche en profondeur d'abord

Questions?

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Bibliographie Sommaire

- JGanascia, L'intelligence Artificielle, Coll. DOMINOS, Flammarion, 1993
- Ginsberg, Essentials of Artificial Intelligence, Morgan Kaufmann, 1997.
- Laurière, Intelligence Artificielle : résolution de problèmes par l'homme et la machine, Eyrolles, 1987
- Russel et Norvig, Artificial Intelligence a Modern Approach, Prentice Hall Series in Artificial Intelligence, 1995
- Cornuéjols et Miclet, Apprentissage artificiel :Concepts et algorithmes, Eyrolles, 2010.
- Alliot et Schiex, Intelligence artificielle et informatique théorique, CEPADUES, 2002.