Rappels / concepts de base de l'IA

D'après le support de Jean-Marc Fouet

Complexité / algorithmes

- s <- 0
  pour i variant de 1 à n
  s <- s + i
  finpour
  imprimer s
  - Complexité O(n)
- lire n
  imprimer (n * (n + 1) / 2)
  - Complexité O(1)
- Illustration

Problème (point de vue IA) [1]

- Solution d'un problème = séquence des états permettant de mener de l'état initial (ce que l'on sait au début du problème) à un état final satisfaisant un but à atteindre
- Modéliser un problème
  - Décrire un état de départ
  - Définir les opérateurs pour passer d'un état à l'autre
  - Pouvoir construire l'espace des états
  - Disposer d'un test de satisfaction du but
  - Savoir construire un chemin de l'état départ à l'état final
  - Disposer d'une fonction donnant le coût de l'application d'un opérateur

Problème (IA) [2]

- Rechercher une solution = explorer l'espace des états pour construire un chemin menant à un état final satisfaisant le but
- Exemple avec l'algorithme A*:
  - http://www.ccg.leeds.ac.uk/james/aStar/
- En savoir plus sur A*
  - http://www.geocities.com/SiliconValley/Lakes/4929/astar.html

Complexité problème = nb d'opérations pour trouver un chemin…

- Jeu d'échec
- Pour la recherche d'une solution gagnante (Mat ou Pat de l'adversaire), la recherche systématique dans l'arbre des états possibles d'un échiquier
- Complexité : $O(p^n)$
- Illustration

Heuristique

- Quelque chose que l'on sait sur le domaine du problème et qui nous permet de choisir un « chemin prometteur » dans l'espace des états.
- D'une manière plus générale, quelque chose que l'on sait qui « oriente » la recherche de la solution.

Comment exprimer ce que l'on sait séparément de l'algorithme qui restera très général…

- Passer d'une programmation « procédurale » (développement incrémental avec essais et erreurs) => pas d'explications faciles à donner, difficulté à faire évoluer…
  - À une programmation « déclarative » autorisant la déclaration de « ce que l'on sait » avec un mécanisme d'inférence simple et général => la logique

Rappels de logique : système formel

- Pour construire une langue (par exemple le français), on a besoin de 4 choses :
  - un alphabet (a, b, ..., z, blanc, virgule, parenthèse ouvrante, ...)
  - un procédé de formation des mots, qui est la concaténation
  - un dictionnaire, qui permet de savoir que "chat" est français, alors que "cat" ne l'est pas
  - des règles de grammaire, qui permettent de savoir que "chattes" est français, alors qu'il n'est pas dans le dictionnaire.
- Pour construire un système formel, nous aurons besoin de 4 choses analogues
  - un alphabet, ensemble de symboles pas nécessairement réduit à des caractères
  - un procédé de formation des expressions, pas nécessairement la concaténation
  - un ensemble d'*axiomes*, c'est-à-dire d'expressions obéissant aux deux premiers points ci-dessus, et dont on décide arbitrairement qu'ils appartiennent au système
  - des règles de dérivation qui, à partir des axiomes, permettent de produire des *théorèmes* (c'est-à-dire des expressions appartenant au système), et peuvent ensuite s'appliquer aux théorèmes pour en produire d'autres

Exemple de système formel

- **Système PEU**
  - alphabet = l'ensemble des trois symboles "p" , "e" , et "u"
  - p.f.e. = concaténation
  - axiome = upueuu
  - règles :
    - R1 : si une expression de la forme AeB est un théorème (où "A" désigne n'importe quelle suite de "u", de "p", et de "e", et B de même), alors l'expression uAeBu est aussi un théorème
    - R2 : si une expression de la forme AeB est un théorème, alors l'expression AueuB est aussi un théorème
- **Questions**
  - Q1 = uupuueuuuu est-il un théorème?
  - Q2 = upuueuuuu ?
  - Q3 = upupueuuu ?
- Réponse Q1
- Ce système est **semi-décidable** car on possède une procédure infaillible pour décider Théorème mais pas non-Théorème.

Système à base de connaissances ?

- Un système formel permettant d'exprimer symboliquement la « connaissance »
  - Alphabet
  - Procédé de formation des expressions
- Des axiomes
- Une règle de dérivation
- Un moteur d'inférence

Un système d'aide au voyageur : Alphabet

- distance.<.2km
  distance.<.300km
  aller.à.pied
  prendre.le.train
  prendre.l'avion
  avoir.le.téléphone
  aller.à.l'agence
  téléphoner.à.l'agence
  acheter.un.billet
  durée.>.2.jours
  être.fonctionnaire
  (
)
  non /*(négation)
  ^ /*(et, ou conjonction)
  -> /*(implique)

Procédé de formation des expressions

- *expression* := *symbole*
- *expression* := (*expression*)
- *expression* := non *expression*
- *expression* := *expression1* ^ *expression2*
- *expression* := *expression1* -> *expression2*

Axiomes

- R1 : distance.<.2km -> aller.à.pied
- R2 : ((non distance.<.2km) ^ distance.<.300km) -> prendre.le.train
- R3 : (non distance.<.300km) -> prendre.l'avion
- R4 : (acheter.un.billet ^ avoir.le.téléphone) -> téléphoner.à.l'agence
- R5 : (acheter.un.billet ^ (non avoir.le.téléphone)) -> aller.à.l'agence
- R6 : prendre.l'avion -> acheter.un.billet
- R7 : (durée.>.2.jours ^ être.fonctionnaire) -> (non prendre.l'avion)
- F1 : (non distance.<.300km)
- F2 : avoir.le.téléphone

Moteur d'inférence

- ça marche
- tant que ça marche
  - ça ne marche pas
  - boucle sur les Ri
    - boucle sur les Fj **non marqués**
      - si Ri est de la forme "Fj -> Fk"
        - ajouter Fk à la BdF
        - **marquer Fj**
        - ça marche
      - sinon
        - boucle sur les Fi
          - si Ri est de la forme "Fj ^ Fi -> ..."
            - ajouter Fm = (Fj ^Fi) à la BdF
            - **marquer Fj**
            - ça marche
          - finsi
        - finboucle
      - finsi
    - finboucle
  - finboucle
- fintant

Faites tourner le moteur en prenant les faits F1 et F2

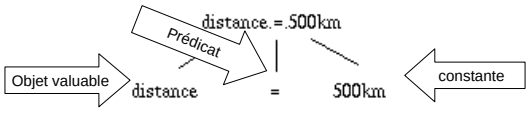
- Que peut-on déduire ?
- A vous…
- ….

Terminologie

- Les axiomes de type R seront appelés des règles
  - Partie gauche (de ->) : Prémisses (conjonction de)
  - Partie droite (de ->) : Conséquents (conjonction de)
- Les axiomes de type F seront appelés des faits
  - Équivalent d'une « partie droite » inconditionnelle (sans prémisses)

Généralisation de 0 à 0+ Des propositions aux prédicats

- Dans l'exemple précédent si on remplace les faits par :
  - F1 : distance.=.500km
  - F2 : avoir.le.téléphone
    - Il ne se passe rien, car aucune règle ne porte sur le symbole « distance = 500km »
- On casse le symbole atomique



De 0+ à 1
Ou Passage à la logique des prédicats du premier ordre

- Introduction de la notion de variable, mais surtout
- Introduction du quantificateur universel : ∀

R1 : ∀ destination, distance(destination) < 2km -> aller.à.pied(destination)

Au-delà de la logique d'ordre 1

Logique d'ordre 2 (règles sur les règles)
=> méta-connaissances
Du monotone au non monotone

- Temporel
- Spatial
- Multivaluation => Flou
- Logiques modales…

Illustration complexité 1

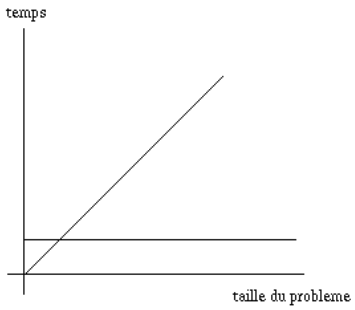
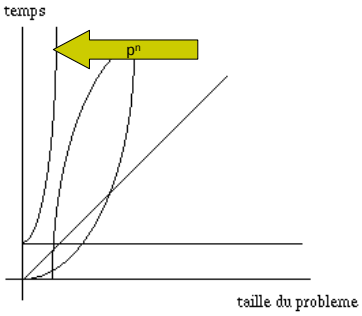
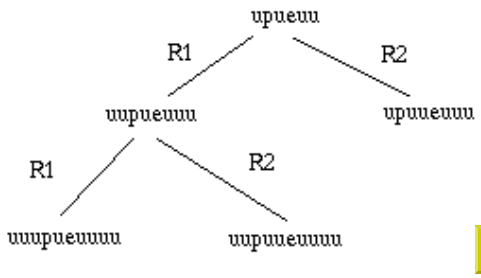


Illustration complexité 2



Réponse



Procédure systématique d'exploration = $2^{n/2}$ avec n longueur de la chaîne