IA- Matthieu Exbrayat

Alexandre Masson

14 Janvier 2013

Table des matières

1 Organisation du cours

Cours de 2h le mardi et TD 2 h le mercredi.

2 Introduction

Questions Qu'est ce que les taches suivantes ont en commun ?

Concevoir des systèmes capables de faire des choses compliquées.
Il y a aussi l'aspect ; apprentissage , le système est il capable d'apprendre de lui même et se débrouille tout seul.
Exemples :
Concevoir un programme qui vire tout seul les spam dans les mails.
Concevoir un super navigateur qui s'occupe tout seul de faire les mise à jour logicielle.

Le point commun ? Posséder un certain degré d'intelligence.
D'où grande question qu'est ce que l'intelligence ?

- Selon Darwin : Ce qui permet l'individu le plus apte, parfaitement adapté a son environnement
- Selon Edison : ce qui fonctionne et qui produit de l'argent.
- Selon Turing : ce qui rend difficile la distinction entre une tache réalisée par un être humain ou une machine

L'IA est une discipline qui systématise et automatise les taches .

- Penser comme un humain
- Agir comme un humain
- Penser rationnellement
- Agir rationnellement

Penser comme un humain Il faut comprendre l'esprit Humain.
Comparaison des différentes étapes d'un programme et du raisonnement humain pour arriver au même problème.
Science cognitive.

Agir comme un humain Test de Turing, on transforme l'ordinateur peut il penser en peut il se comporter intelligemment. Le test est concluent si l'opérateur ne sais pas si la réponse a ses question est donnée par un humain ou une machine

Penser rationnellement : approche logique

Agir rationnellement : agir pour atteindre un objectif Remplir une mission de la meilleure façon.

Agent : unité qui fonctionne de façon autonome, perçoit son environnement , s'adapte aux changements et est capable d'atteindre un objectif.

Agent Rationnel : agent qui agit pour atteindre le meilleur résultats ou du moins le meilleur résultat espéré.

Loebner prize Tous les ans récompense le meilleur système du test de Turing.

- traitement du langage naturel : pouvoir communiquer en un langage naturel.
- représentation de connaissances : stocker ce qu'il sait ou ce qu'il perçoit.
- raisonnement automatique : utiliser des connaissances pour répondre aux questions.

les Lois de la pensée Aristote : quels sont les processus de pensée corrects ?
Plusieurs formes de logique : notation et règles de dérivation pour les pesées , sans relations avec une mécanisation du raisonnement.
Ligne directe entre math et philo -> IA.
Problèmes : tous les comportements intelligents ne sont pas le résultats d'un raisonnement logique.

Agent Rationnel abstrait : un agent est une fonction qui met en correspondance des séquences perceptives P* et des actions A : f : P* -> A.

processus où on ne sais pas tout, ou temps limité trop court pour répondre, donc réponse acceptable mais pas optimale.

Fondements de l'IA Elle repose sur plusieurs domaines.

Philosophie : logique et raisonnement.

Mathématiques : représentations formelles et preuves, algorithmes, (in)décidabilité, probabilité.

Psychologie : adaptation, perception et controle moteur, techniques expérimentales.

Linguistique : représentation de connaissances grammaires.

Neurosciences : substrat physique et biologique de l'activité mentale.

Théorie du controle : systèmes asservis, stabilité, concept d'agent optimal.

3 Histoire de l'IA

- 1943 : modélisation de neurones.
- 1950's : vision complète de l'IA
- 60's : dev des réseaux de neurones.
- 70's : développement des systèmes à base de connaissances, faire de l'aide au diagnostic.
- fin 80's : hiver de l'IA, l'ia s'effondre
- depuis 20 ans : IA évolue et révolutionne sa méthodologie.
- plein de fric à se faire dans la Bourse, algorithme qui réagissent au plus vite et essaye d'analyser la bourse pour établir des règles.

Prédictions et réalité différence prédictions réalité ;, 60's œil électronique, pas encore fait mais ça avance et s'est encourageant.

Robot qui font tout, déjà dans les 70's.

4 Qu'est ce qu'un problème pour l'IA

Problème qui n'as pas de solution analytique connue, objectif : si l'objectif est hors du possible donner une solution acceptable en temps raisonnable.

Certaines taches aisées pour l'humain sont difficiles pour la machine : tout ce qui a besoin de la sensibilité humaine.

Contenu du cours 4 approches différentes de la résolution de problèmes en IA

- recherche de solution dans un espace d'états
- recherche par raisonnement et en présence d'incertitude
- résolution par planification.
- résolution par apprentissage automatique

5 Agents Intelligents

Qu'est ce qu'un agent L'AGENT perçoit des choses de son ENVIRONNEMENT, à laide de SENSORS, il accompli ensuite sa MISSION, et utilise ses ACCUATORS, pour effecteur des actions.

Définition : entité capable de percevoir son environnement par des capteurs et d'agir sur son environnement à l'aide d'effecteurs (actionneurs). AUTONOMIE , PERCEPTION,ACTION,OBJECTIF, sont les maitres-mots qui définissent l'existence de l'agent.

spécification d'un agent le choix d'une action à l'instant t dépend de séquences perceptives .

Agent rationnel Basé sur le raisonnement.

Toute option considérées , faire le meilleur choix pour maximiser les chances de succès.

Mesures de performances : réussir la tâches, quantification maximale d'un objectif.

spécifier l'environnement spécif = problème, agent) solution

PEAS exemple , conduite automatique
- P (mesure de performance) : bon endroit, temps, sécurité
- E (environnement) : rues, voitures, piétons, météo
- A (actions) : tourner , arrêter , klaxonner, etc...
- S (senseurs) : plein.

types d'environnement Observable ou nono , et déterministe ou non., Observable : on a accès à tout moment à l'état complet de l'environnement. Déterministe : on connais l'instant suivant à partir de l'état courant et l'action faite par l'agent.

Épisodique/séquentiel : les états futurs ne dépendent pas des actons effectuées lors de évents précédents.

Statique/dynamique : un environnement ne change pas durant le temps ou l'agent réfléchit.

Discret/continu : environnement discret si le nombre de séquences est fini. Simple agent ou multi-agents. les appli du monde réel sont les plus difficile à mettre en place car l'environnement n'est que partiellement observable, il est séquentiel, dynamique, continu et multi-agent.

types d'agents

- Agent-reflex : presque tout est codé en dur, on applique des règles en fonction de se qu'on observe de l'environnement.
- Agent-reflex avec états : on va conserver des états qui vont servir à estimer des états en fonction de se qu'on viens de capter et ainsi que ce que l'on a sauvegardé. Mettre a jour l'état interne dépend de comment l'env change et comment nos action le font évoluer.
- Agent avec objectif. le Système va être dans un état donné, et après on regarde toutes les solution dispo et on va choisir celle qui nous rapproche le plus de l'objectif. IL a un objectif que décrit les situations désirées. il combien ces informations avec la résultat pour choisir la bonne action. l'agent devrait être capable de considérer des longues séquences pour atteindre l'objectif : **recherche** et **planification**.
- Agent avec la fonction d'utilité : parmi la listes des actions proposées celle qui est la plus prometteuse. On est déjà sur des solutions sophistiquées. Il n'est pas capable de s'améliorer lui même.
- Agent d'apprentissage : il reçoit des signaux et choisis quoi faire et indique ces décisions. il va ensuite essayer de savoir si ça décision est pertinente. Il a des exemples d'actions et de suites d'actions qu'il a effectuées, il sais si ces actions sont bien choisies ou pas

IL est nécessaire d'etre capable de stocker des processus qui vont prendre des décisions. La vision PEAS est une modélisation simple.

6 Problèmes et algorithmes de recherche

6.1 Agents de résolutions de problèmes

Intro aux algorithmes de recherche Les algorithmes de recherche constituent l'une des approches les plus puissantes pour la résolution de problème en IA.

Les algo de recherche sont un mécanisme de recherche général de résolution. On sais toujours tout ce qui peux être fait depuis cet état mais on ne peux pas savoir à l'avance si cet état en bon ou pas. On effectue à chaque action un test de solution. On sais a quoi ressemble un état qui peux être une solution, et il est aussi intéressant de savoir comment nous sommes arrivé a la solution. Plusieurs approche possible en largeur ou en profondeur. Exemple : voyage d'un bout à l'autre de la Bulgarie.

Autre exemple : jeu de taquin : puzzle-8 : on a des cases dans un ordre aléatoire, mais on veux arriver a avoir toutes les cases dans l'ordre.

Définition d'un problème de recherche Formalisons un ptit peu ça : tous les états sont dans un ensemble Q : non vide, il y a parmi ceux la des états initiaux(S). On a aussi un ensemble d'état solutions(G) : définis explicitement, ou implicitement avec des règles de tests. A : un ensemble d'actions . Fonction de successeurs, et parfois une fonction de coût : cela nous permet de pouvoir estimer si une décision est bonne ou pas. Nous avons besoin de définir tous ces éléments pour définir les problèmes de recherche. Exemple :
Aspirateur : problème assez simple. Solution : séquence d'opération .

Il existe aussi des problèmes plus compliqué , les problèmes contingents : effet conditionnel des actions, perception fournit des nouvelles infos de l'état courant, la solution est un arbre ou une stratégie. IL existe aussi les problèmes d'exploration : l'exécution révèle les états, besoin d'expérimenter pour trouver la solution.

6.2 Formulation d'un problème à état unique

Un problème est défini par 4 éléments :
état initial : être à Arad.
Fonction de successeur.
test : implicite : NoDirt(x), ou explicite : "être à Zerind". Tous les problèmes qui peuvent être décrits par : un ensemble fini d'états, un ensemble fini d'actions, un sous-ensemble d'états initiaux et solutions, une relation successeur définie sur l'ensemble des actions et qui renvoie vers l'ensemble des états et une fonction de coût qui est positive.

6.3 Stratégies

Elles sont évaluées sur 4 critères : Complétude : si une solution existe, le système va t il l'atteindre.
complexité en temps : combien de nœuds on est censé explorer. Deux grandes familles, aveugles et heuristiques.

Stratégies Aveugles On utilise la recherche en profondeur limitée, pour profiter de la vitesse de calcul de la largeur, mais aussi de l'utilisation mémoire plus satisfaisante de la recherche en profondeur. Cette solution est complète, sa complexité en temps est $O(b^d)$, sa complexité en espace est O(bd). Optimalité, si le coût pour passer d'une étape à une autre est unitaire, car toues les solution d'un même niveau ont le même coût.

Arbre de recherche bidirectionnelle nécessite de connaître explicitement les états finaux, et des actions sont on connais la fonction inverse, on pars de la source et des solutions, et on construit en fonction de la longueur des chemins tous les nœuds accessibles, et on regarde s'il y a des matchs.

Quelle solution prendre ? Si on a pas trop de problème de mémoire , le bidirectionnel est le plus efficace, si on a des contraintes de mémoire, on aura plus intérêt a prendre la recherche par profondeur itérative.

États répétés L'échec de détection d'états répétés risque de produire des arbres de recherche infinis.
En largeur, garder une trace de tous les états déjà parcouru, si l'état d'un nouveau nœud existe déjà , alors on le supprime.

Recherche heuristiques Nous allons utiliser une fonction heuristique , de l'ensemble des états vers les réels comme une estimation du rapport coût bénéfice qu'il y a a étendre le chemin courant en passant par s.

Idée : Utilisation d'une fonction d'évaluation (heuristique) pour chaque nœud, étendre avec le plus prometteur selon la fonction heuristique. On utilise souvent l'approche gloutonne pour minimiser le coût des transitions. Cette approche gloutonne est elle complète ? : il existe des cas où l'on boucle., mais complet dans un espace fini sans boucle.

En terme de temps , c'est de l'ordre , facteur de branchement avec en exposant la profondeur maximum, pareil en espace, car il faut garder en mémoire tous les nœuds. La performance de cette approche gloutonne, dépend de la précision de la fonction heuristique, il est possible de réduire fortement les contraintes de temps et d'espace.

Une fonction heuristique est admissible (qui ne surestime jamais le coût réel si elle est supérieure a 0 , et inférieur au coût optimal réel, une fonction heuristique est toujours optimiste !

L'idée va être de combiner le greedy search(glouton), réduisant le coût , mais pas toujours optimal, avec le coût uniforme, qui lui est optimal, mais pas très efficace, pour arriver a une solution admissible.

Cela nous mène a l'algorithme A*, éviter de choisir le chemin qui est déjà coûteux, on ne va plus uniquement regarder vers l'avant , mais aussi vers l'arrière, pour prendre n compte le chemin qui mène a un nœud, en plus de sa capacité a nous rapprocher du but.

Complétude du Lemme Oui , sauf dans le cas ou on a un facteur de branchement infini,
Temps Exponentiel,
Espace Exponentiel, il faut garder tous les nœuds en mémoire,
Optimalité Oui , f_{i+1} n'est pas étendu tant que f_i n'est pas fini.
A* étend tous les nœuds ayant f(n) < C*
A* étend quelques nœuds ayant f(n) = C*
A* n'étend aucun nœud ayant f(n) > C*

Qualité de fonction heuristique Facteur de branchement effectif, soit N le nombre total d'états produits pour obtenir la solution, soit d la profondeur à laquelle la solution a été trouvée alors b* est la facteur de branchement d'un arbre fictif parfaitement équilibré tel que

$$N = 1 + b^* + (b^*)^2 + ... + (b^*)^d$$

Une bonne fonction heuristique aura une valeur de b^* proche de 1 (la fonction heuristique idéale aurait $b^* = 1$)

Dominance Si $h_2(n) >= h_1(n), \forall n$ (les deux étant admissibles) alors h_2 domine h_1 et constitue une meilleure fonction heuristique.
problème simplifié Une heuristique admissible peut être dérivée à partir du coût exact de la solution d'une version "simplifiée" du problème
Point clé Vérifier que le coût optimal du problème simplifié n'est pas plus grand que le coût du problème initial.

Variantes de A* Les problèmes réels sot souvetn très complexes
L'espace de recherche devient très grand
Meme les méthodes de recherche heuristique deviennent inefficaces
A* : complexité en espace exponentiel.
IDA* = approfondissement itératif de A*
SMA* = A* avec gestion mémoire.

IDA* IDS effectue l'approfondissement par rapport à la profondeur de la recherche
IDA* utilise comme limite la la valeur de la fonction heuristique f(n), chaque itération de IDA* est une recherche en profondeur, la profondeur limite correspond à la plus petite valeur observé au delà de la limite de l'étape précédente.
IL est complet et optimal
Complexité en espace de IDA* : O(bd), ok pour coût unitaire, mais qu'en est il pour des coûts réels ?

Un alternative : RBFS
Récursive BFS
Approche best-first, mais
Enregistre dans chaque nœud le f du meilleur chemin alternatif

remonte si le cout est supérieur à un f mémorisé
En remontant, mémorise ce cout (pour redescendre si besoin)
+ efficace qu'IDA* mais risque de redévelopper beaucoup de nœuds
Optimale (si h est admissible), espace O(bd) , temps ?
PB : la faible complexité en espace est finalement gênante.

SMA* Simplified Memory bounded A*
SMA* effectue sa propre gestion de mémoire
principe

- si la mémoire (file d'attente) esy pleine, alors faire de la place en éliminant le nœud le moins intéressant (celui avec une valeur f élevée)
- retenir dans le nœud ancêtre la valeur du meilleur descendant oublié

Propriétés

- espace mémoire alloué par avance
- évite les états multipliés
- complet si espace mémoire suffisant