UNIVERSITÉ TOULOUSE III PAUL SABATIER

Rédigé avec LaTeX
Version du 24 septembre 2014

Techniques de base de résolution de problèmes

IA

M1 Informatique – Développement Logiciel
Semestre 7

Cours donné par C. Cayrol
Rédigé par Antoine de Roquemaurel

2014

1 Introduction : Les agents résolveurs de problème

1.1 Les agents rationnels

Il est dans l'environnement et peut le percevoir.

Definition 1.1 L'**agent** est **rationnel** s'il agit toujours au mieux sur l'environnement de manière à atteindre ses objectifs mais en tenant compte de ses compétences ou capacités.

Un agent rationnel peut possèder les fonctions importantes suivantes :

- Perception, Modélisation, Représentation
- Raisonnement (inférence)
- Communication
- Apprentissage
- Élaboration de projets collectifs

On ajoute à un agent une mesure de performance, plus ses performances seront élevés plu il sera rationnel.

Un agent rationnel peut être plus ou moins autonome. S'il n'est pas autonome il va avoir un comportement dit « Réflexe », dans l'autre cas il va apprendre au fur et a mesure qu'il résous sont environnement, ou son comportement, il sera de plus en plus rationnel.

Perceptions	Environnement	Actions	Buts
— GPS — Compteur de vitesse — Radar	— Piéton — Véhicules — Réseau routier — Client	— Accélérer — Freiner	— Satisfaire la démarche

1.2 L'intelligence artificielle : situation d'une discipline

L'intelligence artificielle est à la fois une science et une technique. On cherche à observer, étudier, comprendre, modéliser on les appel les capacités cognitives.

Un agent intelligent est un agent qui peut effectuer des tâches qui serait qualifiée d'intelligentes si un humain les avaient réalisés.

En Intelligence Artificiel, on utilise des notions dans plusieurs domaines :

Philosophique Notamment les travaux de Platon ou Aristote.

Mathématiques Utilisation de théorèmes

Psychologie Étude du comportement humain

Neuro-sciences Essayer de comprendre le fonctionnement du cerveau humain

Ethologie Étude du comportement animal dans son milieu naturel : reproduire des modèles de comportement animals. Fourmis, abeilles, corbeaux, pies, ...

Linguistique Étude des langues naturelles

Économie

Informatique Pour le développement

1.3 Développement de l'Intelligence Artificiel

En 1956, lors d'une conférence assez célèbre, J. Mac Carthy à lancé un projet avec l'idée que tout ce qui relève de l'intelligence peut être modéliser afin qu'une machine puisse le reproduire.

À la fin des années 1960, il y a eu les premiers logiciel d'IA, puis l'algorithme A*

Dans la fin des années 1980, c'était la grande mode, on pensait pouvoir utiliser l'informatique n'importe où, nous étions trop ambitieux, ce qui à provoqué une retombée néfaste due à la déception.

Dans la fin des années 1990, cela repart, souvent couplée à d'autres disciplines tel que la robotique, il y a de nouveau des avancées, mais nous sommes devenus conscient du fait qu'il soit nécessaire de travailler avec d'autres personnes.

> — Recherche dans les jeux difficiles, tel que les Echecs, le Go.
> — Robotique
> — Vision par ordinateur

Table des matières

2 Le formalisme des espaces d'états

2.1 Introduction

Formalisme :
- État initial
- État but (Explicite ou implicite)
- Actions autorisées (ou opérateurs)

Definition 2.1 Un **Un descendant d'un état S** est un état accessible de S par une séquence non ide d'opérateurs fils (descendant immédiats)

Definition 2.2 Un **espace de recherche** possède les états accessibles de l'état initial

Ordonnance des tâches d'un robot

- N Stations
- Temps entre les stations
- Temps de durée d'une tâche

Le problème de fonctionnement est décrit.

Etat Planning partiel

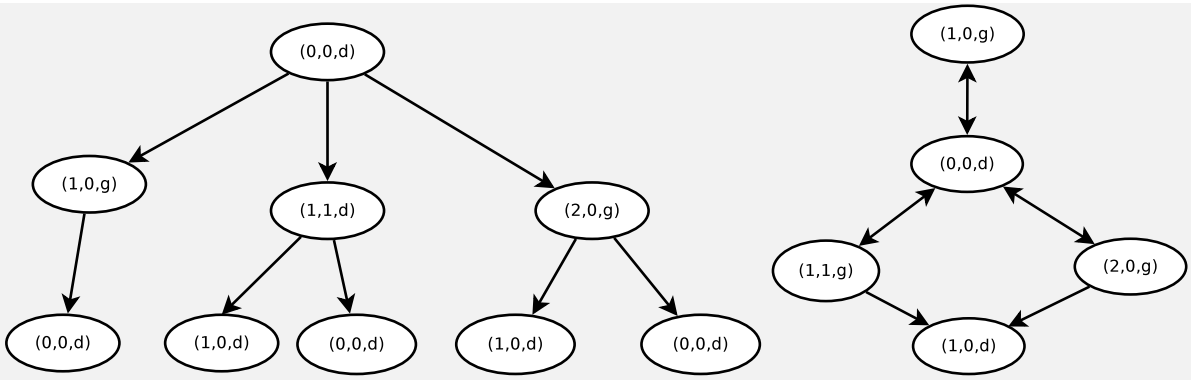
Etat but

Missionnaires et cannibales (cg, mg, sb) : Canibal Gauche, Missionaire Gauche, SB

Etat initial $(0, 0, d)$

Etat but $(3, 3, g)$

Opérateur Traversée en respectant les contraintes



R La taille des espaces d'états pour des problèmes réels est extrêmement importante : nécessité de techniques ou d'algorithmes spécifiques pour ne pas développer tout l'ensemble d'état et pouvoir néanmoins trouver une solution.

2.1.1 Représentation de l'espace d'états

En extension On explicite tous les états possibles [1]

En intention 3 données sont nécessaires :
- État initial
- But
- Les opérateurs de changement d'état : une modélisation astucieuse peut amener un gain de performance [2]

Objectif Utiliser des algorithmes qui vont développer une partie, la plus petite possible de l'espace d'état pour trouver un chemin qui mène de l'état initial à un état qui satisfait le but. Pour ça l'algorithme va développer un espace de recherche

Deux catégories d'algorithmes
- Algorithmes aveugles (non informés) (Profondeur d'abord, largeur d'abord, *Deep First iterative Depending* DFD, ...)
- Algorithmes informés : Nécessite une fonction d'estimation, appelé fonction heuristique (Glouton ou Gradient, Meilleur d'abord, A, A*, IDA*, B, C,...)

2.2 Recherche euristique

C'est une stratégie d'exploration de l'espace de recherche en fonction de choix. Il va nous dire comment choisir le prochain état à examiner.

Une stratégie peut utiliser un heuristique.

Exemples d'heuristiques :
- Information qui classe les opérateurs applicables à un état
- Fonction d'évaluation d'état

1. Pour un taquin $3 \times 3 = 181440$ états, ce qui est une taille trop importante pour l'expliciter, c'est-à-dire le dessiner
2. Taquin 3×3 32 opérateurs $\Leftrightarrow$ 4 opérateurs pour la case vide

Coloration d'une carte Carte planaire avec différentes régions colorée avec N couleurs différentes.

Le problème est de colorer la carte tel que 2 régions adjacentes soient de couleurs différentes.

Etat Coloration partielle. Liste de paires (région, couleur)

Etat but Coloration acceptable complète

Ici on peut évaluer un état en mesurant la distance au but

2.2.1 Stratégies

Definition 2.3 Stratégie
Fonction de choix du prochain état à explorer (Ou à développer)

Definition 2.4 Stratégie aveugle vs stratégie informée
ou non informée une stratégie qui ne dépend pas du problème(Ex : en profondeurd'abord, largeur d'abord)

Inversement, on parle de stratégie informée ou de recherche heuristique.

2.2.2 Critères d'évaluation

Definition 2.5 Complétude
Un algo de recherche est dit complet si l'espace d'état contient l'état but

Definition 2.6 Complexité
On se base sur le facteur de branchement de l'algorithme (Membres max de fils/états, profondeur max, profondeur de l'état but le moins éloigné de la racine)

2.3 Algorithmes de recherche

2.3.1 Version de base

```
1  -- Arguments : départ, test_etat_but, fils_etat, estime_etat, classe.
   -- Variables locales : file_attente, prochain, continuer (booléen)
3  Debut
     file_attente <- {départ};
5    continuer <- vrai;
     tantque continuer et file_attente non vide faire
7      prochain <- pop(file_attente);
       si test_etat_but(prochain) alors
9        continuer <- faux;
       sinon
11       file_attente <-
             classe(file_attente, fils_etat(prochain), estime_etat);
13   fin tantque;

15   si continuer alors
       print(Echec);
17   sinon
```

```
    retourner(prochain);
Fin
```

Listing 2.1 – Recherche sans optimisation

```
-- Arguments : départ, test_etat_but, fils_etat, estime_etat, classe.
-- Variables locales : file_attente, prochain, continuer (booléen), vus.
Debut
  file_attente <- {départ};
  continuer <- vrai;
  tantque continuer et file_attente non vide faire
    prochain <- pop(file_attente);
    vus <- push(prochain,vus);
    si test_etat_but(prochain) alors
      continuer <- faux;
    sinon
      file_attente <-
          classe_b(file_attente, fils_etat(prochain), estime_etat, vus);
  fin tantque;

  si continuer alors
    print(Echec);
  sinon
    retourner(prochain);
Fin
```

Listing 2.2 – Recherche avec optimisation

2.3.2 Recherche en profondeur d'abord

On met toujours le fils du prochain en file d'attente. La complétude est garantie si etseulement si lafonction est optimisée.

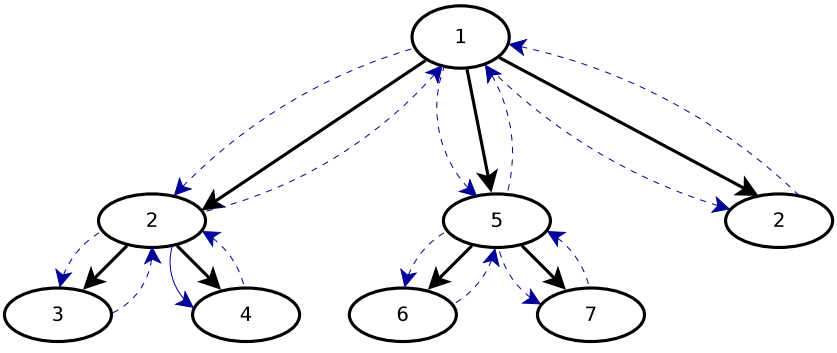


Figure 2.1 – Exemple de parcours en profondeur d'abord

2.3.3 Rercherche en largeur d'abord

On met le fils en queue de la file d'attente, c'est complet mais plus long.

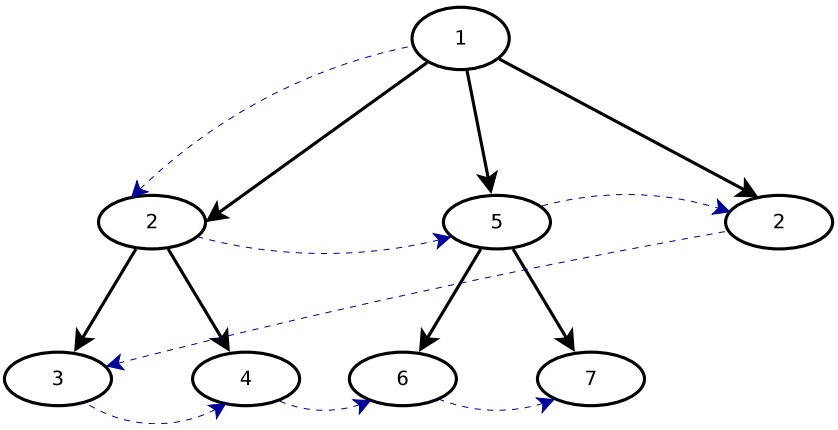


FIGURE 2.2 – Exemple de parcours en largeur d'abord

2.3.4 Recherche informée

Meilleur d'abord : on doit utiliser une fonction d'évaluation d'un état. (ex : nombre de jetons mal placés).

Recherche d'une solution optimale à un problème On appelle $f^*(e)$ le coût d'un chemin optima allant de ei a un but passant par l'état courant e.

Je cherche une fonction f qui estime $f(e) = g(e) + h(e)$, h étant la composante heuristique.

2.3.5 Algorithme A^*

On calcule $f(N) = \underbrace{g(N)}_{Dynamique} + \underbrace{h(N)}_{Statique}$ pour chaque nœud N, ces nœuds sont classés en fonction de $f(N)$ dans une lite d'attente, les plus petites valeurs d'abord. On développe toujours le nœud qui est en tête de la liste.

Pour avoir un A^* il faut que l'heuristique soit minorante, c'est-à-dire qu'elle sous estime toujours la réalité. Dans le cas contraire, nous aurons un algorithme A.

Definition 2.7
h est minorante si on a $h(e) \leq h^*(e)$

2.3.5.1 Propriétés d'un A^*

— Si le nombre de fils par état est fini, il existe un minorant > 0 du coût des opérateurs
— Complétude
— Si l'heuristique h est minorante, ou optimiste, alors A^* est admissible

2.3.5.2 Cas particuliers d'un A^*

On prend $h = 0(cst)$ un coût uniforme.

> Chaque opérateurs coûte 1.

3 Le formalisme des arbres de buts

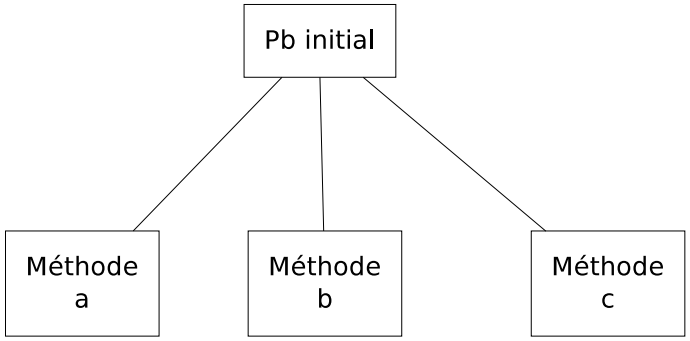


FIGURE 3.1 – Schéma du nœud Ou

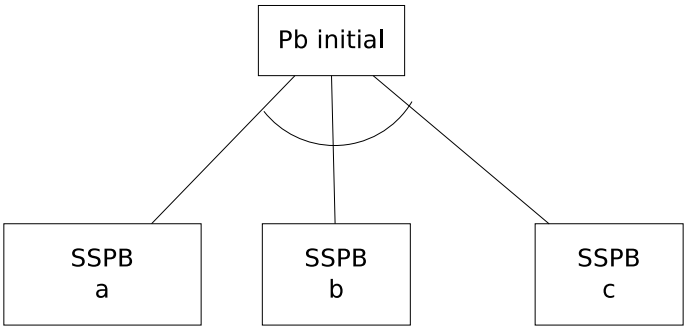


FIGURE 3.2 – Schéma du nœud Et

Si l'on a un problème initial, on suppose que l'on a 3 méthodes de résolutions,ou alors qu'une seule.

On construit alors un arbre de but qui est un cas particulier des arbres Et/Ou.

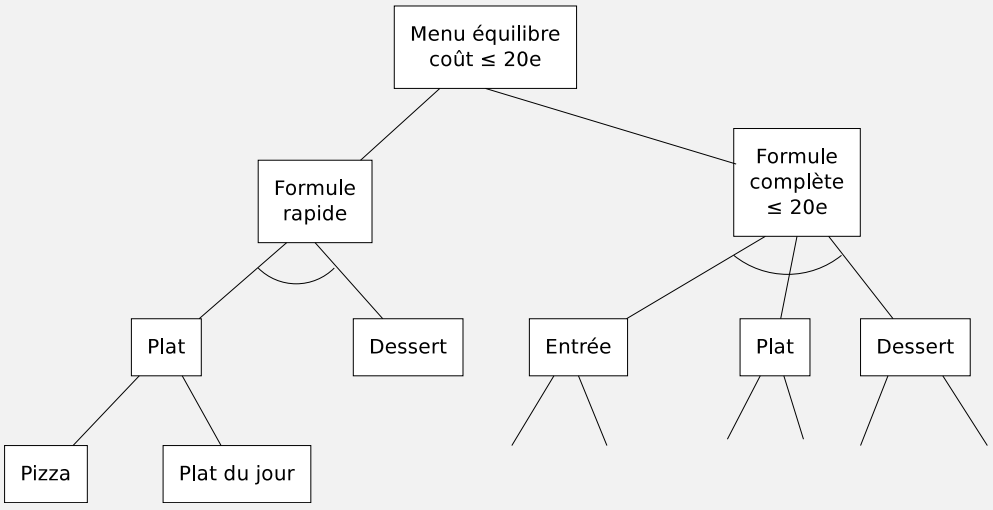


FIGURE 3.3 – Exemple de noeuds Et/Ou

3.1 Définitions

On distingue différents noeuds
- Nœud non terminal ET
- Nœud non terminal OU

— Nœud terminal

On dira qu'un arbre est résolu si sa racine est résolu On dira qu'un arbre est résolu si sa racine est résoluee

Definition 3.1 Un nœud terminal est :

— resolu : quand le problème associé a une solution connu sans continuer le travail
— en echec :

Definition 3.2 Un nœud OU est résolu ssi un de ses fils est résolu.

Definition 3.3 Un nœud ET est résolu ssi tout ses fils sont résolus et la contrainte est satisfaite

R La coloration d'une carte sera mieux formalisée ici car on peut la former en utilisant de arbres Et/Ou.

3.2 Recherche d'une solution

Ici un état est un plan partiel de résolution du but initial

Etat initial Plan vide

Etat but Plan complet

Transition Compléter un plan en ajoutant une action

Definition 3.4 Un plan partiel issu de n est un sous arbre de l'arbre de but de racine n qui contient au plus un fils par nœud Ou.

Definition 3.5 Un plan complet issu de n est un sous arbre de l'arbre de but de racine n contenant un fils exactement par nœud Ou et tous les fils par nœud Et.

3.2.1 Cas d'un environnement non déterministe

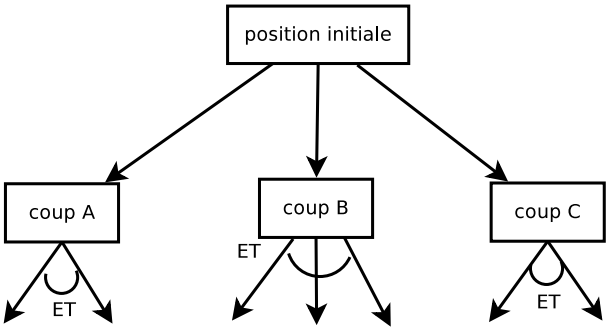
Dans ce cas là, on doit prendre en compte les réactions de l'environnement.

4 Recherche dans les arbres de jeux

4.1 Arbres de jeu et arbres de buts

On appelle joueur de référence le joueur auquel on s'intéresse, c'est celui qui veut gagner.

On construit un arbre de but, les nœreprésentant les différents choix de jeu sont des nœuds OU.



R On a une alternance entre les nœuds ET et les nœuds OU

On a une pile de 7 jetons, il faut choisir une pile et la couper en deux piles de tailles différentes, le premier qui ne peut plus jouer à perdu.

On peut dessiner un arbre de jeu complet, indépendant des joueurs qui représente tous les coups positives.

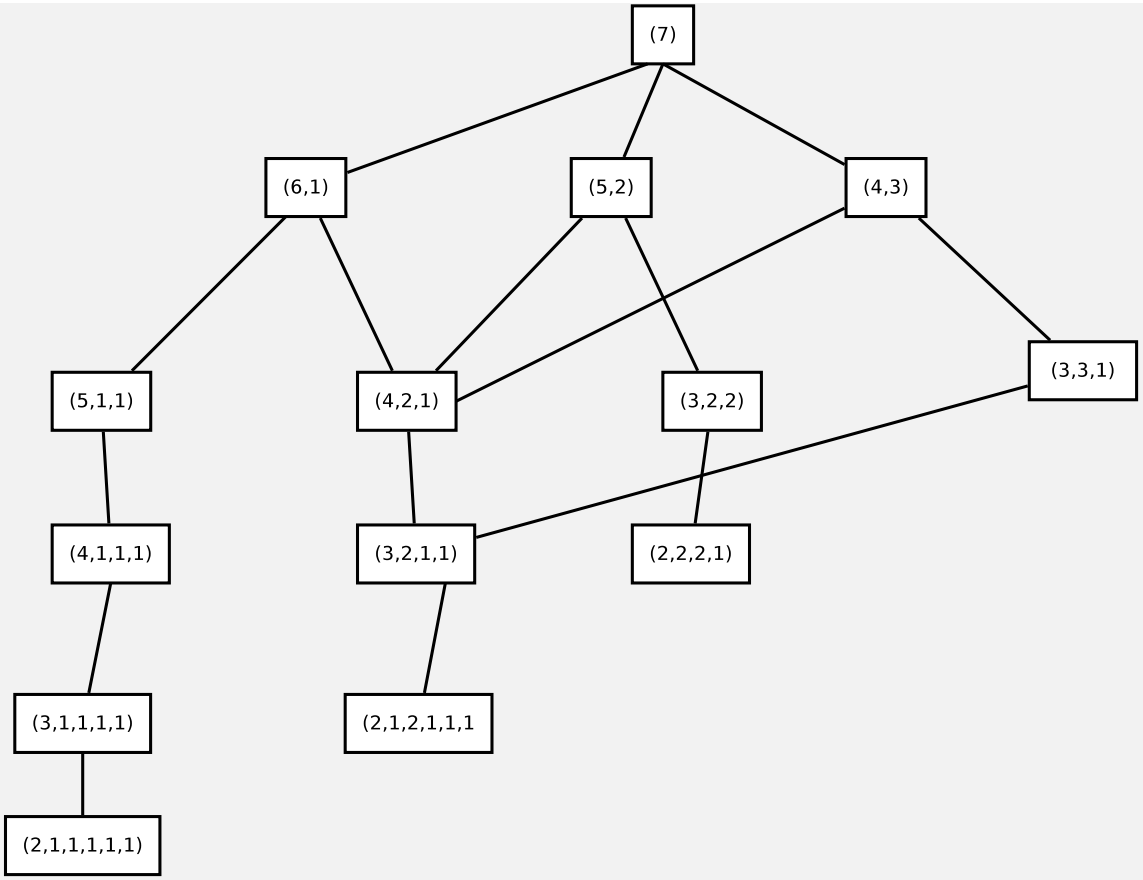


Figure 4.1 – Exemple d'arbre complet

4.1.1 Définitions

Soit A un arbre de jeu étiqueté ET/OU.

Definition 4.1 Une **stratégie partielle** (Sp) est un sous arbre de A de même racine que A contenant au plus un fils / nœud ou.

Definition 4.2 Une **stratégie complète** (pour le joueur de référence) est un sous arbre de A, de même racine, contenant un fils/nœud OU et tous les fils / nœud ET.

Definition 4.3 Une **stratégie gagnante** (pou le joueur de référence) est une stratégie complète dont toutes les feuilles sont gagnantes, on va gagner dans tous les cas

Il existe au moins une stratégie gagnante pour l'un des joueurs

4.1.2 Intéressement au gain

On ajoute un intéressement au gain du jeu.

Definition 4.4 Pour un joueur J, le **gain garanti** est le gain qui est assuré à J quelque soit la réponse de l'adversaire.

Definition 4.5 Le **gain garanti maximal** est le meilleur gain que le joueur puisse avoir.

4.1.2.1 Trouver le gain garanti

Plusieurs méthodes existent :

« Brute Force » On cherche toutes les stratégies gagnantes pour J, on a le gain garanti pour chaque stratégie gagnante, et on peut trouver la meilleure stratégie gagnante associée au gain garanti maximum.

Procédure minimax Soit A un arbre de jeu étiqueté ET/OU, les feuilles sont valuées pour le joueur de référence.

On remonte une valeur à la racine :
- OU max(valeurs des fils) ;
- ET min(valeurs des fils) ;

La valeur minimax de l'arbre est le gain garanti maximum, c'est-à-dire le gain garanti par la meilleur stratégie gagnante.

R Si le joueur de référence commence, on obtient aussi le meilleurs 1er coup.

4.1.3 Cas d'un arbre de jeu incomplet

Les feuilles ne sont plus des situations terminales du jeu.

Une estimation statique de la On applique une fonction à ces feuilles qui estiment l promese de la situation, pour pouvoir comprer 2 situations.

La valeur minmax de l'arbre est la promesse de la situation la plus prométteuse pour J qui peut être atteinte contre toute riposte de l'adversaire.

- Développer l'arbre de jeu à profondeur 4
- Appliquer l'estimation aux feuilles
- Remonter le minmax et voir le coup à jouer
- J Joue
- L'adversaire répond
- Recommencer

4.2 Recherche du minmax

Soit A un arbre de jeu étiqueté ET/OU, les feuilles sont valuées pour le joueur de référence.

On remonte une valeur à la racine de l'arbre :
- OU : max(valeurs des fils)
- ET : min(valeurs des fils)

Cette valeur est le minimax de l'arbre. La valeur minimax de l'arbre est le gain garanti maximum (gain garanti par la meilleure sg.) Si J commence on obtient aussi le meilleur 1^{er} coup.

4.2.1 Cas d'un arbre de jeu incomplet.

Les feuilles ne sont plus des situations terminales du jeu. On applique une fonction a ces feuilles qui estime la promesse de la situation. Pour pouvoir comparer 2 situations.

Definition 4.6 La **valeur minimax** de l'arbre est une promesse de la situation la plus prometteuse pour J qui peut être atteinte contre toute riposte de l'adversaire.

- Développer l'arbre de jeu à profondeur 4
- Appliquer l'estimation aux feuilles
- Remonter le minimax et voir le voup à jouer
- J joue
- L'adversaire répond
- Recommencer

Soit A arbre (complet ou incomplet).

2 classes de méthodes : Espace d'état de stratégies partielles → SSS*

Etat Stratégie partielle

État initial Stratégie partielle réduite à la racine de A

État but stratégie complète

Opérateur étendre la stratégie partielle e, rajoutant d'un fils à un nœud extensible. Fonction évaluation d'état plus petit majorant de la valeur de la meilleure stratégie complète qui peut être atteinte.

> R Il serait possible d'effectuer une optimisation, en effet dans une stratégie partielle, si on a un nœud OU extensible on a autant de successeurs que de fils dans A. Nœud ET extensible on a 1 successeur prochain fils.

<++>

4.2.2 Algorithme Alpha-Beta

C'est l'algorithme le plus ancien et le plus étudié dans les arbres de jeux. On cg

4.2.2.1 Convention Negamax

```
si noeud terminal alors
  negamax <- valeur noeud;
sinon
  developper noeud;
```

```
  i <- 1;
  mp <- −∞
  tanque i <= n faire
    vloc <- -negamax(fils(noeud));
    si vloc > mp alors
      mp <- vloc;
    fin si;
    i <- i + 1;
  fin tantque;
fin si;
```

5 Le formalisme des CSP