Introduction
à
l'intelligence artificielle

(algorithmes avec adversaires)

Antoine Cornuéjols

INAPG

antoine.cornuejols@agroparistech.fr http://www.lri.fr/~antoine

3. Algorithmes avec adversaire : plan

3.1 Introduction

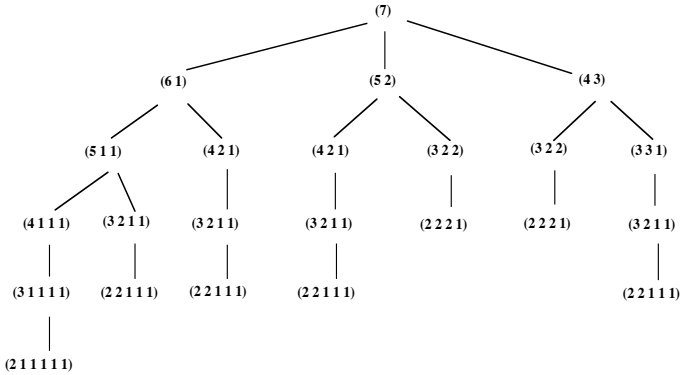
- **Jeux à *information complète*** (et un adversaire)
  - Pas de hasard
  - Chaque joueur connaît toutes les possibilités de jeu de l'adversaire (i.e. disposent de la même information)
    Exclut le bridge, le backgammon, ...
  - **Rq : Les techniques développées peuvent être relaxées à des jeux à plusieurs adversaires et information incomplète ou incertaine (ex: Météo, backgammon, ...)**
  - Historique
  - Etudié depuis 1949 [Morgenstein et von Neumann]
  - CHECKER (~1960)
  - DEEP BLUE (1997 : bat Kasparov en match en 6 parties)
    (Nous verrons comment)

3.1 Introduction

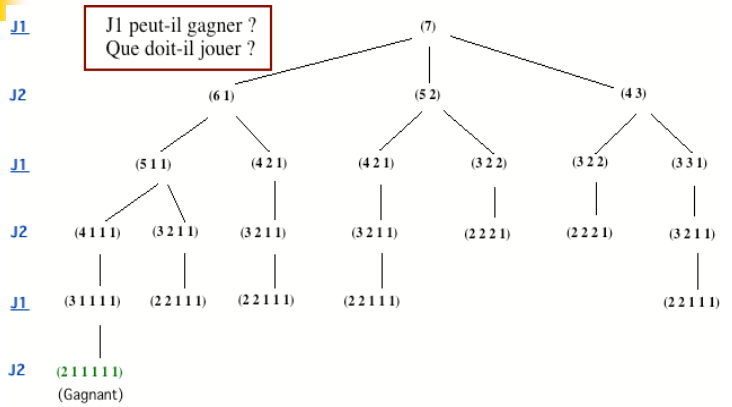
- **Motivation :**
  - Savoir décider en face de situations incertaines
  - Essentiellement, il s'agit de se préparer au pire en cherchant à minimiser le risque maximal

3.2 Jeux à exploration complète. *Ex : le grundy*

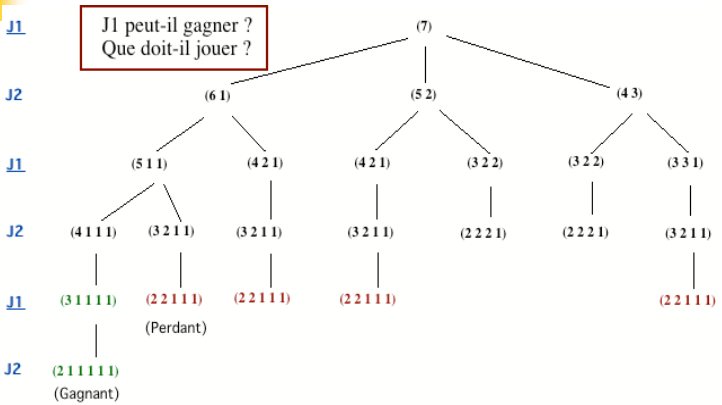
- On dispose initialement de 7 pièces en pile
- Chaque joueur à tour de rôle doit diviser une pile en deux pile inégales
- Si c'est impossible, le joueur a perdu



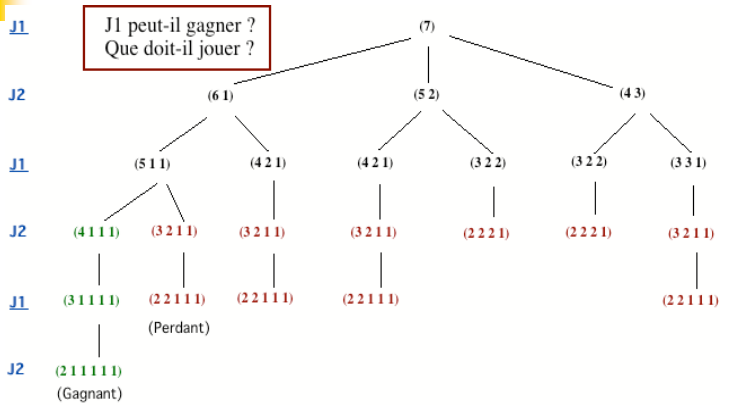
3.2 Jeux à exploration complète. *Ex : le grundy*



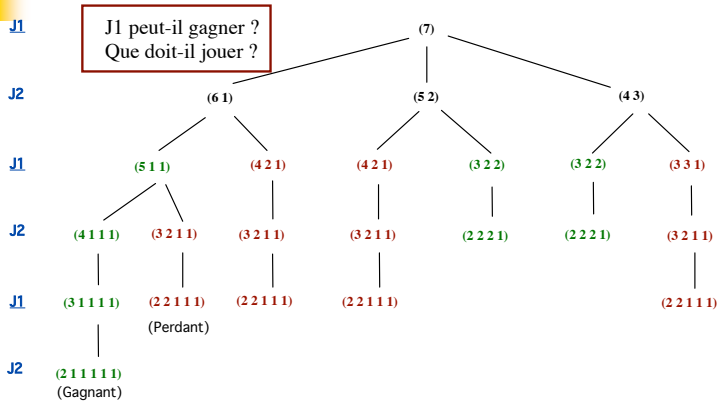
3.2 Jeux à exploration complète. *Ex : le grundy*



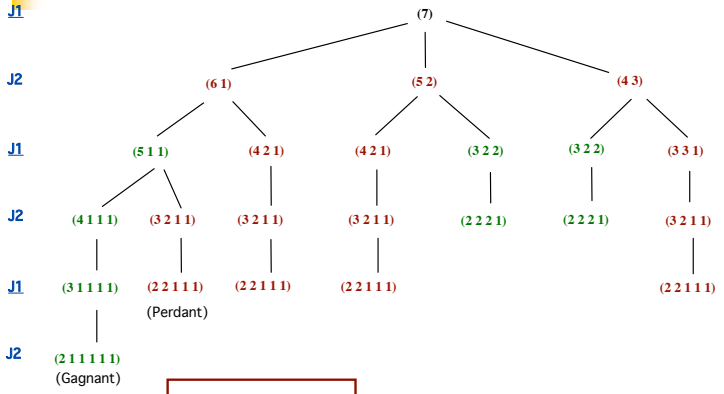
3.2 Jeux à exploration complète. *Ex : le grundy*



3.2 Jeux à exploration complète. Ex : le grundy



3.2 Jeux à exploration complète. Ex : le grundy



J1 peut-il gagner ?
Que doit-il jouer ?

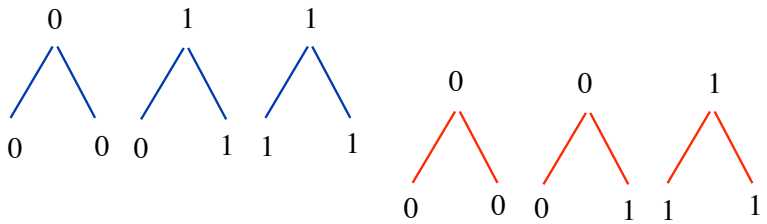
→ J1 ne peut pas gagner

3.2 Jeux à exploration complète.

Algorithmes de remontée des étiquettes

- Perdant = 0 (faux)
- Gagnant = 1 (vrai)

- **Si J1 : fonction OU des successeurs**
- **Si J2 : fonction ET des successeurs**



3.3 L'algorithme MinMax

- **La plupart des jeux (et des situations réelles) ne sont pas complètement explorables**
- **Idée :**
  - On explore aussi loin que possible en avant
  - On étiquette les feuilles avec une évaluation numérique de la position (pour le joueur J1)
  - On remonte ces étiquettes numériques jusqu'à la racine pour savoir quel coup jouer

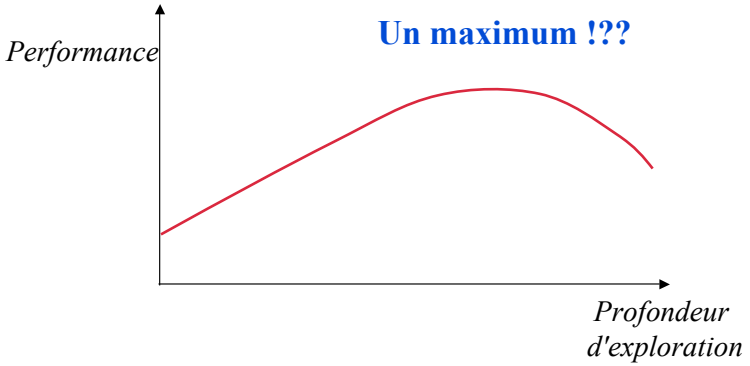
➥ **Algorithme MinMax**

3.3 L'algorithme MinMax

- **Algorithmes de remontée des étiquettes numériques**
  - **Si** J1 : Max **des valeurs des successeurs**
  - **Si** J2 : Min **des valeurs des successeurs**

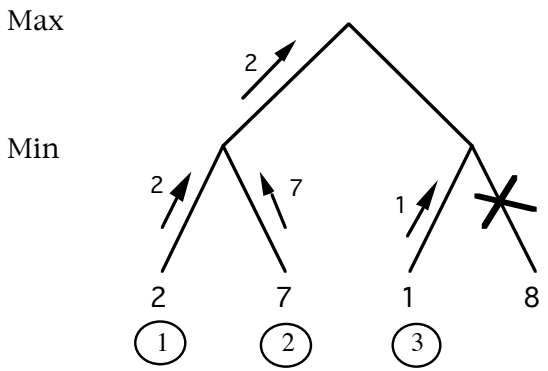
3.3 L'algorithme MinMax. Des jeux pathologiques ?!

- **Analyse des raisons du succès de MinMax (et de ses limites)**
- **Un comportement bizarre**



3.4 Des améliorations possibles

- **On peut économiser des évaluations de feuille (coûteuses en temps)**



3.4 L'algorithme alpha-beta

Maxmin (nœud,***alpha***, ***beta***) → ***alpha***

si feuille(nœud) : évalue(nœud) → ***alpha***

sinon : parcours séquentiel des successeurs de nœud avec :

max(***alpha***, **Minmax**(succ(nœud),***beta***) → ***alpha***

et si ***alpha*** ≥ ***beta*** sortir ***alpha*** (cas d'une ***alpha-coupure***)

Minmax (nœud,***alpha***, ***beta***) → ***beta***

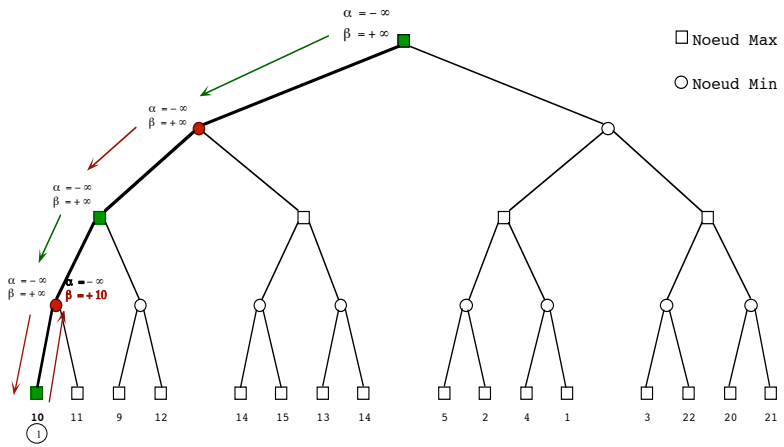
si feuille(nœud) : évalue(nœud) → ***beta***

sinon : parcours séquentiel des successeurs de nœud avec :

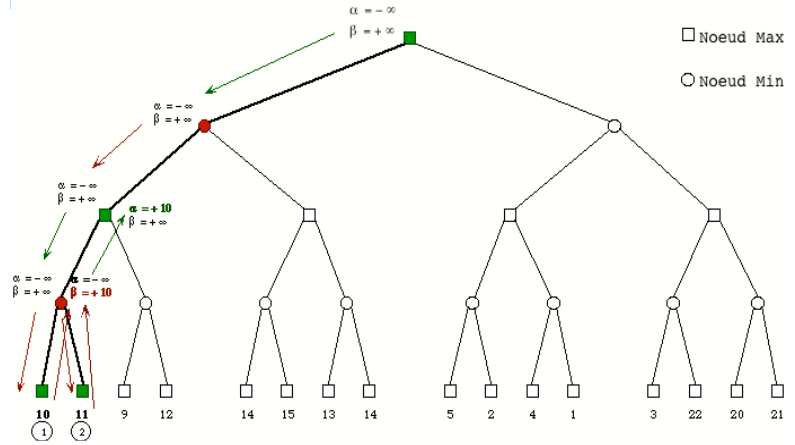
min(***beta***, **Maxmin**(succ(nœud),***alpha***) → ***beta***

et si ***beta*** ≤ ***alpha*** sortir ***beta*** (cas d'une ***beta-coupure***)

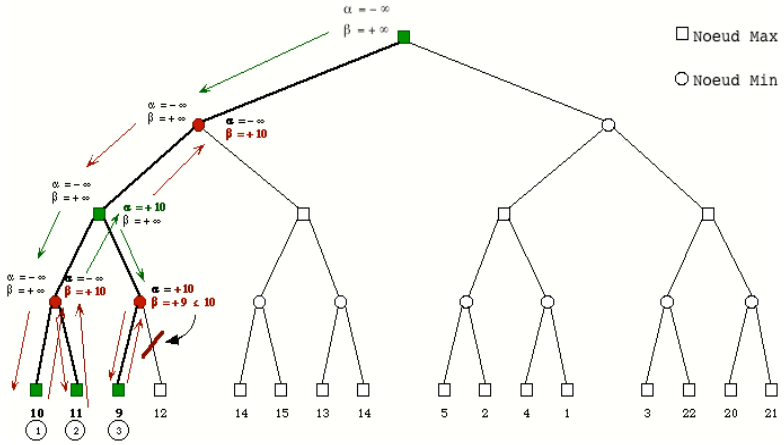
3.4 L'algorithme alpha-beta : Illustration (1)



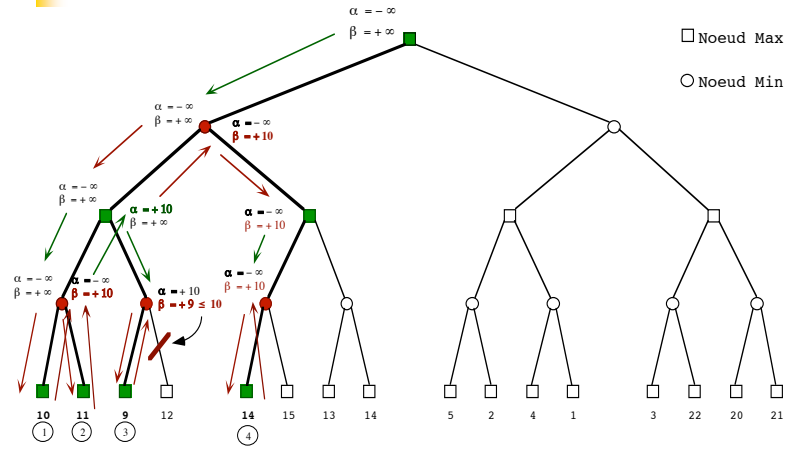
3.4 L'algorithme alpha-beta : Illustration (2)



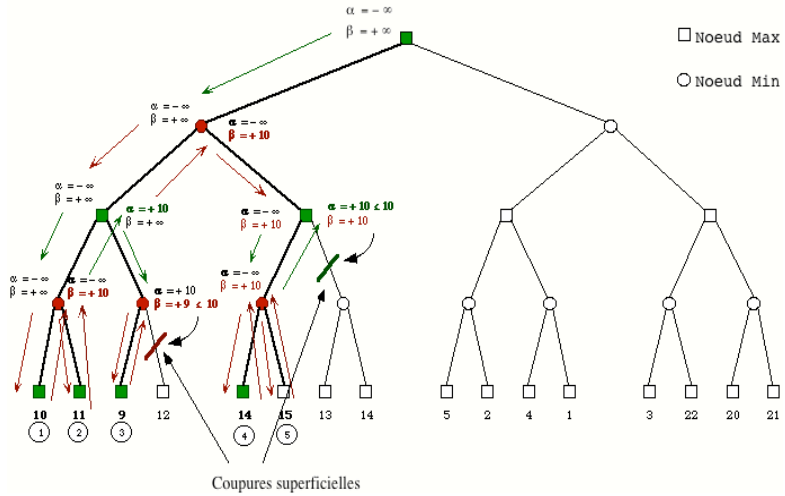
3.4 L'algorithme alpha-beta : Illustration (3)



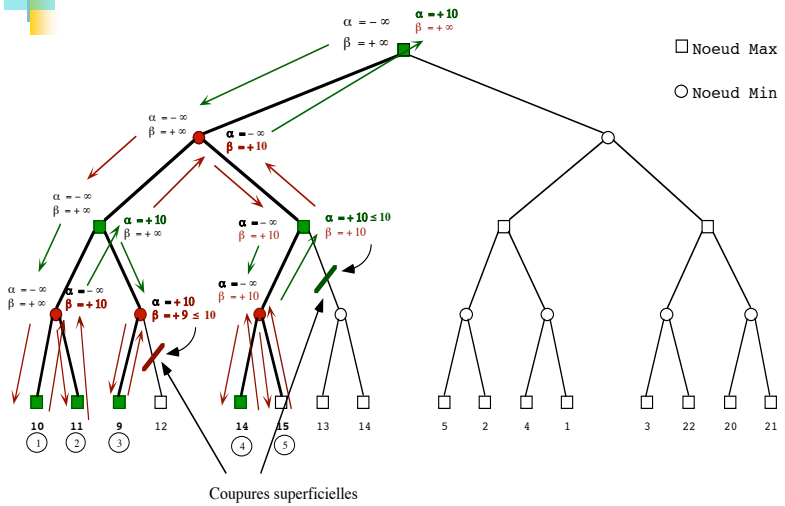
3.4 L'algorithme alpha-beta : Illustration (4)



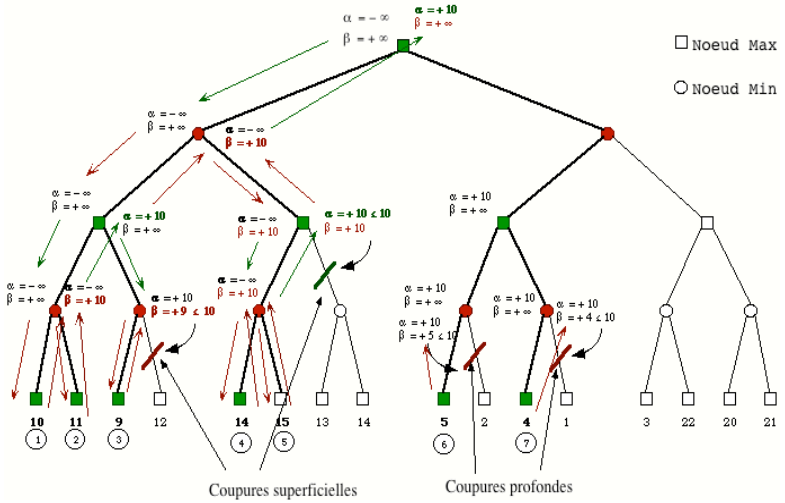
3.4 L'algorithme alpha-beta : Illustration (5)



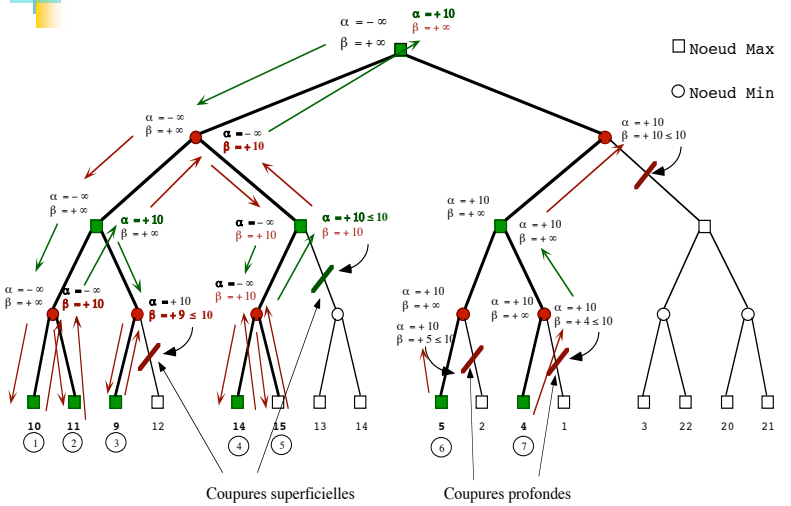
3.4 L'algorithme alpha-beta : Illustration (6)



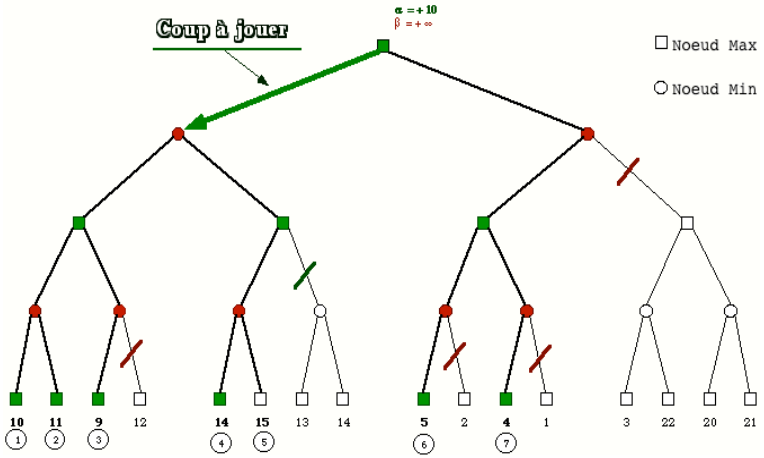
3.4 L'algorithme alpha-beta : Illustration (7)



3.4 L'algorithme alpha-beta : Illustration (8)



3.4 L'algorithme alpha-beta : Illustration (9)



3.4 L'algorithme alpha-beta : Performances

➢ **Le résultat est le même que celui de l'algorithme MinMax**

- *Mais avec **quel avantage en performance ?***
  - Pire cas : $O(b^d)$
  - Cas le plus favorable : $O(b^{d/2})$
    - facteur de branchement effectif = $b^{1/2}$
    - Peut aller deux fois plus loin que Minmax pour la même puissance de calcul
  - Cas asymptotique : $O((b/\log b)^d)$
    - Pour $b > 1000$
  - Cas moyen : $O(b^{3d/4})$
    - Peut aller 4/3 fois plus loin que Minmax pour la même puissance de calcul (pas si mal !!)

$$N_d = \begin{cases} 2\,b^{d/2} - 1 & pour\ d\ pair \\ b^{(d+1)/2} + b^{(d+1)/2} & pour\ d\ impair \end{cases}$$

3.4 L'algorithme alpha-beta : Performances

- ***Comment ordonner les nœuds pour s'approcher du cas optimal ?***
  - Heuristiques
    - Exemple : **d'abord examiner les captures, puis les menaces, puis les mouvements en avant, puis les retraites, …**
  - Recherche en profondeur itérative
    - **Faire un alpha-beta à profondeur 1 : utiliser le résultat pour ordonner les nœuds**
    - **Faire un alpha-beta à profondeur 2 sur l'arbre réordonné à l'étape précédente : utiliser le résultat pour ordonner les nœuds**
    - **Idem à profondeur *n* jusqu'à épuisement des ressources de calcul allouées**

3.5 Les limites

- **Effet d'horizon**
  - Retarder un désastre inévitable en le repoussant au-delà de l'horizon **(parfois au prix de pertes supplémentaires (ex : perdre des pions pour éviter la perte inéluctable de la dame))**
  - Remèdes
    - Attendre des *positions stables* ou à l'équilibre **(ex: sans échecs ou échanges forcés)**
    - *Recherche secondaire* (refaire une petite exploration au-delà de la branche sélectionnée pour vérifier)
- **Pas de plan stratégique**
- **Pas de tentative de piège** (même si possible au prix d'une position éventuellement à peine pire)
- **On suppose que l'adversaire a la même fonction d'évaluation**
  - Remède : apprentissage

3.5 Heuristiques d'amélioration

- **Augmentation des moyens calcul**
  - C'est la source principale des progrès récents
- **Bibliothèque d'ouvertures**
- **Exploration complète en fin de partie**
- **Apprendre la fonction d'évaluation de l'adversaire** (en mémorisant ses parties et en apprenant les conséquences)
  - **Permet d'accélérer l'évaluation et de s'adapter à l'adversaire**
- **Heuristique du coup meurtrier**
  - **Si le coup de l'adversaire *min_1* détruit la position courante au niveau *n*, il est probable qu'il détruit aussi les positions issues de la position courante aux niveaux > *n*, donc *min_1* est à examiner en priorité dans les branches plus bas**
- **Utilisation du temps de réflexion de l'adversaire**
- **… (beaucoup d'autres techniques plus ou moins ad hoc)**

3.5 Le cas du jeu Othello

- **Ouverture**
  - Théorie pauvre sur Othello (contrairement aux échecs)
  - On mémorise position --> coup à jouer (d'après des études statistiques sur les 6 premiers coups)
- **Finale**
  - Le nombre de coups est précisément connu (exactement 60 demi-coups)
  - On lance une recherche exhaustive en fct de la puissance de calcul disponible (les plus gros ordinateurs démarrent 17 coups avant la fin (1998))
  - Rouleau compresseur de la force brute

3.5 Le cas du jeu Othello

- **Milieu de partie**
  - Avoir une bonne fonction d'évaluation
  - Evaluation de la mobilité
    - Combien de degrés de liberté ? **(il vaut mieux peu au début et plus à la fin)**
  - Evaluation statique
    - Poids associé à chaque case

120	–20	20	5	5	20	–20	120
–20	–40	–5	–5	–5	–5	–40	–20
20	–5	15	3	3	15	–5	20
5	–5	3	3	3	3	–5	5
5	–5	3	3	3	3	–5	5
20	–5	15	3	3	15	–5	20
–20	–40	–5	–5	–5	–5	–40	–20
120	–20	20	5	5	20	–20	120

3.5 Le cas du jeu Othello

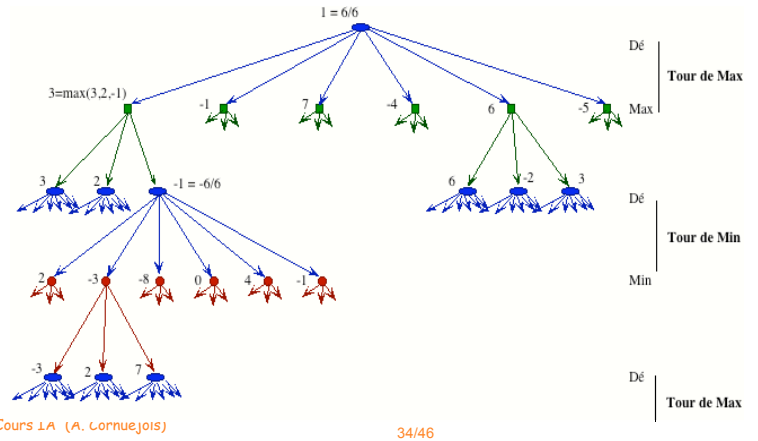
- **Mais il faut modifier cette fonction en cours de jeu**
  - **Exemple: lorsque qu'un coin est pris, modifier les évaluations des cases adjacentes**
- **Tri des nœuds**
  - en fonction de leur valeur statique **(ex: regarder les coups conduisant à l'occupation des coins en premier)** (tri statique)
  - Avec une recherche itérative (tri dynamique : fonction de la situation)
- **Heuristique du coup meurtrier**
- **Optimisations diverses**
  - Occupation de la mémoire (et garbagge collecting)
  - Recherches en parallèle

3.5 Le cas du jeu Othello

- **En 1981,** le champion du monde (Jonathan Cerf) estimait que le meilleur programme d'Othello, Iago, était à son niveau ou meilleur
- **En 1989,** le meilleur programme, Bill, battait le champion du monde, Brian Rose, 56 à 8
  - Iago et Bill utilisent aussi une évaluation très sophistiquée des cases périphériques plus une évaluation du potentiel de mobilité
  - Utilisation d'un alpha-beta à fenêtre **(on fixe l'intervalle [alpha-beta] dans des bornes étroites pour favoriser les coupures)**
  - Iago et Bill économisent du temps en milieu de partie pour l'investir sur la finale
- **Une version simplifiée de Iago/Bill est disponible à l'IIE**

3.5 Elargissement à d'autres types de jeux

- **Jeux avec hasard** (ex: Backgammon)
  - **Notion d'espérance de valeur** (algorithme *expectimax*)



3.5 Elargissement à d'autres types de jeux

- **Jeux avec *n* adversaires (*n* > 2)**

3.5 Leçons

- **Méthodes très différentes du raisonnement humain sur ces jeux**
  - Fonction d'évaluation réduite à un nombre
  - Pas de prise en compte de la stratégie
- **Il est probable qu'elles échoueront sur des jeux comme le Go**

➥ **Raisonnement par reconnaissance des formes et analogie**

➥ **Planification stratégique**

4. Apprentissage de fonction d'évaluation

- **Il y a plusieurs méthodes** (cf. cours d'apprentissage (*apprentissage par renforcement* notamment) de l'option Intelligence Artificielle de 3ème année)
- **Ici, étude d'une méthode particulière en vue d'illustration**
- **Historiquement, Samuel en 1959 est le premier à avoir publié et utilisé une méthode d'apprentissage pour le jeu de dames (américain) : système Checker**

4. Apprentissage de fonction d'évaluation

- **Soit une fonction d'évaluation définie par une combinaison (linéaire) de facteurs**
  - Exemple : **nombre de pièces, occupation du centre ou des angles, mobilité, …**

➥ ***Comment apprendre les bons coefficients ?***

- Exemple : **Samuel utilisait 38 facteurs et se limitait à une combinaison linéaire de ces facteurs avec des coefficients égaux à des puissances de 2 jusqu'à 2^{10}**
  --> 21^{38} fonctions d'évaluation possibles !!

4. Apprentissage de fonction d'évaluation

- **Méthode : recherche par gradient**
  1. Modification de la fonction d'évaluation
  2. et mesure de la performance par rapport à performance précédente
- ***Mesure de la performance ?***
  - Après un ensemble de tournois (très coûteux)
  - En cours de partie, à chaque calcul de coup à jouer, comparer la valeur statique retournée par la fonction d'évaluation avec l'évaluation retournée en utilisant en plus une recherche en avant alpha-beta

    (Si la fonction d'évaluation est correcte, elle devrait retourner un résultat proche de la valeur retournée par alpha-beta)

4. Apprentissage de fonction d'évaluation

Autre méthode :

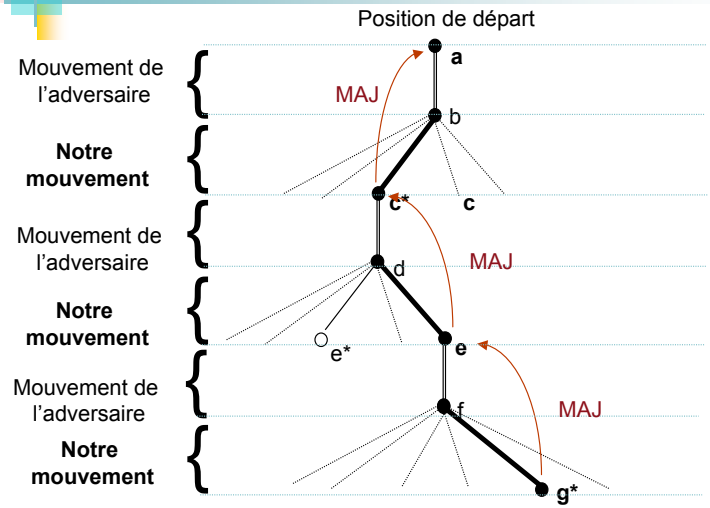
Apprentissage par renforcement par la méthode des différences temporelles (TD-learning)

- **On apprend une fonction *V* : position --> valeur**
  - Initialement, toutes les positions (qu'il faut pouvoir énumérer) sont évaluées à 0,5 sauf les positions perdantes (évaluées à 0) et les positions gagnantes (évaluées à 1)
  - On met à jour l'évaluation de la position *s* en fonction de l'évaluation de la position suivante $\delta(s)$

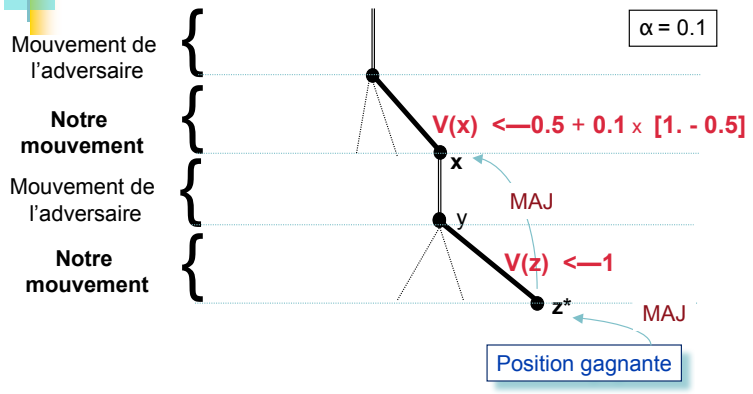
$$V(s_{t+1}) \leftarrow V(s_t) + \alpha[\max(V(\delta(s_t)) - V(s_t)]$$

(α fonct. décroissante)

4. Apprentissage de fonction d'évaluation



4. Apprentissage de fonction d'évaluation



- L'état *x* passe d'une valeur de 0.5 à 0.55
- Si l'adversaire joue toujours *y* dans la position *x* sa valeur ne cessera d'augmenter [même si *x* est, dans l'absolu, une position perdante]

4. Apprentissage de fonction d'évaluation

- ***Questions***
  - *Qu'est-ce qui assure la convergence vers la « politique » optimale ?*
  - *Comment régler les paramètres (e.g. α) ?*
  - *Comment généraliser ?*
    - Ne pas stocker une valeur pour chaque situation possible
  - *A partir de quelles expériences apprendre ?*
    - Machine contre machine ?
    - Machine contre expert ?
    - Machine contre joueur moyen ?
    - Machine contre plusieurs joueurs ?

4. Apprentissage de fonction d'évaluation

- **Attention :**
  - **Le système** s'adapte à l'adversaire
    - Peut devenir très mauvais sur un autre adversaire
  - **Il faut aussi des** coups exploratoires **(risqués) pour éviter de toujours jouer de manière myope ce qui semble le meilleur (optimum local)**
  - **Problèmes si trop de positions :**
    - Il faut généraliser les positions pour apprendre

      On associe une évaluation à un type de position

État de l'art

- **Checkers** (10^32) : Chinook > H
- **Othello** (10^58) : Logistello > H
- **9x9 Go** (10^85) : Meilleur pgm << H
- **Échecs** (10^123) : Deep Blue >= H
- **19x19 Go** (10^400) : Meilleur pgm <<<<< H
- **Backgammon** : TD-Gammon ~ H
- **Pocker** : beaucoup de travaux en cours

http://www.gameai.com/clagames.html

Sources documentaires

- **Ouvrages / articles**
  - Ginsberg M. (93) : *Essentials of Artificial Intelligence*. Morgan Kaufmann 1993.
  - Nilsson N. (98) : *Artificial Intelligence : A new synthesis*. Morgan Kaufmann, 1998.
  - Russel S. & Norvig P. (95) : *Artificial Intelligence : A modern approach*. Prentice Hall, 1995.
- **Sites web**
  - http://www.gameai.com/clagames.html