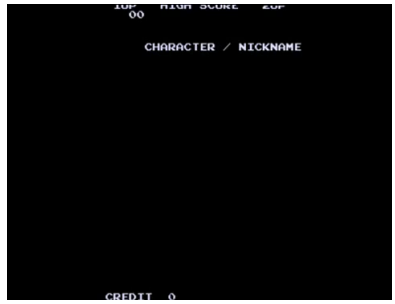
6 De l'I.A. dans les jeux

- L'I.A. des jeux vidéo
- Les vrais jeux
- Fonction d'évaluation
- Espace d'états
- Algorithme du min-max
- Procédure alpha-beta
- Quelques considérations sur les programmes de jeux

L'I.A. de Pong (1979)

L'I.A. de Pac Man (1980)



- Blinky attaque directement Pac Man. Il suit Pac-Man comme son ombre.
- Pinky a tendance à se mettre en embuscade. Elle vise l'endroit où va se trouver Pac-Man.
- Inky est capricieux. De temps en temps, il part dans la direction opposée à Pac-Man.
- Clyde feint l'indifférence. De temps en temps, il choisit une direction au hasard (qui peut être celle de Pac-Man).

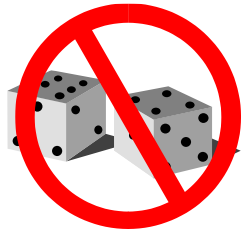
L'I.A. du XX^{ème} au XXI^{ème} siècle

De l'I.A. dans les jeux

- IA dans les jeux et test de Turing ?
  - on n'a pas besoin de croire que l'adversaire est un humain
  - on veut juste s'amuser
- Simuler des adversaires, des partenaires
  - d'un niveau permettant à l'humain de trouver de l'intérêt
    - crédibles, intelligents, à la hauteur
    - pas forcément prévisibles

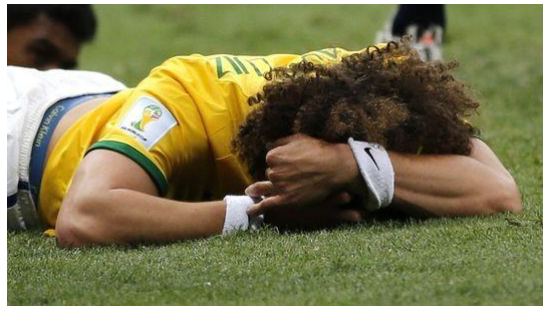
Les vrais jeux (1)

- Jeux considérés
  - jeux à connaissance complète
    - chaque joueur voit l'intégralité du plateau
    - chaque joueur connaît les coups légaux, et leurs conséquences
  - jeux opposant deux adversaires (ami et ennemi)



Les vrais jeux (2)

- Partie dans un état initial
- Situation de fin de parties connues
  - défaite
  - victoire
  - match nul



Les vrais jeux (3)

- Quelques exemples de jeux
  - dames
  - échecs
  - go
  - morpion
  - puissance 4
  - awélé

Les vrais jeux (4)

- Objectif d'une I.A. de « vrai » jeu
  - la machine doit simuler un joueur « intelligent »
    - si la machine perd toujours, c'est pas rigolo
    - si la machine joue au hasard, c'est pas rigolo longtemps
    - si la machine gagne toujours, c'est pas rigolo non plus
      - possibilité de régler le niveau
- Des illusions…
  - en 1957, Newell et Simon annonçaient la victoire de la machine sur le champion du monde d'échecs dans les 10 ans
- … à la réalité
  - … arrivée 40 ans plus tard avec Deep Blue

Fonction d'évaluation (1)

- A chaque fois que le programme a le trait
  - choisir le coup qui le mènera à la victoire
    - avec certitude
    - ou au moins avec la plus forte « probabilité »
- Etre capable d'évaluer le plateau
  - espérance de victoire

  ⇒ fonction d'évaluation

$f($  $) = 20/20$!

Fonction d'évaluation (2)

- Si l'on dispose d'une telle fonction
  - le programme doit jouer
    - étudie pour chaque coup légal la valeur du plateau
    - choisit le coup qui conduit à la valeur la plus favorable
- Critères sur lesquels construire une fonction
  - défaite : valeur faible
  - victoire : valeur forte
  - pièce ennemie menacée (ou prise) : valeur forte
  - pièce amie menacée : valeur faible
  - puissance 4 : 2 pions alignés < 3 pions alignés

Fonction d'évaluation (3)

- Problèmes liés à la fonction d'évaluation
  - ne peut pas « prévoir l'avenir »
    - un coup nous semble très favorable
    - le coup suivant, l'adversaire peut gagner

Espace d'états (1)

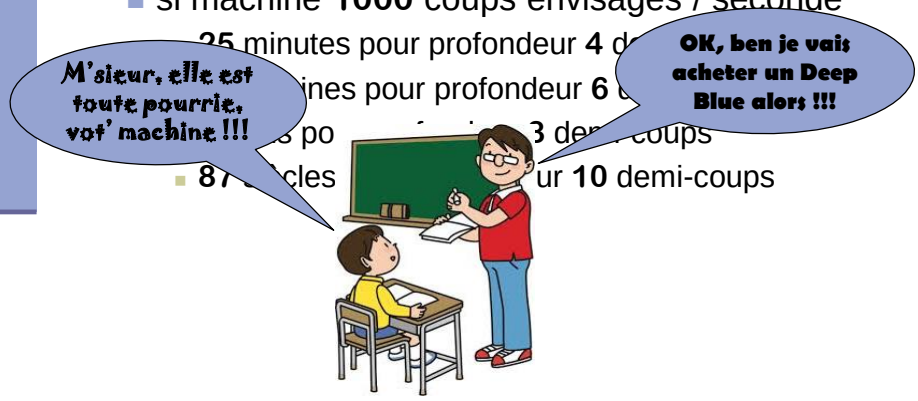
- Les fonctions d'évaluation ne sont pas extralucides
- Idée
  - envisager toutes les parties théoriquement possibles
    - pour chacun de mes coups légaux, j'envisage tous les coups de l'adversaire et ainsi de suite
    - choisir un coup qui mène à la victoire
  - arbre des coups légaux

Espace d'états (2)

- Aux échecs
  - facteur de branchement moyen : **35**
  - si machine **1000** coups envisagés / seconde
    - **25** minutes pour profondeur **4** demi-coups
    - **3** semaines pour profondeur **6** demi-coups
    - **70** ans pour profondeur **8** demi-coups
    - **87** siècles pour profondeur **10** demi-coups

Espace d'états (2)

- Aux échecs
  - facteur de branchement moyen : **35**
  - si machine **1000** coups envisagés / seconde
    - **25** minutes pour profondeur **4** d
    - ines pour profondeur **6**
    - s po 3 de coups
    - **87** cles ur **10** demi-coups



Espace d'états (3)

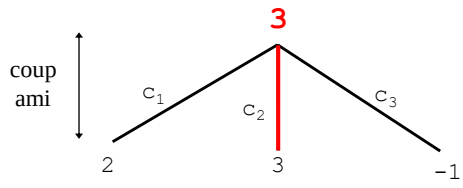
- Deep Blue
  - **3 milliards** coups envisagés / seconde
    - **0,5 ms** pour profondeur **4** demi-coups
    - **0,6 sec** pour profondeur **6** demi-coups
    - **12 min** pour profondeur **8** demi-coups
    - **10 jours** pour profondeur **10** demi-coups
    - **35 ans** pour profondeur **12** demi-coups
    - **437 siècles** pour profondeur **14** demi-coups
  - partie de grand maître : environ **100** demi-coups !
- Explosion combinatoire
  - ≈ impossible de construire tout l'arbre

Algorithme du min-max (1)

- Vision nécessairement partielle de l'avenir
  - choisir le meilleur coup possible
- Exemple
  - facteur de branchement : **3**
  - fonction d'évaluation **-10** ≤ **h(x)** ≤ **+10**

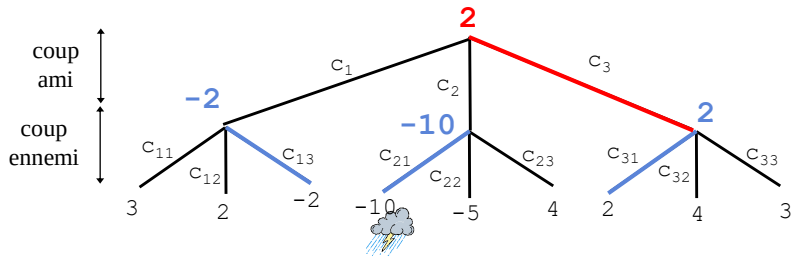
Algorithme du min-max (2)

- Si le programme « voit » à un demi-coup



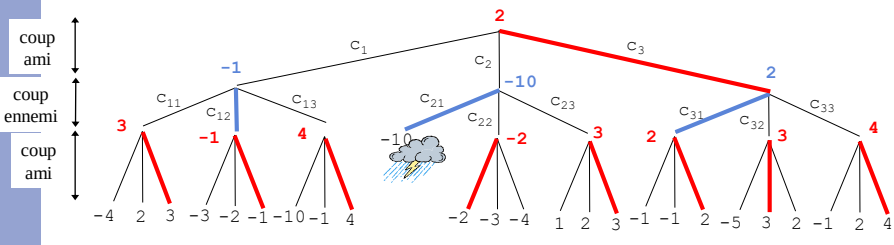
Algorithme du min-max (3)

- Si le programme « voit » à deux demi-coups



Algorithme du min-max (4)

- Si le programme « voit » à 3 demi-coups



Algorithme du min-max (5)

```
fonction min-max (prof : in entier; damier : in t_dam; coup :
t_coup) : entier
début
       d ← jouer(damier, coup)
       si (prof = MAXPROF) alors retourner évaluation(d)
       si (prof % 2 = 0)  // nœud max
       alors
              maxi ← -∞
              pour tout c tel que coup-légal(c) faire
                     maxi ← max(maxi,min-max(prof+1, d, c))
              finpour
              retourner maxi
       sinon // nœud min
              mini ← +∞
              pour tout c tel que coup-légal(c) faire
                     mini ← min(mini,min-max(prof+1, d, c))
              finpour
              retourner mini
       finsi
fin
```

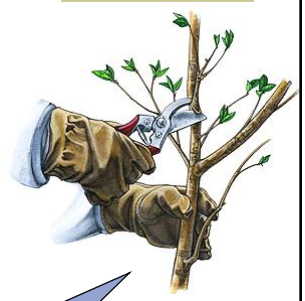
Algorithme du min-max (6)

```
Appel :

maxi ← -∞
pour tout c tel que coup-légal(c) faire
       val ← min-max(1, damier, c)
       si val > maxi
       alors
              meilleur ← c
              maxi ← val
finpour
jouer(damier, meilleur)
```

Procédure alpha-beta (1)

- Échecs
  - **1 million** de coups testés / seconde
  - **60** secondes par coup
  - **4** à **5** demi-coups de profondeur (médiocre)
- Ne pas examiner certaines branches
  - examiner d'autres plus en profondeur
- Coupes
  - heuristiques
  - correctes

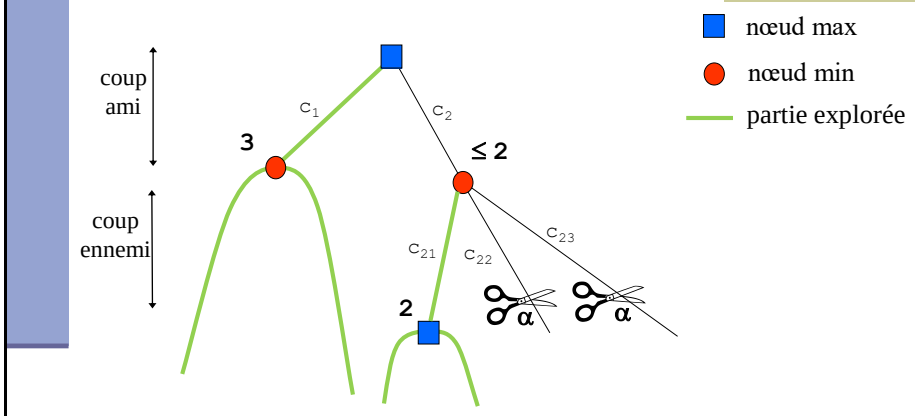


pruning

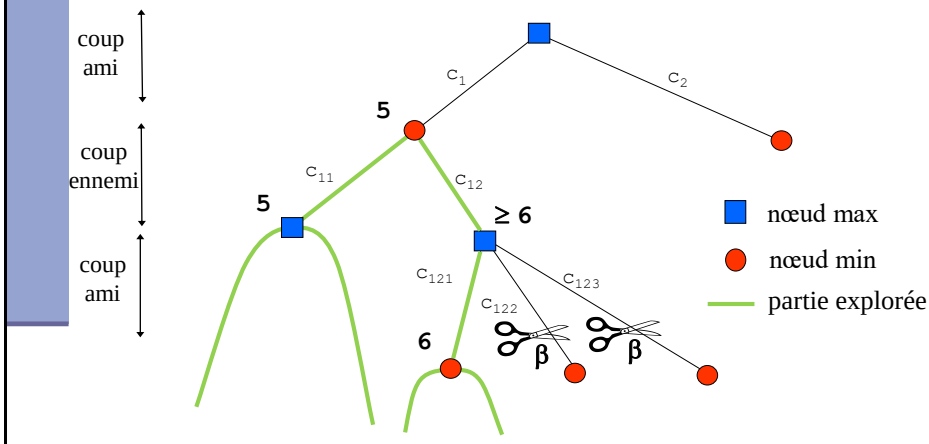
Procédure alpha-beta (2)

- alpha-beta effectue des coupes correctes
  - le résultat est le même que le min-max
  - certaines branches ne sont pas explorées
    - on sait qu'elles ne modifieront pas le résultat
- Le gain de temps peut être important
  - on peut donc, si le temps est limité, développer l'arbre des coups légaux plus en profondeur

Procédure alpha-beta (3)



Procédure alpha-beta (4)



Quelques considérations...

- Défauts du min-max
  - aucune stratégie
  - effet d'horizon

Améliorations du min-max (1)

- Maximiser les coupes
  - en ordonnant les branches en fonction de la valeur qu'elles avaient lors du dernier coup
    - paradoxal
  - en lançant la fonction d'évaluation à chaque niveau de l'arbre (jusqu'à une profondeur donnée)
    - efficace

Améliorations du min-max (2)

- Imiter le comportement humain
  - ne pas développer toutes les branches
  - développer quelques branches en profondeur
- Programme min-max approximation
  - développer le nœud qui provoque la plus grande augmentation de la racine
  - évaluation lancée à chaque coup
  - utilise des dérivées partielles de la fonction d'évaluation

IBM Deep Blue : force brute

- IBM Deep Blue aux échecs
  - 10/02/1996 bat Garry Kasparov sur une partie
  - mais Kasparov gagne le match 4-2
  - mai 1997 : Kasparov battu 3 ½ - 2 ½

- Caractéristiques
  - architecture massivement parallèle fondée sur une technologie de type RS/6000 à 32 cœurs
  - capable d'évaluer 200 millions de positions par seconde

Google AlphaGo : deep learning

- AlphaGo
  - octobre 2015 bat un joueur professionnel
  - mai 2017 bat le champion du monde Ke Jie
- Deep learning
  - entraîné sur des dizaines de milliers de parties
  - millions de parties contre lui-même

AlphaGo

La planification (1)

- Grand Maître aux échecs
  - 35 coups analysés
  - 6,8 demi-coups de profondeur
- Bon amateur
  - 30,8 coups analysés
  - 6,6 demi-coups de profondeur
- La différence ne se fait pas sur la capacité de calcul !

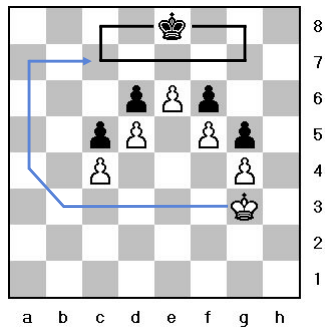
La planification (2)

- Il manque à la machine de la stratégie !!!

« Il est préférable de suivre rigoureusement un plan même si ce n'est pas le meilleur, plutôt que de jouer sans plan du tout. »

- Planification
  - technique utilisée en robotique
    - but ⇒ plan
    - plan ⇒ séquence de sous-plans…
  - limite l'explosion combinatoire
    - en orientant la progression dans l'espace d'états

La planification (3)



- plan : soutenir le pion qui peut aller à promotion tout en menaçant les pions de l 'adversaire
- min-max : 3000 ans de calcul
  - 20 coups à jouer, facteur de branchement de 5

Bibliographie

- Claude Berge, Graphes
- Didier Müller, Introduction à la théorie des graphes
- Eric Sopena, Eléments de théorie des graphes
- Jean-Charles Régin, Arnaud Malapert, Théorie des Graphes
- David Harel, Algorithmics: The Spirit of Computing, Addison Wesley, 2004