Intelligence Artificielle et Jeux

ECM 2A 2018-2019

Thierry Artières

CENTRALE MARSEILLE

Partie II
Exploration avec adversaire

Jeux vs. problèmes de recherche

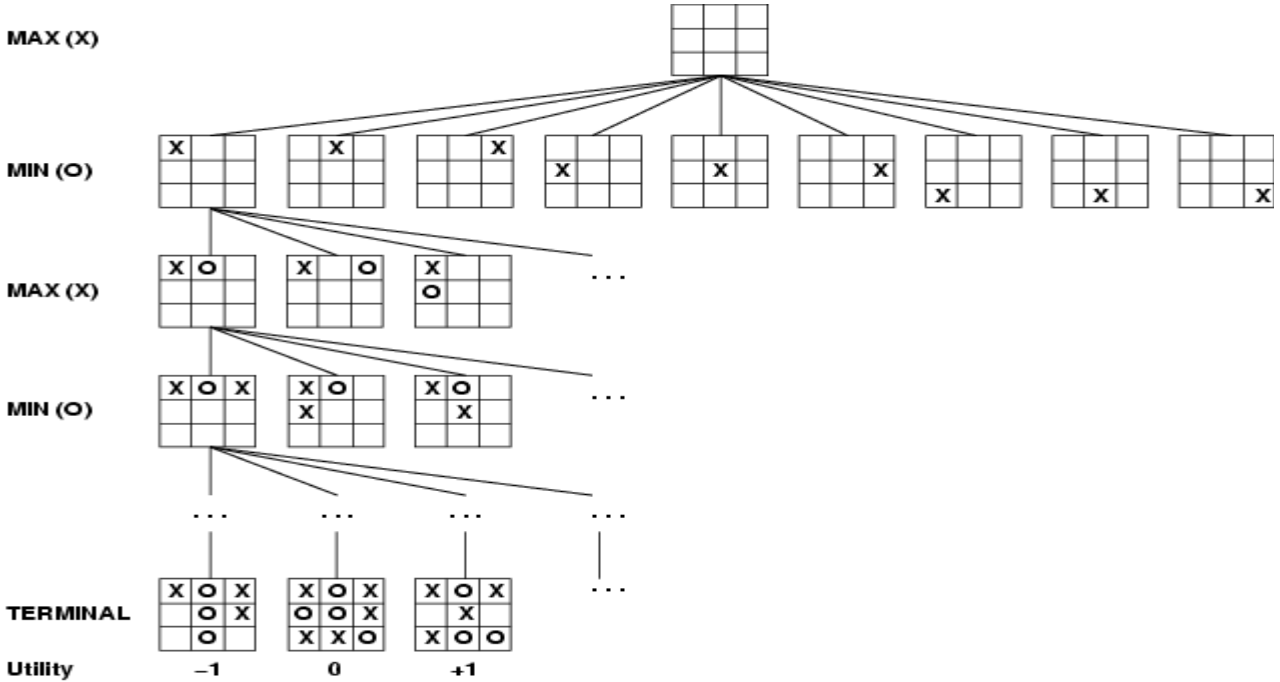
- Adversaire “imprédictible"

⇒ Prévoir un coup pour chaque coup possible de l’adversaire

- Limites de temps

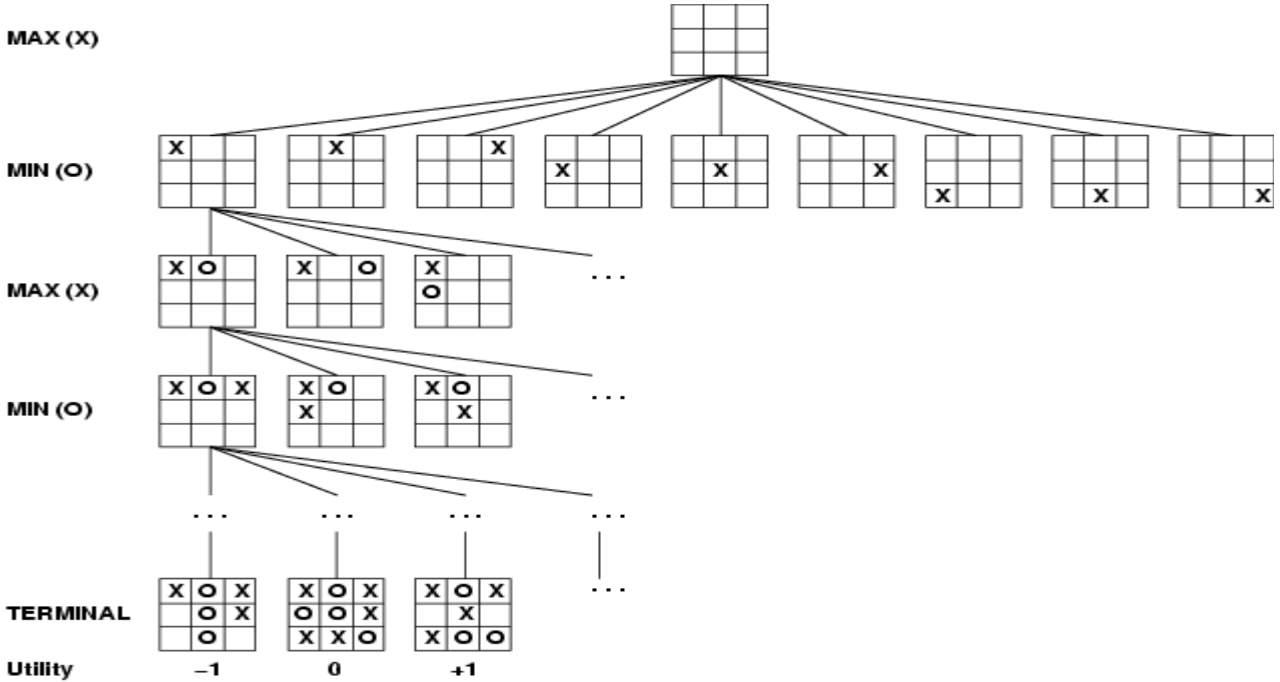
⇒ Peu de chances de trouver le but, donc on approxime

Arbre de jeu (2 joueurs, déterministe, tours)



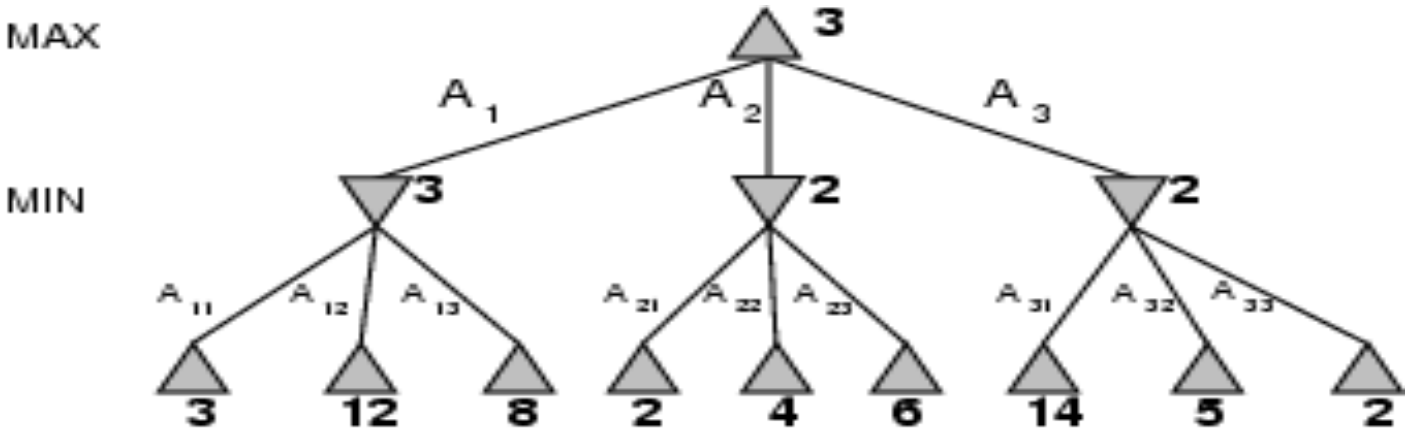
- Décisions optimales
- Elagage par α-β
- Imparfait, mais temps réel possible

Arbre de jeu (2 joueurs, déterministe, tours)



Minimax

- Jeu parfait pour des jeux déterministes
- Idée: Choisir le coup avec la meilleure valeur de minimax
  = meilleur situation atteignable contre le meilleur coup
- E.g., jeu à 2 joueurs:



Algorithm Minimax

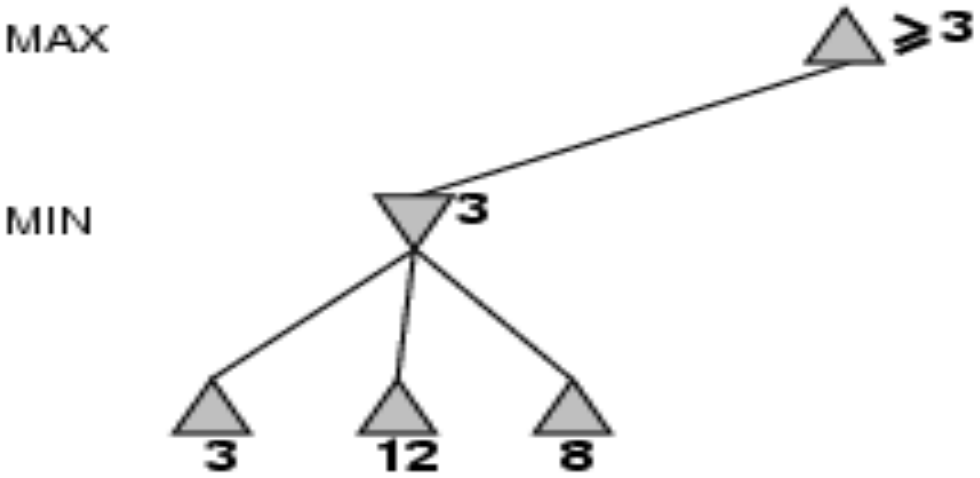
```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Propriétés de minimax

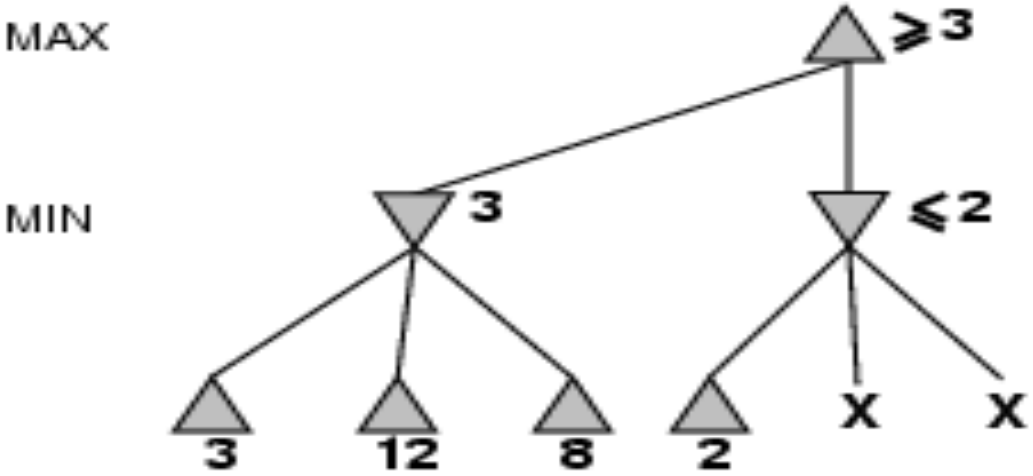
- Complet? Oui (si l'arbre est fini)
- Optimal? Oui (contre un adversaire optimal)
- Complexité en temps ? $O(b^m)$
- Complexité en mémoire ? O(bm) (profondeur d'abord)

- Echecs : b ≈ 35, m ≈100 pour des parties "raisonnables"
  → solution exacte non calculable

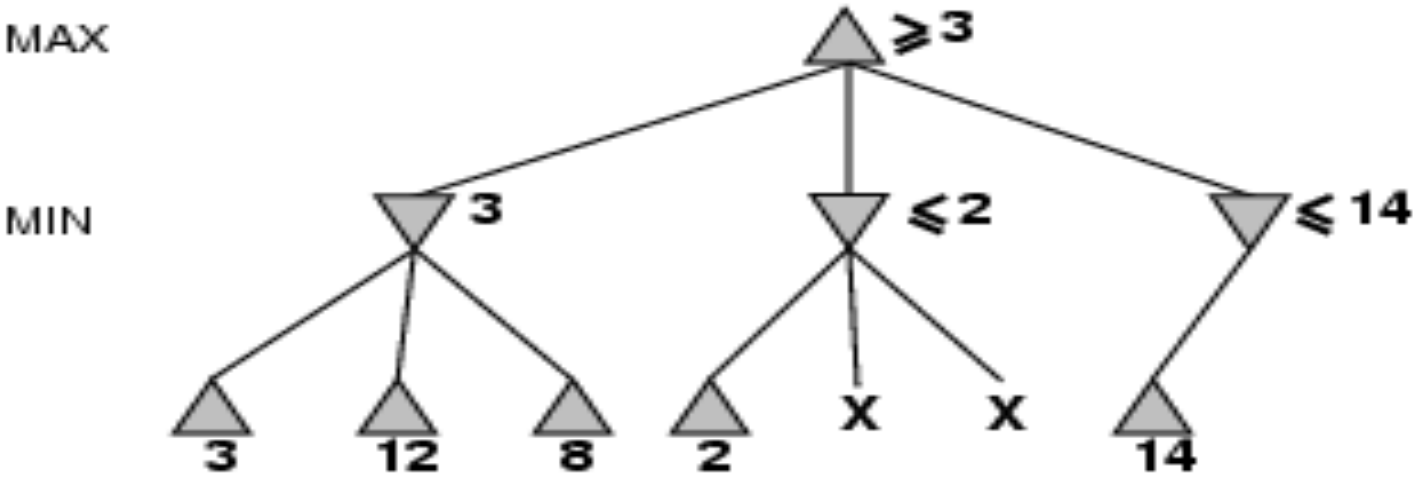
Elagage α-β : exemple



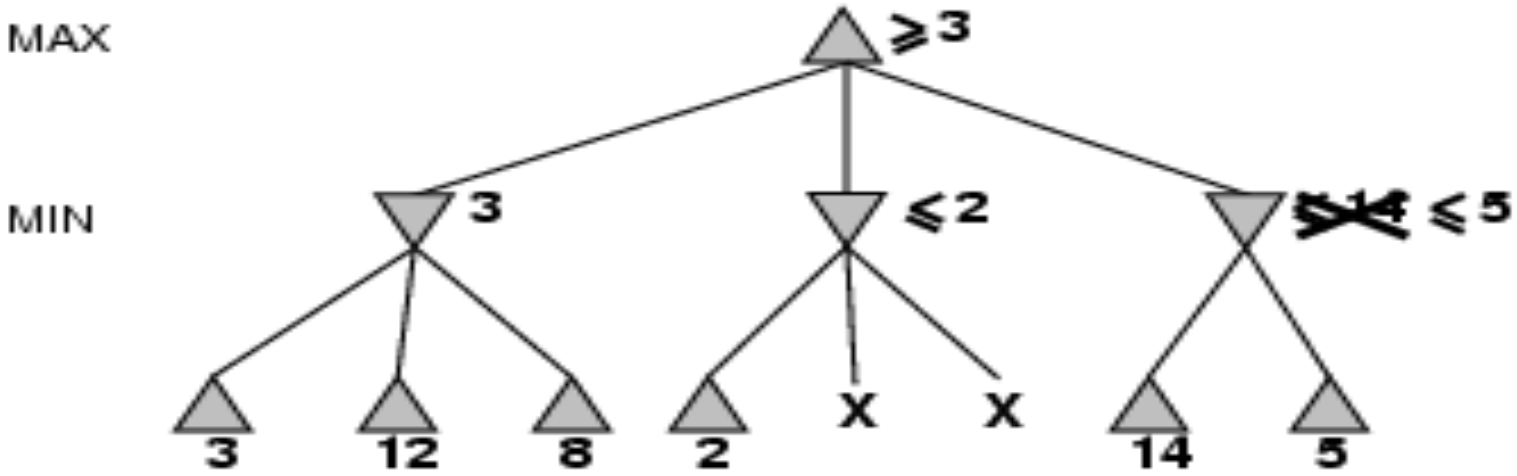
Elagage α-β : exemple



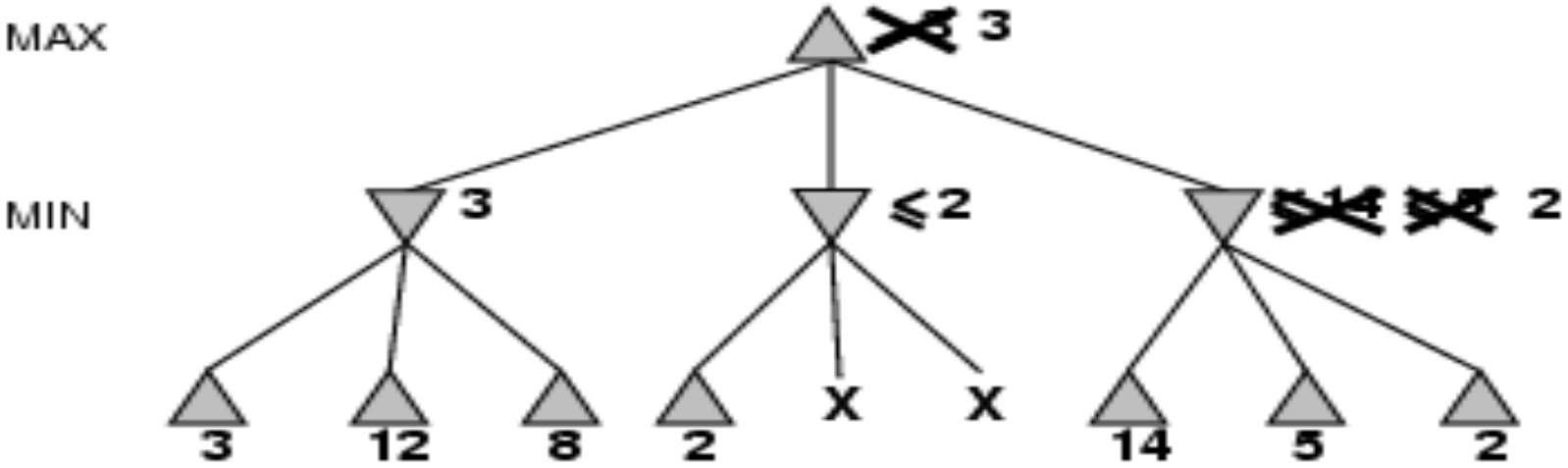
Elagage α-β : exemple

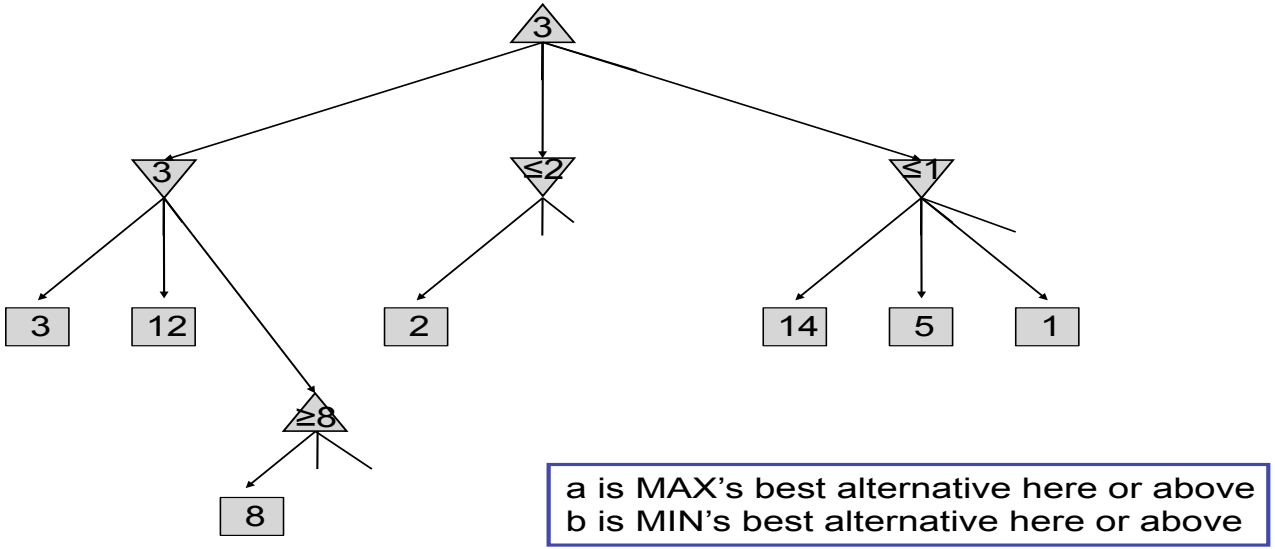


Elagage α-β : exemple



Elagage α-β : exemple




3
3
≤2
≤1
3
12
2
14
5
1
≥8
8
a is MAX's best alternative here or above
b is MIN's best alternative here or above

Pourquoi α-β?

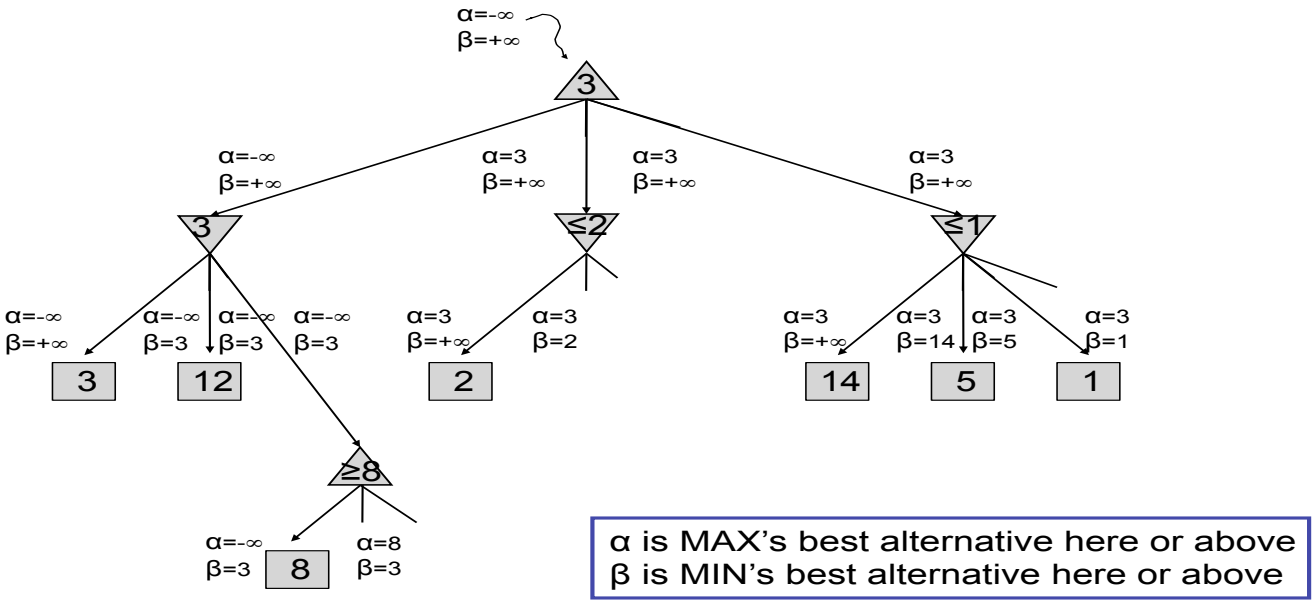
- α est la valeur de la meilleure (i.e., + grande valeur) alternative trouvée pour le joueur *max* jusqu'ici le long du chemin jusqu'au noeud courant
- β est la valeur de la meilleure (i.e., + petite valeur) alternative trouvée pour le joueur *min* jusqu'ici le long du chemin jusqu'au noeud courant

- Si en explorant les fils d'un noeud *max* on remonte une valeur supérieure à β, on arrête l'exploration et on remonte cette valeur
- Car le noeud calcule un max des valeurs remontées par ses fils

$\Longrightarrow$ On sait qu'on remontera une valeur moins favorable que celle qui a été trouvée par ailleurs

$\Longrightarrow$ On élimine la branche

Pourquoi α-β?



α is MAX's best alternative here or above
β is MIN's best alternative here or above

L'algorithme α-β

```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game
    v ← MAX-VALUE(state, −∞, +∞)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s, α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

L’ algorithme α-β

```
function MIN-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s, α, β))
        if v ≤ α then return v
        β ← MIN(β, v)
    return v
```

Propriétés de α-β

- L'élagage n'affecte pas le résultat final
- Un bon ordonnancment des coups améliore l'efficacité de l'élagage
- Avec "un tri parfait," compléxité en temps = $O(b^{m/2})$

  → permet de doubler la profondeur de recherche

- Exemple de raisonnement sur la pertinence de certains calculs (sorte de métaraisonnement)

Problème de ressources limitées

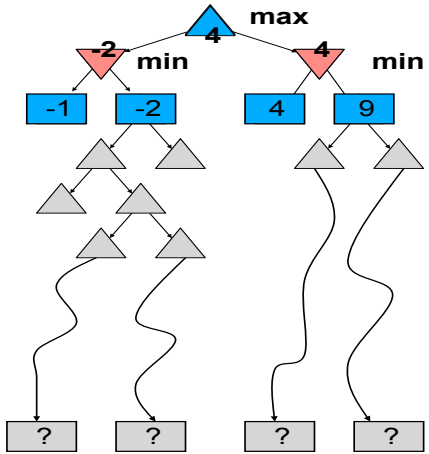
Supposons qu'on dispose de 100 secs, et qu'on explore 10^4 noeuds/sec
→ 10^6 noeuds par coup

Approche standard:

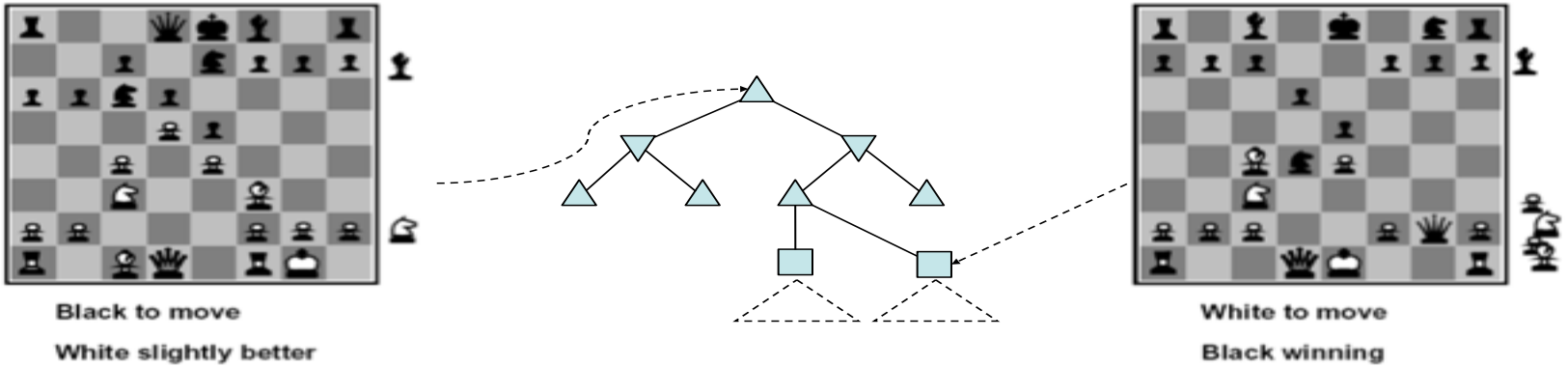
- Test d'arrêt :

  e.g., profondeur limite
- Fonction d'évaluation

  = estimation de l'utilité d'une position
- Perte de garantie sur l'optimalité



=> Fonctions d'évaluation



- Pour les échecs, typiquement une combinaison linéaire de caractéristiques

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

- e.g., $w_1 = 9$ avec
  $f_1(s)$ = (Nombre de tours noires) – (nombre de tours blanches), etc.

Ressources limitéess

Supposons qu'on dispose de 100 secs, et qu'on explore 10^4 noeuds/sec
→ 10^6 noeuds par coup

Approche standard:

- Test d'arrêt :

  e.g., profondeur limite

- Fonction d'évaluation

  = Estimation de l'utilité d'une position

Recherche avec test d'arrêt

MinimaxCutoff est identique à *MinimaxValue* sauf que

1. *Final?* est remplacé par *Arrêt?*
2. *Utilité* est remplacé par *Eval*

En pratique:

$b^m = 10^6$, b=35 → m=4

Exploration à un horizon de 4 coups => très mauvais joueur d'échecs

- Horizon à 4 coups ≈ novice
- Horizon à 8 coups ≈ typique PC
- Horizon à 12 coups ≈ Deep Blue, Kasparov

Jeux déterministes en pratique

- Dames: Chinook a gagné le championnat du monde en 1994. utilise une base des coups idéaux pour toutes les positions avec moins de 8 pièces, 444 milliards de positions.
- Echecs: Deep Blue a battu Garry Kasparov en 1997. Deep Blue évalue 200 million positions par seconde, et utilise des stratégies explorant jusqu'à 40 coups.
- Othello: Les champions humains de jouent pas contre les ordinateurs, trop bons.
- Go [2010]: Les champions humains de jouent pas contre les ordinateurs, trop mauvais. Au go, $b > 300$.
- Go [2015] : AlphaGo (Google Deepmind) bat le champion du monde de Go

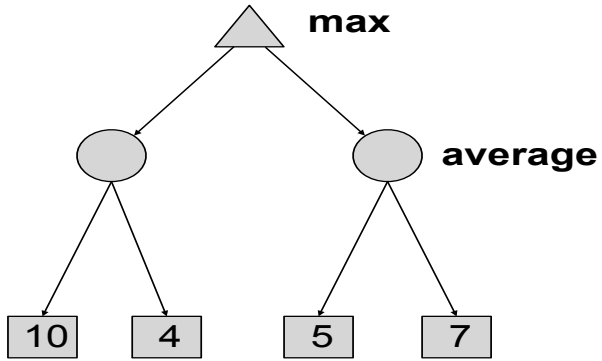
Jeux avec résultats d'actions non déterministes

Cas d'un joueur aléatoire

- Résultat d'une action non déterministe

⇒ Ajout de noeuds *Chance* pour lesquels on moyenne les scores des fils

⇒ Sinon algos identiques à Minimax



Cas d'un jeu avec aléatoire (par ex. BackGammon)

⇒ Algo Expectminimax

⇒ L'environnement est un joueur (noeuds Chance) qui joue après chaque joueur et qui calcule l'espérance.