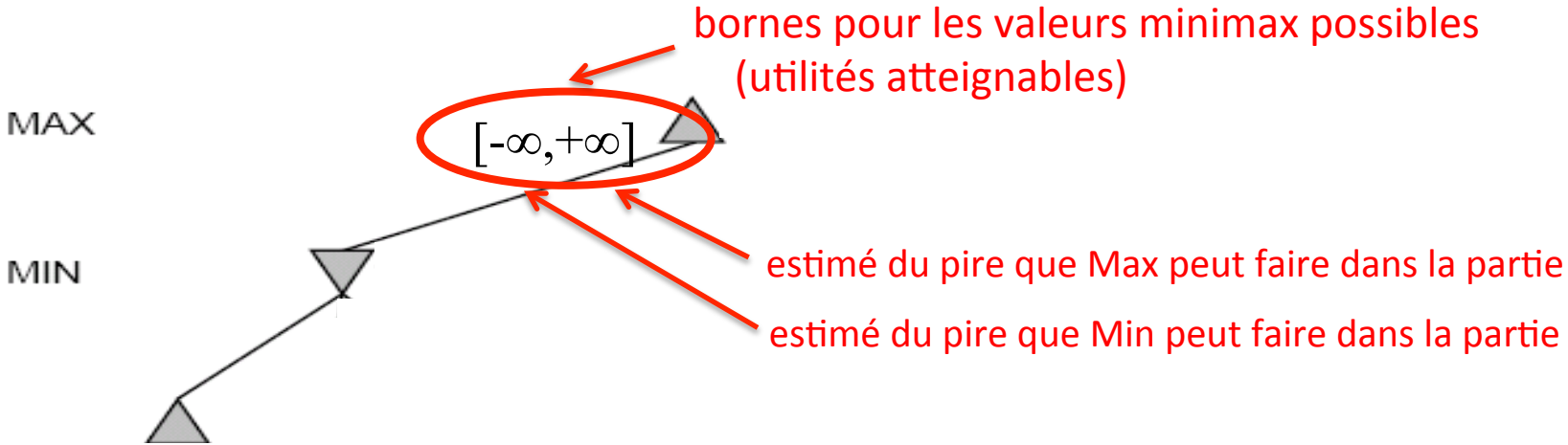
Comment accélérer la recherche

- Deux approches
  - la première maintient l'exactitude de la solution
  - la deuxième introduit une approximation

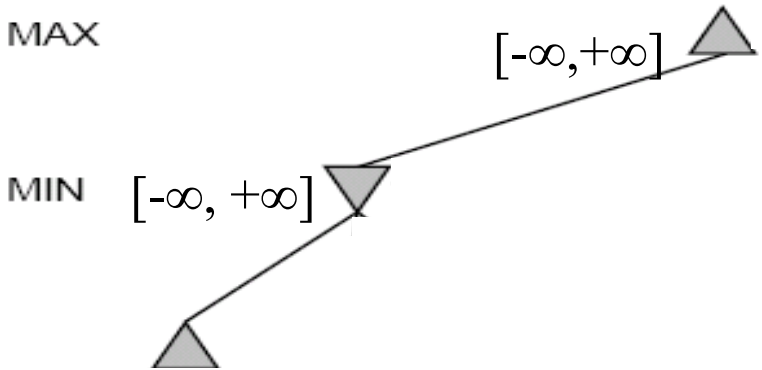
1. Élagage alpha-bêta (*alpha-beta pruning*)
  - **idée :** identifier des chemins dans l'arbre qui sont explorés inutilement

2. Couper la recherche et remplacer l'utilité par une fonction d'évaluation heuristique
  - i**dée :** faire une recherche la plus profonde possible en fonction du temps à notre disposition et tenter de prédire le résultat de la partie si on n'arrive pas à la fin

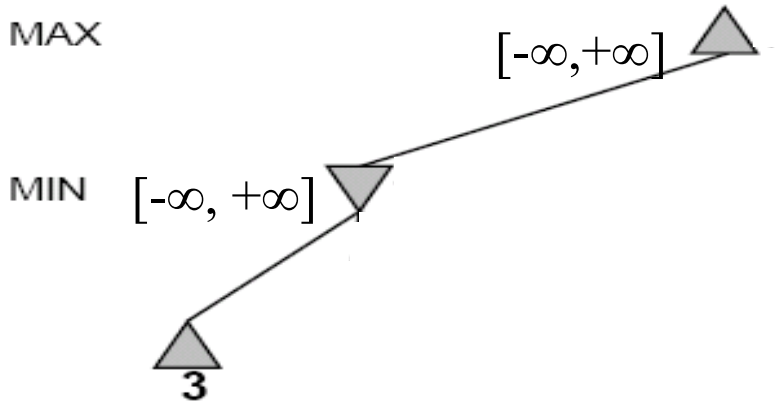
Alpha-beta pruning



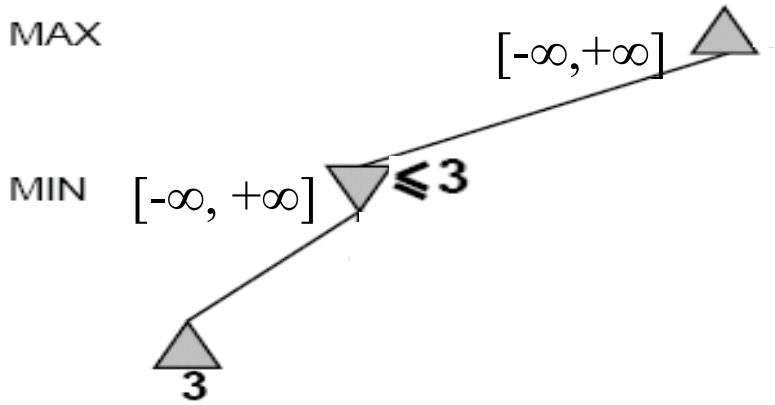
Alpha-beta pruning



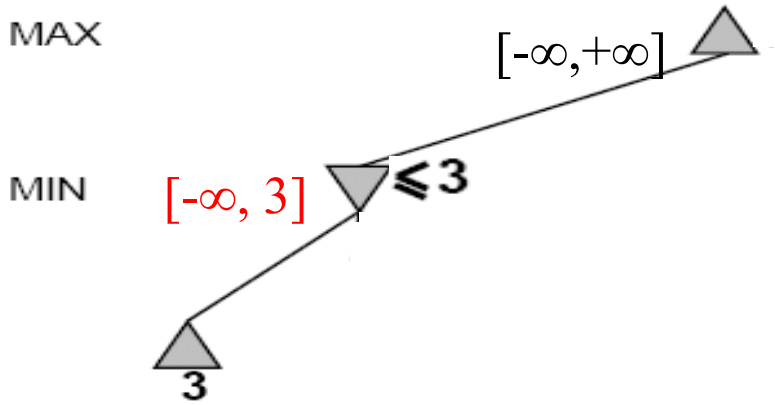
Alpha-beta pruning



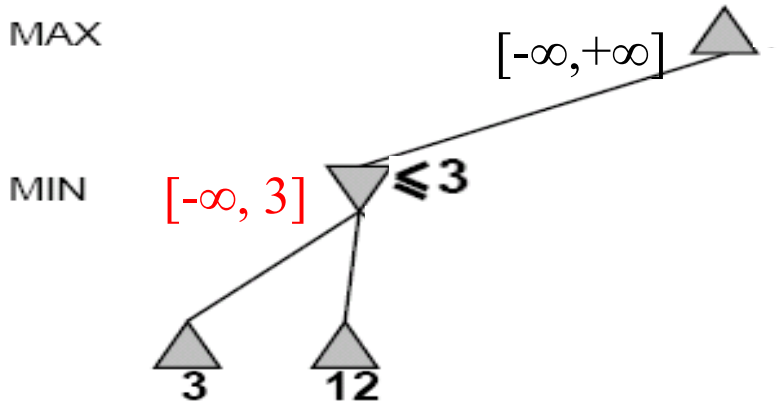
Alpha-beta pruning



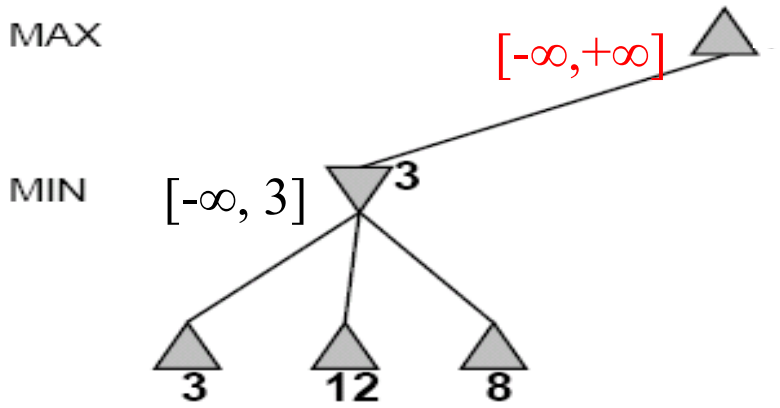
Alpha-beta pruning



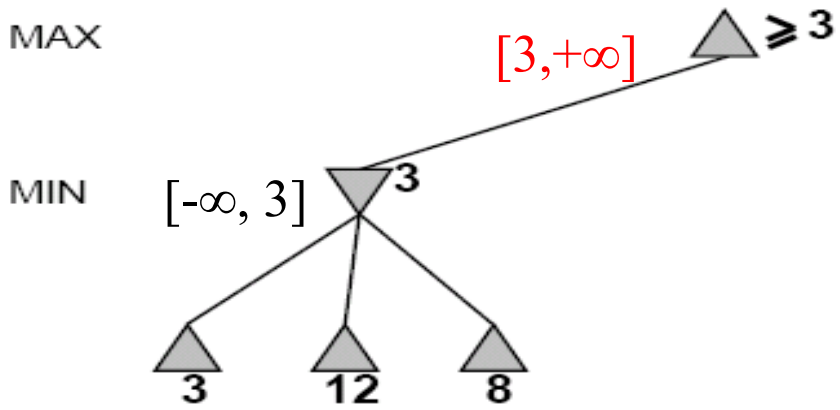
Alpha-beta pruning



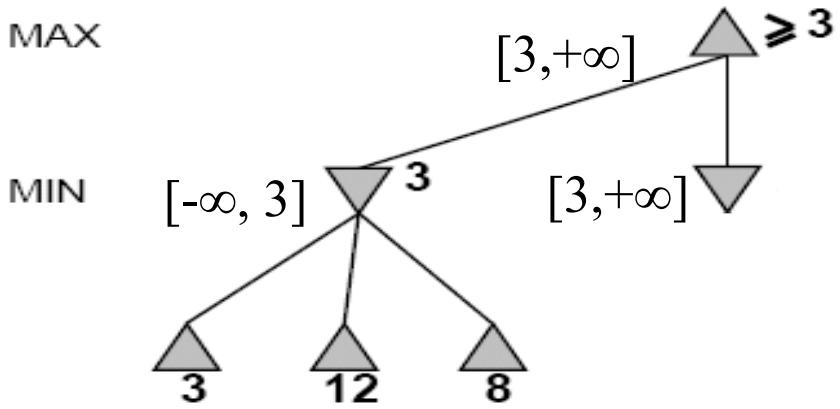
Alpha-beta pruning



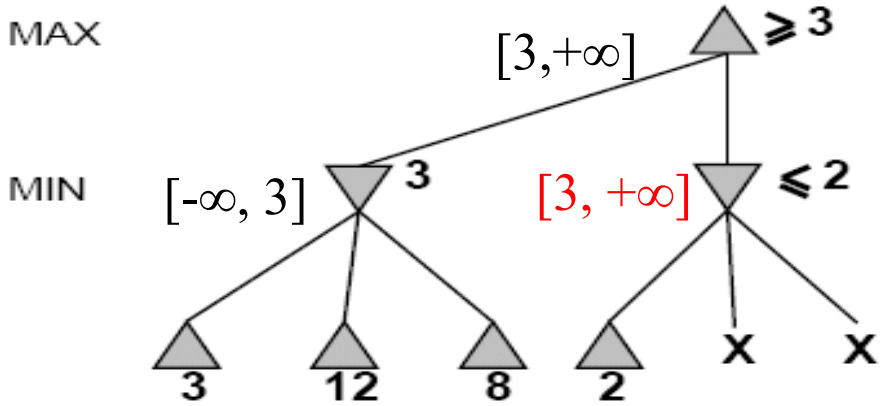
Alpha-beta pruning



Alpha-beta pruning

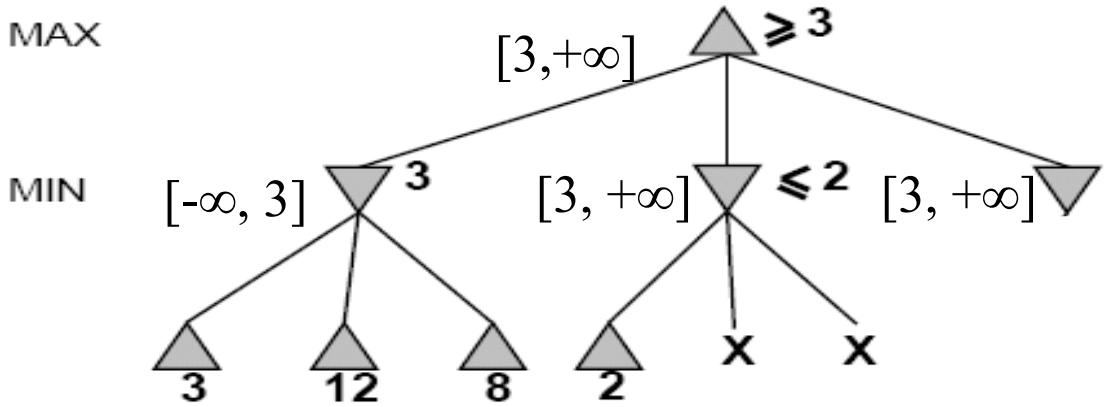


Alpha-beta pruning

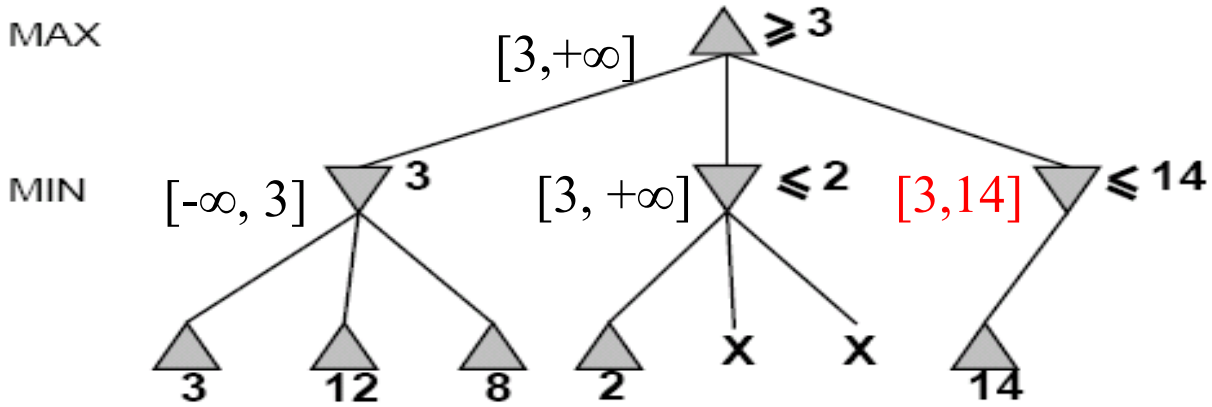


Contradiction entre partie globale et partie à ce état

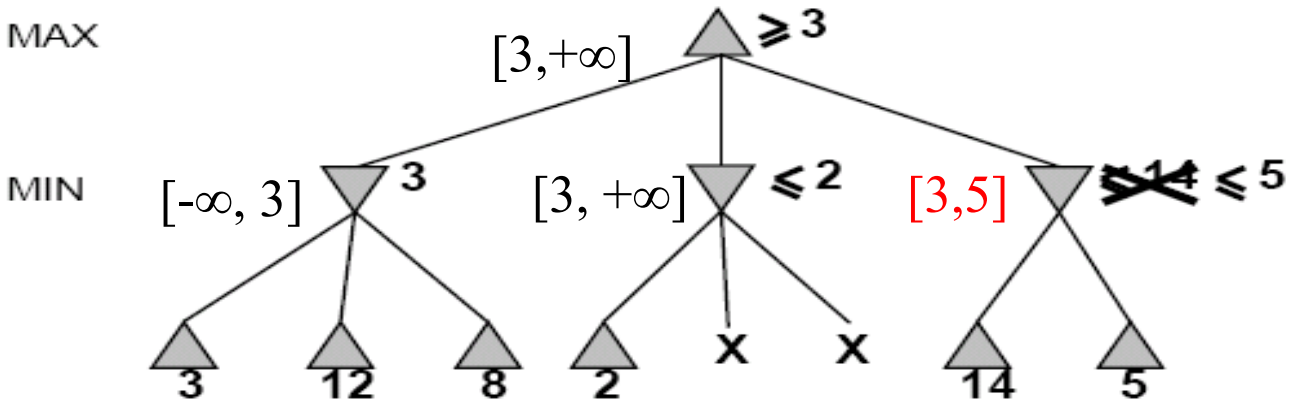
Alpha-beta pruning



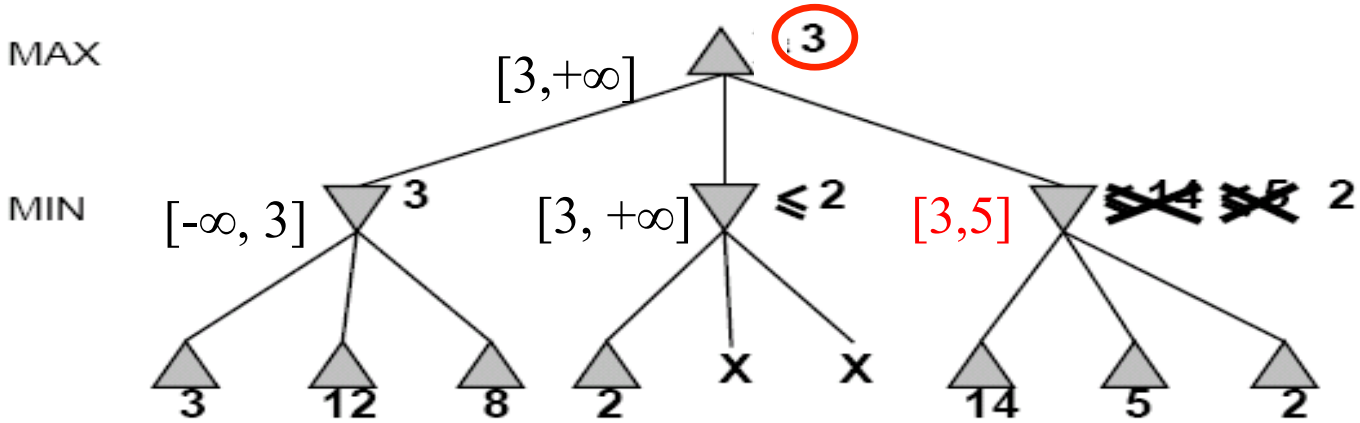
Alpha-beta pruning



Alpha-beta pruning



Alpha-beta pruning



Algorithme *élagage alpha-bêta*

Algorithme ÉLAGAGE-ALPHA-BÊTA(*noeudInitial*)

1. retourne l'action choisie par TOUR-MAX(*noeudInitial*,-∞, ∞)

Algorithme TOUR-MAX(*n*, α, β)

1. si *n* correspond à une fin de partie, alors retourner UTILITÉ(*n*)
2. *u* = -∞, *a* = void
3. pour chaque paire (*a'*,*n'*) donnée par TRANSITION(*n*)
4. si l'utilité de TOUR-MIN(*n'*, α, β) ≥ *u* alors affecter *a* = *a'*, *u* = utilité de TOUR-MIN(*n'*, α, β)
5. si *u* ≥ β alors retourne l'utilité *u* et l'action *a*
6. α = max(α, *u*)
4. retourne l'utilité *u* et l'action *a*

Algorithme TOUR-MIN(*n*, α, β)

1. si *n* correspond à une fin de partie, alors retourner UTILITÉ(*n*)
2. *u* = ∞, *a* = void
3. pour chaque paire (*a'*,*n'*) donnée par TRANSITION(*n*)
4. si l'utilité de TOUR-MAX(*n'*, α, β) < *u* alors affecter *a* = *a'*, *u* = utilité de TOUR-MAX(*n'*, α, β)
5. si *u* ≤ α alors retourne l'utilité *u* et l'action *a*
6. β = min (β, *u*)
4. retourne l'utilité *u* et l'action *a*

ce qui a changé...

Propriétés de *alpha-beta pruning*

- L'élagage n'**affecte pas le résultat final** de minimax

- Dans le pire des cas, *alpha-beta pruning* ne fait aucun élagage; il examine b^m nœuds terminaux comme l'algorithme minimax :
  - » b : le nombre maximum d'actions/coups légales à chaque étape
  - » m : nombre maximum de coup dans un jeu (profondeur maximale de l'arbre)

- Un bon ordonnancement des actions à chaque nœud améliore l'efficacité
  - dans le meilleur des cas (ordonnancement parfait), la complexité en temps est de $O(b^{m/2})$
    - » si le temps de réflexion est limité, la recherche peut être jusqu'à deux fois plus profonde comparé à minimax!
  - dans le cas moyen d'un ordonnancement aléatoire : $O(b^{3m/4})$