Introduction à l'Intelligence Artificielle

Cours 3 – mercredi 12 octobre 2022

Adrien Revault d'Allonnes

ara@up8.edu

Université Paris 8 – Vincennes à Saint-Denis

IIA – sept. à déc., 2022

Les jeux à deux joueurs

- Jeux où deux joueurs s'affrontent
  - chaque joueur joue de son mieux pour battre son adversaire
  - les joueurs jouent chacun leur tour
  - e.g. morpion, échecs, othello, dames, go...
- Comment programmer une IA capable de jouer de la sorte ?
  - raisonner en fonction (et dans) un univers changeant
  - raisonner en tenant compte de son adversaire
- Idée
  - construire l'arborescence des coups réalisables
    - combinatoire : limitation à une certaine profondeur
  - évaluer chaque position en fin de branche
    - utilisation d'un moyen pour évaluer une position
  - choisir le chemin permettant d'atteindre la meilleure position finale
    - considérer les coups menant à la position souhaitée

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$
- Exemple de complexité spatiale d'arbre de jeu :
  le nombre de Shannon

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$
- Exemple de complexité spatiale d'arbre de jeu :
  le nombre de Shannon
  - moyenne de 40 coups par partie

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$
- Exemple de complexité spatiale d'arbre de jeu :
  le nombre de Shannon
  - moyenne de 40 coups par partie
  - à chaque demi-coup un joueur à, en moyenne 30 choix possibles

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$
- Exemple de complexité spatiale d'arbre de jeu :
  le nombre de Shannon
  - moyenne de 40 coups par partie
  - à chaque demi-coup un joueur à, en moyenne 30 choix possibles

$\Longrightarrow$ nombre de parties possibles : $(30 \times 30)^{40} \approx 10^{120}$

Complexité spatiale de jeux à deux joueurs

- E.g. le notaire et l'héritier : c'est combien, un milliard, en secondes ?
  - $10^1 s = 10s$
  - $10^2 s = 1min, 40s$
  - $10^3 s = 16min, 40s$
  - $10^4 s = 2h, 46min, 40s$
  - $10^5 s = 1jour, 3h, 46min, 40s$
  - $10^6 s = 11jours, 13h, 46min, 40s$
  - $10^9 s = 31ans, 8mois, 7jours, 7h, 46min, 40s$
- Exemple de complexité spatiale d'arbre de jeu :
  le nombre de Shannon
  - moyenne de 40 coups par partie
  - à chaque demi-coup un joueur à, en moyenne 30 choix possibles

  $\Longrightarrow$ nombre de parties possibles : $(30 \times 30)^{40} \approx 10^{120}$
  - équivalent pour le go : 10^{600}

Jeux à 2 joueurs : principe de résolution

- Trouver la suite de coups de la position initiale à un état final
  - étude de l'arborescence de coups : **arbre de jeu**
- Examen d'un **arbre de recherche**
  - sous-ensemble de l'arbre de jeu
- Estimation d'une position p
  - soit $\mathcal{P}$ l'ensemble des positions légales
  - soit $\mathcal{P}^*$ l'ensemble des positions légales et terminales ($\mathcal{P}^* \subset \mathcal{P}$)
  - idéalement
    - **fonction de décision** : $h^* : \mathcal{P}^* \longrightarrow \{-\infty, 0, +\infty\}$
    - $h^*(p)$ dit quel joueur gagne dans la position p ou si la partie est nulle
    - $h^*(p)$ ne s'applique qu'à des positions terminales
  - on a besoin d'une **estimation** de h^*
    - **fonction d'évaluation** : $h : \mathcal{P} \longrightarrow \mathbb{R}$
  - h est d'autant plus fiable que p est proche d'une position terminale
    - **si p est terminale, alors $h(p) \equiv h^*(p)$**

Exemple de fonction d'évaluation

- **Jeu d'échecs** : évaluation de la position p
  - h : fonction qui évalue les chances de gain pour un camp
  - h est une **estimation** de la « véritable » fonction h^*
- $h(p) = \alpha_1 S(p) + \alpha_2 R(p) + \alpha_3 M(p) + \alpha_4 C(p) + \alpha_5 P(p) + \alpha_6 A(p)$
- Où
  - $S(p)$: valeur des pièces
  - $R(p)$: sûreté des rois
  - $M(p)$: mobilité des pièces
  - $C(p)$: contrôle des cases centrales
  - $P(p)$: structure des pions
  - $A(p)$: possibilités d'attaque

- chacun des critères est évalué du point de vue des blancs $(+)$ et du point de vue des noirs $(-)$
  - par exemple : $S(p) = S_{\text{blancs}}(p) - S_{\text{noirs}}(p)$

Stratégie pour évaluer une position p

- Utiliser seulement h : problème, car h n'est pas h^*
  - h est une **heuristique**
- Idée : contruire un arbre de recherche, à partir de p
  - suite de n coups : $p = p_0 \longrightarrow p_1 \longrightarrow \ldots \longrightarrow p_n$
  - évaluer $h(p_n)$
    - p_n **plus proche** d'une position finale que p
    - $h(p_n)$ plus fiable que $h(p)$
  - « remonter » l'évaluation de p_n pour évaluer p
- Hypothèses
  - les deux adversaires utilisent la **même fonction d'évaluation**
  - les deux joueurs jouent pour gagner
  - le joueur qui a les blancs cherche à **maximiser** son évaluation
  - le joueur qui a les noirs cherche à **minimiser** son évaluation

Algorithme minimax pour évaluer une position p

- Calcul de $f(p)$: évaluation de la position p à une profondeur n
  - on connaît la fonction de décision : $h^* : \mathcal{P}^* \longrightarrow \{-\infty, 0, +\infty\}$
  - on considère donnée une fonction d'évaluation : $h : \mathcal{P} \longrightarrow \mathbb{R}$
  - deux joueurs : MAX et MIN
- **minimax**(n, p, j) : évaluation de p à une profondeur n (joueur j)
  - si p est terminale : $f(p) \longleftarrow h^*(p)$
  - sinon, si $n = 0$: $f(p) \longleftarrow h(p)$
  - sinon (i.e. si $n > 0$)
    - soit $p_1, ..., p_m$ les m positions **accessibles en un coup** depuis p
    - si $j =$ MAX : $f(p) \longleftarrow \max_{i=1,\ldots,m}(\text{minimax}(n-1, p_i, \text{MIN}))$
    - si $j =$ MIN : $f(p) \longleftarrow \min_{i=1,\ldots,m}(\text{minimax}(n-1, p_i, \text{MAX}))$
  - retourner $f(p)$

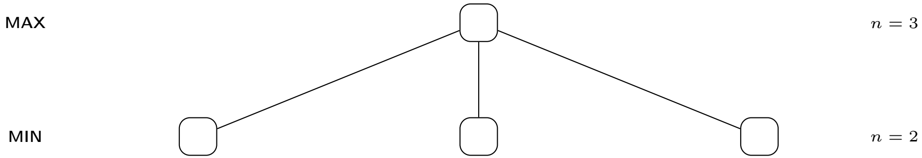
Exemple d'application

$\mathsf{minimax}(3, p, \mathsf{MAX})$

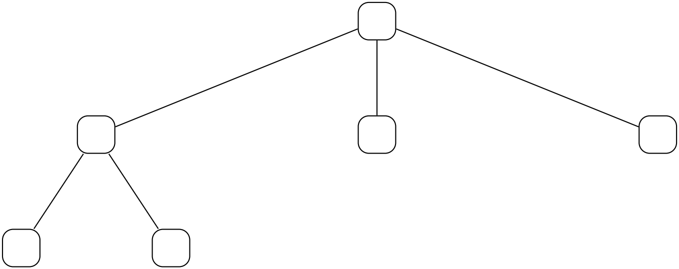
Exemple d'application

MAX  $n = 3$

Exemple d'application



Exemple d'application

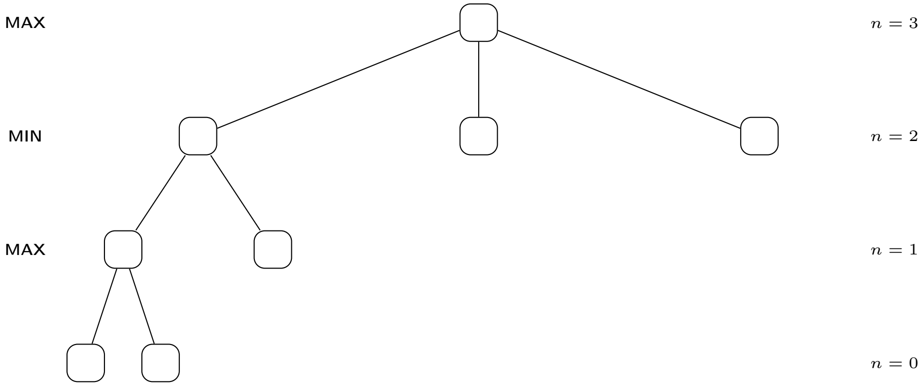


MAX $n = 3$

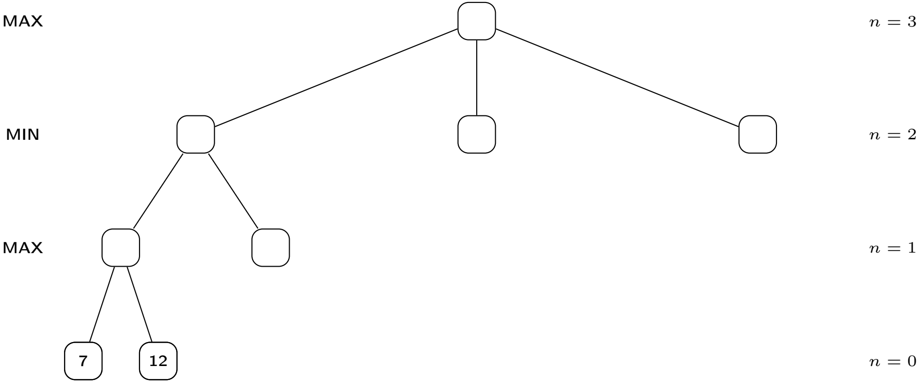
MIN $n = 2$

MAX $n = 1$

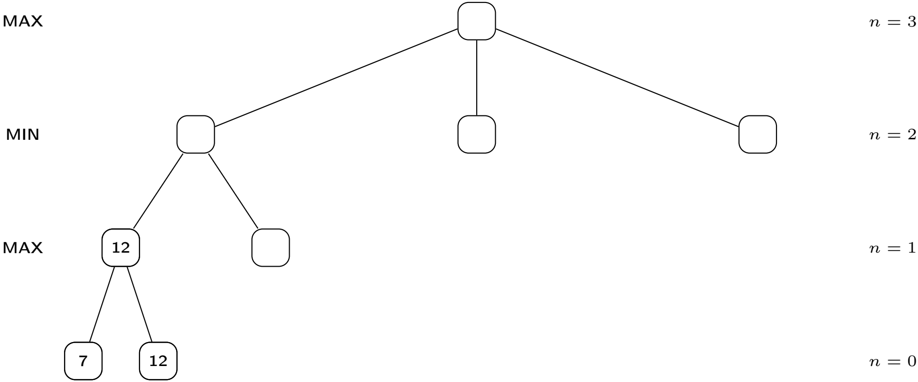
Exemple d'application



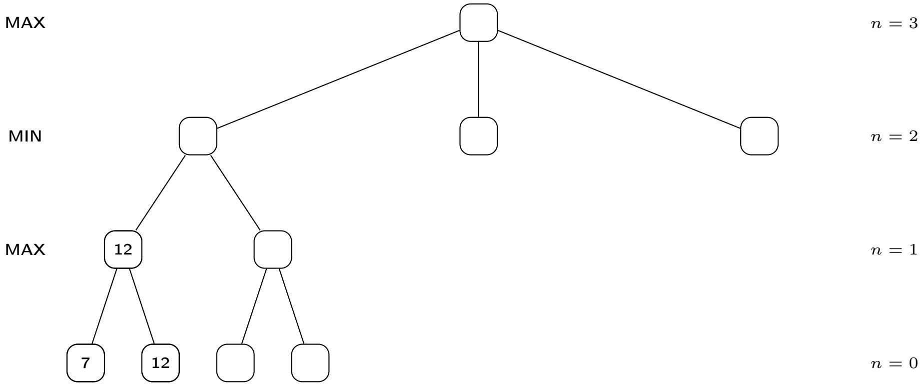
Exemple d'application



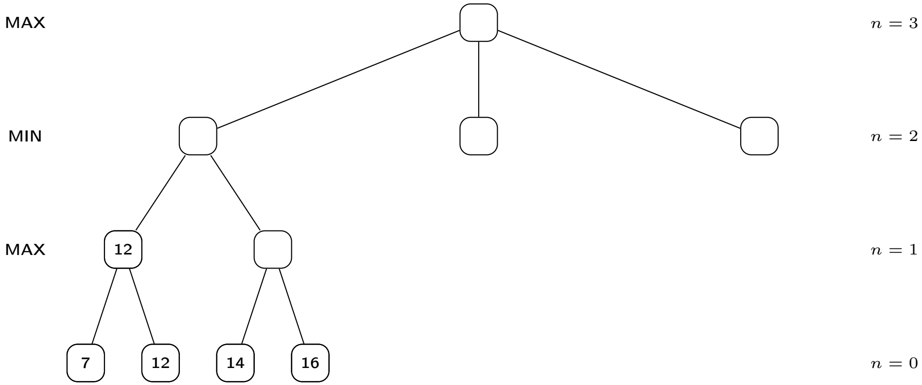
Exemple d'application



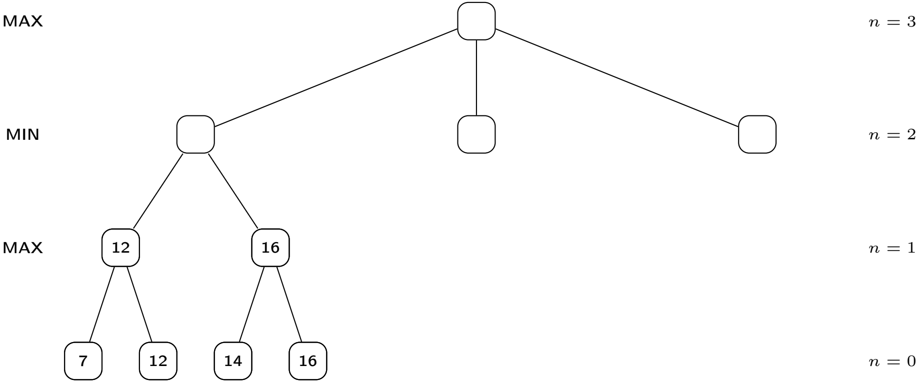
Exemple d'application



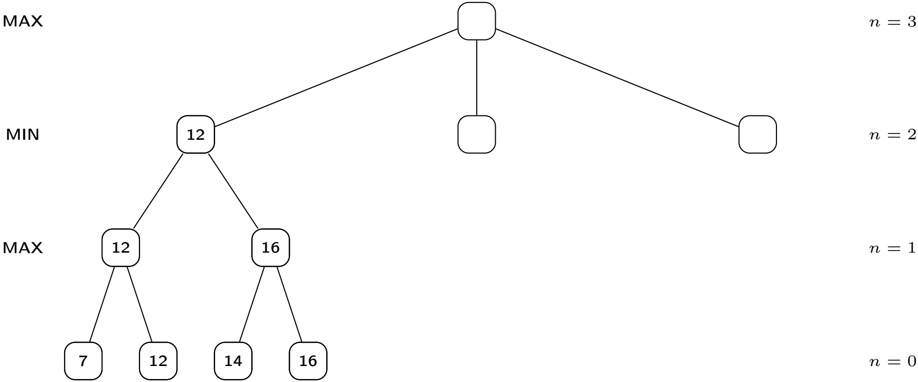
Exemple d'application



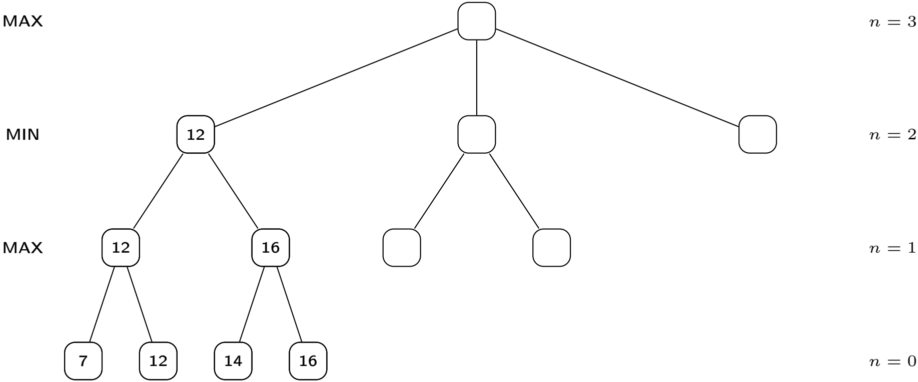
Exemple d'application



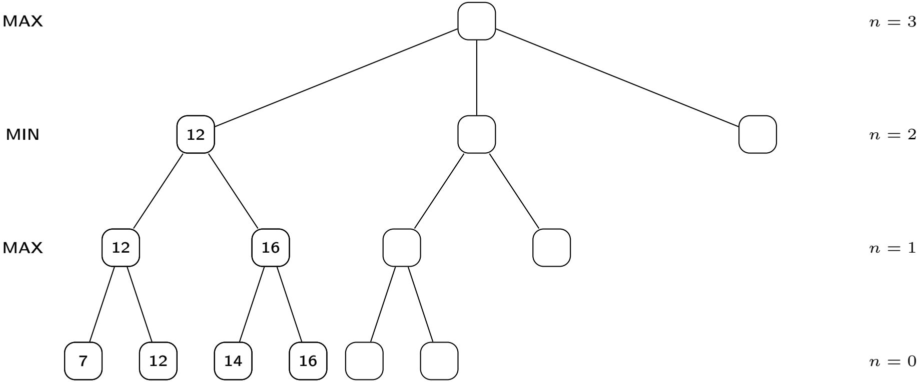
Exemple d'application



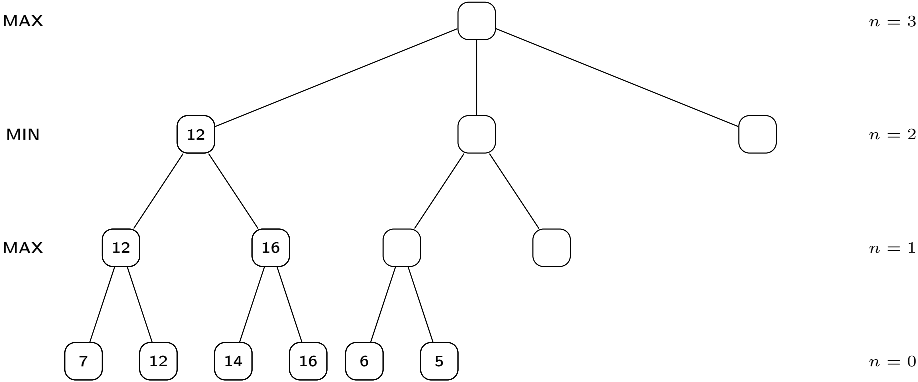
Exemple d'application



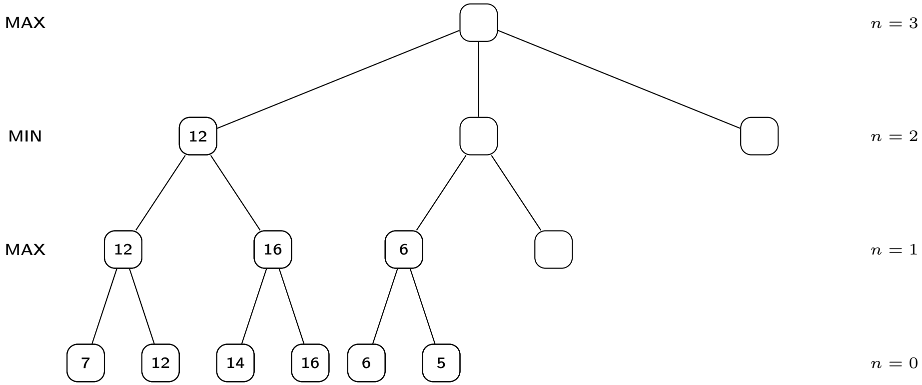
Exemple d'application



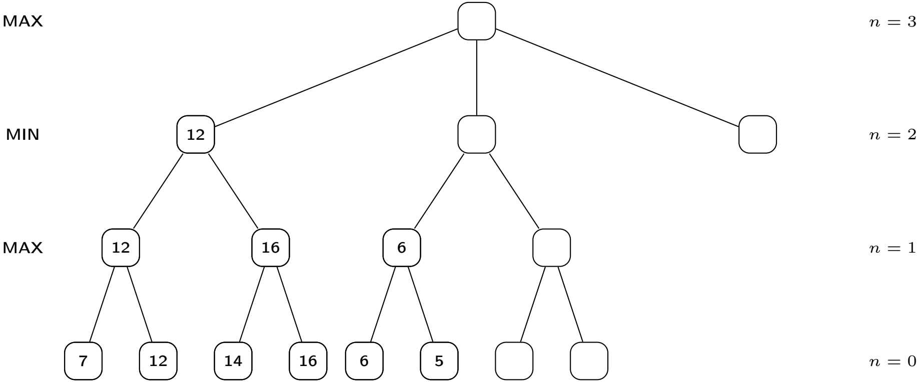
Exemple d'application



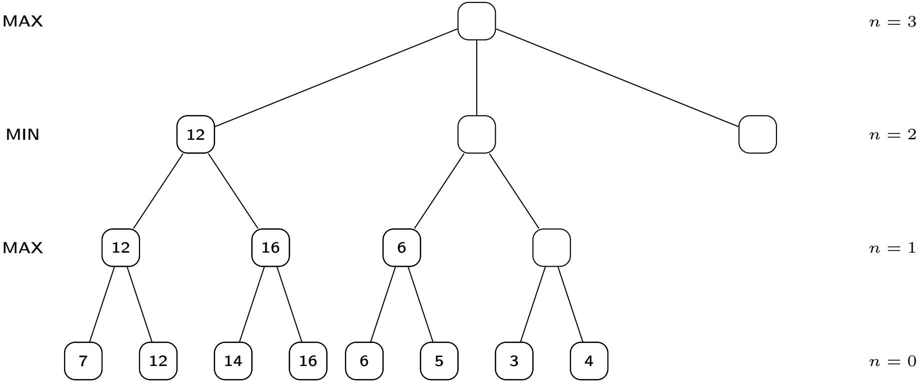
Exemple d'application



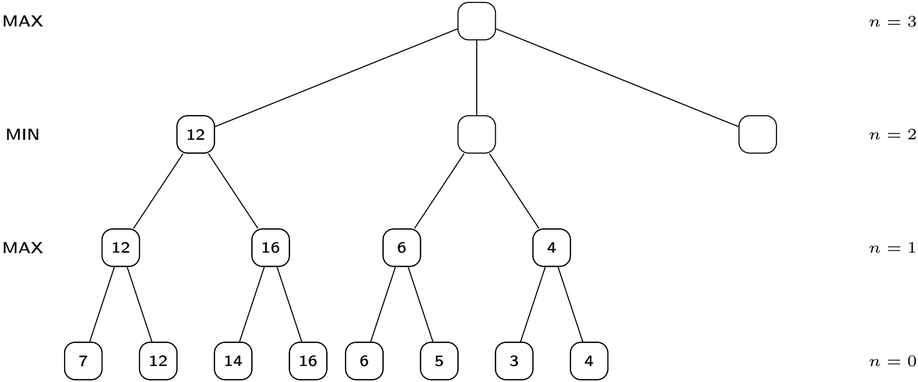
Exemple d'application



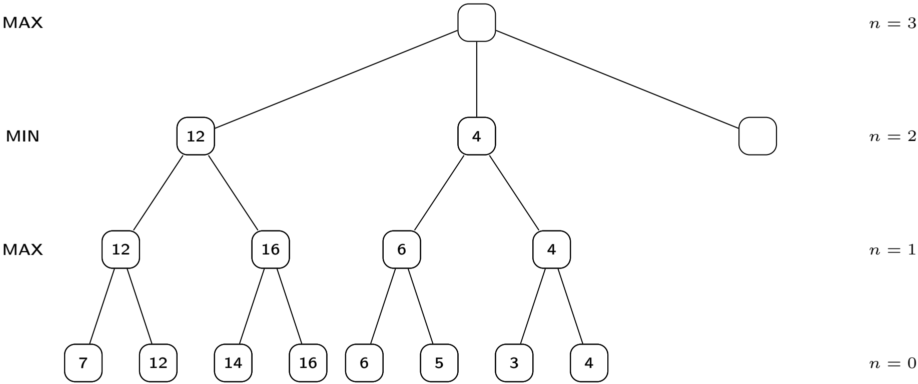
Exemple d'application



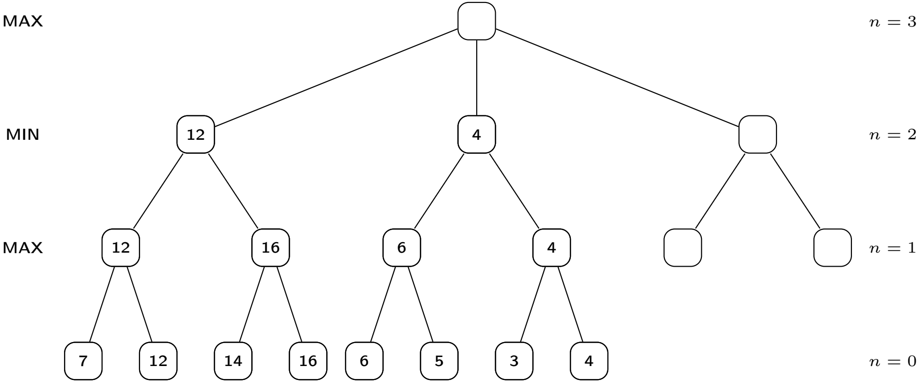
Exemple d’application



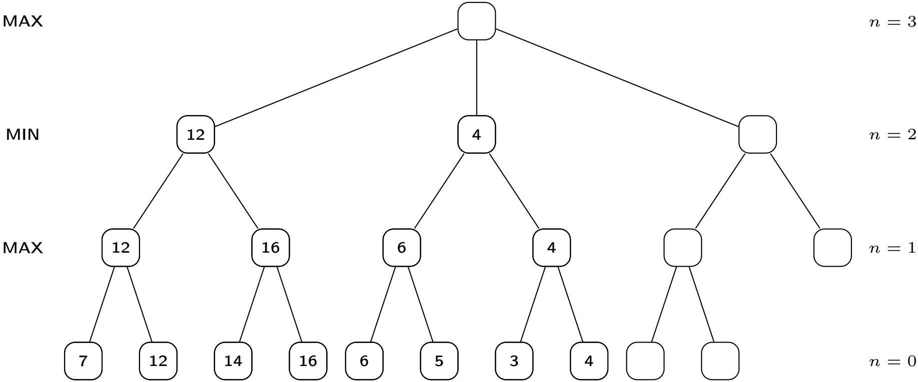
Exemple d'application



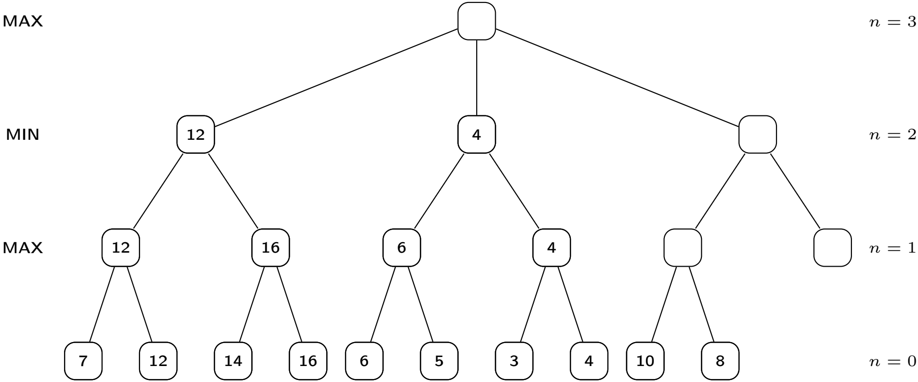
Exemple d'application



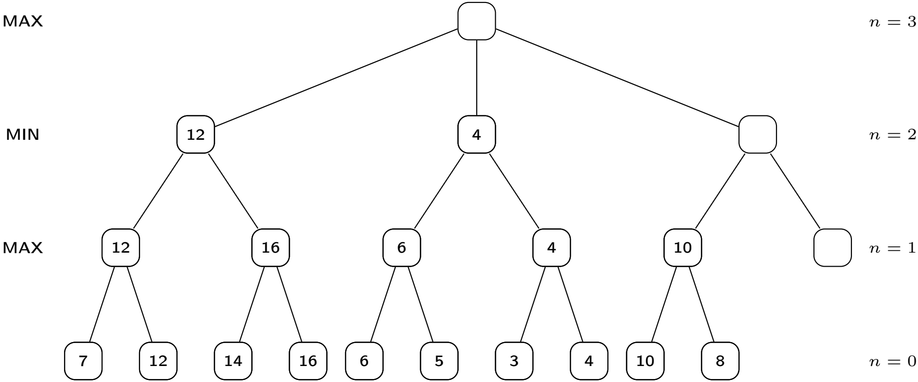
Exemple d'application



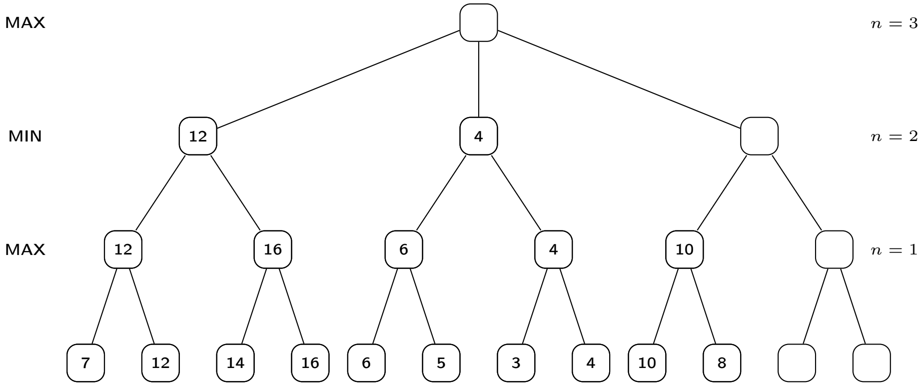
Exemple d'application



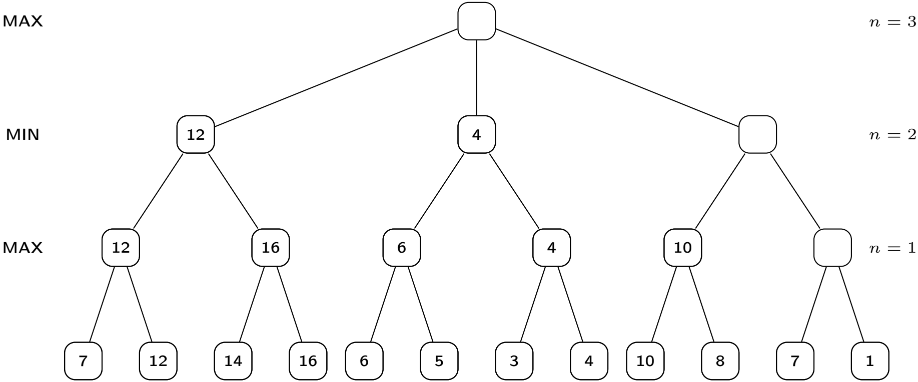
Exemple d'application



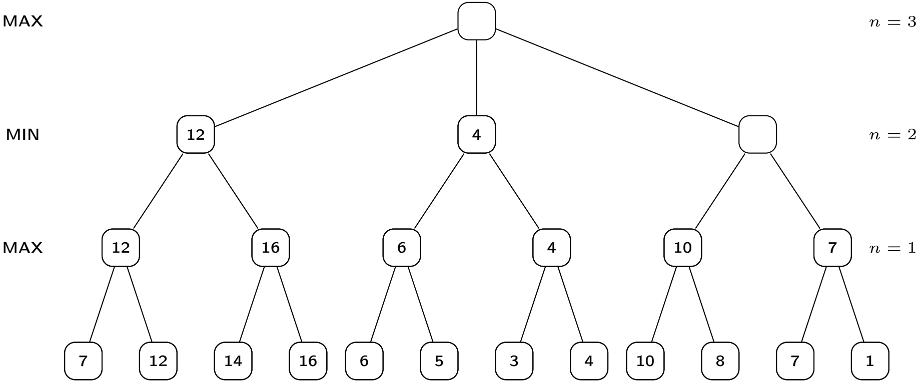
Exemple d'application



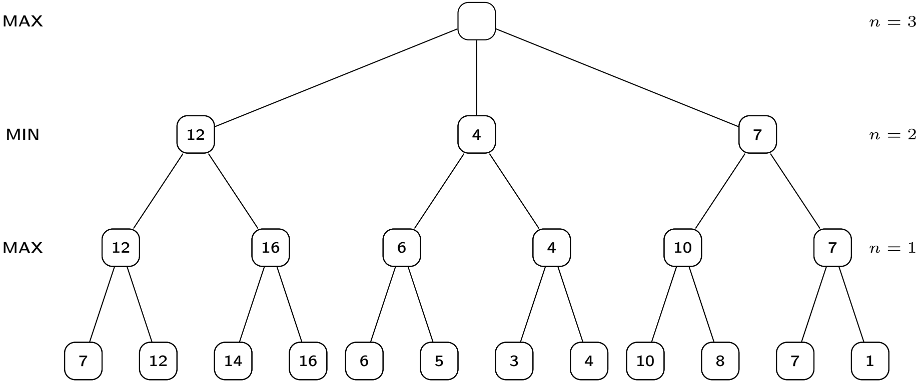
Exemple d'application



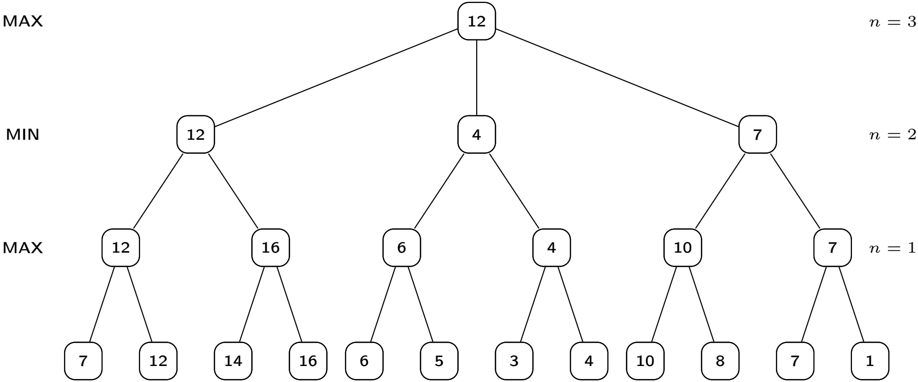
Exemple d'application



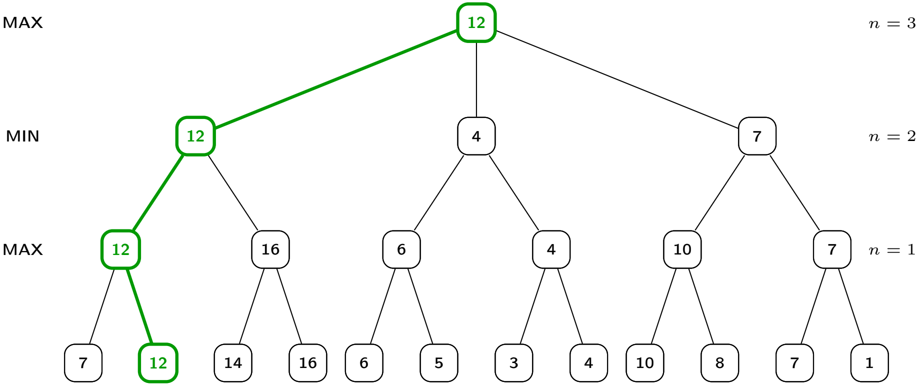
Exemple d'application



Exemple d'application



Exemple d'application



Algorithme minimax : quelques limites

- Complexité élevée
  avec c coups légaux pour chaque position et à profondeur n :
  - temps : $O(c^n)$
  - espace : $O(c \times n)$
- Plus grave : **effet d'horizon**
  - pour minimax(n, p, j) : il se peut que p_{n+1} soit perdante, alors que p_n est favorable
    - exemple : ne pas s'arrêter pendant une suite de prises

Exemple d'effet d'horizon

$f(p_0) = 0$

Exemple d'effet d'horizon

$f(p_0) = 0$

$f(p_1) = -5$

Exemple d'effet d'horizon

$f(p_0) = 0$

$f(p_1) = -5$

$f(p_2) = 0$

Exemple d'effet d'horizon

$f(p_0) = 0$

$f(p_1) = -5$

$f(p_2) = 0$

$f(p_3) = 0$

Exemple d'effet d'horizon

$f(p_0) = 0$

$f(p_1) = -5$

$f(p_2) = 0$

$f(p_3) = 0$

Évaluation de p_0

- pour p_3 on a 0
- **MAIS**

Exemple d'effet d'horizon

$f(p_0) = 0$

$f(p_1) = -5$

$f(p_2) = 0$

$f(p_3) = 0$

$f(p_4) = +\infty$

Évaluation de p_0

- pour p_3 on a 0
- **MAIS**
- pour p_4 on a $+\infty$

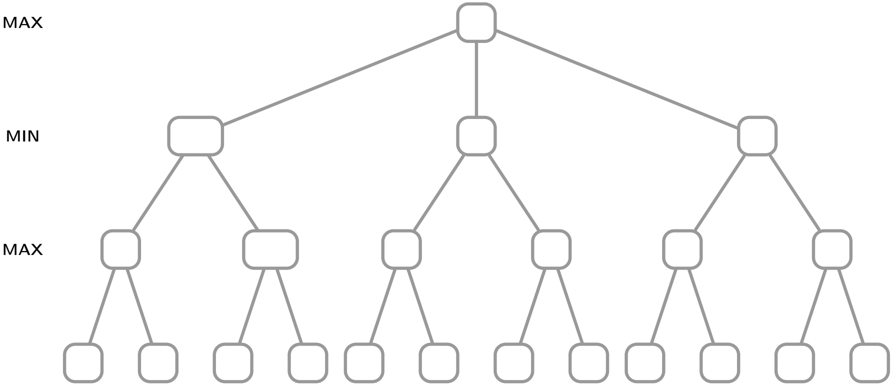
Algorithme minimax : quelques limites

- Coût : complexité éleveée
  - $o(c^n)$: avec c coups légaux pour chaque position et à profondeur n
- Plus grave : **effet d'horizon**
  - pour minimax(n, p, j) : il se peut que p_{n+1} soit perdante, alors que p_n est favorable
    - exemple : ne pas s'arrêter pendant une suite de prises
  - impossible à éliminer dans un arbre de recherche
- Une amélioration
  - éliminer les évaluations de positions superflues
  - **élagage de l'arbre de recherche**

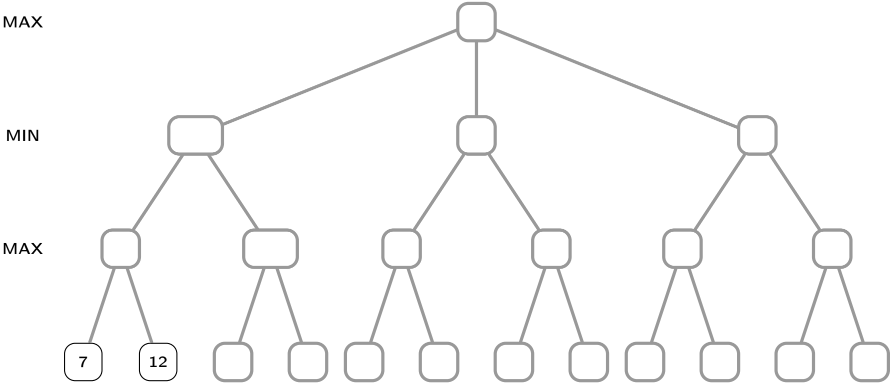
$\Longrightarrow$ algorithme **alpha-bêta** $(\alpha - \beta)$

  - cela n'enlève pas l'effet d'horizon mais permet d'augmenter la profondeur de recherche pour en limiter les méfaits

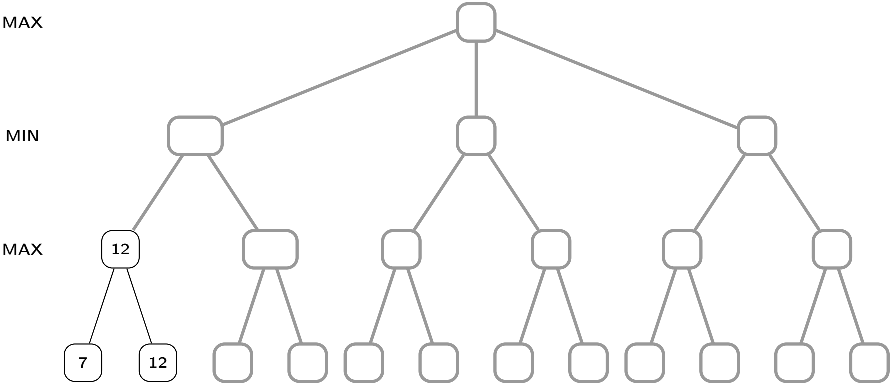
Exemple de calculs superflus



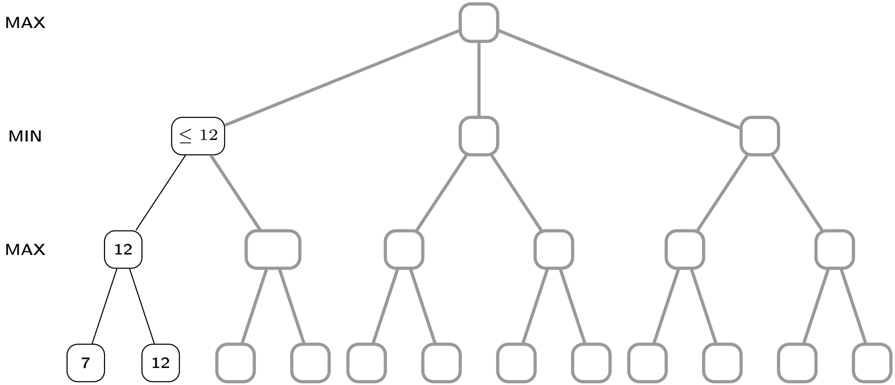
Exemple de calculs superflus



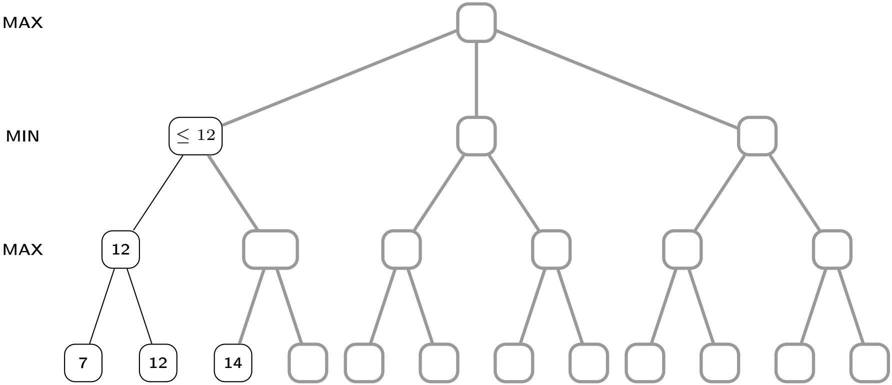
Exemple de calculs superflus



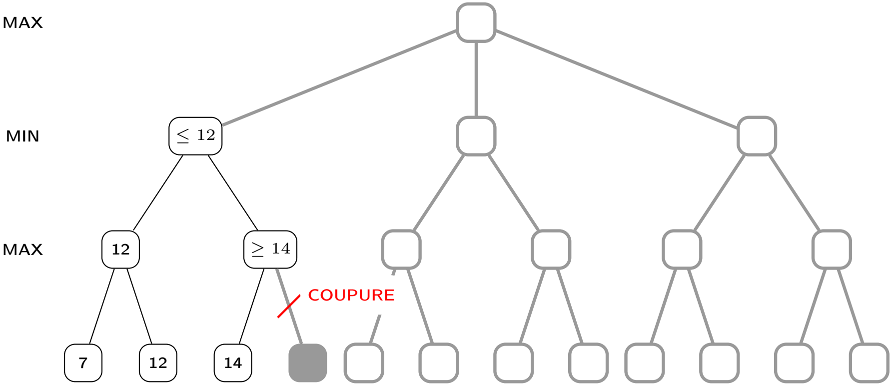
Exemple de calculs superflus



Exemple de calculs superflus



Exemple de calculs superflus



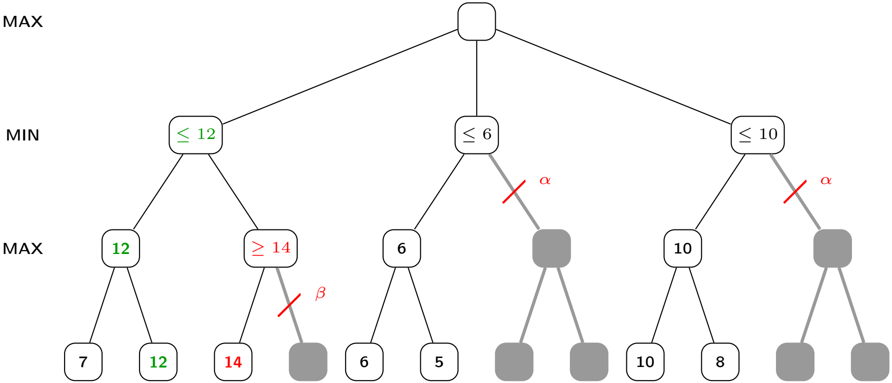
Algorithme alpha-bêta

- Calcul de $f(p)$: évaluation de la position p à une profondeur n
  - on connaît la fonction de décision : $h^* : \mathcal{P}^* \longrightarrow \{-\infty, 0, +\infty\}$
  - on considère donnée une fonction d'évaluation : $h : \mathcal{P} \longrightarrow \mathbb{R}$
  - deux joueurs : MAX et MIN
- Pour évaluer p à une profondeur n, pour j, on conserve
  - α : meilleure évaluation trouvée pour le joueur MAX dans les branches déjà vues
  - β : meilleure évaluation trouvée pour le joueur MIN dans les branches déjà vues
  - au départ : $\alpha = -\infty$ et $\beta = +\infty$

Algorithme alpha-bêta

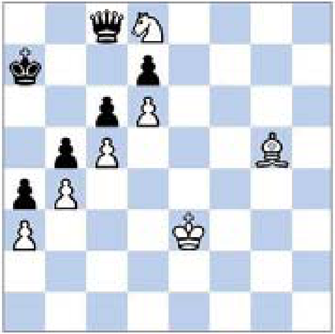
- **alphabeta**(n, p, j, α, β)
  - si p est terminale : $f(p) \longleftarrow h^*(p)$
  - sinon, si $n = 0$: $f(p) \longleftarrow h(p)$
  - sinon (i.e. si $n > 0$)
    - soit $p_1, ..., p_m$ les m positions **accessibles en un coup** depuis p
    - si $j =$ MIN
      - $e \longleftarrow$ alphabeta$(n-1, p_1,$ MAX$, \alpha, \beta)$
      - si $e \leq \alpha$: rendre $\alpha \Longrightarrow$ **coupe** α
      - sinon $\beta \longleftarrow \min(\beta, e)$ puis, recalculer e pour $p_2 \ldots$
    - si $j =$ MAX
      - $e \longleftarrow$ alphabeta$(n-1, p_1,$ MIN$, \alpha, \beta)$
      - si $e \geq \beta$: rendre $\beta \Longrightarrow$ **coupe** β
      - sinon $\alpha \longleftarrow \max(\alpha, e)$ puis, recalculer e pour $p_2 \ldots$
  - retourner $f(p)$

Exemple d'application



La résolution des jeux

- Exploration systématique d'un arbre de coups
- Toute la connaissance est dans la fonction d'évaluation
- Connaissance locale à une position
  - manque possible de vision à long terme
- Absence totale de stratégie
  - exemple (P. Nolot) : position difficile à traîter pour une IA



Les Blancs jouent et gagnent

Quelques améliorations possibles

- S'inspirer des humains
- Améliorer la sélection des coups examinés
  - utilisation d'**oracles** pour planifier les coups
  - utiliser des tactiques typiques du jeu
- Doter le programme de capacités d'apprentissage
  - éviter de reproduire ses erreurs
  - apprendre à s'améliorer