Intelligence Artificielle (IA)

Les jeux, recherche avec horizon (I)

Akka Zemmari

LaBRI, Université de Bordeaux

2021 - 2022

Introduction d'*heuristiques*

Idée

On ne développe l'arbre de recherche que jusqu'à une certaine profondeur maximale p. Les feuilles de l'arbre ne sont plus forçément des positions finales du jeu.
On ne peut donc plus se contenter de l'information « gagné » ou « perdu ». Il faut **évaluer** les positions.

Introduction d'*heuristiques*

Idée

On ne développe l'arbre de recherche que jusqu'à une certaine profondeur maximale p. Les feuilles de l'arbre ne sont plus forçément des positions finales du jeu.
On ne peut donc plus se contenter de l'information « gagné » ou « perdu ». Il faut **évaluer** les positions.

Définition intuitive

Les heuristiques sont des fonctions statiques associant un réel aux plateaux de jeu. Dans les jeux avec adversaire, plus l'heuristique est grande et positive, plus on est proche de la victoire. Plus elle est négative et petite, plus on est proche de la défaite.

Exemple des dominos 3×3

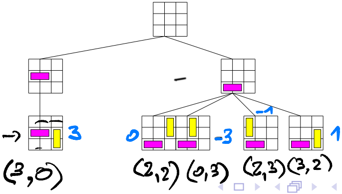
Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*

Exemple des dominos 3×3

Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*



Exemple d'heuristiques : les Échecs dès 1950

(1) The relative values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, 1, respectively. Thus other things being equal (!) if we add the numbers of pieces for the two sides with these coefficients, the side with the largest total has the better position.

(2) Rooks should be placed on open files. This is part of a more general principle that the side with the greater mobility, other things equal, has the better game.

(3) Backward, isolated and doubled pawns are weak.

(4) An exposed king is a weakness (until the end game).

These and similar principles are only generalizations from empirical evidence of numerous games, and only have a kind of statistical validity. Probably any chess principle can be contradicted by particular counter examples. However, form these principles one can construct a crude evaluation function. The following is an example: -

$$
\begin{aligned}
f(P) = {} & 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - \\
& 0.5(D-D'+S-S'+I-I') + \\
& 0.1(M-M') + \ldots
\end{aligned}
$$

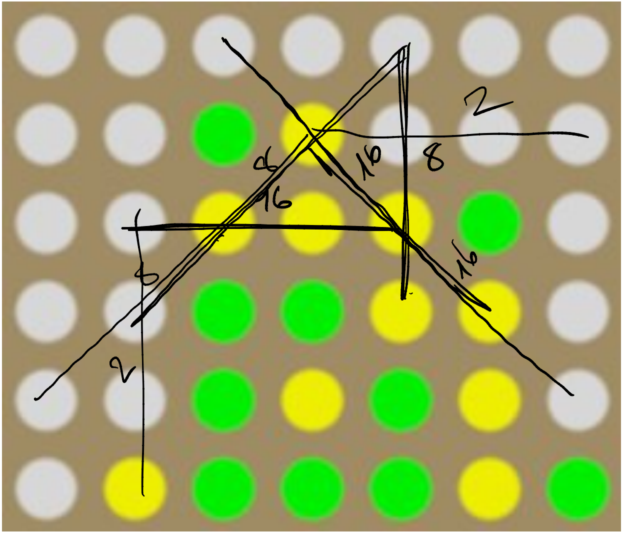
in which: -

(1) K,Q,R,B,B,P are the number of White kings, queens, rooks, bishops, knights and pawns on the board.

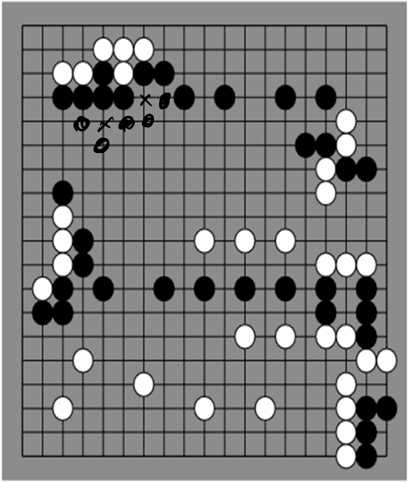
(2) D,S,I are doubled, backward and isolated White pawns.

(3) M= White mobility (measured, say, as the number of legal moves available to White).

Exemple d'heuristiques : Puissance 4



Exemple d'heuristiques : Go



Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

Supposition forte

On suppose que les deux joueurs jouent « bien » s'ils suivent les indications de l'heuristique.

Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

Supposition forte

On suppose que les deux joueurs jouent « bien » s'ils suivent les indications de l'heuristique.

En pratique, il faudra prendre en compte cela pour des raisons de performances. À chaque feuille de l'arbre, on devra calculer sa valeur heuristique

Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

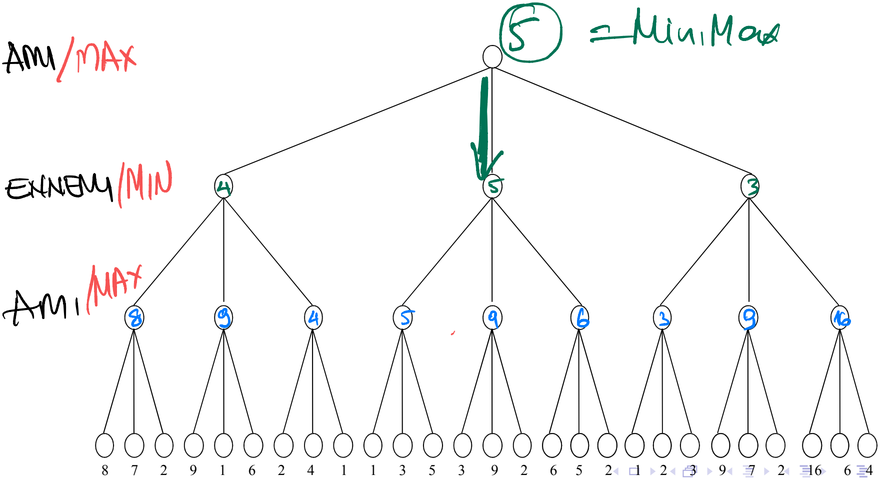
Supposition forte

On suppose que les deux joueurs jouent « bien » s'ils suivent les indications de l'heuristique.

En pratique, il faudra prendre en compte cela pour des raisons de performances. À chaque feuille de l'arbre, on devra calculer sa valeur heuristique

Problème : Comment faire remonter la meilleur feuille à la racine ?

Exemple d'arbre de jeu



Algorithme MiniMax – Fonction *MaxMin*

Idée

On appelle *MaxMin* pour évaluer la valeur MiniMax de la racine.
On doit aussi remonter*le meilleur coups à jouer* qui lui est associé.

```
Fonction MaxMin(etat)                          ▷ Évaluation niveau AMI
    etat : Plateau de jeu courant
    Si EstFeuille(etat) Alors
        Retourner evalue(AMI, etat)            ▷ Évaluation heuristique
    Fin Si
    Meilleur ← −∞
    Pour Tout successeur s de etat Faire
        Meilleur ← max(Meilleur, MinMax(s))
    Fin Pour
    Retourner Meilleur
Fin Fonction
```

Algorithme MiniMax – Fonction *MinMax*

Attention

Si l'évaluation de la position a lieu sur un niveau impair (c'est à *ENNEMI* de jouer), la fonction heuristique doit en tenir compte !

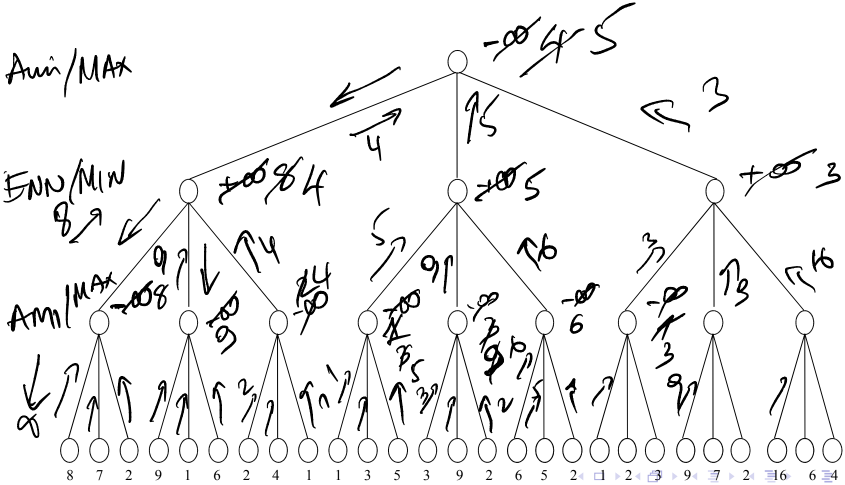
Fonction *MinMax*(*etat*) ▷ Évaluation niveau ENNEMI
    etat : Plateau de jeu courant
    Si *EstFeuille*(*etat*) **Alors**
        Retourner *evalue*(*ENNEMI*, *etat*) ▷ Évaluation heuristique
    Fin Si
    $Pire \leftarrow +\infty$
    Pour Tout successeur *s* de *etat* **Faire**
        $Pire \leftarrow min(Pire, MaxMin(s))$
    Fin Pour
    Retourner *Pire*
Fin Fonction

Les deux fonctions MiniMax

```
Fonction MaxMin(etat)                                  ▷ Évaluation niveau AMI
    etat : Plateau de jeu courant
    Si EstFeuille(etat) Alors
        Retourner evalue(AMI, etat)                    ▷ Évaluation heuristique
    Fin Si
    Meilleur ← −∞
    Pour Tout successeur s de etat Faire
        Meilleur ← max(Meilleur, MinMax(s))
    Fin Pour
    Retourner Meilleur
Fin Fonction

Fonction MinMax(etat)                                  ▷ Évaluation niveau ENNEMI
    etat : Plateau de jeu courant
    Si EstFeuille(etat) Alors
        Retourner evalue(ENNEMI, etat)                 ▷ Évaluation heuristique
    Fin Si
    Pire ← +∞
    Pour Tout successeur s de etat Faire
        Pire ← min(Pire, MaxMin(s))
    Fin Pour
    Retourner Pire
Fin Fonction
```

Exemple d'arbre de jeu



Limitations des principes MiniMax

- Que se passe-t-il si un joueur

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Recherche dans les arbres de jeux

L'espace des états atteignables peut être gigantesque. Il dépend :

- Facteur de branchement b du jeu
- Nombre de coups n d'une partie

Idée des tailles de certains jeux

- Échecs : $p \sim 35$ et $n \sim 30$ coups au minimum
  $|\text{graphe d'états}| = 35^{30}$
  Soit 2099139642966190174953146230280399322509765625
- Othello : $5 \leq b \leq 15$
- Go : $b \sim 360$

Il faut retrouver l'idée d'élagage de l'arbre de recherche.

$\alpha\beta$

Élagage efficace de l'arbre

Élagage admissible

On veut trouver la même valeur d'évaluation finale du noeud racine sans développer tout l'arbre. Il faut donc élaguer des parties de l'arbre de recherche qui sont sans conséquence sur l'évaluation d'un noeud.

$\alpha\beta$

Élagage efficace de l'arbre

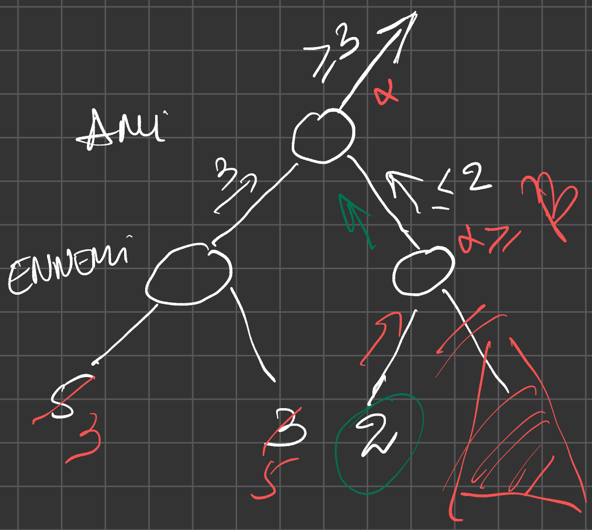
Élagage admissible

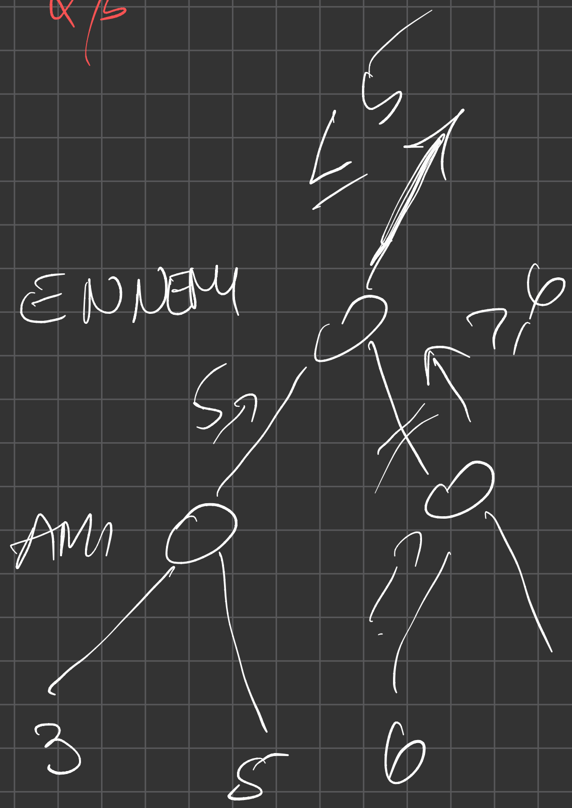
On veut trouver la même valeur d'évaluation finale du noeud racine sans développer tout l'arbre. Il faut donc élaguer des parties de l'arbre de recherche qui sont sans conséquence sur l'évaluation d'un noeud.

Intuitivement

Soit un noeud *n* dans l'arbre de recherche, tel que *Joueur* peut joueur en *n*.
S'il existe pour *Joueur* un choix *m* meilleur que *n* (soit à partir du noeud parent de *n*, soit plus haut dans l'arbre), *n* ne sera jamais effectivement joué.

$\alpha\ \beta$





$\alpha\beta$

Elagage efficace de l'arbre II

Deux types de coupes : α et β

- α : le meilleur choix à un instant donné pour Max sur le chemin développé. **La valeur α est croissante**
- β : le meilleur choix à un instant donné pour Min sur le chemin développé. **La valeur β est décroissante**

Les coupes auront lieu dès que α est supérieur à β

Algorithme $\alpha\beta$

La première fonction : *MaxValue*

Fonction $MaxValue(etat, \alpha, \beta)$ ▷ Évaluation niveau AMI
    $etat$: Plateau de jeu courant
    α : Meilleur évaluation courante pour AMI
    β : Meilleur évaluation courante pour ENNEMI
    Si $EstFeuille(etat)$ **Alors**
        Retourner $evalue(etat)$ ▷ Évaluation heuristique
    Fin Si
    Pour Tout successeur s de $etat$ **Faire**
        $\alpha \leftarrow max(\alpha, MinValue(s, \alpha, \beta))$
        Si $\alpha \geq \beta$ **Alors** ▷ Coupe β
            Retourner β
        Fin Si
    Fin Pour
    Retourner α
Fin Fonction

Algorithme $\alpha\beta$

La suite : *MinValue*

Remarque. - Pour évaluer un plateau, on appelle *MaxValue* avec : plateau à évaluer, $\alpha = -\infty$ et $\beta = +\infty$. Les variables α et β sont bien *locales*, elles sont changées par l'intermédiaire des valeurs de retour.

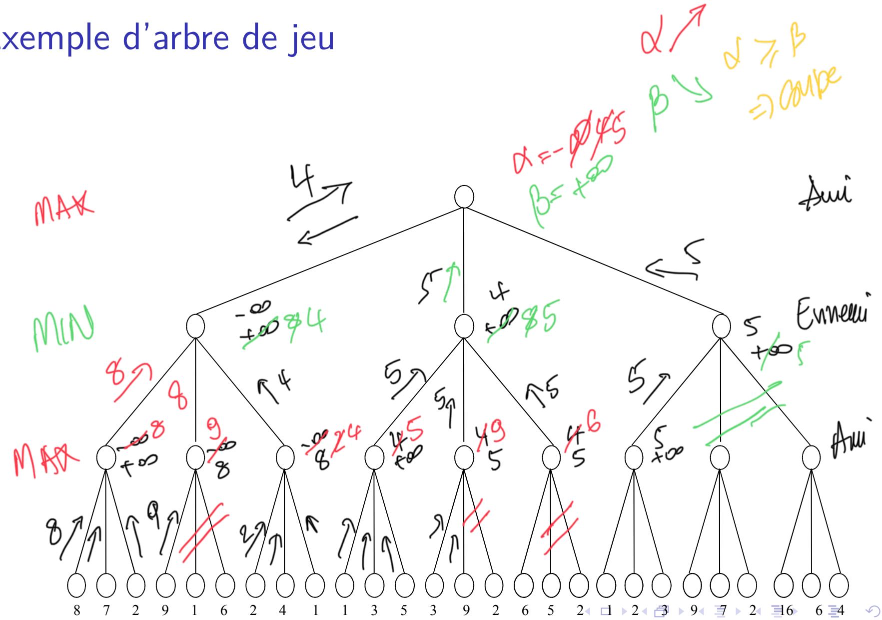
Fonction *MinValue*(*etat*, α, β)) ▷ Évaluation niveau ENNEMI
    etat : Plateau de jeu courant
    α : Meilleur évaluation courante pour AMI
    β : Meilleur évaluation courante pour ENNEMI
    Si *EstFeuille*(*etat*) **Alors**
        Retourner *evalue*(*etat*) ▷ Évaluation heuristique
    Fin Si
    Pour Tout successeur *s* de *etat* **Faire**
        $\beta \leftarrow$ *min*(β, *MaxValue*(*s*, α, β))
        Si $\alpha \geq \beta$ **Alors** ▷ Coupe α
            Retourner α
        Fin Si
    Fin Pour
    Retourner β
Fin Fonction

Algorithme $\alpha\beta$, Les deux fonctions ensembles

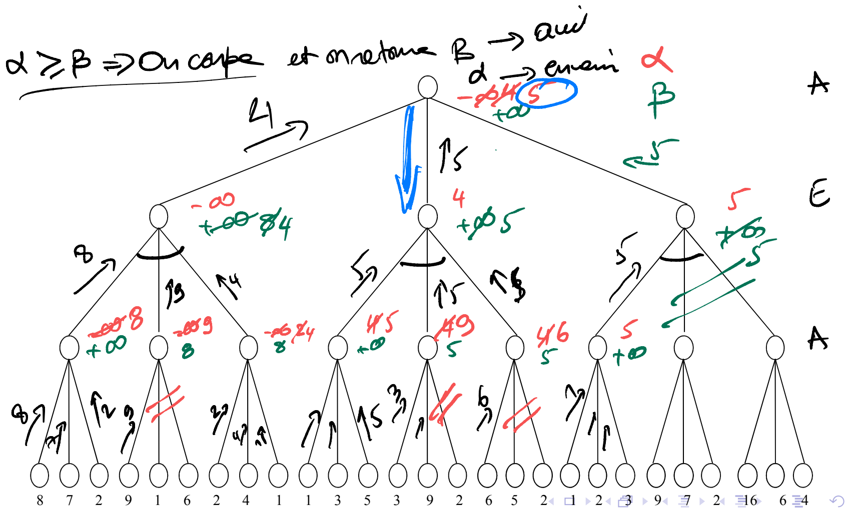
```
Fonction MaxValue(etat, α, β))                              ▷ Niveau AMI
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                              ▷ Évaluation heuristique
    Fin Si
    Pour Tout successeur s de etat Faire
        α ← max(α, MinValue(s, α, β))
        Si α ≥ β Alors                                      ▷ Coupe β
            Retourner β
        Fin Si
    Fin Pour
    Retourner α
Fin Fonction

Fonction MinValue(etat, α, β))
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                              ▷ Niveau ENNEMI
    Fin Si
    Pour Tout successeur s de etat Faire
        β ← min(β, MaxValue(s, α, β))
        Si α ≥ β Alors                                      ▷ Coupe α
            Retourner α
        Fin Si
    Fin Pour
    Retourner β
Fin Fonction
```

Exemple d'arbre de jeu



Exemple d'arbre de jeu



Propriétés de $\alpha\beta$

Efficacité théorique

Si on suppose que les fils sont ordonnées idéalement (ou presque), le coût de l'exploration est de $O(b^{d/2})$ au lieu de $O(b^d)$

- Le facteur de branchement théorique passe de b à $\sqrt{b}$
- La recherche peut aller deux fois plus loin dans l'arbre

Améliorations possibles

- Comment couper encore plus lors de la recherche ?
  - Recherche aspirante : α et β sont initialisés.
  - Si la valeur finale est bien dans $[\alpha, \beta]$ la recherche reste admissible
- ... D'autres encore à venir !