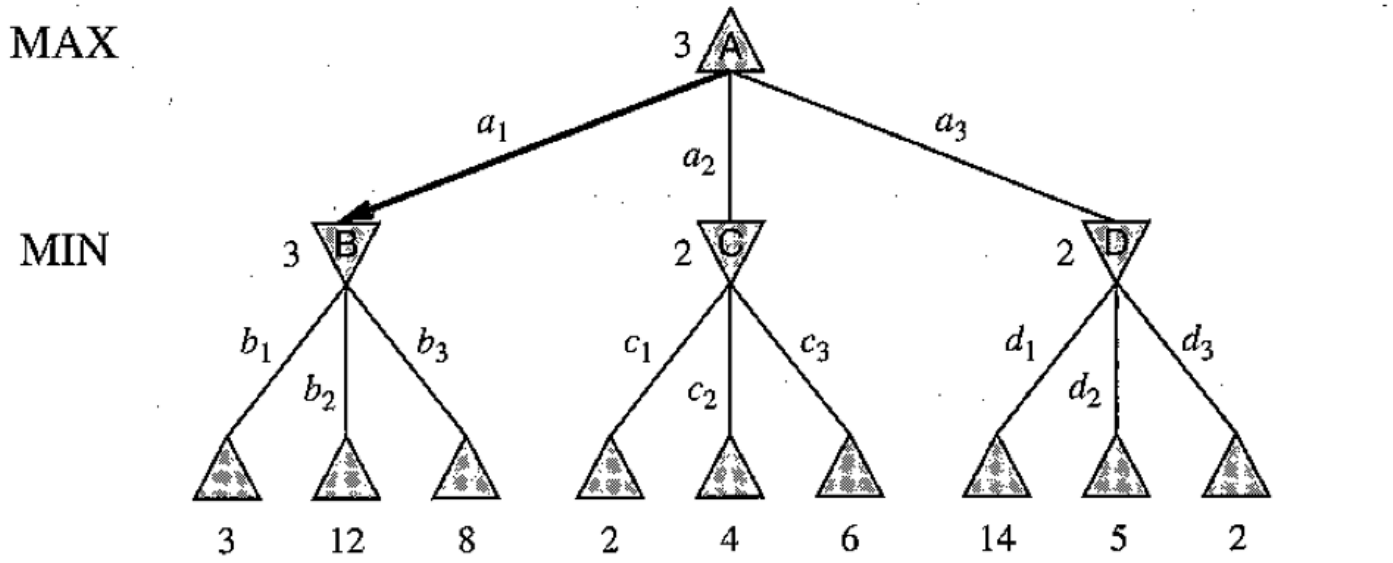
Jeux III

Yue Ma

ma@lisn.fr

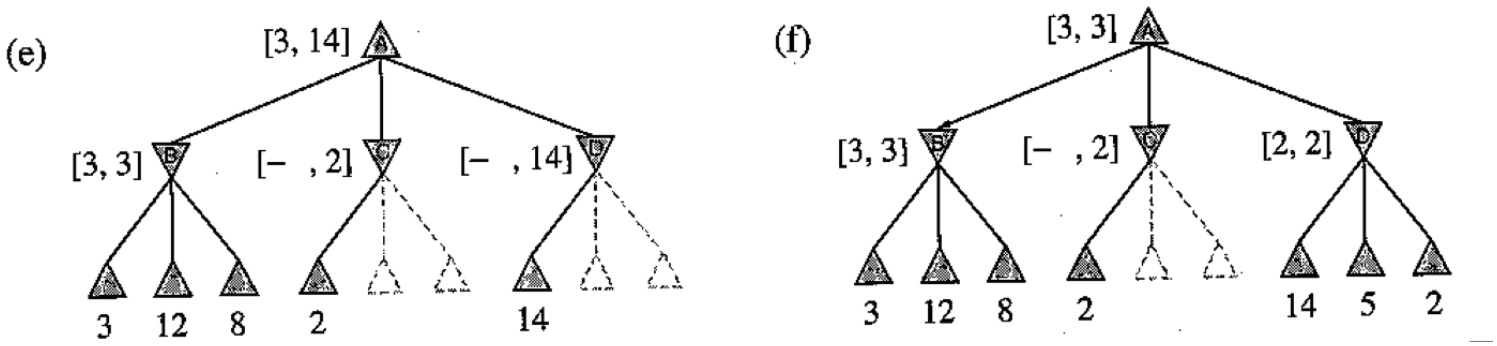
LISN - CNRS, Université Paris-Saclay

MINIMAX



$$
\begin{aligned}
\mathit{MINIMAX}(\mathit{root}) &= \max(\min(3,12,8),\min(2,x,y),\min(14,5,2)) \\
&= \max(3,\mathit{min}(2,x,y),2) \\
&= \max(3,\leq 2,2) \\
&= 3
\end{aligned}
$$

LE RÉORDONNANCEMENT DES NOEUDS DANS $\alpha\beta$



Pas d'élagage des fils de noeud D, parce que ... ? (à vous)

Problématique du temps réel

Questions liées au choix de la profondeur de recherche

La profondeur de recherche doit être fixée avant de lancer la recherche (L'horizon doit être équilibré sur tout l'arbre de recherche). Comment choisir la profondeur ?

- Comment garantir au joueur que l'ordinateur ne va pas passer trop de temps à réflechir ?
- Comment le garantir avec des machines différentes ?
- Comment lier le temps passé à réflechir avec la profondeur maximale de l'horizon ?
- Que se passe-t-il si l'arbre n'est pas homogène (hypothèse (très) réaliste) ?

Comment jouer au mieux dans un temps donné ?

IDÉE : ÉTENDRE PEU À PEU L'HORIZON DE $\alpha\beta$

Mais doit-on mémoriser tout l'arbre pour repartir des feuilles ?

ITERATIVE DEEPENING (ID)

Soit n initialisé à un horizon *immédiat*.

- Faire une recherche à horizon n
- S'il reste du temps, tout oublier (sauf le coup à jouer) et incrémenter n.

ET LES PERFORMANCES ?

ID semble refaire beaucoup de fois la même chose. Est-ce pour autant très inefficace ?

Iterative Deepening

UN ALGORITHME PAS SI *inefficace*

La croissance exponentielle des arbres montre intuitivement que la majorité des noeuds (donc de la difficulté) se situent sur les feuilles... Développer le dernier niveau coûte le plus cher, d'autant plus si on a un grand facteur de branchement.

QUI VEUT DES CHIFFRES ?

Une recherche à profondeur d et facteur de branchement b coûte

$$1 + b + b^2 + \ldots + b^{d-1} + b^d$$

Pour $(d = 4, b = 10)$ on a 11 111 noeuds. Si on fait un ID, on aurait développé 12 344 noeuds au total (11 pour d=1, 111 pour d=2, 1 111 pour d=3, 11 111 pour d=4;), soit 11%[a] de noeuds supplémentaires (seulement).

[a] $(11 + 111 + 1111 + 1111)/11111 = 11.1\%$

Avantages

- Grande souplesse dans la gestion du temps. Le programme peut donner la meilleur valeur trouvée à n'importe quel moment.
- Couplé à $\alpha\beta$, ID peut même devenir plus efficace qu'un seul $\alpha\beta$

COMMENT ?

$\alpha\beta$ élague d'autant plus que les meilleurs coups sont développés en premier. On profite donc de l'ancien $\alpha\beta$ à profondeur n pour ordonner les fils à profondeur $n+1$.

IDÉES

Plutôt que de tout oublier après chaque relancement de ID, on va essayer de mémoriser pour réordonner l'arbre ensuite.

PREMIÈRE MÉTHODE

Garder tout l'arbre de recherche en mémoire, l'ordonner à chaque niveau puis relancer $\alpha\beta$ directement sur le nouvel arbre *optimal*.

SECONDE MÉTHODE

Réserver une zone mémoire pour associer à chaque plateau de jeu déjà vu le meilleur des fils et la valeur heuristique associée (ainsi que sa profondeur).