Intelligence artificielle (IF06M100) • TD 4 • Livre : chapitre 5

Programmation des jeux de réflexion

Exploration en situation d'adversité

- **Le chapitre 2 a introduit la notion d'environnassent multi agents** dans lesquels chaque agent doit tenir compte des actions d'autres agents et de la manière dont celles-ci peuvent le concerner.
- **Dans ce chapitre, nous couvrons les environnements concurrentiels,** dans lesquels les buts des agents sont en conflit, donnant lieu à des problèmes exploration en situation d'adversité - souvent appelés Jeux.

La théorie mathématique des Jeux

- **Branche des sciences économiques.**
- **Considère les environnements multi agents comme un jeu,** dès lors que l'impact de chaque agent sur les autres est « significatif », et indépendamment du fait que les agents coopèrent ou rivalisent !

- **En IA, les jeux les plus fréquents sont d'un type particulier :** les théoriciens des jeux les qualifient de :
  - → Jeux alternés
  - → Déterministes
  - → A somme nulle
  - → A Information parfaite

Exemple : les échecs

Un jeu à somme nulle est défini comme un jeu où la somme des gains de tous les joueurs est identique pour chaque instance du jeu.

Aux échecs, l'issue est la victoire, la défaite ou le nul, dont les valeurs respectives 1, 0 et 1/2.

Les échecs sont à somme nulle parce que chaque jeu a un gain de ou 0+ 1, 1+0 ou 1/2 + 1/2.

L'expression « Jeu à somme constante » aurait été meilleure, mais - jeu à somme nulle » est le terme consacré et semble raisonnable si on imagine que chaque joueur verse un droit d'entrée de 1/2.

Différents types de jeux

Information	Déterministe	Hasard
Parfaite	échecs, reversi, go, ...	backgammon, monopoli, ...
Imparfaite	bataille navale, mastermind, ...	bridge, poker, scrabble, ...

- **Dans notre terminologie, cela se traduit par :**
  - → Des environnements déterministes
  - → Complètement observables,
  - → Les deux agents agissent à tour de rôle
  - → Les valeurs de la fonction d'utilité obtenues en fin de partie sont toujours égales et opposées.
    Par exemple, si un joueur gagne une partie d'échecs, l'autre joueur perd nécessairement. C'est en raison de cet antagonisme entre les fonctions d'utilité des joueurs que l'on parle de « situation d'adversité ».

- **L'état d'un jeu est facile à représenter et les agents sont généralement limités à un petit nombre d'actions dont les résultats sont définis par des règles précises.**
- **Les jeux, comme les problèmes du monde réel, nécessitent la capacité de prendre une décision même lorsque le calcul de la décision optimale est impossible.**
- **En outre, l'inefficacité y est lourdement pénalisée.**

Les jeux à deux joueurs

- **Nous abordons en premier lieu les jeux à deux joueurs,**
  que nous appellerons MAX et MIN.
- **Max joue le premier**, puis les deux joueurs jouent à tour de rôle jusqu'à La fin du jeu.
- **Quand le jeu est terminé**, des points sont attribués au vainqueur et des pénalités au perdant.

- **On peut définir formellement un jeu comme un problème d'exploration**
  caractérisé par les éléments suivants :

Définition d'un jeu

Un jeu peut être formellement défini comme un problème de recherche, avec :

S_0	L'état initial, qui spécifie l'état du jeu au début de la partie
Player(s) Joueur(s)	Définit quel joueur doit jouer dans l'état s
Action(s)	Retourne l'ensemble d'actions possibles dans l'´etat s (L'ensemble des coups autorisés dans un état donné).
Result(s, a)	Fonction de transition, qui définit quel est le résultat de l'action a dans un état s (c'est le modèle de transition, qui définit le résultat d'un coup.)
Terminal – Test(s)	Test de terminaison. Vrai si le jeu est fini dans l'état s, faux sinon. Les états dans lesquels le jeu est terminé sont appelés états terminaux.
Utility(s, p) Utilité(s, p)	Une fonction d'utilité (aussi appelée fonction objectif ou fonction de gain) associe une valeur numérique a chaque état terminal s pour un joueur p

Arbre de jeu vs Arbre d'exploration

L'état initial S_0, la fonction Action(s) et la fonction Result(s, a) définissent **l'arbre de jeu** pour une partie, un arbre dont les nœuds sont des états du jeu et les arêtes sont des coups

Nous utilisons le terme **arbre d'exploration** pour désigner un arbre qui se superpose à l'arbre complet du jeu, et comprend suffisamment de nœuds pour permettre à un joueur de choisir un coup.

Décisions optimales dans les jeux

Si l'on avait affaire à un problème d'exploration habituel, la solution optimale consisterait en une séquence d'actions conduisant à un état final, en l'occurrence un état terminal qui serait une victoire.

Les jeux à deux joueurs

- **Nous abordons en premier lieu les jeux à deux joueurs,** que nous appellerons MAX et MIN.
- **Max joue le premier**, puis les deux joueurs jouent à tour de rôle jusqu'à La fin du jeu.

- **Dans une exploration en situation d'adversité, MIN a son mot à dire.**
- Il faut donc que Max trouve une stratégie contingente qui spécifie le coup de Max à l'état initial.
- Les coups de MAX dans les états résultant de toutes les réponses possibles de MIN, puis ceux que provoqueront les états résultant de toutes les réponses possibles de MIN à ces coups, etc.

- En gros, une stratégie optimale a des résultats au moins aussi bons que toute autre stratégie menée face à un adversaire infaillible.

- Étant donné un arbre de jeu, on peut déterminer la stratégie optimale à partir de **la valeur minimax de chaque nœud** que nous écrivons **MINIMAX(n)**.

- La valeur minimax d'un nœud est l'utilité (pour MAX) associée à un état, en supposant que les deux joueurs jouent de manière optimale depuis l'état considéré jusqu'à la fin de la partie.

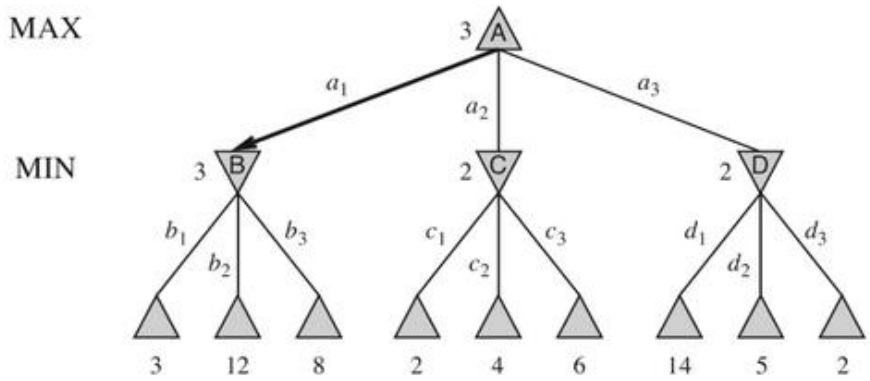


Figure 5.2 : Un arbre de jeu à deux demi-coups. Les nœuds △ sont des « nœuds MAX », dans lesquels MAX a l'initiative, et les nœuds ▽ sont des « nœuds MIN ». Les nœuds terminaux indiquent les valeurs d'utilité pour MAX ; les autres nœuds sont étiquetés par leurs valeurs minimax. À la racine, le meilleur coup possible pour MAX est a_1, parce qu'il mène à l'état ayant la plus forte valeur minimax, et la meilleure réponse de MIN est b_1, parce qu'il mène à l'état ayant la plus faible valeur minimax.

- Bien sûr, la valeur minimax d'un état terminal se résume à son utilité
- En outre, devant un choix, MAX préfère aller vers un état caractérisé par une utilité maximale, tandis que MIN préfère un état de valeur minimale, d'où :

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITÉ}(s) & \text{si TEST-TERMINAL}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RÉSULTAT}(s, a)) & \text{si JOUEUR}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RÉSULTAT}(s, a)) & \text{si JOUEUR}(s) = \text{MIN} \end{cases}$$

Appliquons ces définitions à l'arbre de jeu de la figure 5.2.

- La fonction **Utilité(s, p)** du jeu donne leur valeur aux nœuds terminaux du niveau inférieur.
- Le premier nœud Min, intitulé B, à trois états successeurs dont les valeurs sont 3, 12 et 8. Sa valeur minimax est donc 3.
- De même, les deux autres nœuds Min ont une valeur minimax égale à 2.

- Le nœud racine est un nœud MAX ; ses états successeurs ont les valeurs minimax 3,3 et 2 : en conséquence, sa valeur minimax est égale à 3. On peut également identifier la décision minimax à la racine : l'action a, constitue le choix optimal pour MAX, car elle conduit à l'état ayant la valeur minimax la plus élevée.

Cette définition du coup optimal pour MAX suppose que MIN joue aussi de manière optimale, c'est-à-dire qu'il maximise la pire issue possible pour MAX.

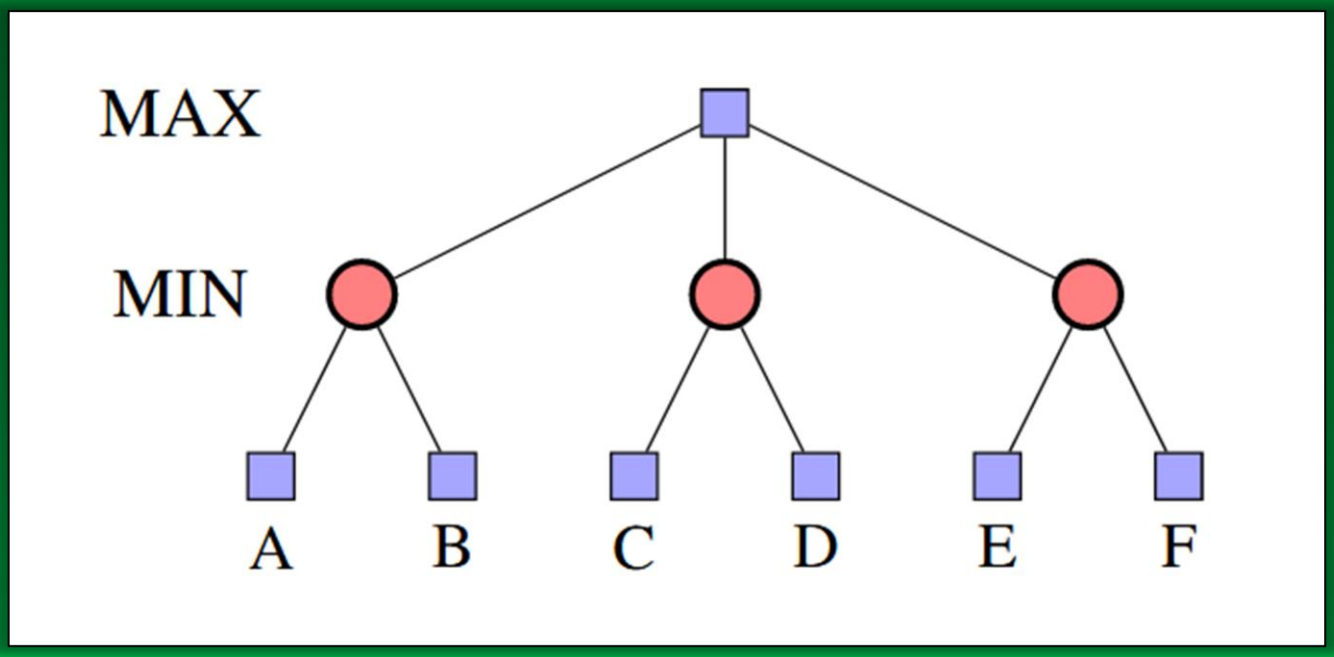
Que se passe-t-il si MIN ne joue pas de manière optimale ? Il est alors facile de montrer que Max peut obtenir un gain encore plus élevé. Face à des adversaires sous-optimaux, il peut y avoir des stratégies plus intéressantes que la stratégie minimax, mais elles seront moins bonnes contre des adversaires optimaux.

Sources :

Artificial Intelligence: A Modern Approach by S. Russell and P. Norvig (French Version)
Cours UE Intelligence artificielle / E. Bonzon

Exercice 1

Considérez l'arbre de jeux suivant.



Question 1

Soit A=3; B=5; C=1; D=8; E=2; F=1

(a) Appliquez l'algorithme MINIMAX sur cet arbre

L'algorithme minimax

- L'algorithme minimax calcule la décision minimax pour l'état courant.
- Il utilise un calcul récursif simple des valeurs minimax de chaque état successeur par le biais d'une implémentation directe des équations du minimax.
- Dans un premier temps, la récursivité développe toutes les feuilles de l'arbre, puis elle fait remonter les valeurs minimax le long de l'arbre niveau par niveau, lorsque les appels récursifs sont dépilés.

- Par exemple, à la figure 5.2, l'algorithme commence par développer récursivement la partie gauche de l'arbre jusqu'à ce qu'il atteigne les nœuds en bas à gauche, puis il utilise la fonction **Utilité(s, p)** sur ceux-ci pour découvrir que leurs valeurs sont respectivement 3, 12 et 8.

- Il prend ensuite la plus basse de ces valeurs et la retourne comme valeur pour le nœud B. On obtient à l'aide du même processus les valeurs 2 pour C et 2 pour D. Enfin, on retient la plus grande valeur entre 3,2 et 2 pour assigner la valeur 3 au nœud racine.

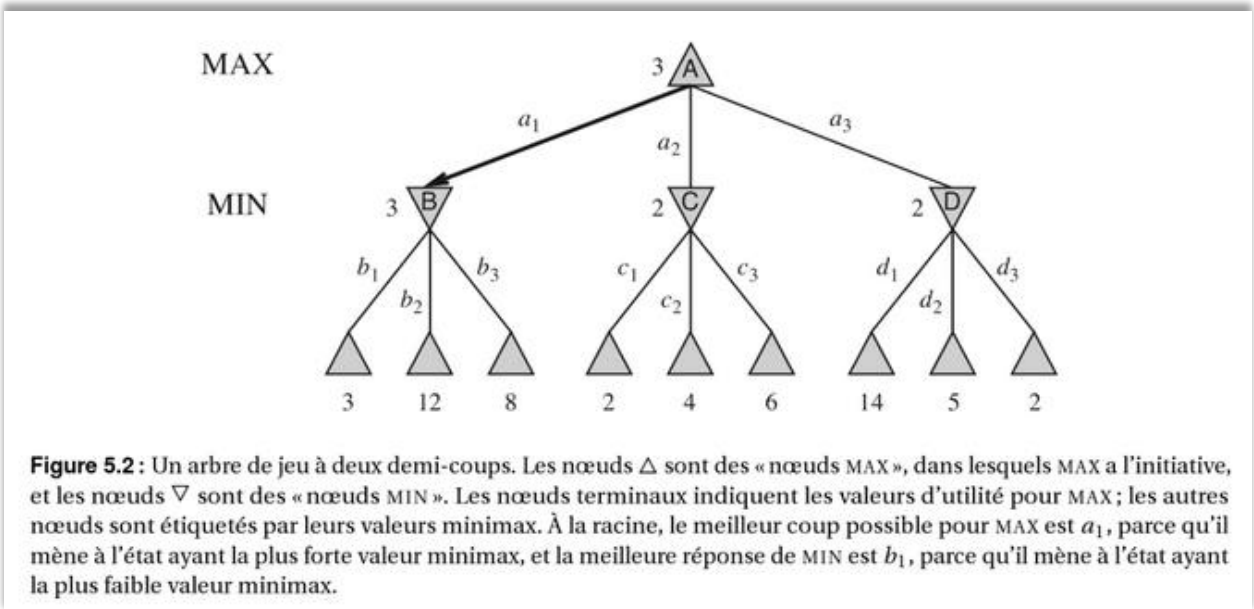


Figure 5.2 : Un arbre de jeu à deux demi-coups. Les nœuds △ sont des « nœuds MAX », dans lesquels MAX a l'initiative, et les nœuds ▽ sont des « nœuds MIN ». Les nœuds terminaux indiquent les valeurs d'utilité pour MAX ; les autres nœuds sont étiquetés par leurs valeurs minimax. À la racine, le meilleur coup possible pour MAX est a_1, parce qu'il mène à l'état ayant la plus forte valeur minimax, et la meilleure réponse de MIN est b_1, parce qu'il mène à l'état ayant la plus faible valeur minimax.

- L'algorithme minimax réalise une exploration en profondeur d'abord (DFS) complète de l'arbre de jeu.

Si la profondeur maximale de l'arbre est m et qu'il y a b coups légaux à chaque point, alors :

- La complexité en temps de l'algorithme minimax est $O(b^m)$.
- La complexité en espace est $O(bm)$

- Pour des jeux réels, cet algorithme nécessite bien sûr un temps qui le rend impraticable, mais il est à la base de l'analyse mathématique des jeux et également à la base d'algorithmes plus pratiques.

> Pour les échecs par exemple : $b \sim 35$, $m \sim 100 \Rightarrow$ solution exacte impossible

fonction DÉCISION-MINIMAX(*état*) **retourne** *une action*
  retourner $\arg\max_{a \in \text{ACTIONS}(s)}$ VALEUR-MIN(RÉSULTAT(*état*, *a*))

---

fonction VALEUR-MAX(*état*) **retourne** *une valeur d'utilité*
  si TERMINAL-TEST(*état*) **alors retourner** UTILITY(*état*)
  $v \leftarrow -\infty$
  pour chaque *a* **dans** ACTIONS(*état*) **faire**
    $v \leftarrow$ MAX(v, VALEUR-MIN(RESULT(s, a)))
  retourner v

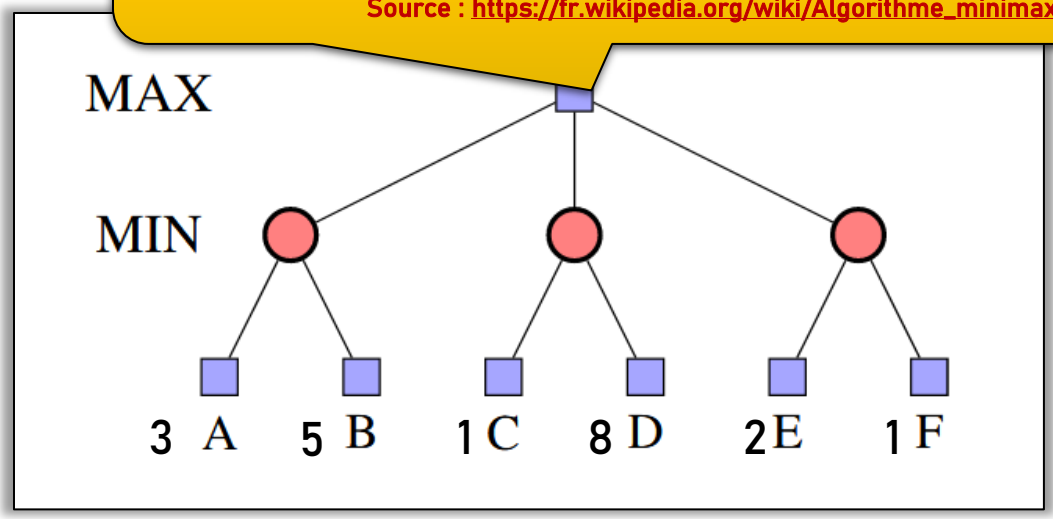
---

fonction VALEUR-MIN(*état*) **retourne** *une valeur d'utilité*
  si TEST-TERMINAL(*état*) **alors retourner** UTILITÉ(*état*)
  $v \leftarrow \infty$
  pour chaque *a* **dans** ACTIONS(*état*) **faire**
    $v \leftarrow$ MIN(v, VALEUR-MAX(RESULT(s, a)))
  retourner v

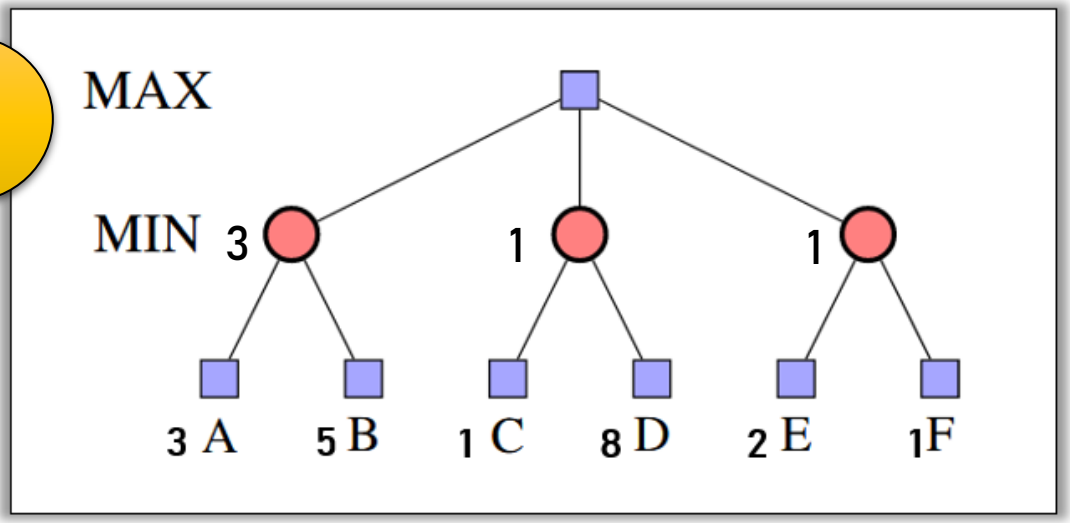
Figure 5.3 : Un algorithme de calcul de décisions minimax. Il retourne l'action qui correspond au meilleur coup possible, autrement dit le coup qui conduit à l'issue caractérisée par la meilleure utilité dans l'hypothèse où l'adversaire joue de manière à minimiser l'utilité. Les fonctions VALEUR-MAX et VALEUR-MIN traversent la totalité de l'arbre de jeu jusqu'à atteindre les feuilles, de manière à déterminer la valeur à remonter pour caractériser un état. La notation $\text{argmax}_{a \in S} f(a)$ correspond au calcul de l'élément a de l'ensemble S qui a la valeur de $f()$ maximale.

Pour déterminer la valeur du premier nœud, on choisit la valeur maximum de l'ensemble des nœuds de la 2em ligne. Il faut donc déterminer les valeurs des nœuds de la 2em ligne qui reçoivent chacun la valeur minimum stockée dans leurs fils. Les nœuds de la 3em ligne sont des feuilles, leur valeur peut donc être calculée par la fonction d'évaluation.
Source : https://fr.wikipedia.org/wiki/Algorithme_minimax

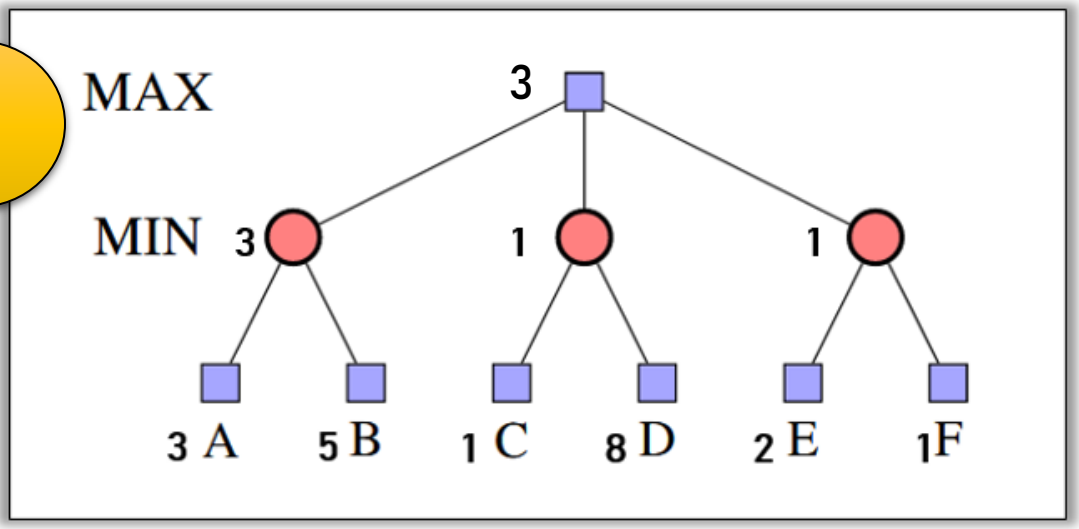
1

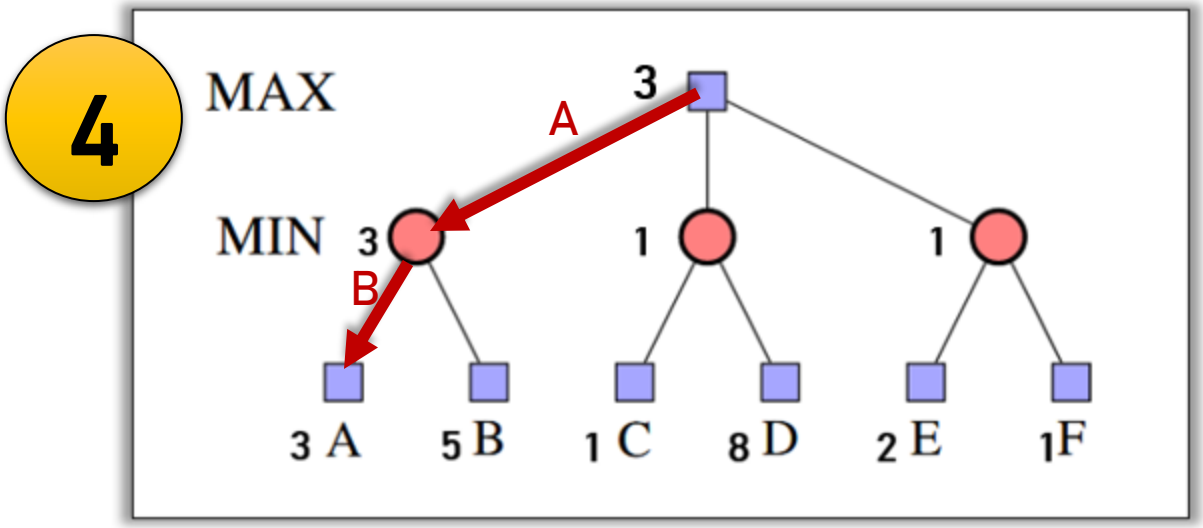


2



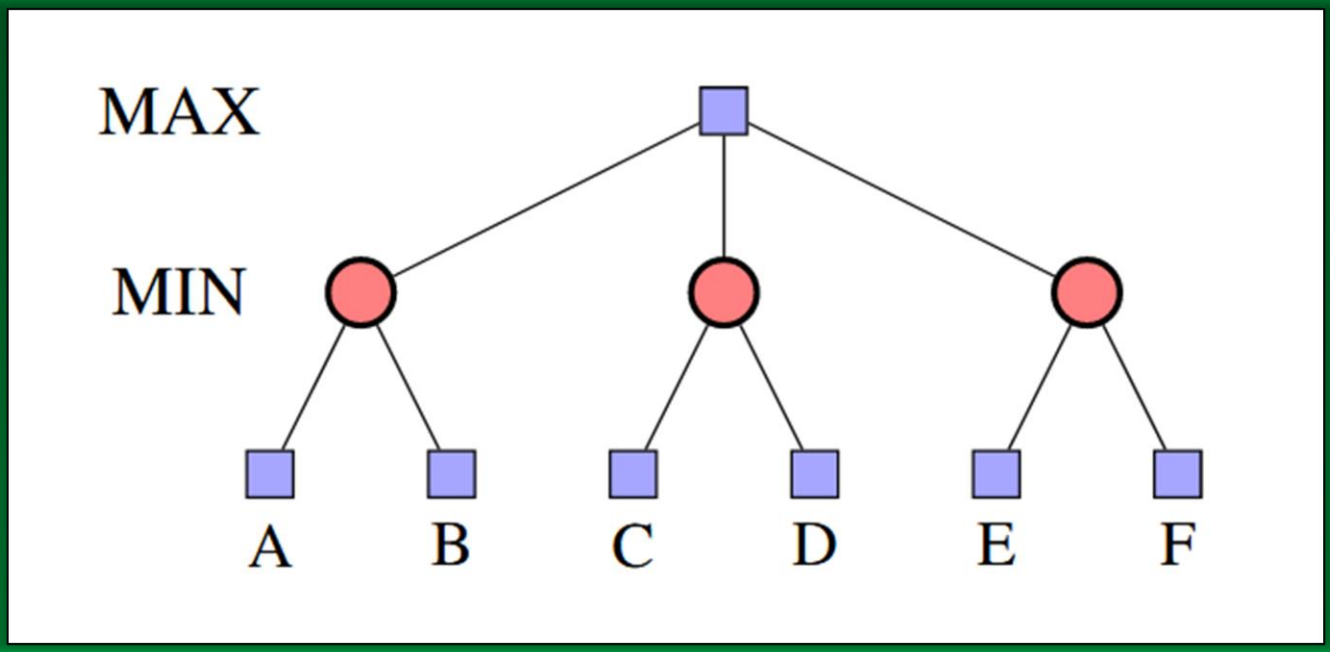
3





Exercice 1

Considérez l'arbre de jeux suivant.



Question 1

Soit A=3; B=5; C=1; D=8; E=2; F=1

(b) Appliquez l'algorithme α-β sur cet arbre

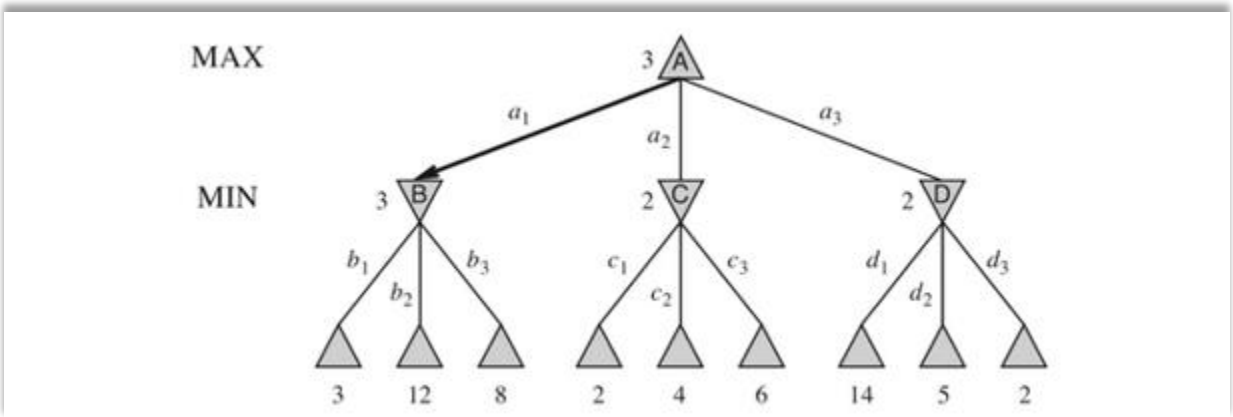
Élagage alpha-bêta

- **Le problème de l'exploration minimax** est que le nombre d'états à examiner dépend exponentiellement de la profondeur de l'arbre.
- S'il est malheureusement impossible d'éliminer l'exposant, on peut toutefois le diviser par deux.

- **En fait, il est possible de calculer la décision minimax appropriée sans examiner tous les nœuds de l'arbre de jeu.**
- Autrement dit, on peut prendre **l'idée d'élagage** ***(pruning)*** pour éviter d'explorer de grandes parties de l'arbre.
- **Cette technique particulière se nomme élagage alpha-bêta.**

- Appliquée à un arbre minimax standard, elle retourne le même coup que le ferait minimax, mais en **élaguant des branches qui ne risquent pas d'influer sur la décision finale**.

- **Considérons de nouveau l'arbre de jeu à deux tours de la figure 5.2**



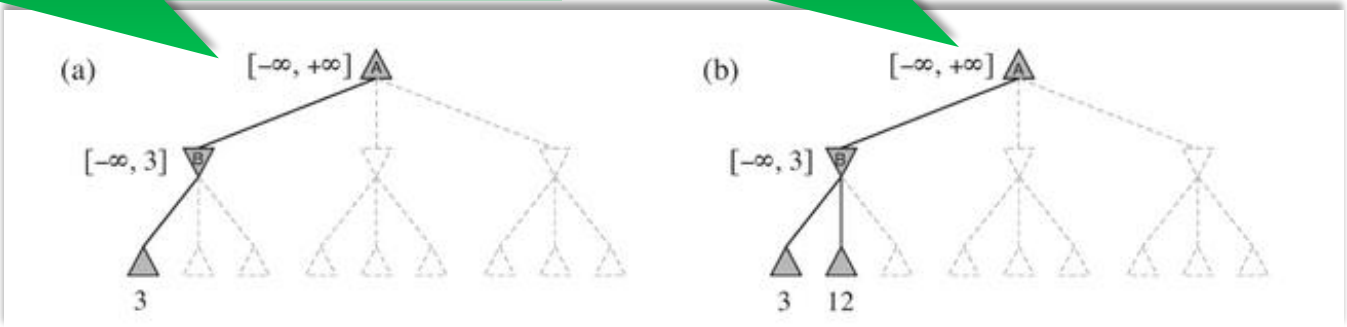
- Calculons la décision optimale, en tenant compte cette fois de ce que l'on sait à chaque étape du processus :

Étapes du calcul de la décision optimale pour l'arbre de jeu de la figure 5.2

À chaque point, on a représenté le domaine de valeurs possibles pour chaque nœud.

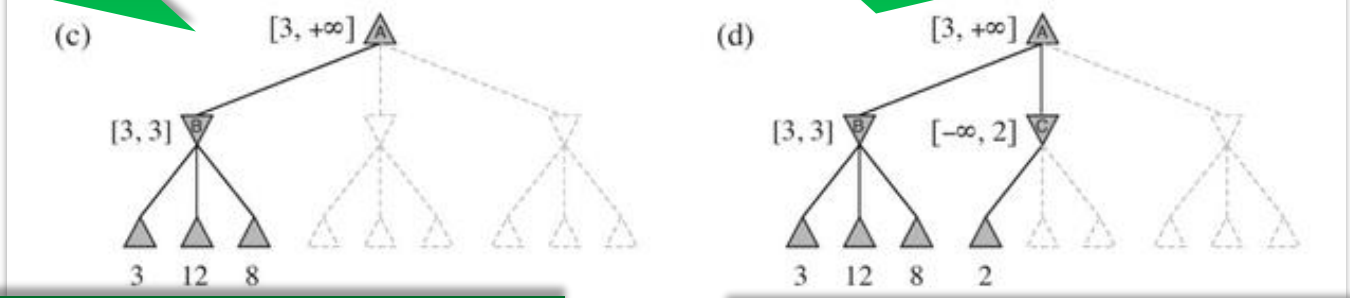
(a) La première feuille sous B a la valeur 3. En conséquence, B, qui est un nœud MIN, a une valeur maximale de 3.

(b) La deuxième feuille sous B a une valeur de 12. MIN évitera ce coup et on en déduit que la valeur de B demeure au plus de 3.



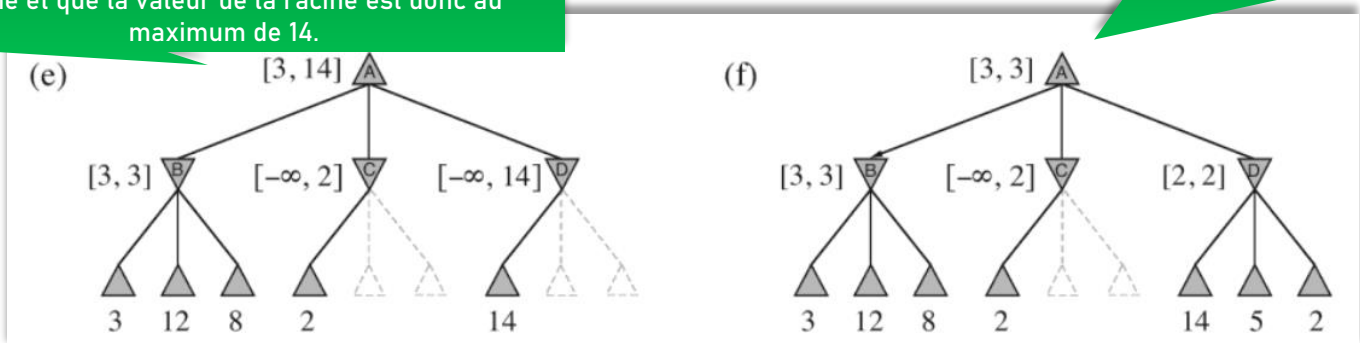
(c) La troisième feuille sous B a une valeur de 8. Comme tous les états successeurs de B ont été passés en revue, la valeur de B est exactement 3. On peut maintenant inférer que la valeur de la racine est au minimum de 3, car MAX a un choix auquel est associée la valeur 3 à la racine.

(d) La première feuille sous C a la valeur 2. En conséquence, C, qui est un nœud MIN, a une valeur maximale de 2. Or, nous savons que B vaut 3, c'est pourquoi MAX ne choisira jamais C. Il est donc inutile d'examiner les autres états successeurs de C. C'est un exemple d'élagage alpha-béta.



(e) La première feuille sous D a la valeur 14 et, par suite, D vaut au maximum 14. Comme cette valeur est supérieure à la meilleure solution de MAX {c'est-à-dire 3), il faut continuer à explorer les états successeurs de D. Remarquez aussi qu'on dispose désormais de limites pour les successeurs de la racine et que la valeur de la racine est donc au maximum de 14.

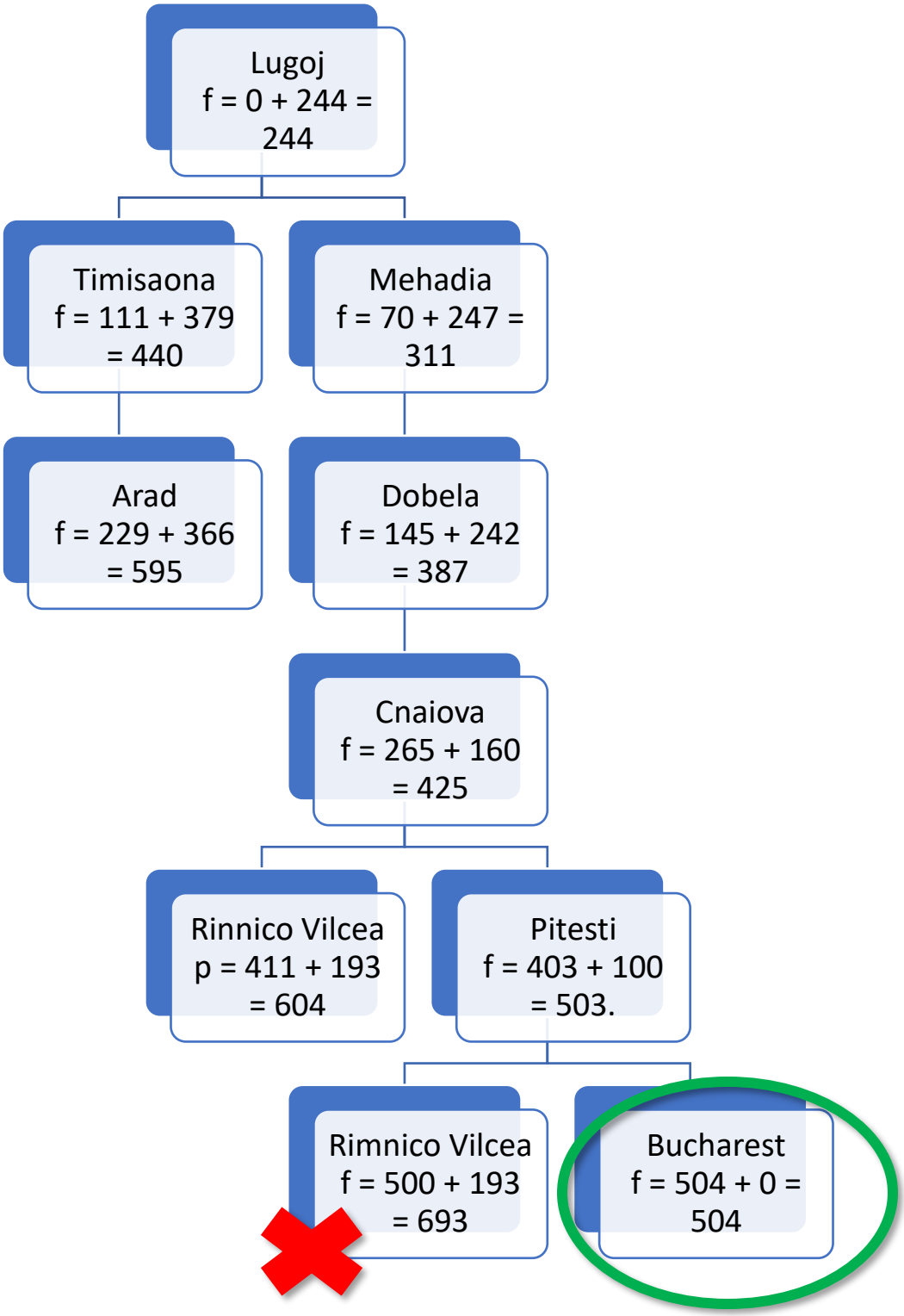
(f) Le deuxième successeur de vaut 5 et on en déduit la nécessité de poursuivre l'exploration. Le troisième successeur vaut 2, ce qui fait que D vaut désormais exactement 2. La décision de MAX à la racine est de se déplacer en B, ce qui donne une valeur de 3.



Il est possible d'identifier la décision minimax sans même évaluer deux des nœuds feuilles.

- **Une autre manière d'envisager la chose tient en une simplification de la formulation du MINIMAX.**
- Supposons que les deux successeurs non évalués du nœud C ont les valeurs x et y.
- La valeur du nœud racine est alors donnée par

$$
\begin{aligned}
\text{MINIMAX}(\textit{racine}) &= \max(\min(3,12,8),\min(2,x,y),\min(14,5,2)) \\
&= \max(3,\min(2,x,y),2) \\
&= \max(3,z,2) \qquad \text{où } z = \min(2,x,y) \le 2 \\
&= 3.
\end{aligned}
$$


Lugoj
f = 0 + 244 = 244
Timisaona
f = 111 + 379 = 440
Mehadia
f = 70 + 247 = 311
Arad
f = 229 + 366 = 595
Dobela
f = 145 + 242 = 387
Cnaiova
f = 265 + 160 = 425
Rinnico Vilcea
p = 411 + 193 = 604
Pitesti
f = 403 + 100 = 503.
Rimnico Vilcea
f = 500 + 193 = 693
Bucharest
f = 504 + 0 = 504