Exploration systématique de graphes

2. Application dans les arbres de jeu

2. 1. Min-Max

Jeu des X et des O

Jeu des billes

2.2 Elagage(Coupe) alpha-béta

Exemple

C. Exploration en largeur

Passage de a à b avec application d'un nombre minimal de fonctions

Backtracking dans les arbres de jeux.

Principe du MIN-MAX

- On considère les jeux de stratégie entre deux joueurs. Les deux joueurs sont soumis aux mêmes règles(symétrie). Le jeu est déterministe(le hasard n'intervient pas).
- Ce type de jeu peut être représenté sous forme d'une arborescence. Chaque nœud représente une configuration possible du jeu. un arc représente une transition légale entre deux configurations. La racine constitue la configuration initiale, les feuilles les configurations finales(gagné, perdu, nul).
- L'un des joueurs est la machine.

Backtracking dans les arbres de jeux.

Principe du MIN-MAX

- A chaque nœud de l'arbre est associé une valeur. On commence par attribuer des valeurs aux feuilles.(+1 si A gagne, -1 si A perd, 0 si nul)
- Les valeurs sont propagées vers les noeuds ascendants, jusqu'à arriver à la racine de la façon suivante:
- - si c'est au tour de A de jouer, le noeud correspondant prend la plus grande des valeurs des ses fils(le coup le plus profitable pour A)
- - si c'est au tour de B de jouer, le noeud correspondant prend la plus petite des valeurs des ses fils.(le coup le plus profitable pour B)

Backtracking dans les arbres de jeux.

Principe du MIN-MAX

- Si la racine prend comme valeur 1, le joueur A peut gagner s'il ne fait pas d'erreurs.
- Si la racine prend la valeur -1, le joueur A est assuré de perdre si B ne fait pas d'erreurs.
- Si la racine prend la valeur 0, aucun des deux joueurs n'a de stratégie gagnante, mais tous deux peuvent s'assurer, au pire, d'un match nul en jouant aussi bien que possible.(Cas du jeu des croix et des cercles décrits plus loin).

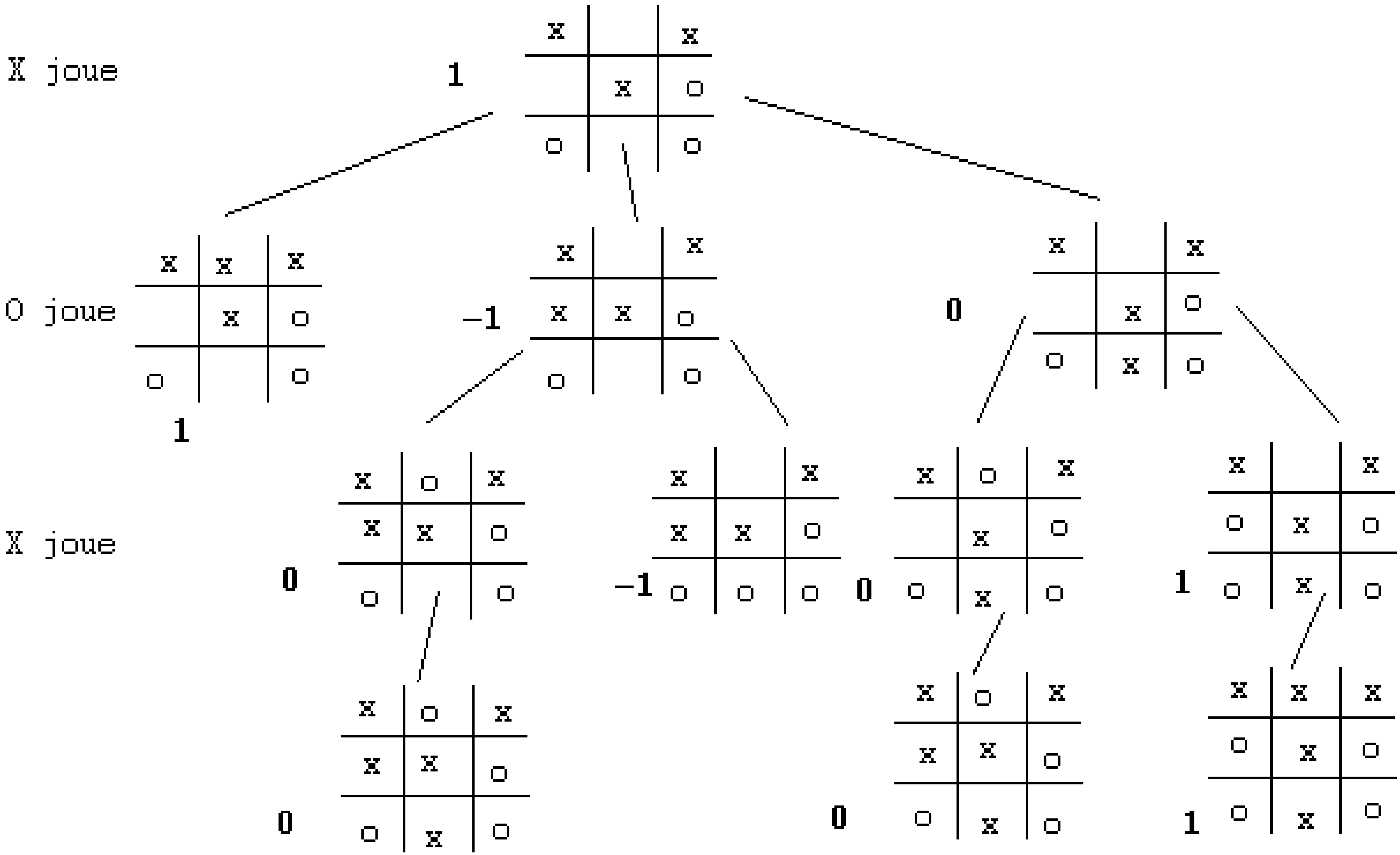
Backtracking dans les arbres de jeux.

Exemple 1 : Tic-Tac-Toe(jeu des croix et des cercles)

- **consiste à placer des croix et des cercles sur une grille de 9 cases, jusqu'à ce que l'un des joueurs aligne trois de ses symboles.**

- **Partant d'une situation donnée j et c'est au tour du joueur X de jouer par exemple. Quel est le bon choix ? Il faut donc calculer le MinMax pour se décider. Donc X avant de jouer doit évaluer les fils et prendre le max.**

Backtracking dans les arbres de jeux.



Backtracking dans les arbres de jeux.

Exemple 2 : Jeu des billes

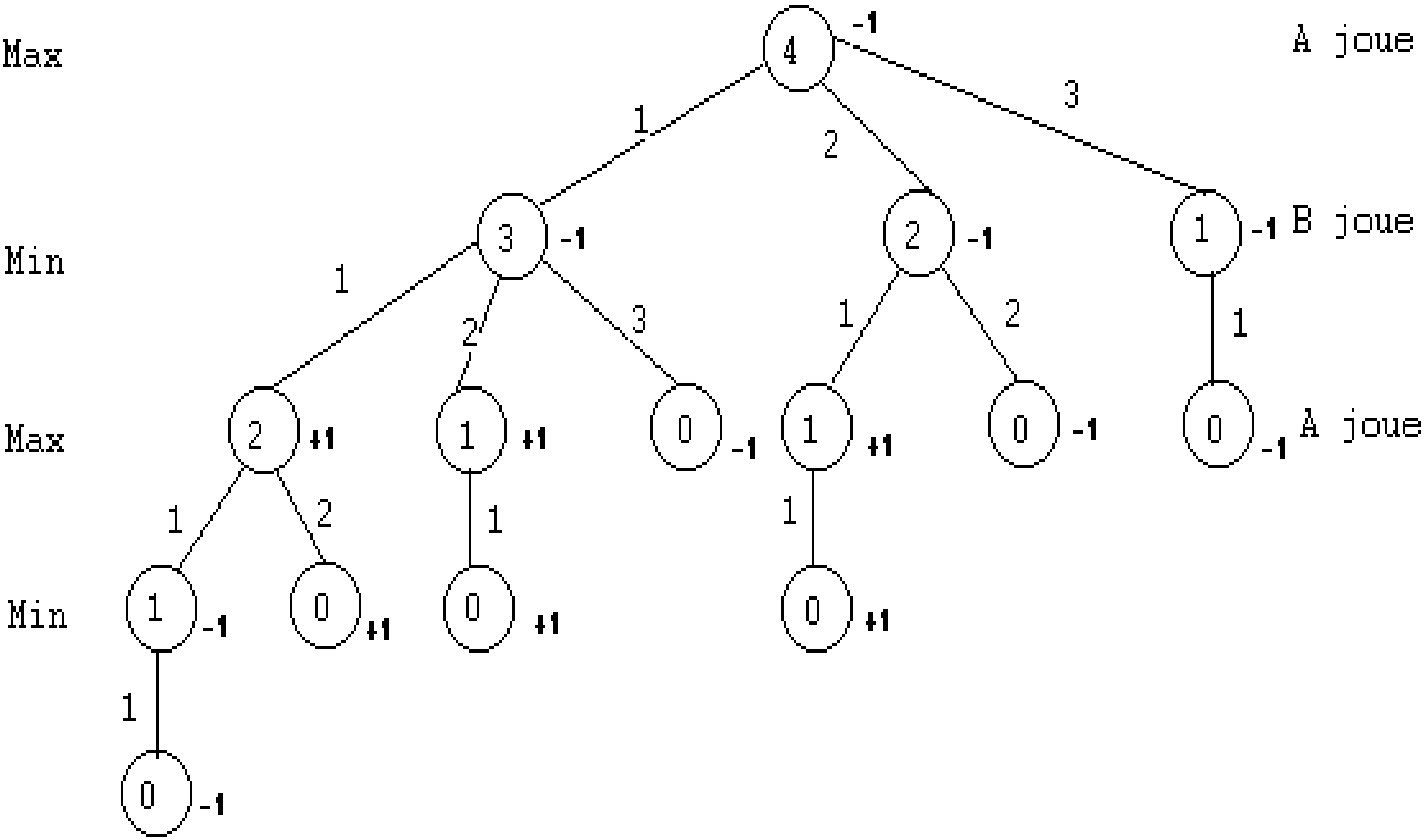
- 1. On dispose de n billes sur une table.
- 2. A prend un nombre de billes entre 1 et k. S'il reste 0 bille A gagne.
- 3. B prend un nombre de billes entre 1 et k. S'il reste 0 bille B gagne.
- 4. Si non fin allera 2.

Backtracking dans les arbres de jeux.

Exemple 2 : jeu des billes

- **Pour n=4 et k=3, le joueur B a une stratégie gagnante comme le montre l'arbre suivant.**
- **Pour n=4 et k=2 le joueur A a une stratégie gagnante.**

Exemple pour n=4 et k=3



Backtracking dans les arbres de jeux.

Algorithme du MIN-MAX

- **utilise un parcours d'arbre post-fixé.**
- **Fonction Minmax(j, mode)**

  évalue le coût de la situation j, sachant que si mode = Max c'est au premier joueur de jouer et au second si mode = Min. La fonction retourne le coût de la situation.

Backtracking dans les arbres de jeux.

```
Algorithme du MIN-MAX
Si j est une feuille :   Retourner le coût de j
Sinon
  Si mode = Max  :     Val := - infini
  Sinon Val := + infini Fsi
  Pour chaque fils K du jeu j:
    Si mode = Max
         Ret Max(Val,Minmax(K,Min))
    Sinon
         Ret Min(Val,Minmax(K, Max))
    Fsi
  Finpour
Fsi
```

Backtracking dans les arbres de jeux.

Algorithme du MIN-MAX

- **<u>Remarque 1 :</u>**

Si l'arbre de jeu n'est pas très grand, le construire et le garder totalement en mémoire centrale. Donc éviter de calculer à chaque le MinMax des nœuds

- **<u>Remarque 2 :</u>**

— Les valeurs attribuées aux nœuds peuvent être généralisées à des nombres quelconques (appelés coût).

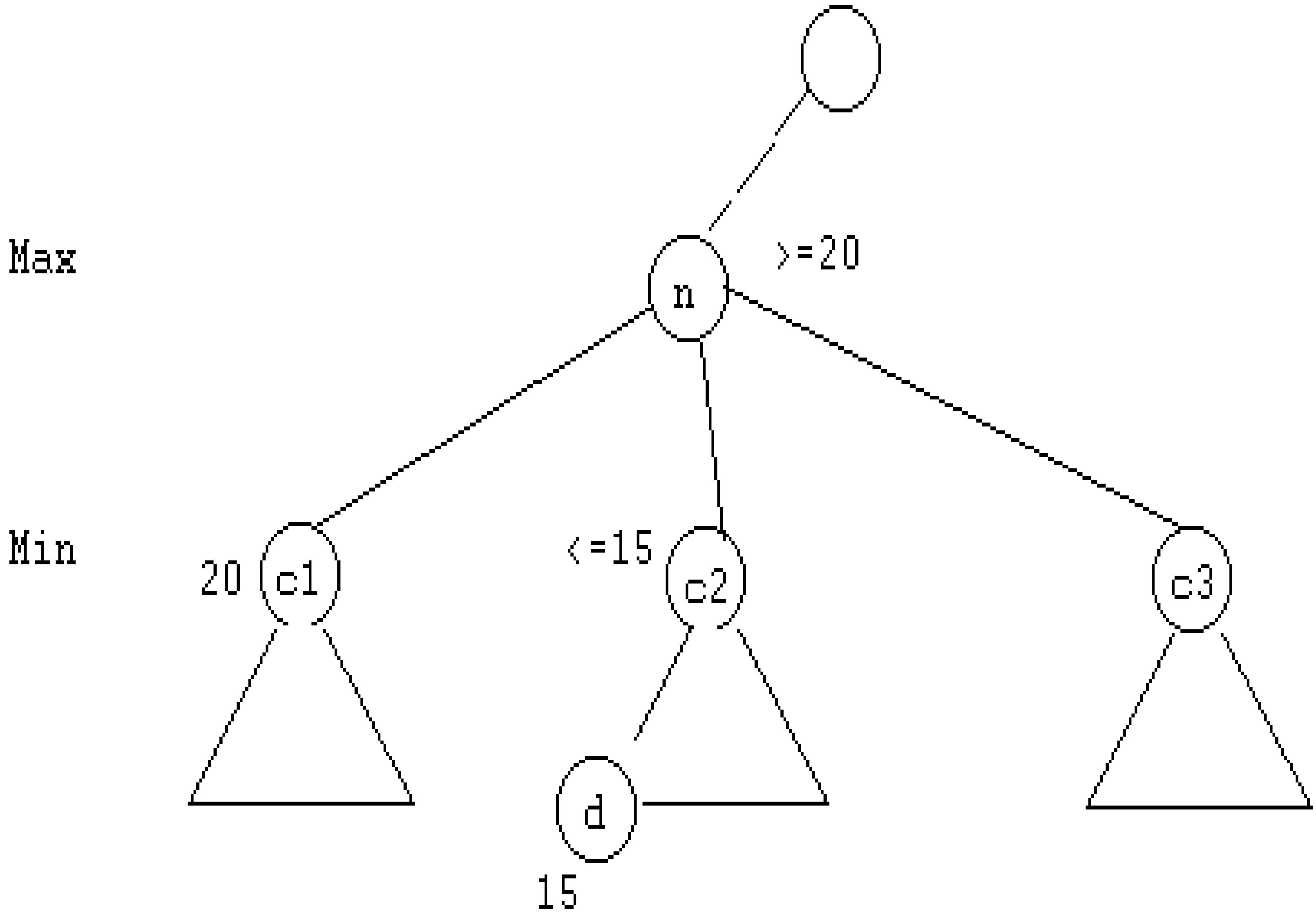
— Par exemple dans le jeu des échecs l'arbre de jeu est trop grand. Pour cela, la profondeur est fixée en fonction de la capacité de calcul de la machine. Chaque programme utilise une fonction de coût permettant d'estimer, suivant la disposition de l'échiquier(nombre de pièces, densité de défense autour du roi,...) la probabilité que la machine gagne à partir de cette position.

Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **Par une simple remarque, on peut éliminer une grande partie des arbres de recherche en ignorant certains descendants d'un nœud.**

Backtracking dans les arbres de jeux.



Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- Par une simple remarque, on peut éliminer une grande partie des arbres de recherche en ignorant certains descendants d'un nœud.
- 1) En parcourant tous les descendants de c1, on trouve que c1 a une valeur égale à 20.
- 2) Avant d'explorer les prochains fils de n, on peut déjà dire que n >= 20. (car n est un nœud maximisant)
- 3) En explorant c2 et ses fils, on tombe sur un nœud d de valeur 15. Comme d est un fils de c2, on aura c2 <= 15(nœud minimisant)
- 4) A ce niveau, on peut abandonner l'exploration des autres fils de c2 et passer directement à c3.

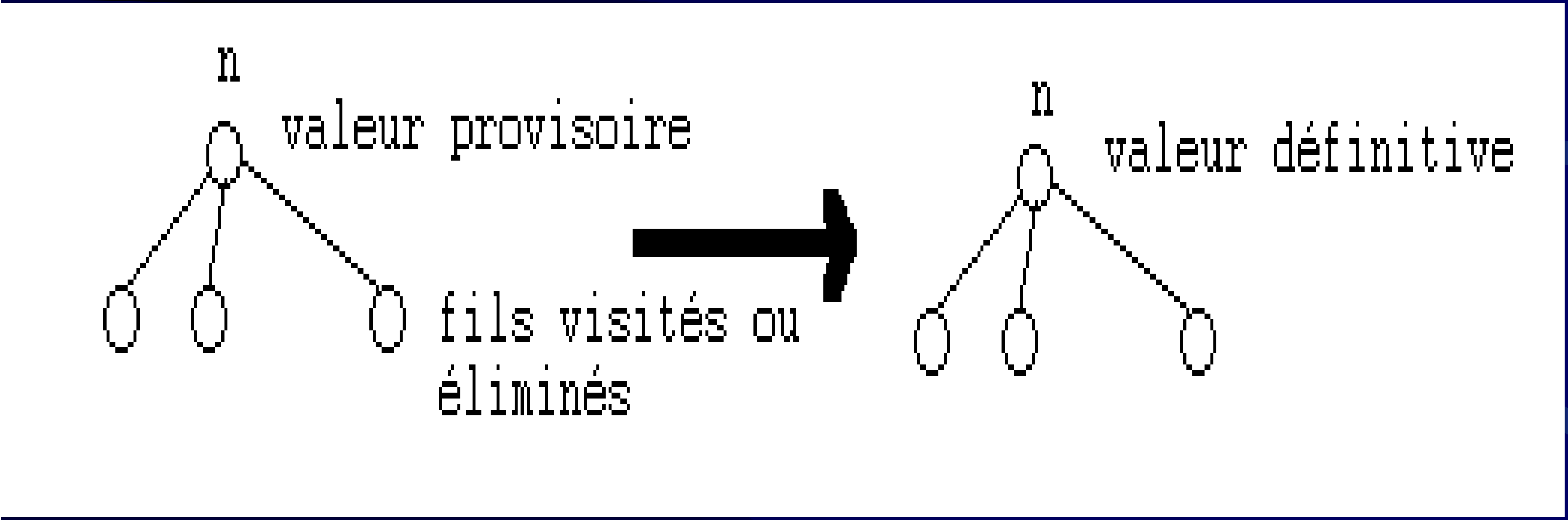
Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **<u>Règles de calcul</u>**

1. Si tous les enfants d'un nœud n ont été examinés ou éliminés, transformer la valeur de n (jusqu'ici provisoire) en une valeur définitive.

Backtracking dans les arbres de jeux.



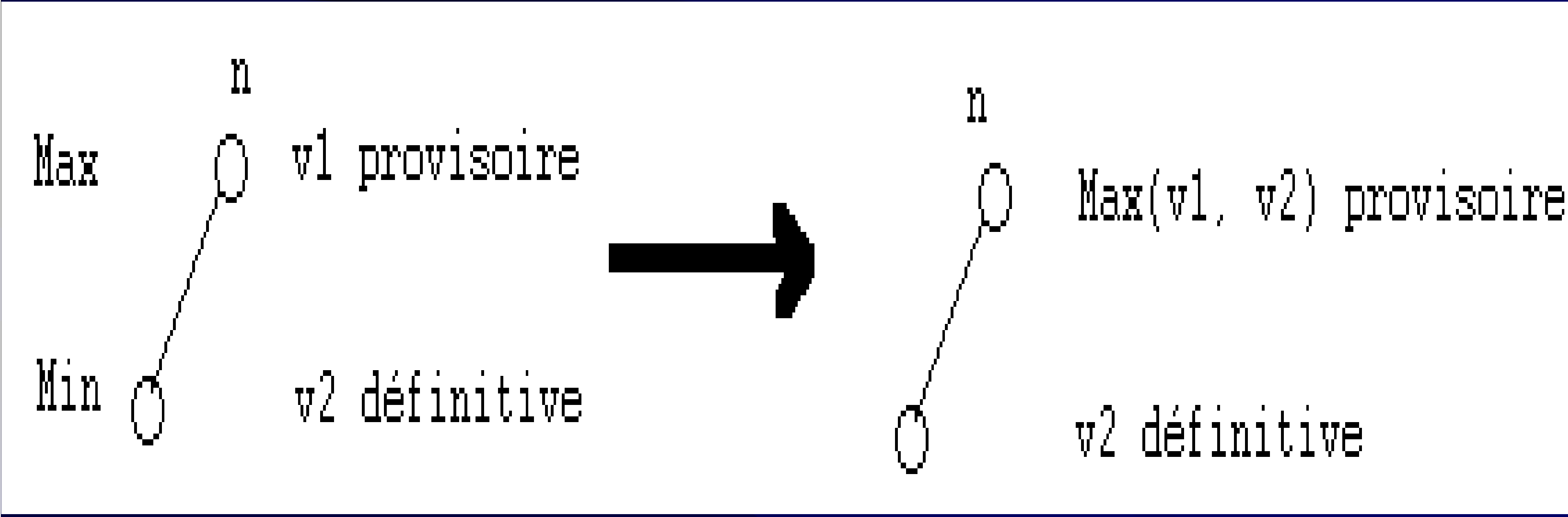
Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **Règles de calcul :**

2. Si un nœud maximisant n a une valeur provisoire v1 et un fils de valeur définitive v2, donner à n la valeur provisoire Max(v1, v2). Si n est minimisant on lui donne la valeur provisoire (dans les mêmes conditions) Min(v1, v2).

Backtracking dans les arbres de jeux.



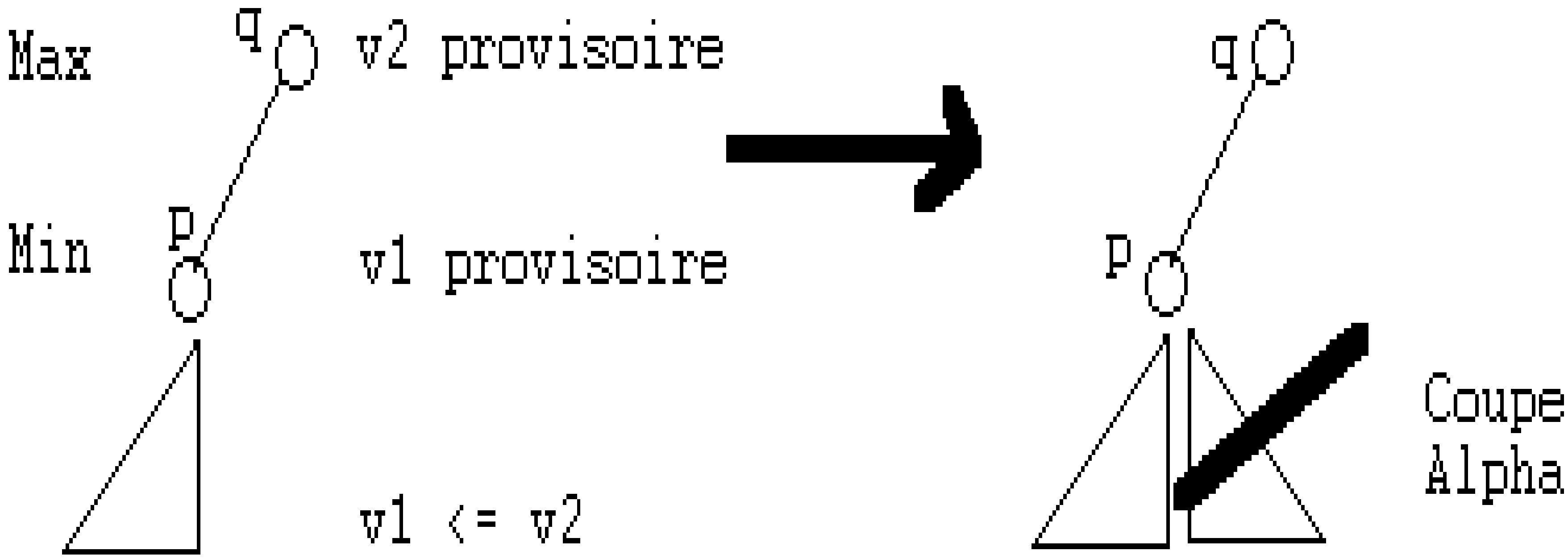
Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **<u>Règles de calcul :</u>**

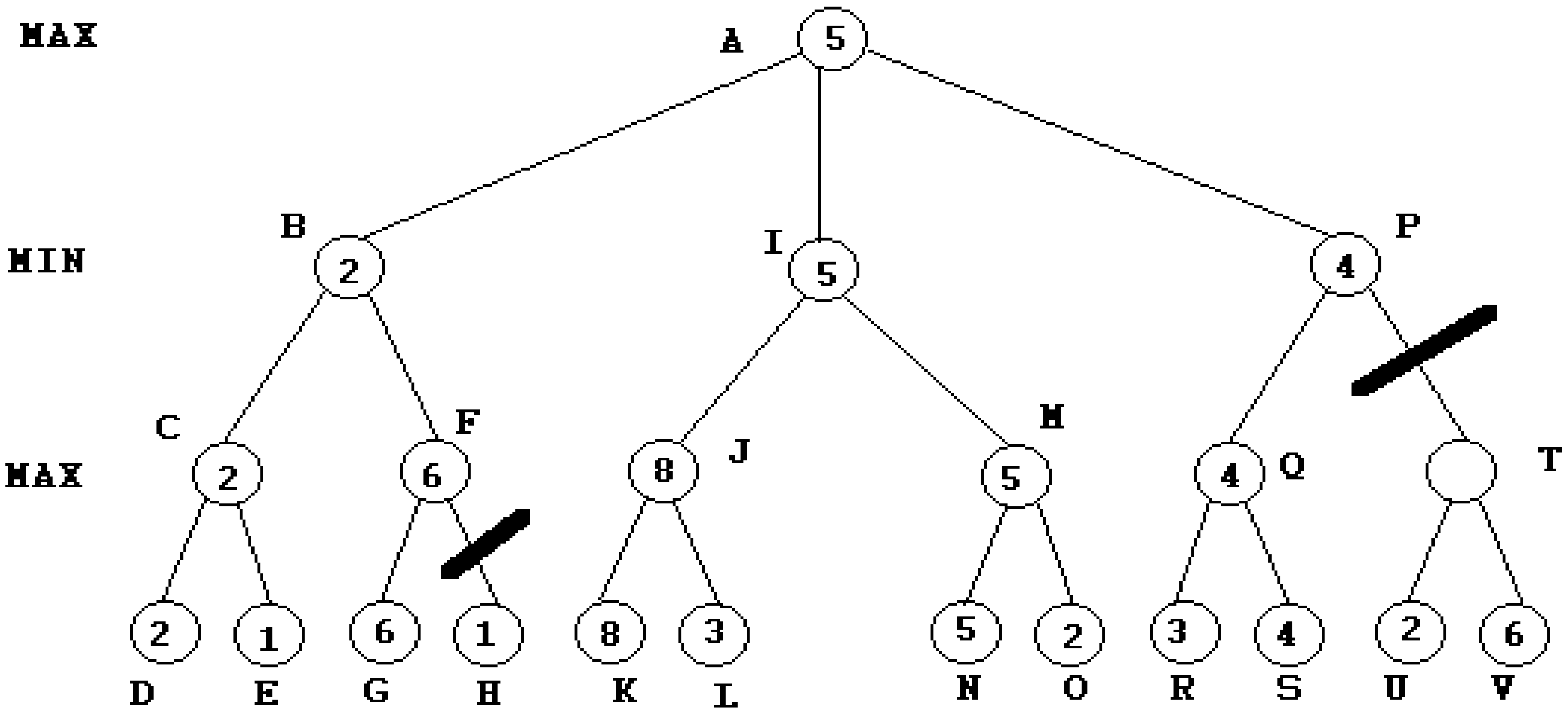
  3. Si p est un nœud minimisant de père q maximisant avec des valeurs provisoires respectives v1 et v2 avec v1 <= v2, alors ignorer toute la descendance encore inexplorée de p (coupe de type Alpha). Une coupe de type Béta est définie de manière analogue dans le cas où p est maximisant et q minimisant et v1 >= v2.

Backtracking dans les arbres de jeux.



Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning") **: Exemple**



Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **<u>Scénario :</u>**

Noeud visité en postordre : D avec valeur définitive 2

R2 : V(C) = 2 provisoire

Noeud visité E avec valeur définitive 1

R1 : V(C) = 2 définitive

R2 : V(B) = 2 provisoire

Noeud visité G de valeur 6

R2 : V(F) = 6

R3 : Coupe Béta sur la branche H

R2 : V(A) = 2 provisoire

Noeud visité K de valeur 8

R2 : V(J) = 8 provisoire

Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **<u>Scénario :</u>**

  Noeud visité L de valeur 3

  R1 : V(j) = 8 définitive

  R2 : V(I) = 8 provisoire

  Noeud visité N de valeur 5

  R2 : V(M) = 5 provisoire

  Noeud visité O de valeur 2

  R1 : V(M) = 5 définitive

  R1 : V(I) = 5 définitive

  R2 : V(A) = 5 provisoire (Max de 2 et 5)

  Noeud visité R de valeur 3

Backtracking dans les arbres de jeux.

Principe de l'élagage Alpha-Béta (coupe de branche " pruning")

- **Scénario :**

R2 : V(O) = 3

Noeud visité S de valeur 4

R1 : V(Q) = 4

R2 : V(P) = 4 provisoire

R3 : Coupe Alpha sur branche TUV

Exploration en largeur

Exploration en largeur

- **Si le graphe est très grand(ou infini) ou si l'on veut plutôt chercher une solution avec peu de manipulations, on utilise un parcours en largeur.**
- **Exemple : Passage de 15 à 4 en appliquant un nombre minimal de fonctions f et g définies par f(x) = 3x et g(x) = x div 2.**
- **$4 = gfg^2(15)$**

Exploration en largeur