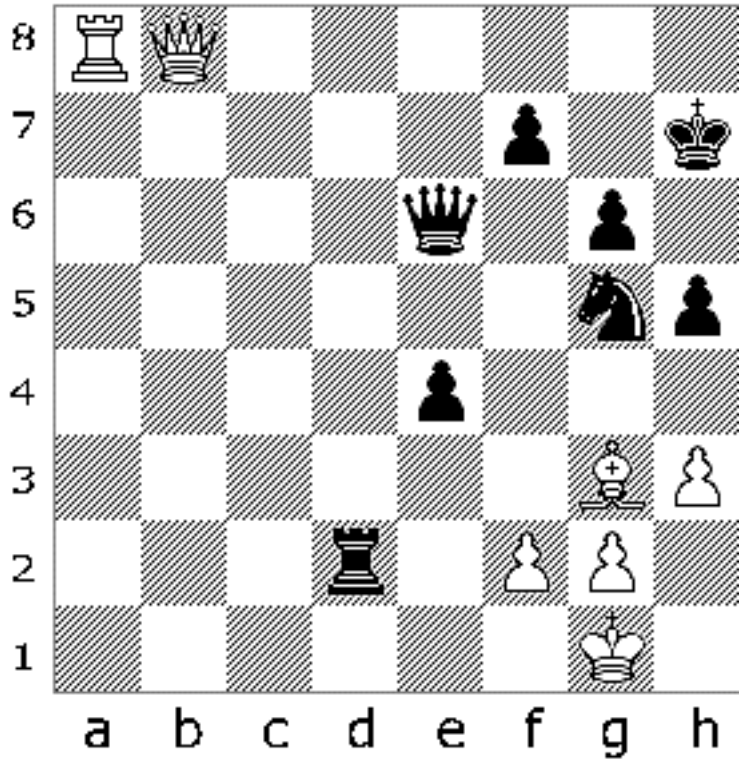
Exploration d'espace d'état

Une classe particulière d'algorithmes,
utilisés dans de nombreuses situations, permettent de résoudre
les problèmes qui se ramènent à **l'*exploration* d'un espace d'états**.

En particulier dans le cas des jeux à information ouverte
(i.e. des jeux où toute l'information nécessaire est constamment accessible),
les nœuds de l'arbres des états sont les différentes situations du jeu, associées
à l'indication du joueur dont c'est le tour de jouer

Par exemple, aux échecs, un état est la donnée d'une configuration
sur l'échiquier, associée à une information du type «c'est aux blancs de jouer».

Exemple d'états pour le jeu d'échecs



- Si c'est aux blancs de jouer, ils gagnent en 1 coup (Dame-H8)
- Si c'est aux noirs de jouer, ils *peuvent* gagner en 3 coups...

Le jeu des allumettes (1)

Comme exemple d'illustration, nous allons considérer un jeu très simple, **le jeu des allumettes**.[6]

Ce jeu se joue à deux.
La situation de départ est un nombre donné d'allumettes (ce nombre sera appelé la dimension du jeu), associé à l'indication du joueur qui doit commencer le jeu.

Chacun des joueurs retire alors à tour de rôle 1, 2 ou 3 allumettes parmi les allumettes encore disponibles.

Le joueur qui gagne est celui qui réussit à retirer la (ou les) dernière(s) allumette(s).

6. Ce jeu, connu sous bien d'autres noms - par exemple «le dernier des mohicans», admet de nombreuses variantes.

Le jeu des allumettes (2)

Exemple: déroulement d'un jeu de dimension 11, le joueur 1 débute

Coup joué	***Etat***	
	11 allumettes disponibles	la main est au joueur 1
le joueur 1 retire 3 allumettes	8 allumettes disponibles	la main est au joueur 2
le joueur 2 retire 2 allumettes	6 allumettes disponibles	la main est au joueur 1
le joueur 1 retire 2 allumettes	4 allumettes disponibles	la main est au joueur 2
le joueur 2 retire 1 allumettes	3 allumettes disponibles	la main est au joueur 1
le joueur 1 retire 3 allumettes	0 allumettes disponibles	fin du jeu

le joueur 1 gagne...

Le jeu des allumettes (3)

Représentation en graphe (en arbre) des étapes du jeu:

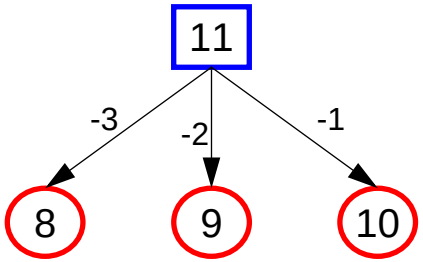
Chaque niveau de l'arbre correspond aux coups possibles d'un joueur, en alternance (cette alternance est représentée ci-après par des sommets en forme de cercles ou de rectangles).

Un **sommet** représente le **nombre d'allumette encore disponible**
pour le joueur associé à la profondeur du sommet (mis en évidence par sa forme)

Les fils d'un sommet sont alors les situations accessibles en un coup
à partir de la situation décrite par le nœud père.

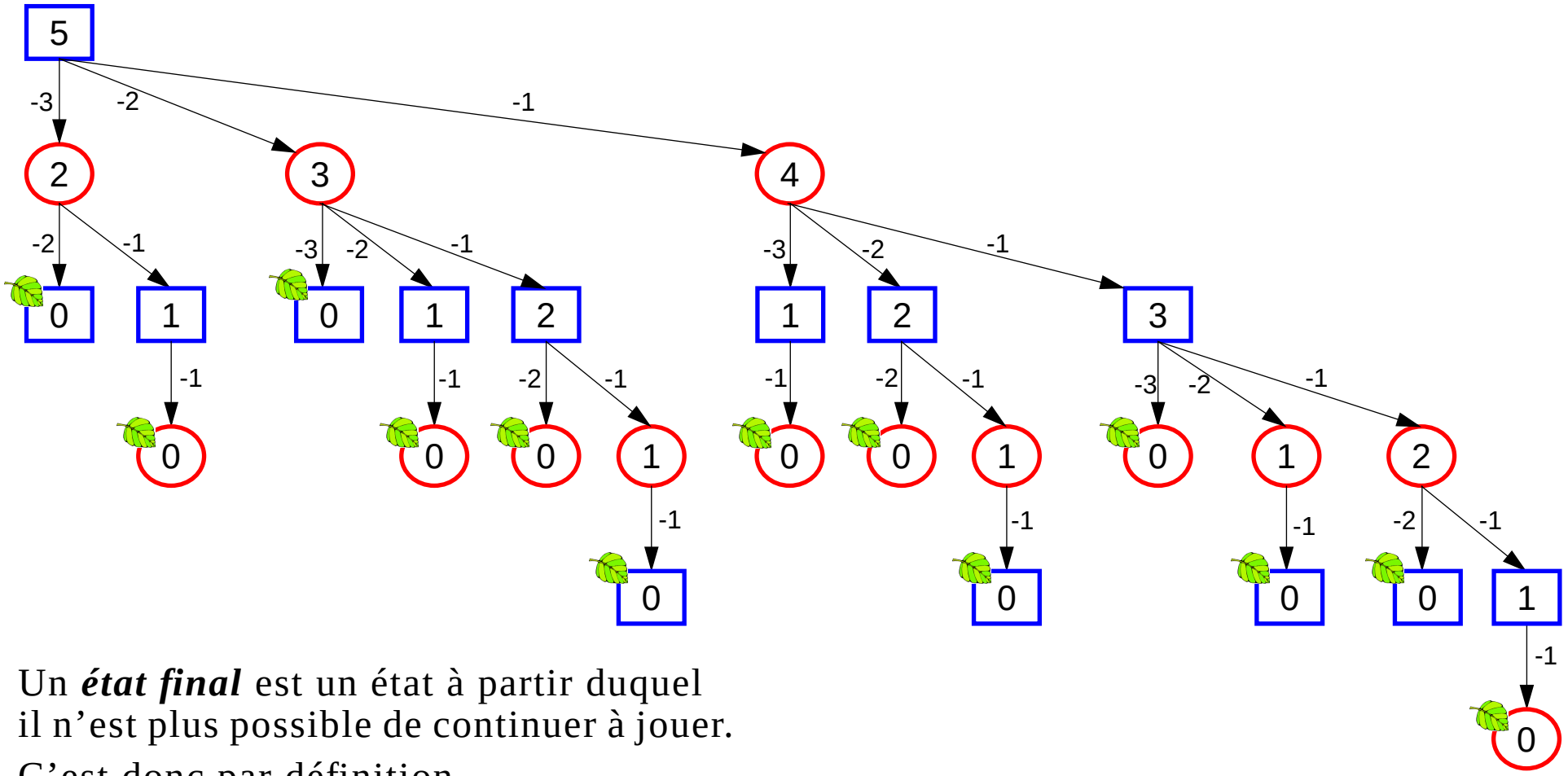
Exemple:

à partir de l'état «11 allumettes disponibles, le joueur 1 doit jouer»,
les trois états accessibles sont:



Le jeu des allumettes (4)

Arbre complet des états d'un jeu de dimension 5:



Un ***état final*** est un état à partir duquel il n'est plus possible de continuer à jouer.

C'est donc par définition **une feuille** de l'arbre des états.

Score d'un état final

Pour pouvoir identifier et caractériser – *gagnant*, *perdant*, *nul* – les états finaux d'un jeu, il faut disposer d'une fonction permettant **d'associer un *score* à tout état final**.

Le *score* sera, pour un état final et un joueur donné, une valeur numérique, comprise entre `-maxScore` et `+maxScore`:

⇨ la valeur `-maxScore` indique que la situation est perdue pour ce joueur (et donc gagnée pour son adversaire);

⇨ la valeur `+maxScore` indique que la situation est gagnée pour ce joueur.

Les valeurs comprise dans les intervalles `]-maxScore,0[` et `]0,+maxScore[` permettent de gérer des notions comme «le résultat du jeu est plutôt favorable (ou défavorable) à ...».

Stratégie de jeu

Une ***stratégie* de jeu** est une méthode permettant à un joueur de choisir les coups à jouer au cours de la partie.

Chaque joueur va bien sûr s'efforcer de mettre en œuvre une ***stratégie optimale***, i.e. une *stratégie lui permettant d'arriver à une situation finale la plus favorable possible*

Une stratégie optimale doit donc permettre au joueur qui l'applique de faire évoluer le jeu vers un état dont le score lui est aussi favorable que possible.

Du fait de la définition de la fonction de score:

- ce joueur va s'efforcer d'atteindre un état final dont le score lui est aussi favorable que possible, c'est-à-dire, en regard de sa fonction de score, aussi grand que possible (idéalement `+maxScore`): ce joueur sera de ce fait appelé le ***maximiseur***;

alors que:

- son adversaire va également s'efforcer d'atteindre un état final dont le score lui est aussi favorable que possible, c'est-à-dire (au sens de la fonction de score du *maximiseur*), aussi petit que possible (idéalement `-maxScore`): ce joueur sera de ce fait appelé le ***minimiseur***.

Fonction d'évaluation MIN-MAX

Une **stratégie optimale** consiste à systématiquement choisir le coup le plus favorable, i.e. le coup permettant d'arriver à l'état final le plus favorable, **quelle que soit la stratégie utilisée par l'adversaire** (et en particulier si l'adversaire choisit lui aussi systématiquement le coup qui lui est le plus favorable).

Pour implémenter une telle stratégie, on peut utiliser une fonction d'évaluation des sommets de l'arbre des états, appelée **fonction d'évaluation MIN-MAX**, et définie de la manière suivante:

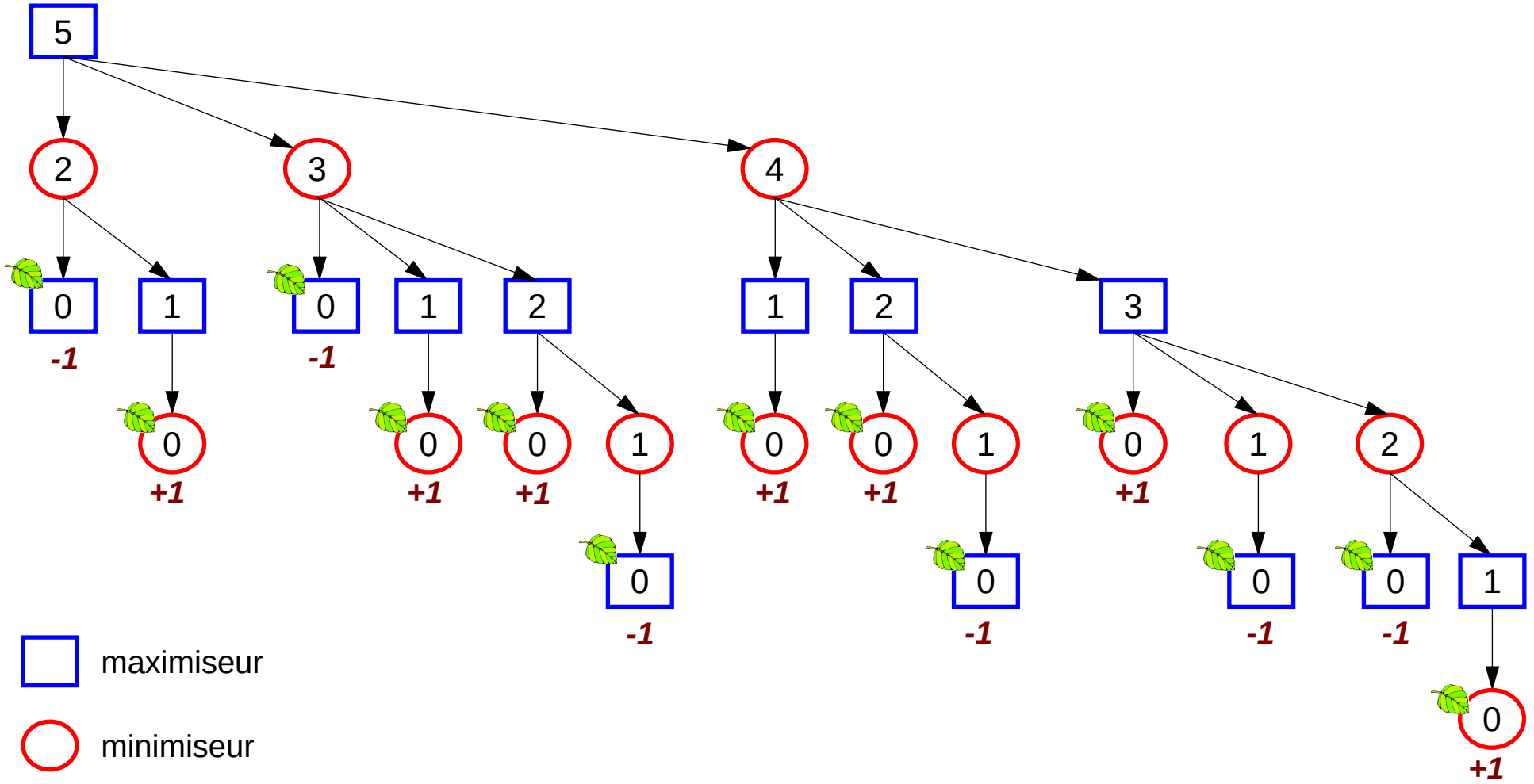
*A chaque sommet de l'arbre, on associe la plus favorable des valeurs des scores des état finaux que le joueur associé au sommet **est certain d'atteindre**, **quelle que soit** la stratégie utilisée pas son adversaire.*

Le joueur *maximiseur* étant fixé, la fonction MIN-MAX revient à:

1. associer à **chaque feuille** le score du joueur maximiseur;
2. associer, à chaque sommet non terminal **correspondant au joueur maximiseur**, la plus grande des valeurs des fils du sommet considéré (i.e. *la plus favorable situation atteignable*);
3. associer, à chaque sommet non terminal **correspondant au joueur minimiseur**, la plus petite des valeurs des fils du sommet considéré (i.e. *la plus défavorable situation atteignable*).

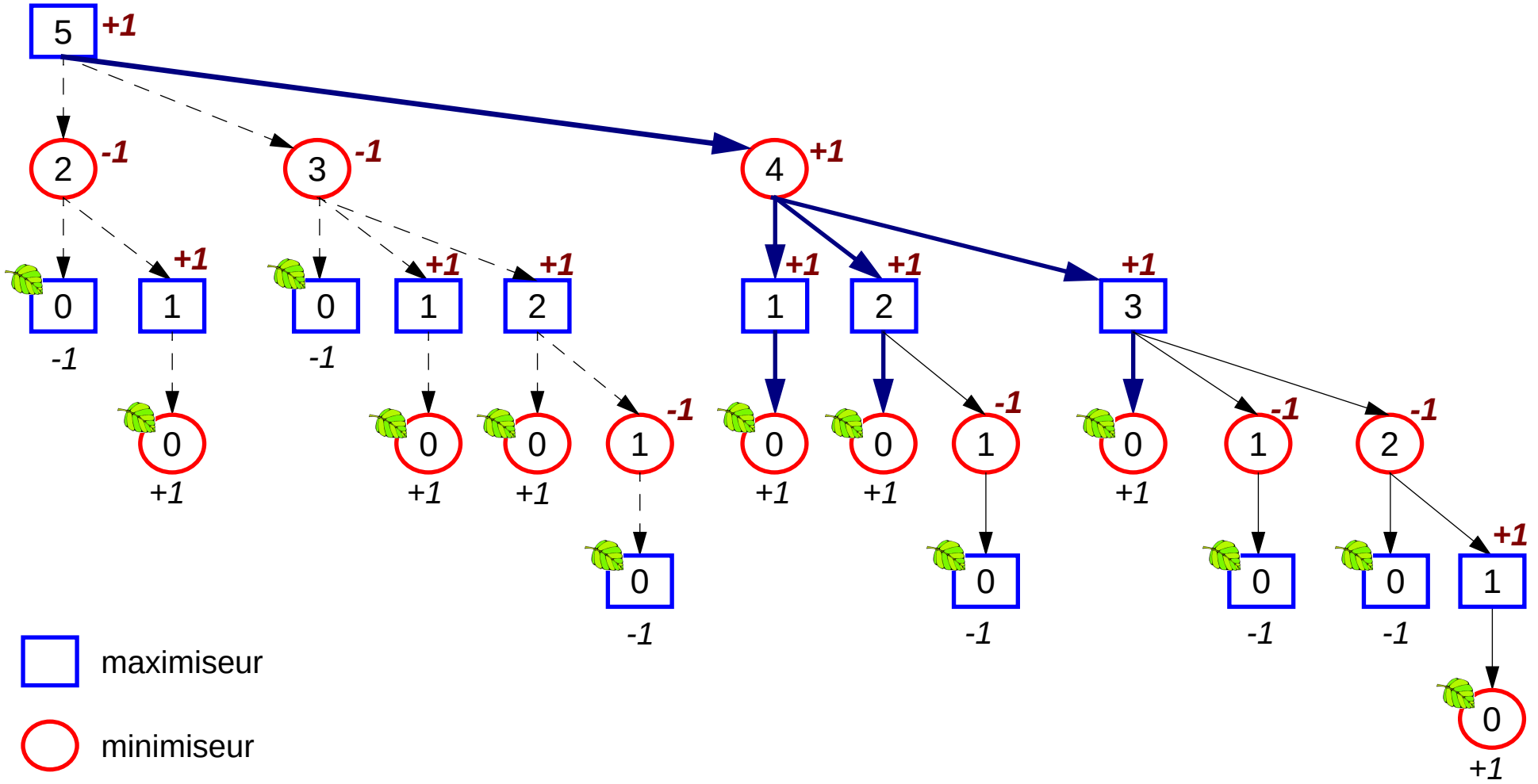
Evaluation MIN-MAX (1)

Exemple du calcul des valeurs de la fonction d'évaluation MIN-MAX pour l'arbre du jeu des allumettes de dimension 5 où le maximiseur commence:



Evaluation MIN-MAX (2)

En remontant alors les scores des feuilles vers les sommets parents, en suivant la définition opératoire de la fonction d'évaluation MIN-MAX, on obtient:



Stratégie MIN-MAX

La stratégie de jeu associée à la fonction d'évaluation MIN-MAX, dite stratégie MIN-MAX, est alors extrêmement simple:

- ☞ le joueur maximiseur choisit systématiquement le coup menant au sommet de valeur MIN-MAX maximale («le maximiseur maximise»);
- ☞ le joueur minimiseur choisit systématiquement le coup menant au sommet de valeur MIN-MAX minimale («le minimiseur minimise»).

La nature de cette stratégie est une autre justification des nom *minimiseur* et *maximiseur*.

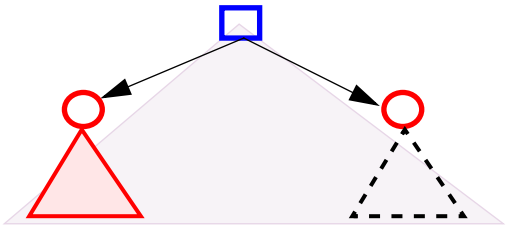
Dans le cas de notre exemple du jeu des (5) allumettes, on constate que le joueur qui commence est certain de pouvoir gagner (score sur la racine = `+maxScore`): il lui suffit de retirer 1 allumette; quelque soit le coup joué par le second joueur (le minimiseur), le maximiseur pourra alors gagner la partie, en choisissantle coup menant à un sommet de score `+maxScore`.

[c.f. arcs du schéma précédent]

Evaluation MIN-MAX exhaustive

Le problème posé par une implémentation exhaustive de la fonction d'évaluation MIN-MAX est que pour déterminer le score associé à un sommet donné, il faut parcourir l'ensemble de la portion de l'arbre des états dominée par ce sommet:

En particulier, pour calculer la valeur de la fonction d'évaluation MIN-MAX associée à la racine de l'arbre (nécessaire pour jouer le 1er coup), il faut explorer tout l'arbre des états, soit l'ensemble des configurations possibles du jeu, et donc **l'ensemble des parties possibles**.



Or, pour la majorité des jeux [intéressants], l'arbre des états est d'une taille suffisamment considérable pour que la mise en œuvre de l'implémentation exhaustive **ne soit pas réalisable**, le temps de calcul devenant prohibitif (par exemple, l'arbre du jeu des allumettes de dimension 20 contient plus de 250'000 sommets !)...

il faut donc trouver des moyens pour accélérer le calcul des valeurs de la fonction d'évaluation, c'est-à-dire trouver des algorithmes permettant de réaliser l'évaluation MIN-MAX **à partir d'explorations partielles** de l'arbre des états.

Nous allons considérer 2 méthodes:

- celle à **profondeur d'exploration limitée**;
- et celle **avec élagage** (*alpha-bêta*)

Exploration à profondeur limitée

Le principe de l'évaluation avec profondeur limitée repose sur l'idée suivante:

☞ l'implémentation simple de l'évaluation MIN-MAX entraîne l'exploration de tout l'arbre des états en raison du fait que la fonction d'évaluation ne peut affecter [directement] un score qu'aux situations terminales du jeu (les feuilles).

⇨ Par conséquent, si l'on disposait d'une fonction (appelée ***fonction d'évaluation***) permettant de **calculer le score d'une situation quelconque** (et pas seulement terminale), alors l'évaluation MIN-MAX d'une situation donnée ne nécessiterait que l'exploration de ses sommets fils (profondeur d'exploration limitée à 1 ⇒ fonction d'évaluation *exacte*)

Les jeux pour lesquels il existe une fonction d'évaluation *exacte* sont appelés
jeux calculables
et sont en fait **sans véritable intérêt**, puisqu'il suffit de connaître la fonction d'évaluation pour pouvoir jouer de manière optimale.

En fait, pour aucun des jeux intéressants on ne connaît de fonction d'évaluation *exacte*.

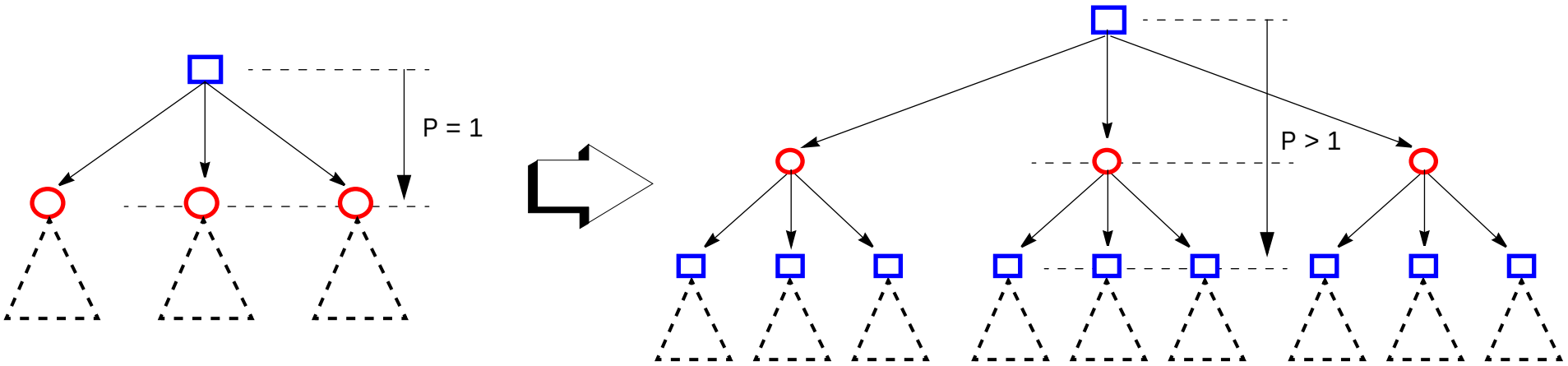
Comment peut-on alors mettre un œuvre une méthode à profondeur limitée ?

Fonctions d'évaluation heuristiques (1)

La solution est de rechercher **des fonctions d'évaluations *heuristiques***, i.e. des fonctions d'évaluation pour lesquelles on espère (sans pouvoir le démontrer) qu'elles produisent de **bonnes approximations** des scores des situations non terminales.

Bien entendu, la qualité de la fonction d'évaluation heuristique va très fortement conditionner la qualité de la stratégie de jeu qu'elle permet de mettre en œuvre.

Pour cette raison, on associe généralement à l'utilisation d'une fonction d'évaluation heuristique le fait **d'explorer** l'arbre des états **à une profondeur supérieure à 1**, comme ce serait le cas avec une fonction d'évaluation exacte.



fonction exacte *fonction heuristique*

Fonctions d'évaluation heuristiques (2)

Il est important de noter que la principale difficulté pour la mise en œuvre d'une évaluation à profondeur limitée est la recherche d'une ***bonne* fonction heuristique**.

En effet, la modification de l'algorithme implémentant la fonction MIN-MAX standard (exploration de la totalité de l'arbre), qui permet d'intégrer une fonction d'évaluation à profondeur d'exploration maximale donnée est extrêmement simple:

il suffit de *modifier la condition d'arrêt* de la définition récursive, de façon à intégrer le fait que, *lorsque la* ***profondeur maximale*** *d'exploration est* ***atteinte***, le *score de la situation courante est directement estimé par la* `fonction d'évaluation` (au lieu de la fonction de score).

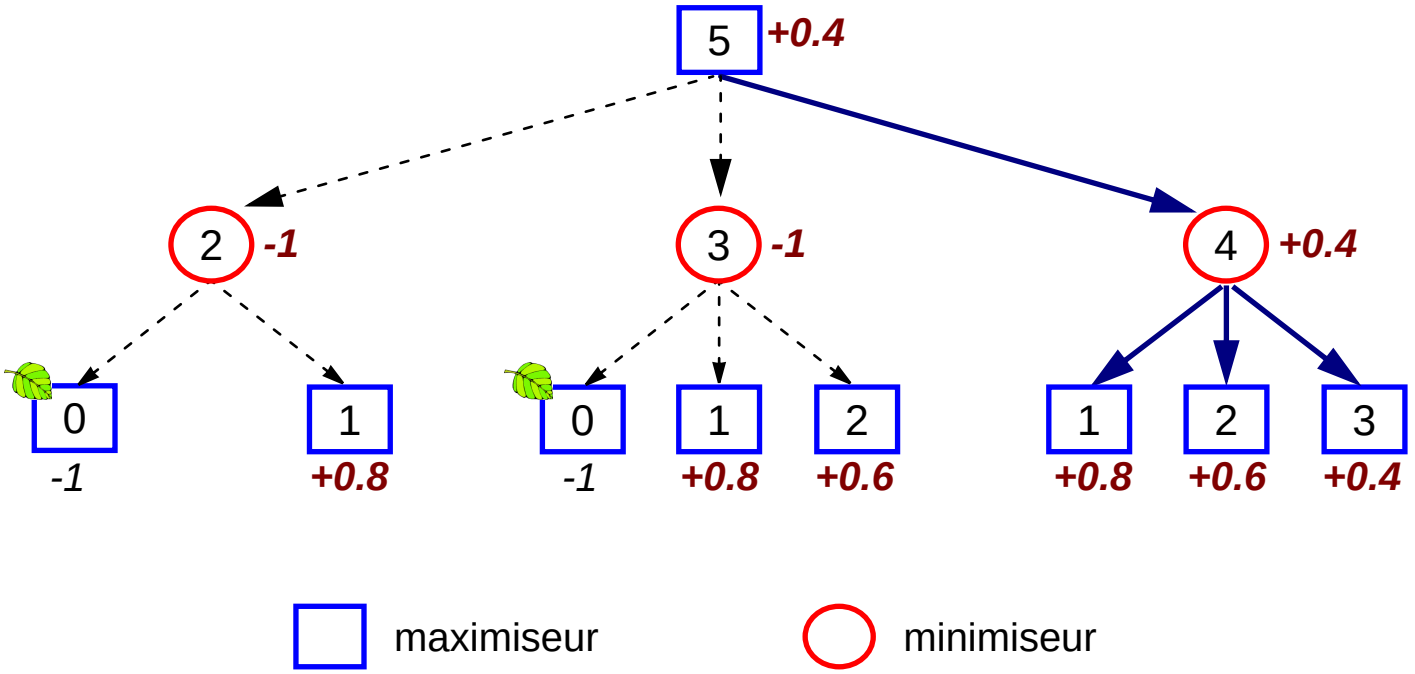
Pour le reste, l'algorithme ne change pas, et l'on remonte les scores estimés vers les sommets parents en suivant la même procédure que précedemment.

MIN-MAX avec profondeur limitée

Pour le jeu des allumettes, on pourrait, par exemple, prendre comme fonction d'évaluation:

```
+maxScore - (Nb allumettes restantes / Nb allumettes)
```

Avec une telle fonction, et une profondeur d'évaluation 3, on aura l'arbre suivant:



Remarque sur le jeu des allumettes

En fait, le jeu des allumettes est un jeu ***calculable***.

Il suffit de voir que:

- ⇨ pour les situations où il reste 1,2 ou 3 allumettes, le joueur qui doit jouer gagne;
- ⇨ par conséquent, pour la situation où il reste 4 allumettes, le joueur qui doit jouer perd (quoi qu'il joue, il place forcément son adversaire dans la situation précédente);
- ⇨ on peut alors dire que le joueur qui arrive dans la situation où il ne reste que 0 ou 4 allumettes à perdu. Donc, pour les situations où il reste 5,6 ou 7 allumettes, le joueur qui doit jouer gagne;
- ⇨ et ainsi de suite, on trouve que pour toutes les situations où il reste un multiple de 4 allumettes, le joueur qui doit jouer perd, sinon il gagne (toujours dans l'hypothèse que l'adversaire ne commet pas d'erreur).

Par conséquent, une telle fonction d'évaluation est une fonction *exacte*, et une exploration à profondeur 1 est suffisante.

Exploration avec élagage (1)

L'exploration avec *élagage* (*pruning*) est une autre méthode pour mettre en œuvre une exploration partielle. Une différence importante avec l'algorithme d'exploration à profondeur limitée est cependant qu'ici, **l'exploration mène au résultat exact**.

On a vu que le principe de base de l'évaluation MIN-MAX consiste à calculer la valeur du score de la situation associée à un sommet non terminal à partir des valeurs des scores des situations associées à ses sommets-fils.

La technique de l'évaluation avec élagage (aussi appelée **algorithme *alpha-bêta***) repose sur la généralisation de l'idée suivante:

si, en calculant les scores des situations associées aux fils d'un sommet donné de l'arbre d'état, **on trouve une situation optimale** [pour le joueur correspondant au sommet], **alors il n'est pas nécessaire d'explorer les autres fils du sommet**.

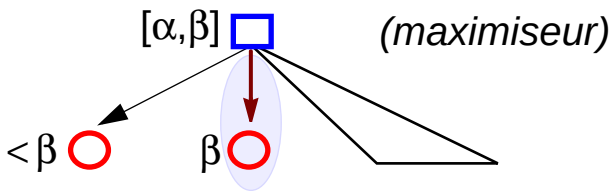
Exploration avec élagage (2)

Mais qu'est-ce qu'une **situation *optimale*** ?

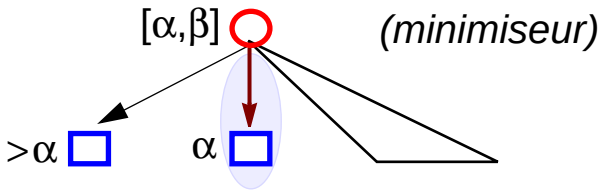
Une situation est optimale (par rapport à une situation-mère dont elle dérive) si le score qui lui est associé correspond au meilleur score que l'on peut associer à la situation-mère.

Si l'on désigne par ***alpha*** respectivement ***bêta***, les bornes minimales respectivement maximale pour le score pouvant être associé à une situation donnée, on a alors les conditions suivantes:

- une sitation associée à un sommet **fils d'un sommet du <u>maximiseur</u>** est optimale si son score est égal à la borne [maximale] **bêta** de la situation-mère;



- une situation associée à un sommet **fils d'un sommet du <u>minimiseur</u>** est optimale si son score est égal à la borne [minimale] **alpha** de la situation-mère.



Exploration avec élagage (3)

De part la définition de l'évaluation MIN-MAX, on a d'autre part:

- ☞ le score d'une situation associée à un sommet fils d'un sommet du **maximiseur** est une valeur possible pour la borne **alpha** de la situation-mère, d'autant meilleure que cette valeur est grande;

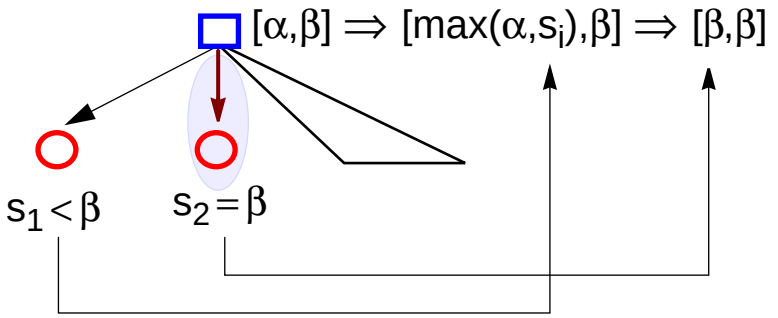
- ☞ le score d'une situation associée à un sommet fils d'un sommet du **minimiseur** est une valeur possible pour la borne **bêta** de la situation mère, d'autant meilleure que cette valeur est petite.

Le processus d'évaluation peut être initialisé avec les valeur `-maxScore`, `+maxScore` pour alpha, respectivement beta.

Exploration avec élagage (4)

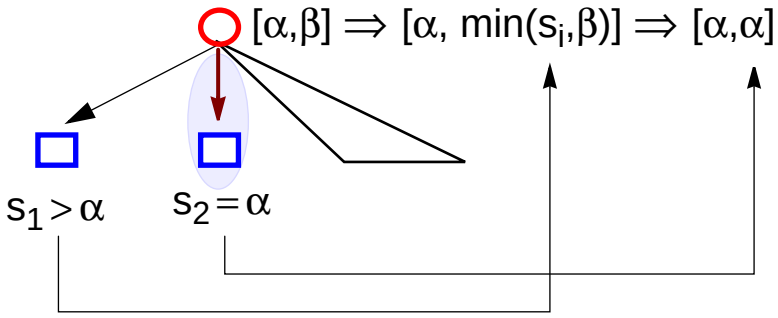
A partir de ces propriétés, on peut dériver la méthode suivante pour l'évaluation MIN-MAX avec élagage:

☞ lors de l'exploration des situations dérivées d'un sommet associé au **joueur <u>maximiseur</u>**:



- ⇨ chacun des scores s_i obtenus peut être utilisé pour mettre à jour la valeur *alpha* du sommet parent, par: $\alpha = \mathbf{max(\alpha, s_i)}$
- ⇨ si, en cours d'exploration, on trouve $\alpha = \beta$, alors le score du sommet maximiseur est **s = [β, β]** et l'on peut arrêter l'exploration des sommets fils (élagage)

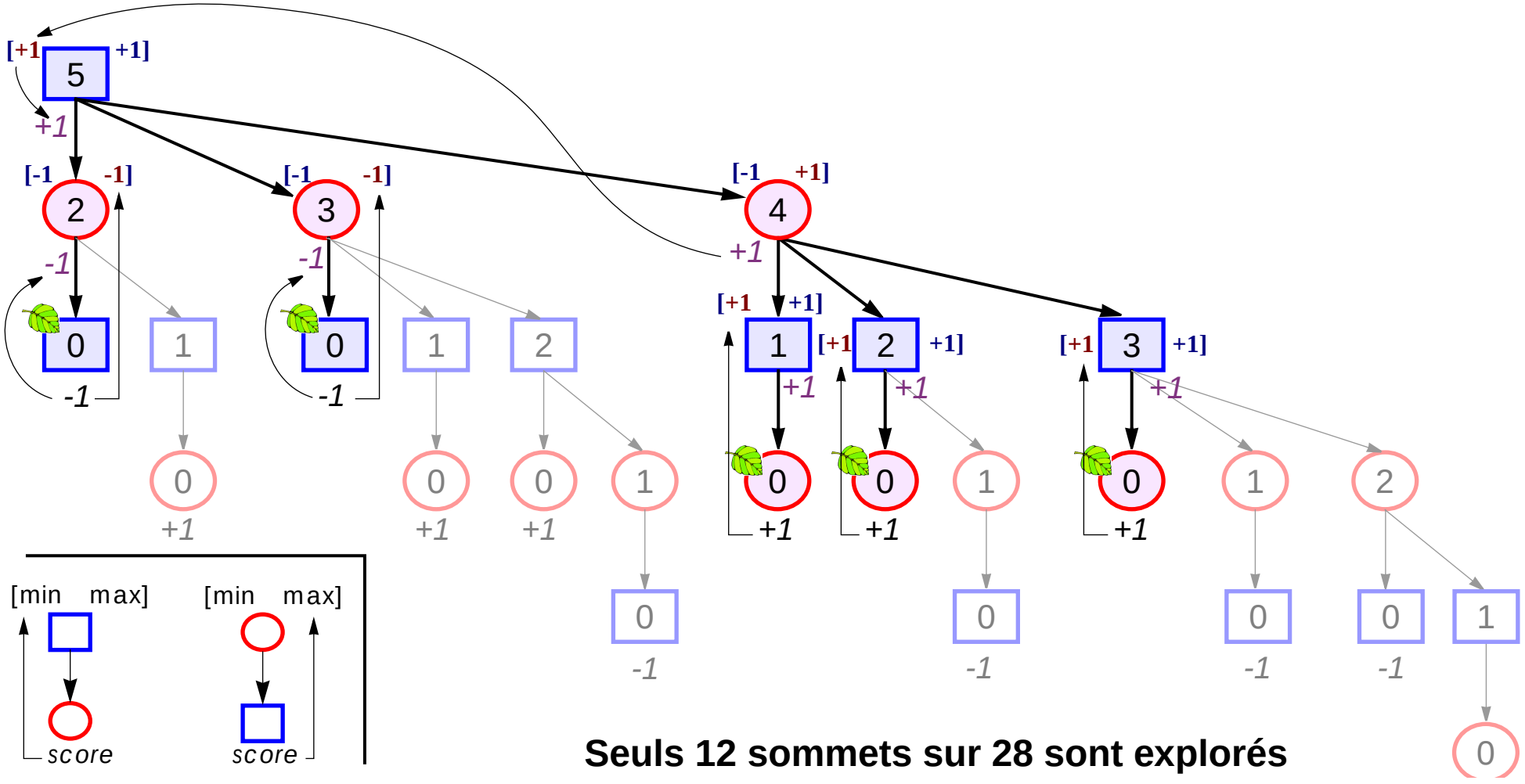
☞ lors de l'exploration des situations dérivées d'un sommet associé au **joueur <u>minimiseur</u>**:



- ⇨ chacun des scores s_i obtenus peut être utilisé pour mettre à jour la valeur *bêta* du sommet parent, par: $\beta = \mathbf{min(\beta, s_i)}$
- ⇨ si, en cours d'exploration, on trouve $\beta = \alpha$, alors le score du sommet maximiseur est **s = [α, α]** et l'on peut arrêter l'exploration des sommets fils (élagage)

Evaluation MIN-MAX avec élagage α-β

Exemple d'évaluation MIN-MAX avec élagage a-b pour le jeu des allumettes de dimension 5, dans le cas où le joueur maximiseur commence.

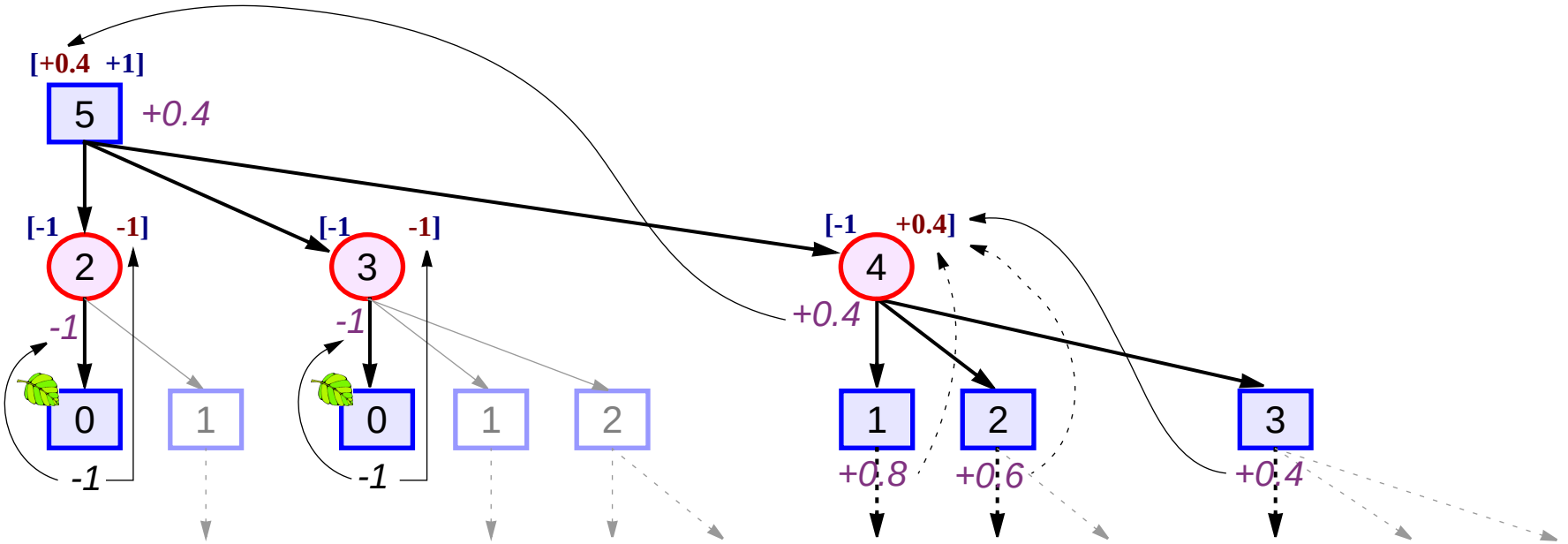


Seuls 12 sommets sur 28 sont explorés

Profondeur limitée et élagage

Les deux techniques (évaluation avec profondeur limitée et élagage) peuvent naturellement être combinées...

ici évaluation avec élagage et profondeur limitée à 2:



Seuls 9 sommets sur 28 sont explorés