

Introduction

à

l' intelligence artificielle

(algorithmes avec adversaires)

Antoine Cornuéjols

AgroParisTech

antoine.cornuejols@agroparistech.fr

Algorithmes avec adversaire(s) : plan

1. Introduction
2. Cas des jeux entièrement explorables
3. L' algorithme du MinMax
4. Amélioration : l' algorithme alpha-beta
5. Etat de l' art (Deep Blue & Co)

Types de jeux

	Déterministe	Non déterministe
Information complète	Échecs, Go	Backgammon
Information incomplète	Bataille navale	Poker, scrabble

Introduction

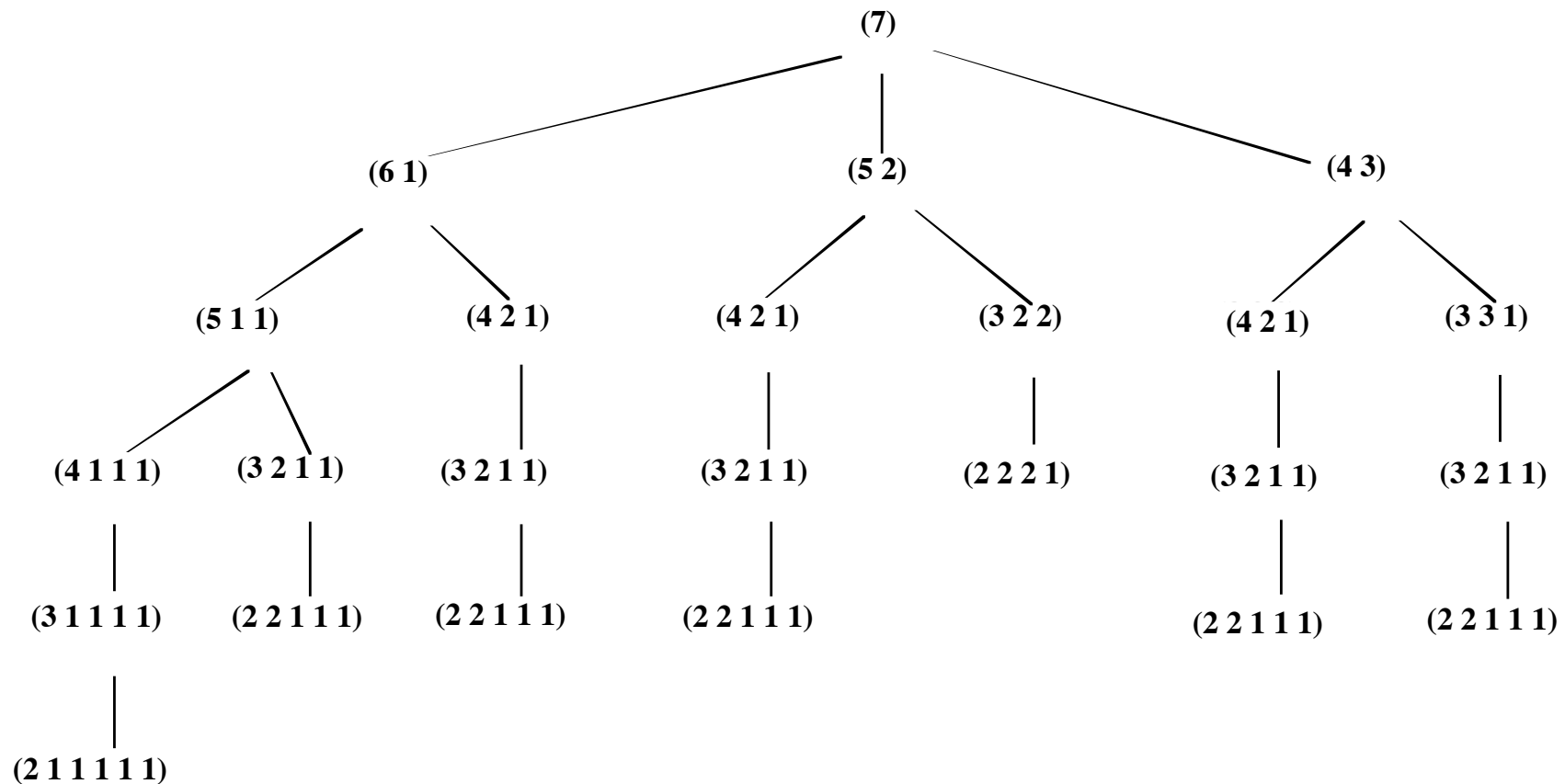
- Jeux à *information complète* (et un adversaire)
 - Pas de hasard
 - Chaque joueur connaît toutes les possibilités de jeu de l'adversaire (i.e. disposent de la même information)
 - Exclut le bridge, le backgammon, ...
 - **Rq** : Les techniques développées peuvent être relaxées à des jeux à plusieurs adversaires et information incomplète ou incertaine (ex: Météo, backgammon, ...)
 - Historique
 - Algorithme MinMax (Ernst Zermelo, 1912)
 - Morgenstein et von Neumann (1949) : théorie des jeux
 - Shannon (1950) « *Programming a computer for playing chess* ». Introduit toutes les idées majeures : état, fonction d'évaluation, exploration d'un arbre de recherche ...
 - CHECKER (~1960)
 - Deep Blue (1997 : bat Kasparov en match en 6 parties)
 - AlphaGo (2016) : bat le champion du monde

Introduction

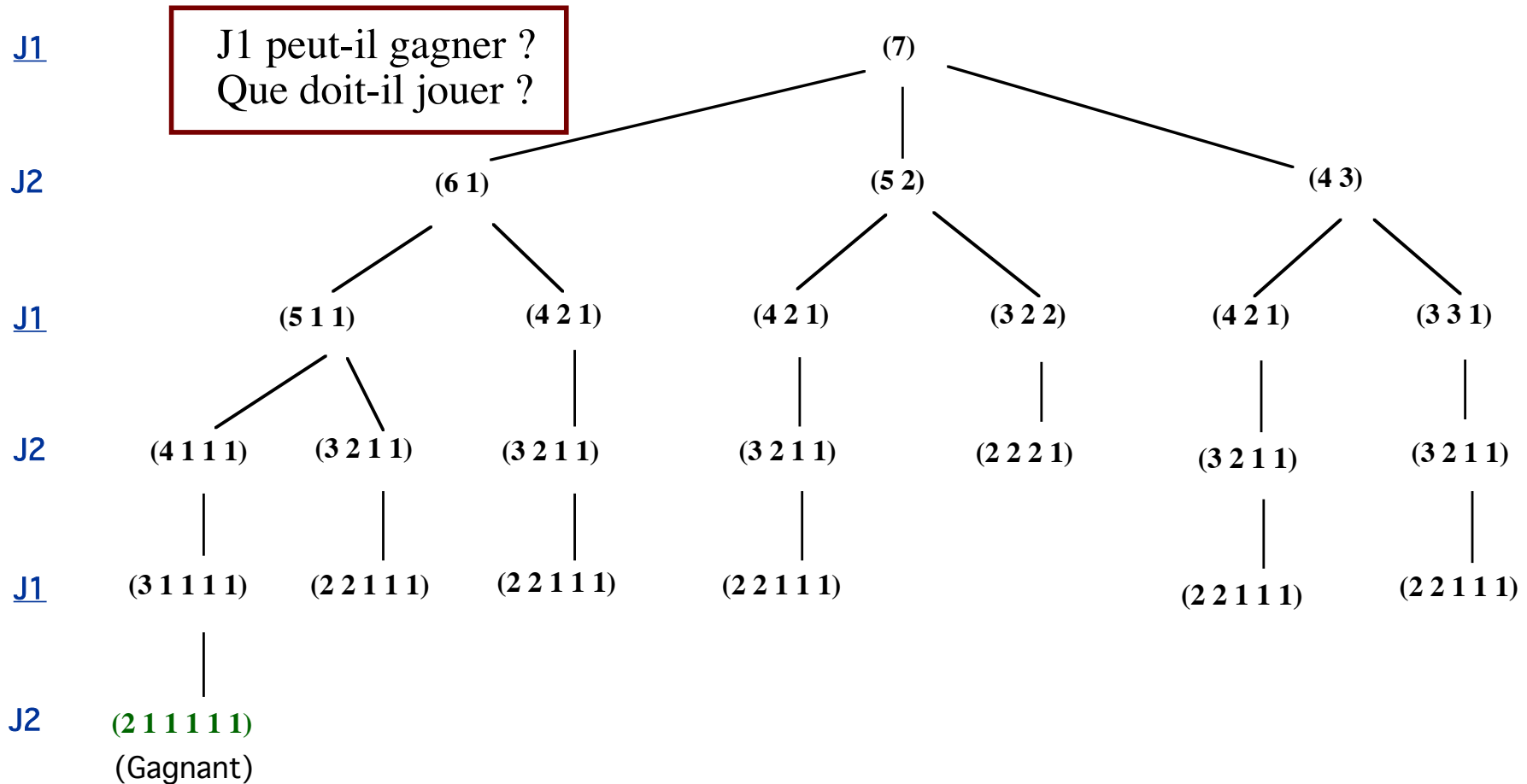
- Motivation :
 - Savoir décider en face de situations incertaines
 - Essentiellement, il s'agit de se préparer au pire en cherchant à minimiser le risque maximal

Jeux à exploration complète. Ex : le Grundy

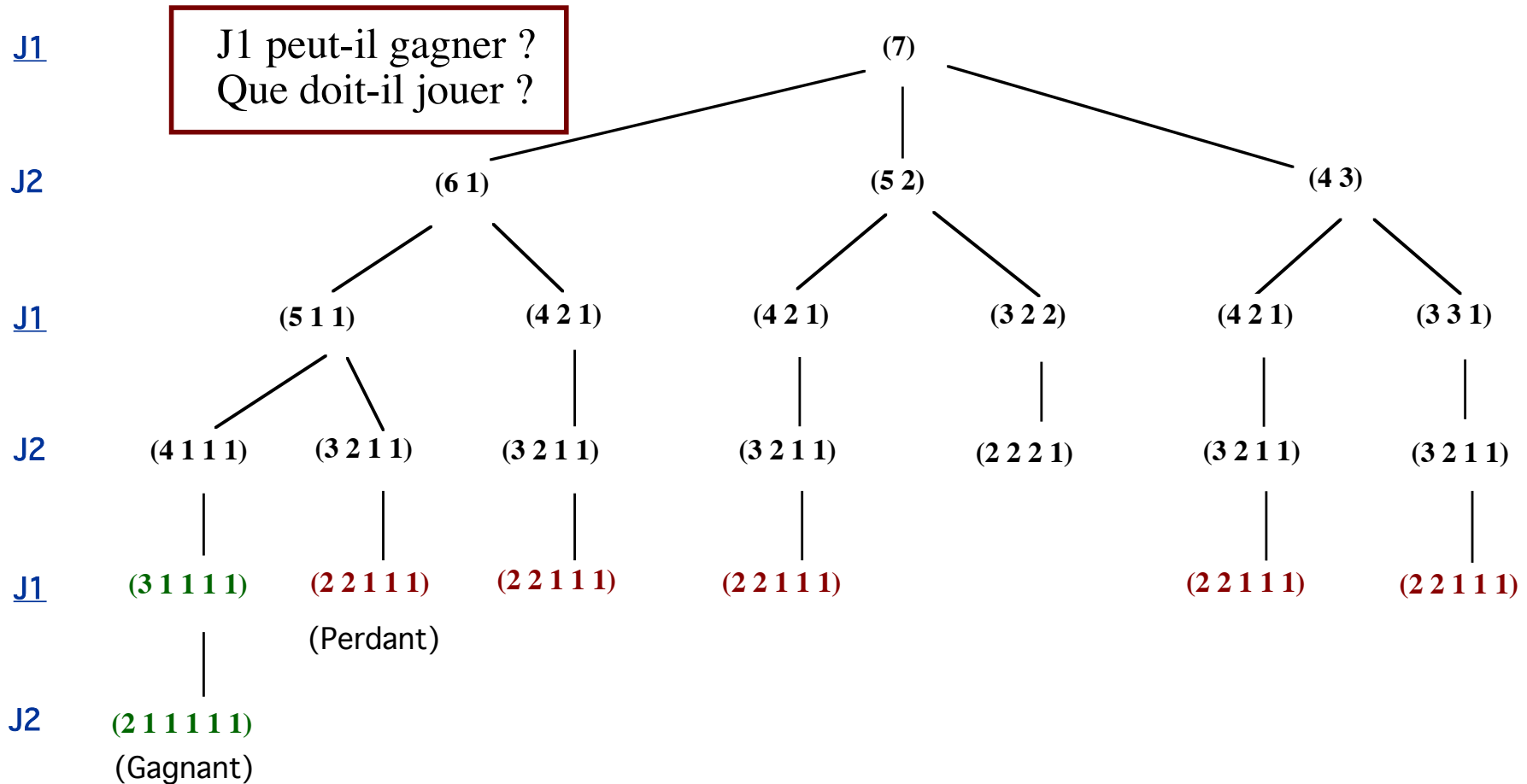
- On dispose **initialement** de 7 pièces en une pile
- Chaque joueur à tour de rôle doit **diviser une pile en deux piles inégales**
- Si c'est impossible, le joueur a perdu



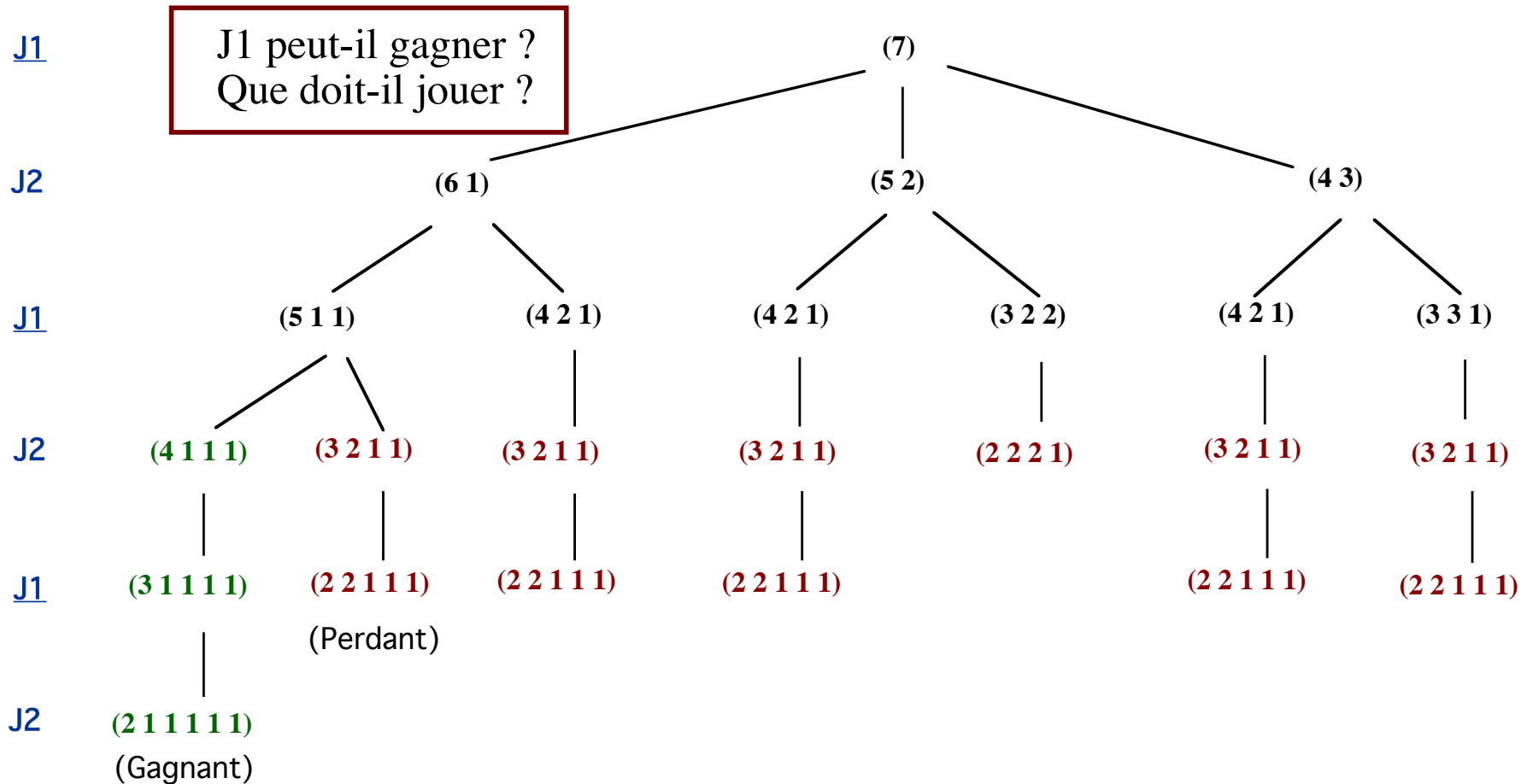
Jeux à exploration complète. Ex : le grundy



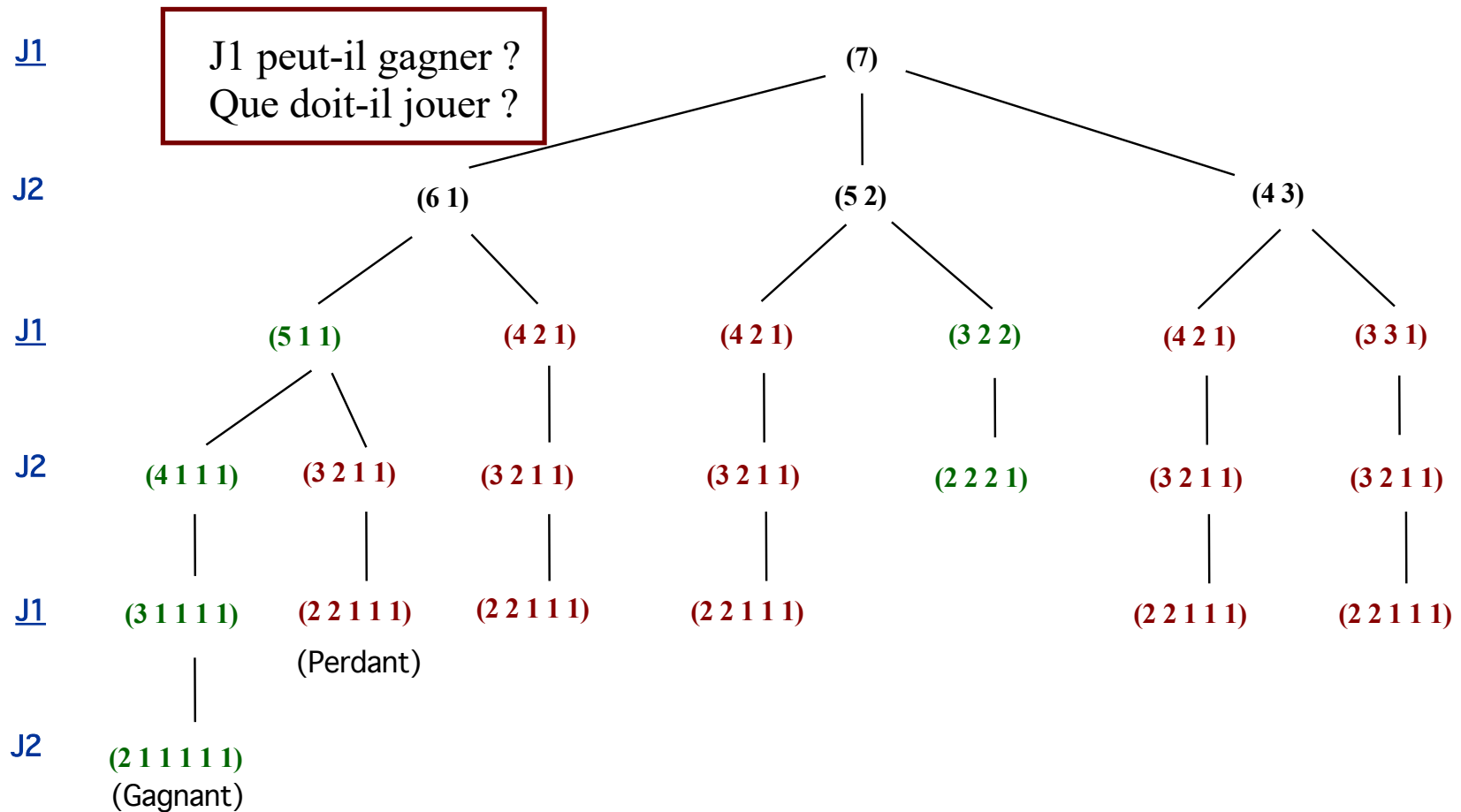
Jeux à exploration complète. Ex : le Grundy



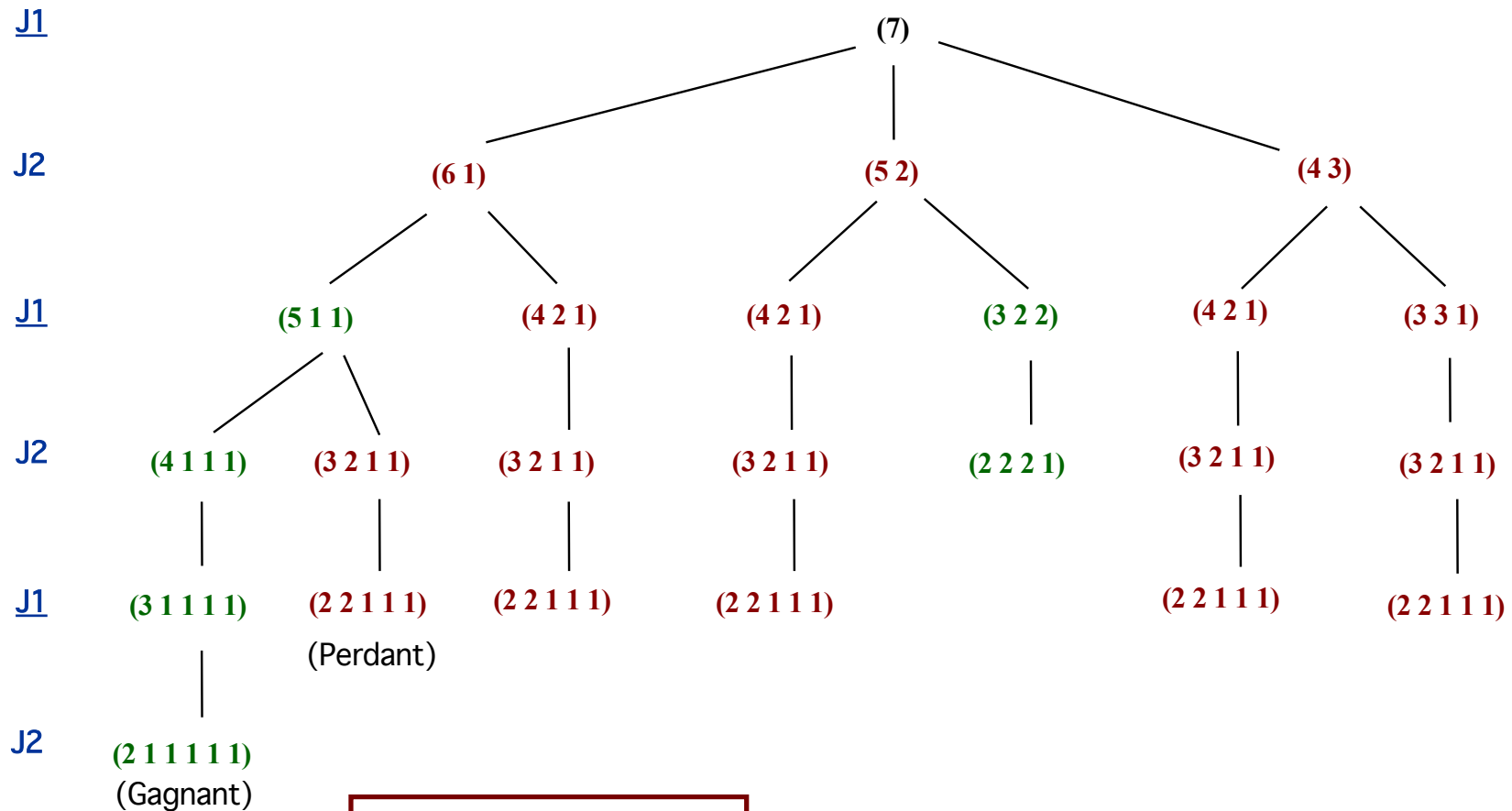
Jeux à exploration complète. Ex : le Grundy



Jeux à exploration complète. Ex : le grundy



Jeux à exploration complète. Ex : le grundy



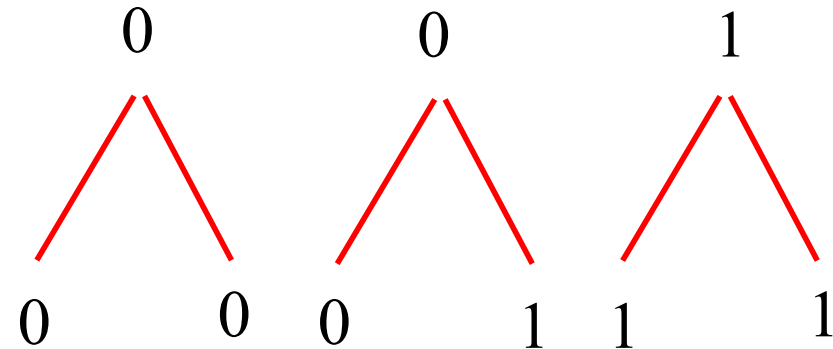
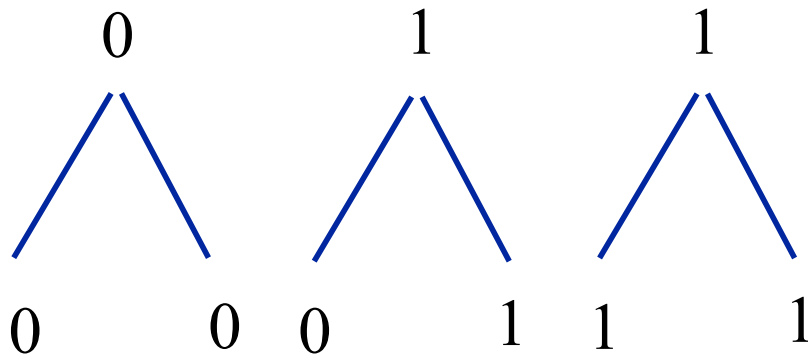
J1 peut-il gagner ?
 Que doit-il jouer ?

➔
 J1 ne peut pas gagner

Jeux à exploration complète

Algorithme de remontée des étiquettes

- Perdant = 0 (faux)
- Gagnant = 1 (vrai)
- Si **J1** : fonction **OU** des successeurs
- Si **J2** : fonction **ET** des successeurs



L' algorithme MinMax

- La plupart des jeux (et des situations réelles) ne sont pas complètement explorables
- Idée :
 - On explore aussi loin que possible en avant
 - On étiquette les feuilles avec une évaluation numérique de la position (pour le joueur J1)
 - On remonte ces étiquettes numériques jusqu'à la racine pour savoir quel coup jouer

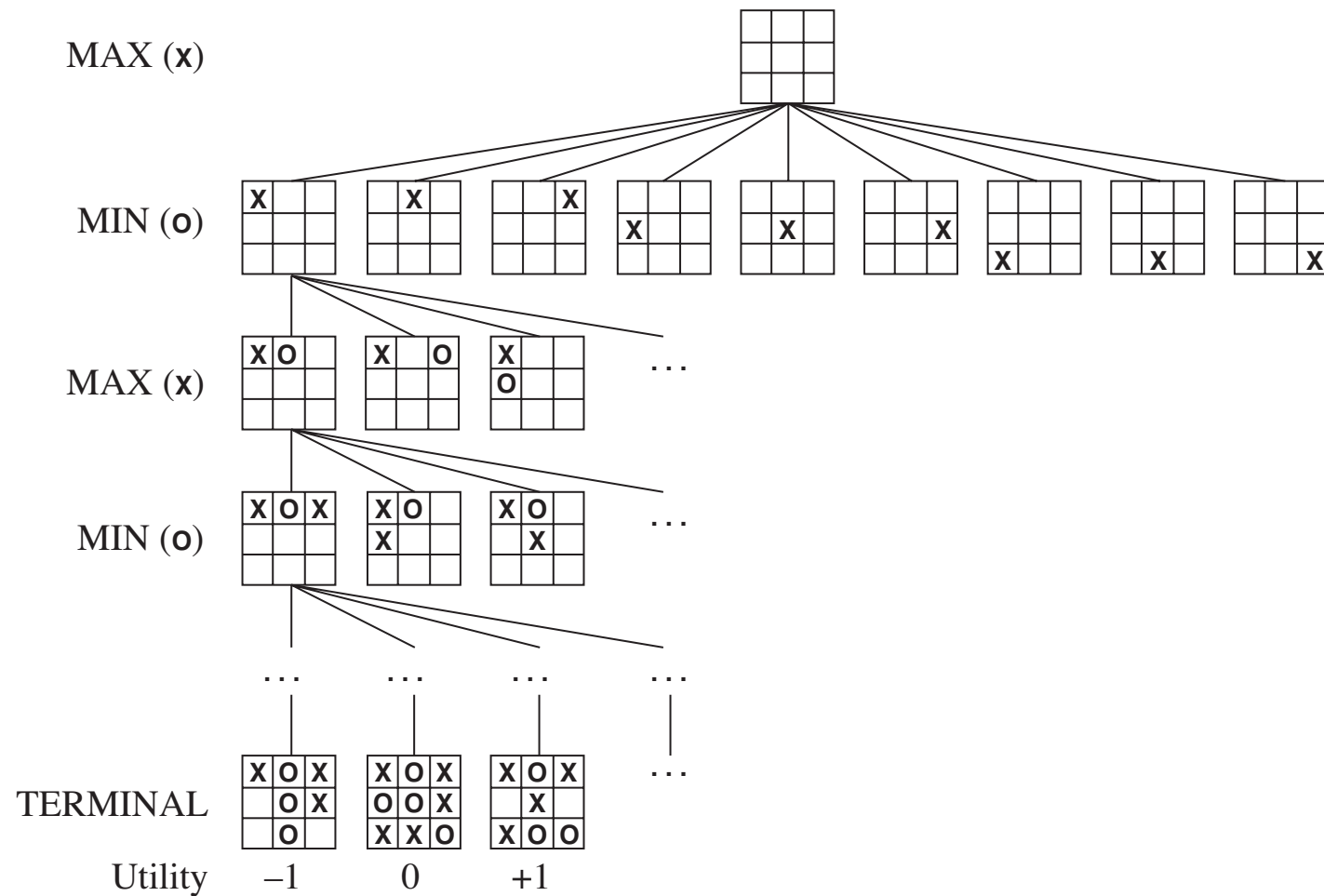
↳ Algorithme MinMax

L' algorithme MinMax

- Algorithme de remontée des étiquettes numériques
 - Si $J1$: Max des valeurs des successeurs
 - Si $J2$: Min des valeurs des successeurs

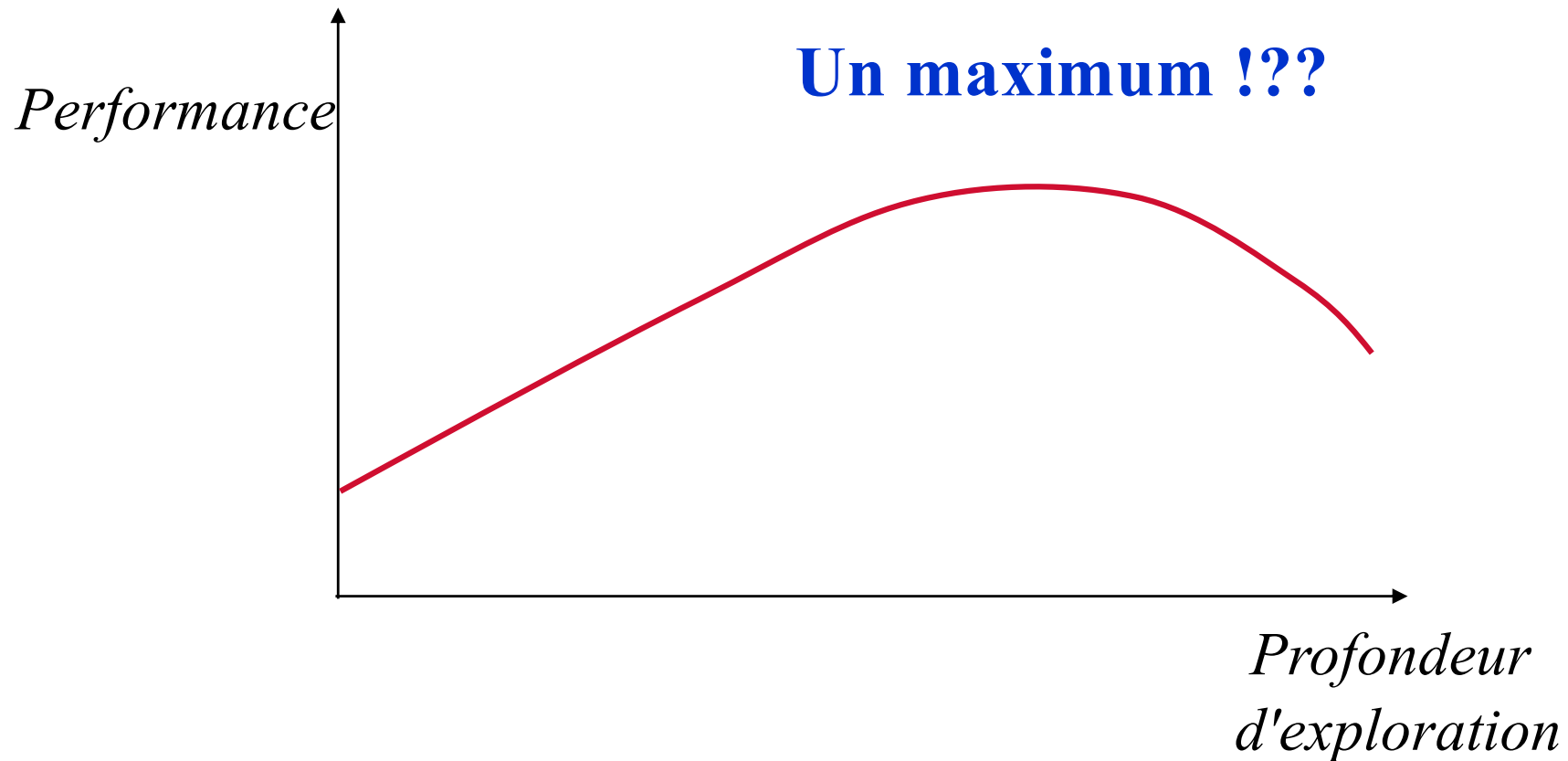
L' algorithme MinMax

- Tic-Tac-Toe



L'algorithm MinMax. Des jeux pathologiques ?!

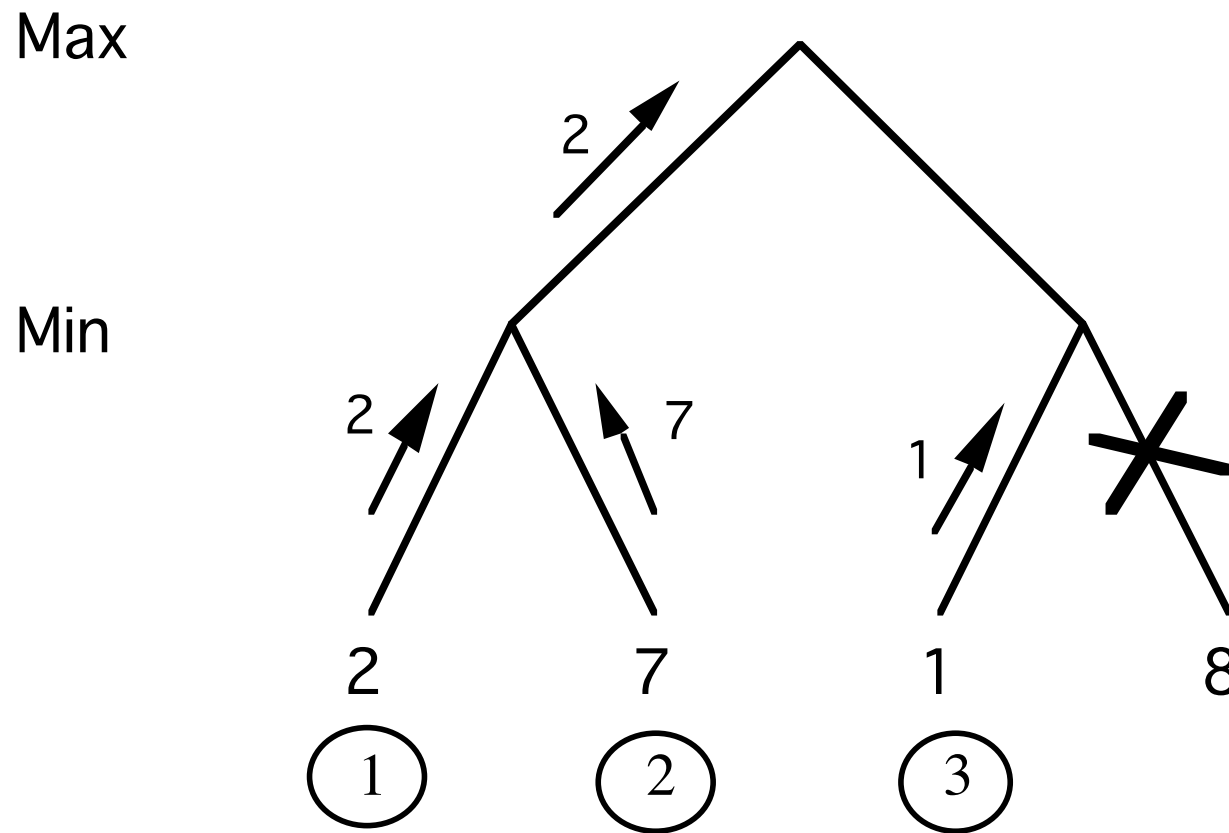
- Analyse des raisons du succès de MinMax (et de ses limites)
- Un comportement bizarre



→ Importance de la fonction d'évaluation

Explorer moins pour faire autant ?

- On peut économiser des évaluations de feuille (coûteuses en temps)



L' algorithme alpha-beta

Maxmin (nœud, **alpha**, **beta**) → **alpha**

si feuille(nœud) : évalue(nœud) → **alpha**

sinon : parcours séquentiel des successeurs de nœud avec :

max(**alpha**, **Minmax**(succ(nœud), **beta**) → **alpha**

et si **alpha** ≥ **beta** sortir **alpha** (cas d'une *alpha-coupure*)

Minmax (nœud, **alpha**, **beta**) → **beta**

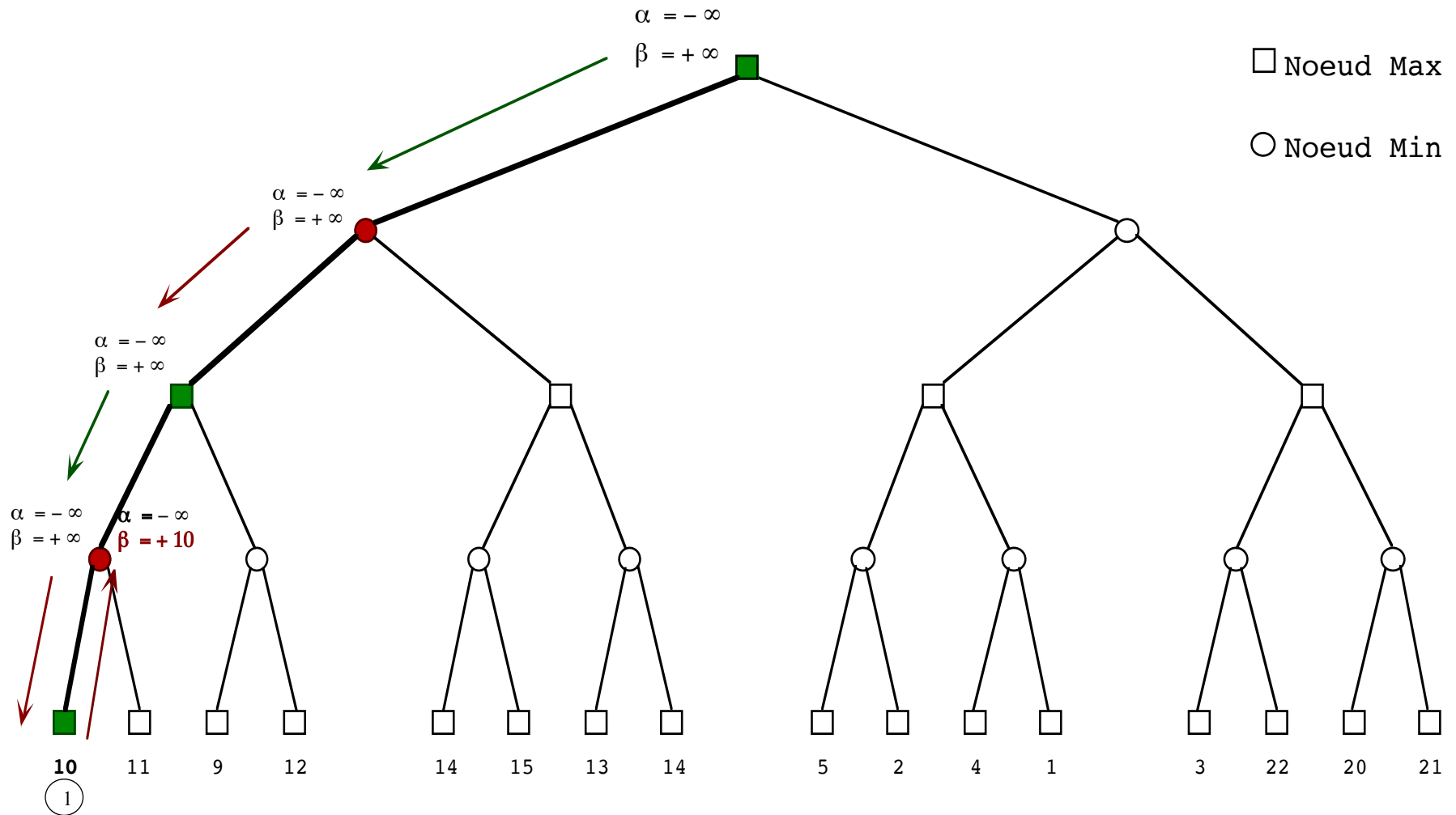
si feuille(nœud) : évalue(nœud) → **beta**

sinon : parcours séquentiel des successeurs de nœud avec :

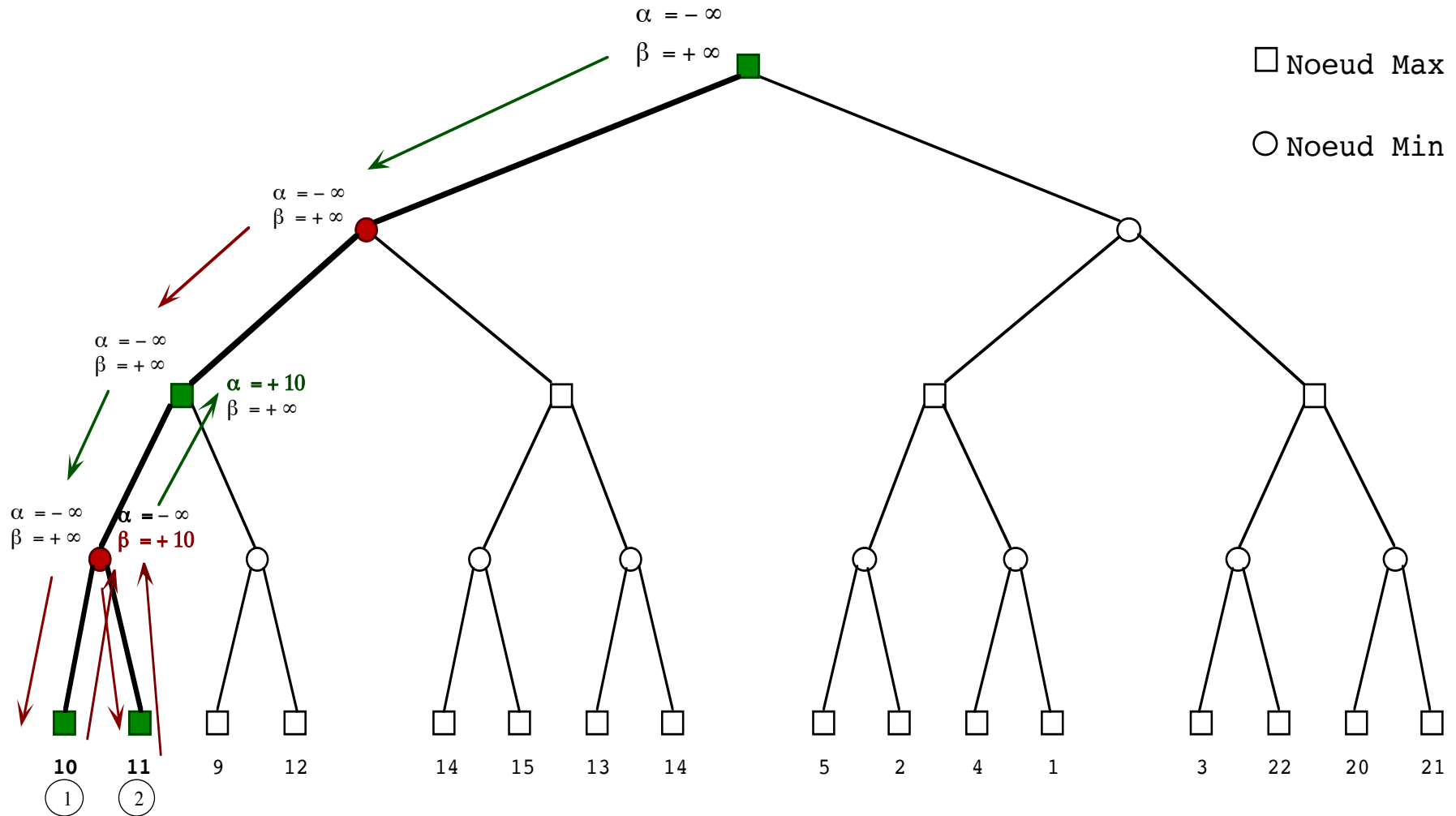
min(**beta**, **Maxmin**(succ(nœud), **alpha**) → **beta**

et si **beta** ≤ **alpha** sortir **beta** (cas d'une *beta-coupure*)

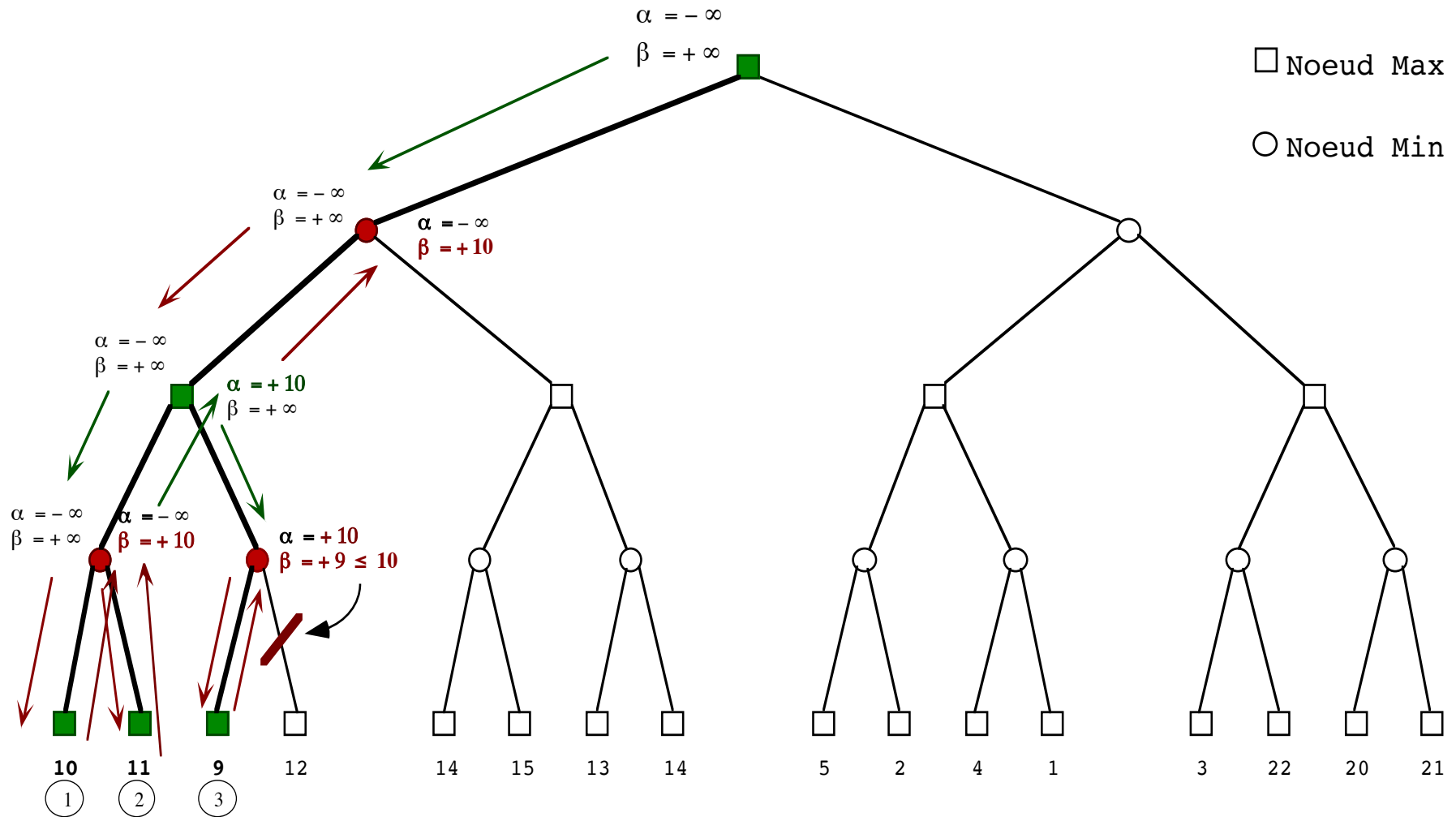
L' algorithme alpha-beta : Illustration (1)



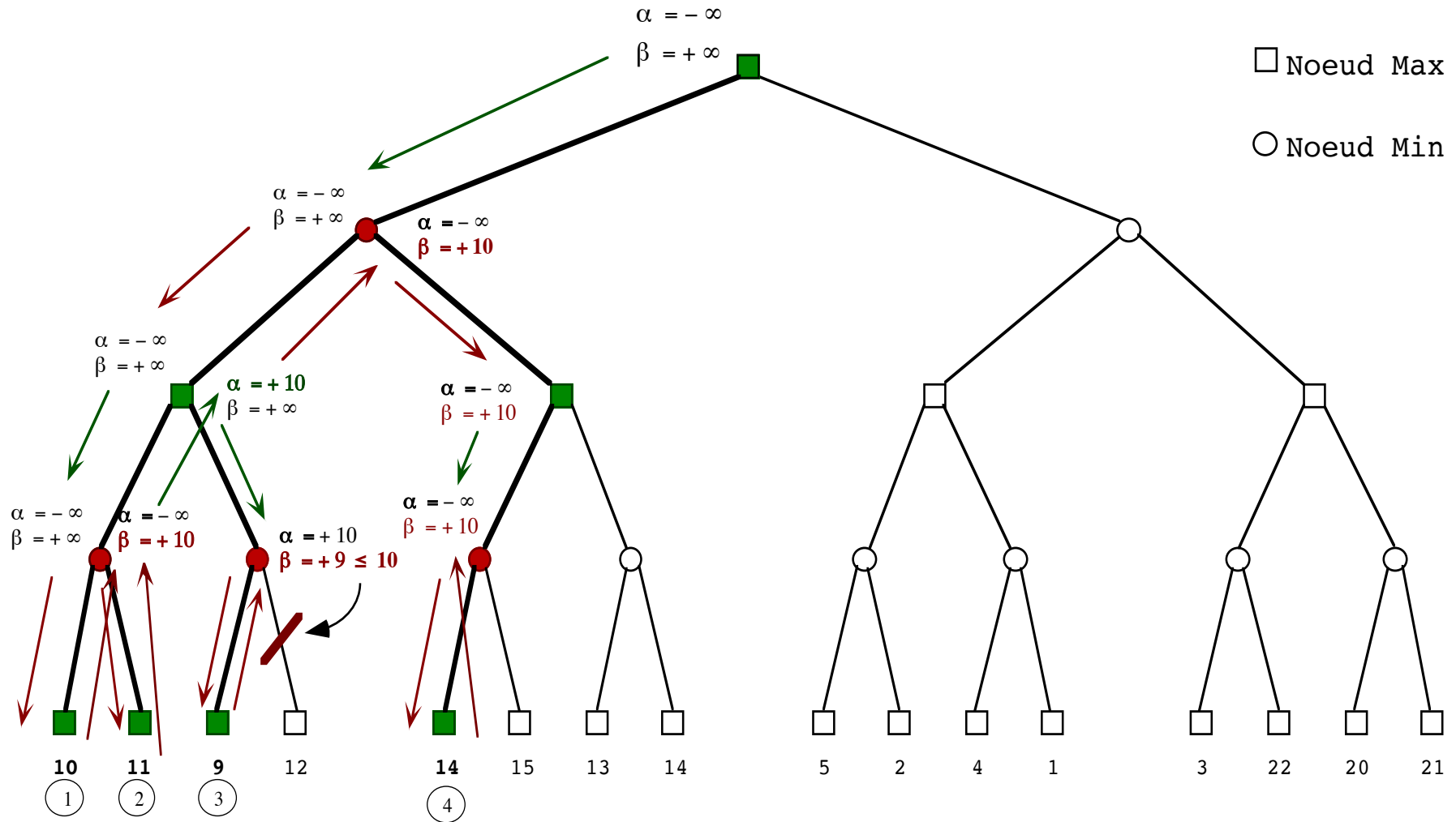
L' algorithme alpha-beta : Illustration (2)



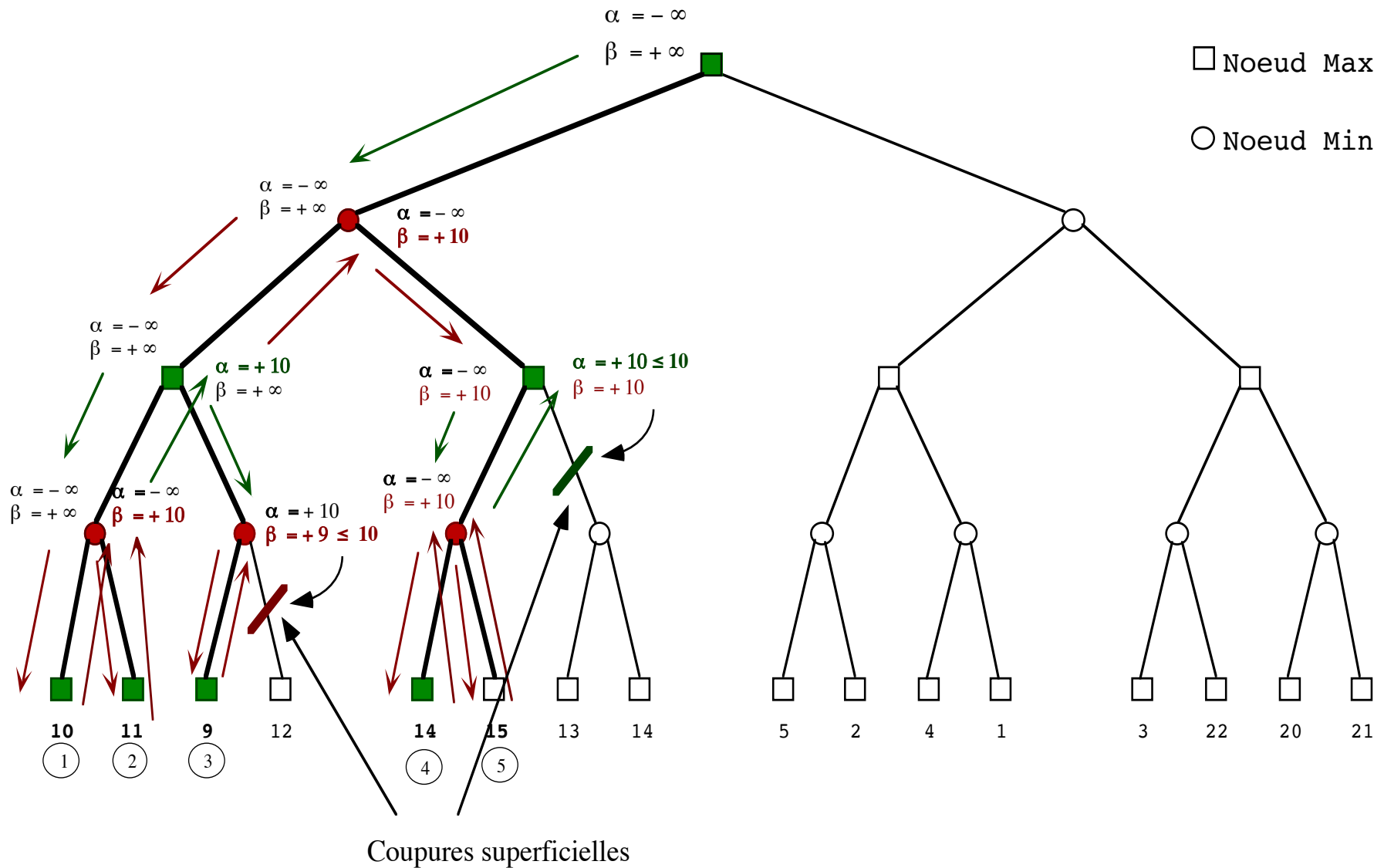
L' algorithme alpha-beta : Illustration (3)



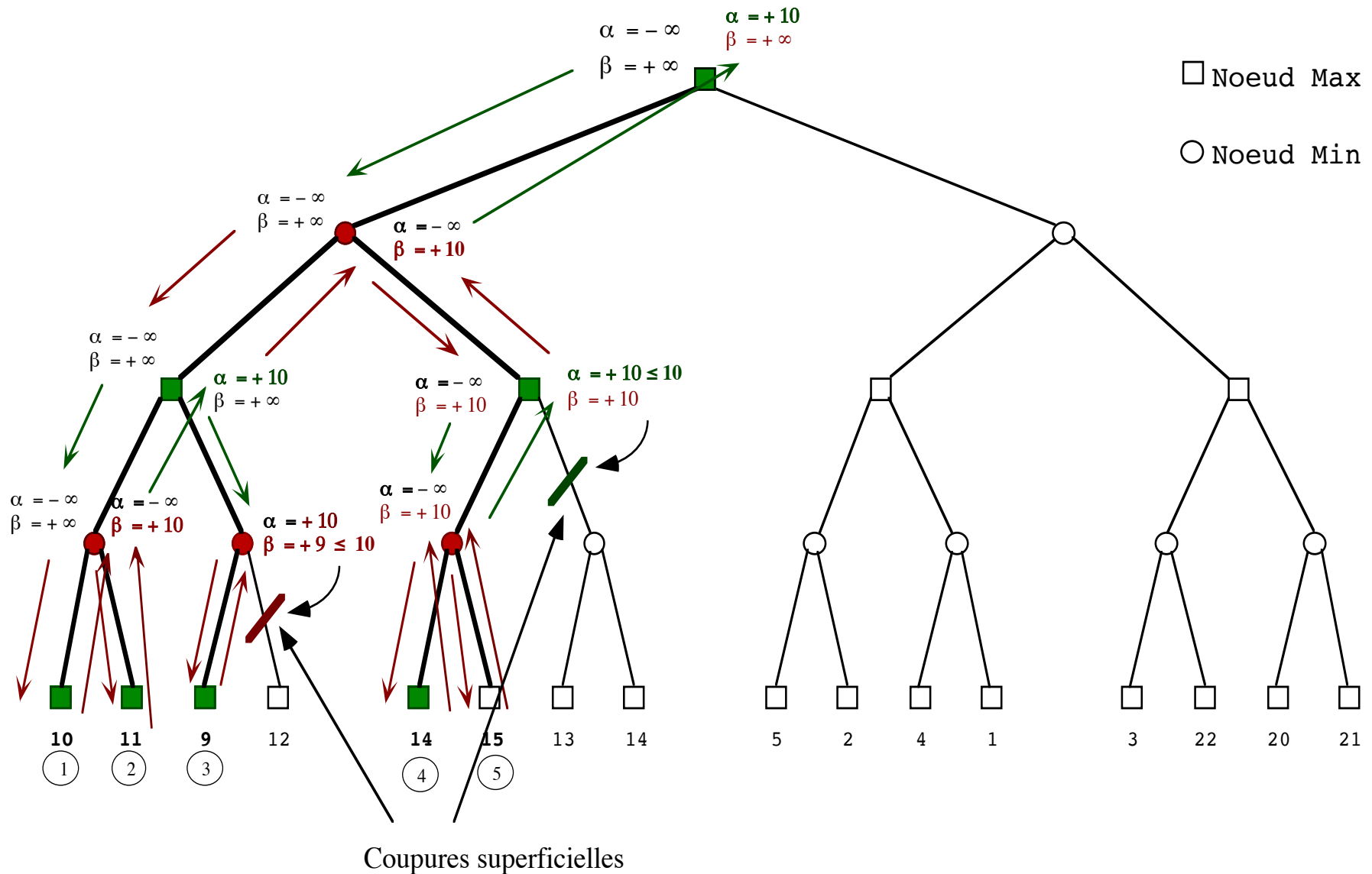
L' algorithme alpha-beta : Illustration (4)



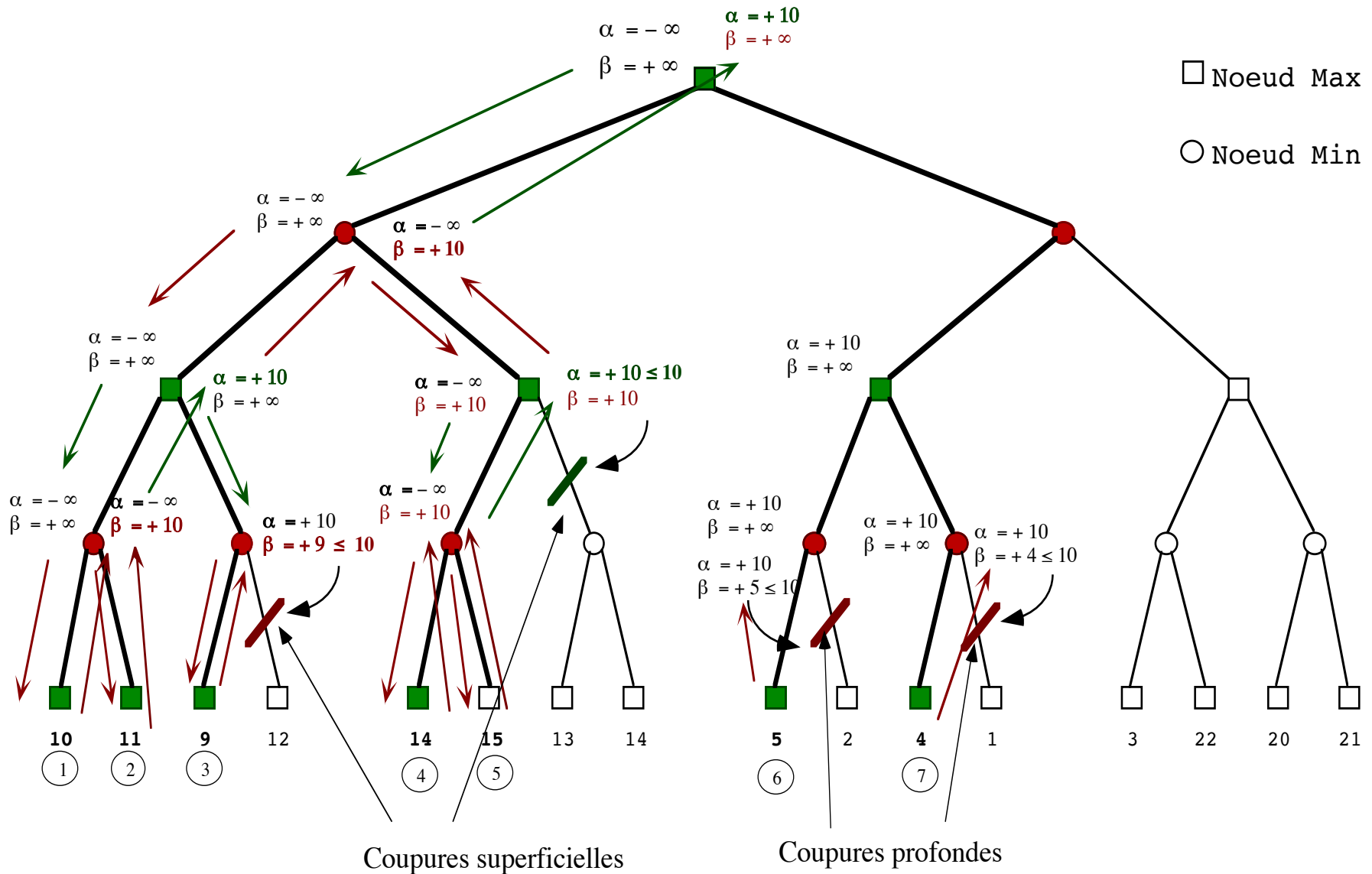
L' algorithme alpha-beta : Illustration (5)



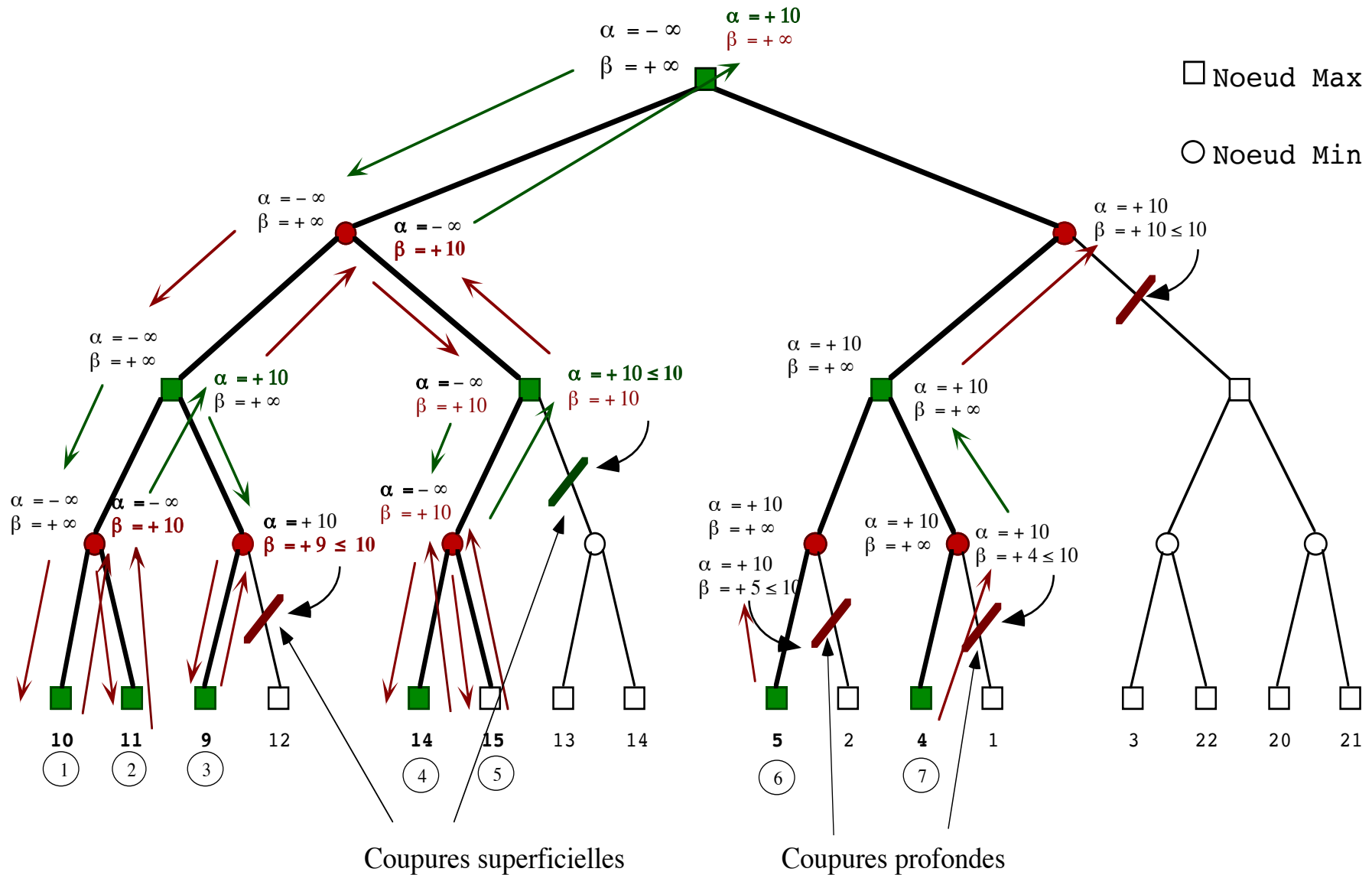
L' algorithme alpha-beta : Illustration (6)



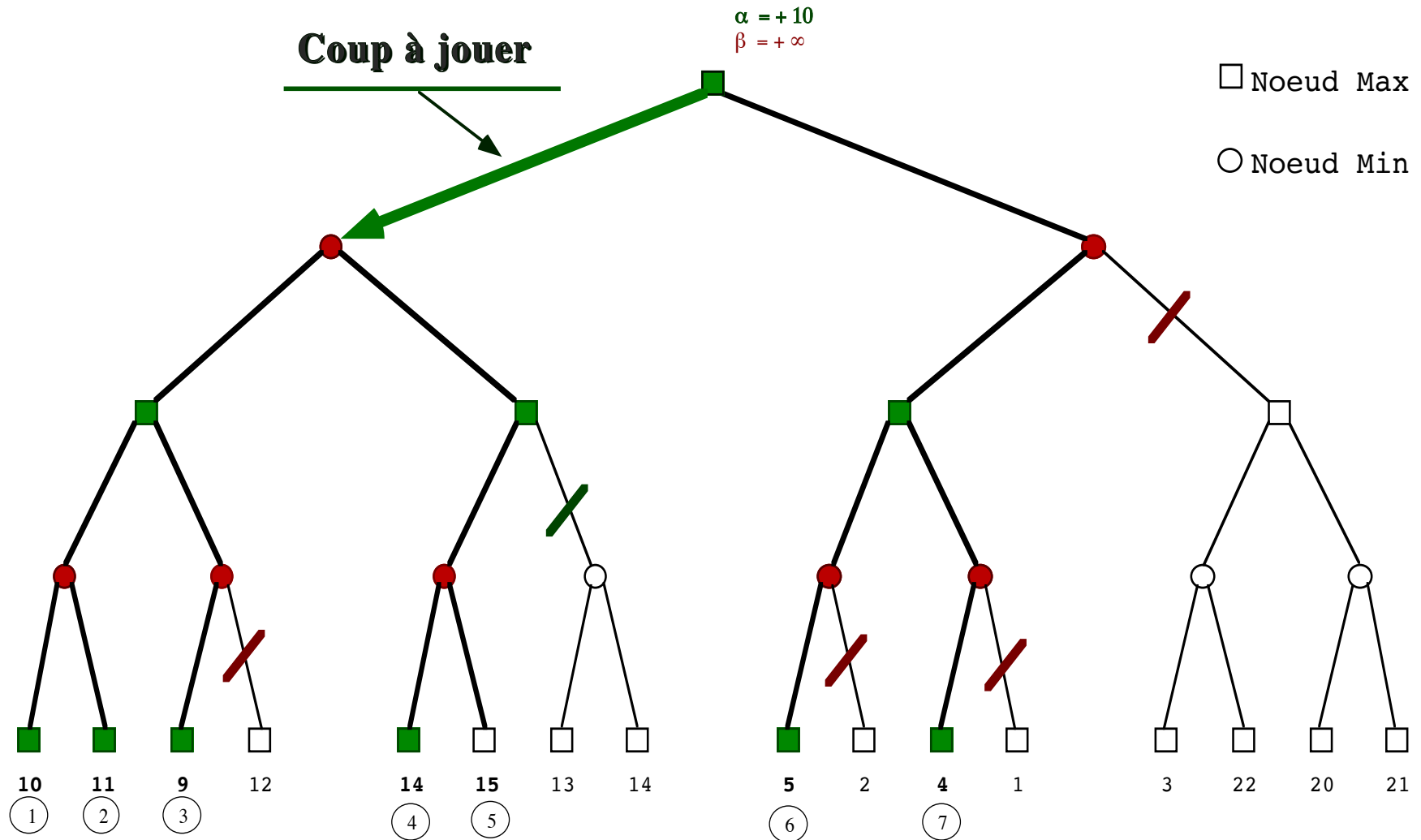
L' algorithme alpha-beta : Illustration (7)



L' algorithme alpha-beta : Illustration (8)



L' algorithme alpha-beta : Illustration (9)



L' algorithme alpha-beta : Performances

↓ Le résultat est le même que celui de l'algorithme MinMax

○ Mais avec quel avantage en performance ?

□ Pire cas : $O(b^d)$

□ Cas le plus favorable : $O(b^{d/2})$

○ facteur de branchement effectif = $b^{1/2}$

○ Peut aller deux fois plus loin que Minmax pour la même puissance de calcul

□ Cas asymptotique : $O((b/\log b)^d)$

○ Pour $b > 1000$

□ Cas moyen : $O(b^{3d/4})$

○ Peut aller 4/3 fois plus loin que Minmax pour la même puissance de calcul (pas si mal !!)

$$N_d = \begin{cases} 2b^{d/2} - 1 & \text{pour } d \text{ pair} \\ b^{(d+1)/2} + b^{(d+1)/2} & \text{pour } d \text{ impair} \end{cases}$$

L' algorithme alpha-beta : Performances

Comment ordonner les nœuds pour s'approcher du cas optimal ?

- Heuristiques

- Exemple : **d'abord examiner les captures, puis les menaces, puis les mouvements en avant, puis les retraites, ...**

- Recherche en profondeur itérative

- **Faire un alpha-beta à profondeur 1** : utiliser le résultat pour ordonner les nœuds
- **Faire un alpha-beta à profondeur 2 sur l'arbre réordonné à l'étape précédente** : utiliser le résultat pour ordonner les nœuds
- Idem à profondeur n **jusqu'à épuisement des ressources de calcul allouées**

Les limites

- Effet d'horizon
 - Retarder un désastre inévitable en le repoussant au-delà de l'horizon (**parfois au prix de pertes supplémentaires (ex : perdre des pions pour éviter la perte inéluctable de la dame)**)
 - Remèdes
 - Attendre des *positions stables* ou à l'équilibre (**ex: sans échecs ou échanges forcés**)
 - *Recherche secondaire* (refaire une petite exploration au-delà de la branche sélectionnée pour vérifier)
- Pas de plan stratégique
- Pas de tentative de piège (même si possible au prix d'une position éventuellement à peine pire)
- On suppose que l'adversaire a la même fonction d'évaluation
 - Remède : apprentissage

Heuristiques d'amélioration

- Augmentation des moyens calcul
 - Source très importante des progrès récents
- Bibliothèque d'ouvertures
- Exploration complète en fin de partie
- Apprendre la fonction d'évaluation de l'adversaire (en mémorisant ses parties et en apprenant les conséquences)
 - Permet d'accélérer l'évaluation et de s'adapter à l'adversaire
- Heuristique du coup meurtrier
 - Si le coup de l'adversaire min_1 détruit la position courante au niveau n , il est probable qu'il détruit aussi les positions issues de la position courante aux niveaux $> n$, donc min_1 est à examiner en priorité dans les branches plus bas
- Utilisation du temps de réflexion de l'adversaire
- ... (beaucoup d'autres techniques plus ou moins ad hoc)

Le cas du jeu Othello

- Ouverture
 - Théorie pauvre sur Othello (contrairement aux échecs)
 - On mémorise position --> coup à jouer (d'après des études statistiques sur les 6 premiers coups)
- Finale
 - Le nombre de coups est précisément connu (exactement 60 demi-coups)
 - On lance une recherche exhaustive en fct de la puissance de calcul disponible (les plus gros ordinateurs démarrent 17 coups avant la fin (1998))
 - Rouleau compresseur de la force brute

Le cas du jeu Othello

- Milieu de partie
 - Avoir une bonne fonction d'évaluation
 - Evaluation de la mobilité
 - Combien de degrés de liberté ? (il vaut mieux peu au début et plus à la fin)
 - Evaluation statique
 - Poids associé à chaque case

120	-20	20	5	5	20	-20	120
-20	-40	-5	-5	-5	-5	-40	-20
20	-5	15	3	3	15	-5	20
5	-5	3	3	3	3	-5	5
5	-5	3	3	3	3	-5	5
20	-5	15	3	3	15	-5	20
-20	-40	-5	-5	-5	-5	-40	-20
120	-20	20	5	5	20	-20	120

Le cas du jeu Othello

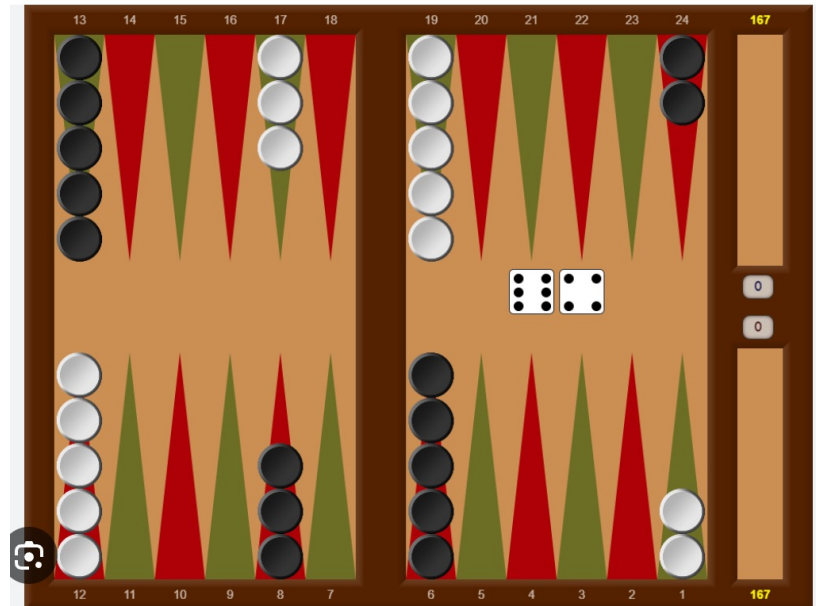
- Mais il faut modifier cette fonction en cours de jeu
 - Exemple: lorsque qu'un coin est pris, modifier les évaluations des cases adjacentes
- Tri des nœuds
 - en fonction de leur valeur statique (ex: regarder les coups conduisant à l'occupation des coins en premier) (tri statique)
 - Avec une recherche itérative (tri dynamique : fonction de la situation)
- Heuristique du coup meurtrier
- Optimisations diverses
 - Occupation de la mémoire (et garbage collecting)
 - Recherches en parallèle

Le cas du jeu Othello

- En 1981, le champion du monde (Jonathan Cerf) estimait que le meilleur programme d'Othello, Iago, était à son niveau ou meilleur
- En 1989, le meilleur programme, Bill, battait le champion du monde, Brian Rose, 56 à 8
 - Iago et Bill utilisent aussi une évaluation très sophistiquée des cases périphériques plus une évaluation du potentiel de mobilité
 - Utilisation d'un alpha-beta à fenêtre (on fixe l'intervalle [alpha-beta] dans des bornes étroites pour favoriser les coupures)
 - Iago et Bill économisent du temps en milieu de partie pour l'investir sur la finale
- Une version simplifiée de Iago/Bill doit être disponible sur le net

Jeux stochastiques

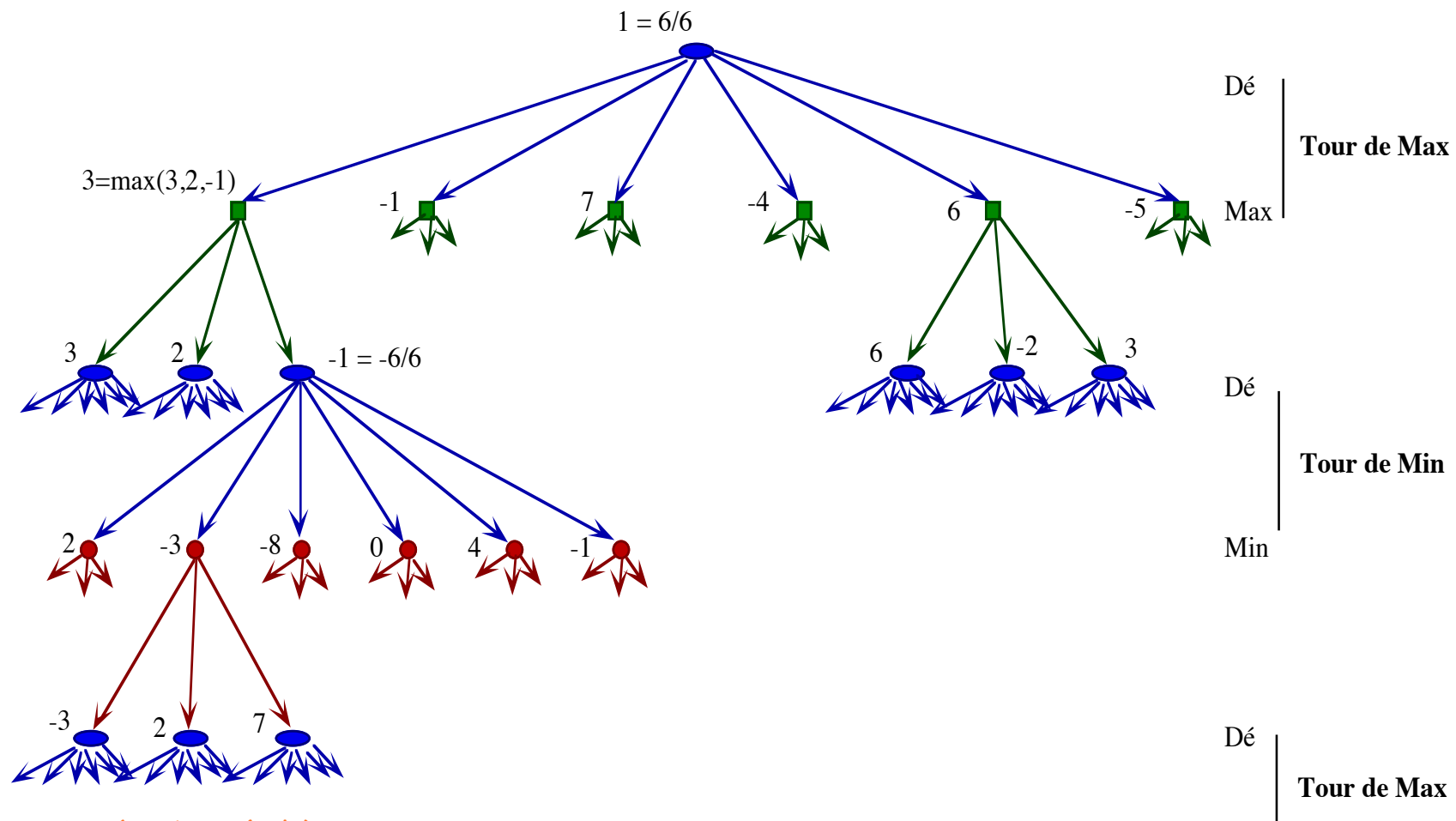
- E.g. le backgammon



$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITÉ}(s, \text{MAX}) & \text{si TEST-TERMINAL}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RÉSULTAT}(s, a)) & \text{si JOUEUR}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RÉSULTAT}(s, a)) & \text{si JOUEUR}(s) = \text{MIN} \\ \sum_r P(r) \text{EXPECTIMINIMAX}(\text{RÉSULTAT}(s, r)) & \text{si JOUEUR}(s) = \text{HASARD}, \end{cases}$$

Élargissement à d'autres types de jeux

- Jeux avec hasard (ex: Backgammon)
 - Notion d'espérance de valeur (algorithme *expectimax*)



Élargissement à d'autres types de jeux

- Jeux avec n adversaires ($n > 2$)

Leçons

- Méthodes très différentes du raisonnement humain sur ces jeux
 - Fonction d'évaluation réduite à un nombre
 - Pas de prise en compte de la stratégie
- ↙ Raisonnement par reconnaissance des formes et analogie
- ↙ Planification stratégique AlphaGo : mars 2016

Apprentissage de fonction d'évaluation

- Il y a plusieurs méthodes (cf. cours d'apprentissage (*apprentissage par renforcement* notamment) de l'option Intelligence Artificielle de 3ème année)
- Ici, étude d'une méthode particulière en vue d'illustration
- Historiquement, Samuel en 1959 est le premier à avoir publié et utilisé une méthode d'apprentissage pour le jeu de dames (américain) : système Checker

Apprentissage de fonction d'évaluation

- Soit une fonction d'évaluation définie par une combinaison (linéaire) de facteurs
 - Exemple : nombre de pièces, occupation du centre ou des angles, mobilité, ...
- ↳ *Comment apprendre les bons coefficients ?*
 - Exemple : Samuel utilisait 38 facteurs et se limitait à une combinaison linéaire de ces facteurs avec des coefficients égaux à des puissances de 2 jusqu'à 2^{10}
 - > 21^{38} fonctions d'évaluation possibles !!

Apprentissage de fonction d'évaluation

- Méthode : **recherche par gradient**
 1. Modification de la fonction d'évaluation
 2. et mesure de la performance par rapport à performance précédente
- *Mesure de la performance ?*
 - Après un ensemble de tournois (très coûteux)
 - En cours de partie, à chaque calcul de coup à jouer, comparer la valeur statique retournée par la fonction d'évaluation avec l'évaluation retournée en utilisant en plus une recherche en avant alpha-beta

(Si la fonction d'évaluation est correcte, elle devrait retourner un résultat proche de la valeur retournée par alpha-beta)

Apprentissage de fonction d'évaluation

Autre méthode :

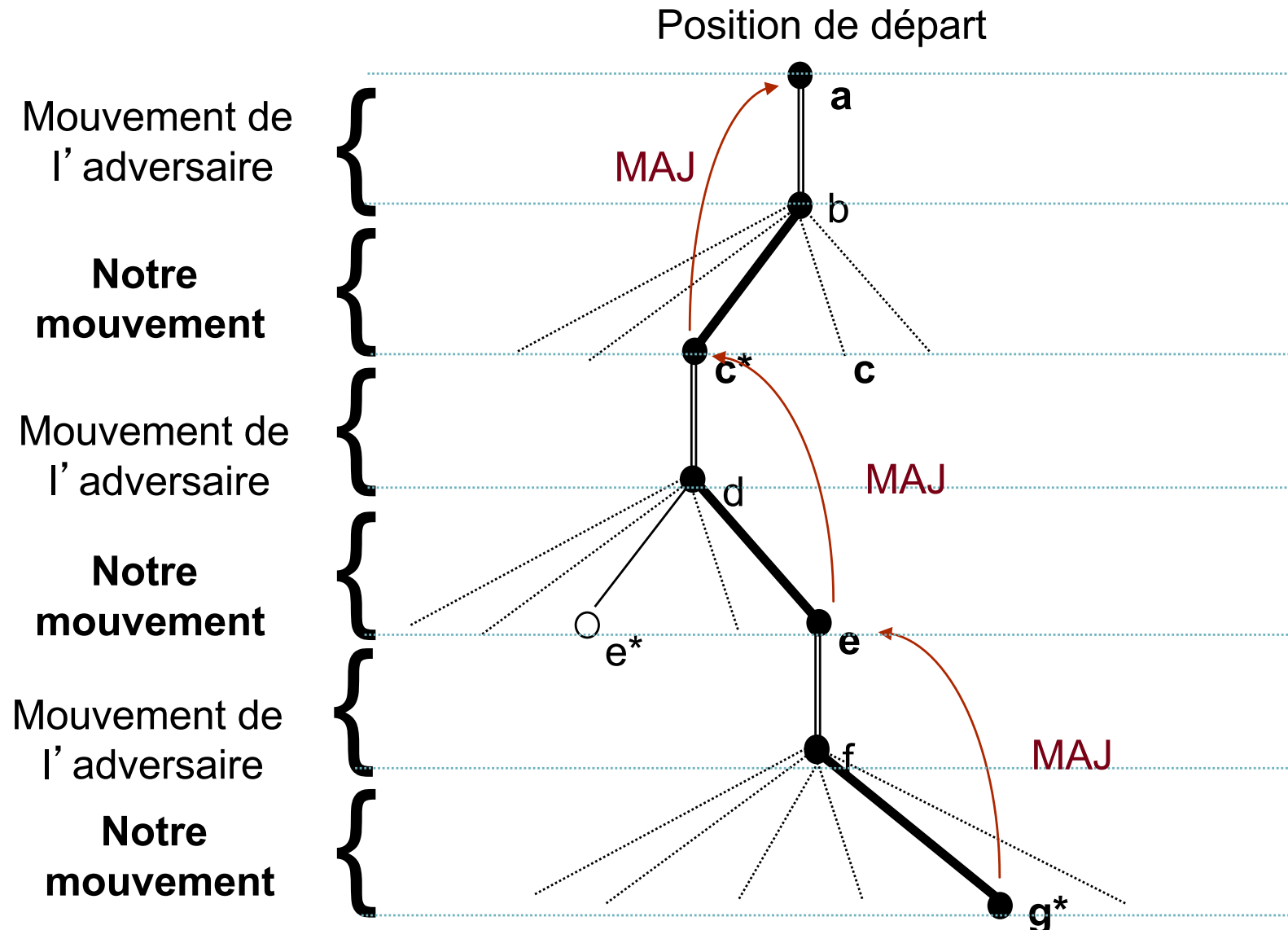
Apprentissage par renforcement par la méthode des différences temporelles (TD-learning)

- On apprend une fonction V : position \rightarrow valeur
 - Initialement, toutes les positions (qu'il faut pouvoir énumérer) sont évaluées à 0,5 sauf les positions perdantes (évaluées à 0) et les positions gagnantes (évaluées à 1)
 - On met à jour l'évaluation de la position s en fonction de l'évaluation de la position suivante $\delta(s)$

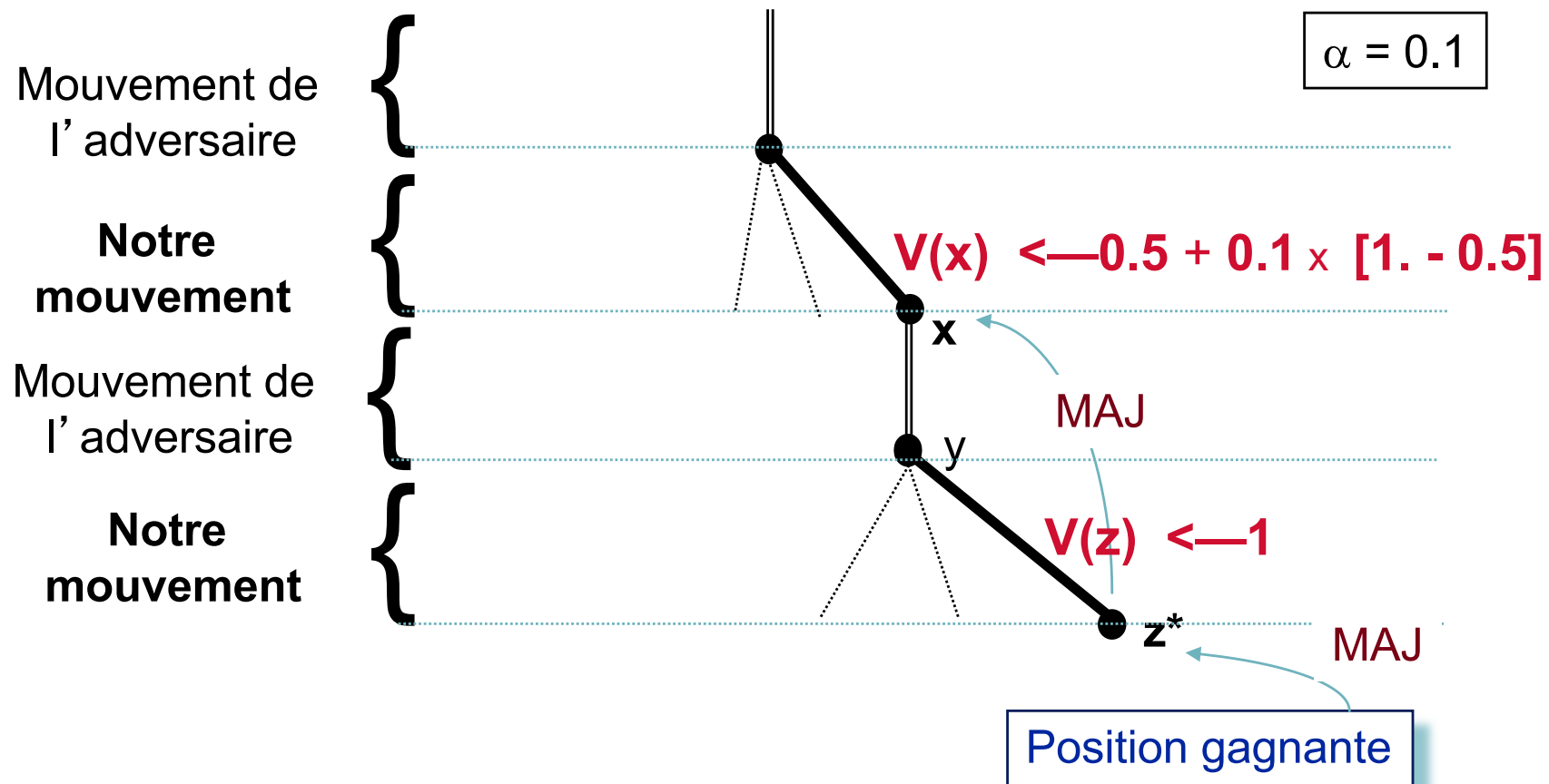
$$V(s_{t+1}) \leftarrow V(s_t) + \alpha [\max(V(\delta(s_t)) - V(s_t))]$$

(α fonct. décroissante)

Apprentissage de fonction d'évaluation



Apprentissage de fonction d'évaluation



- L'état x passe d'une valeur de 0.5 à 0.55
- Si l'adversaire joue toujours y dans la position x sa valeur ne cessera d'augmenter [même si x est, dans l'absolu, une position perdante]

Apprentissage de fonction d'évaluation

○ *Questions*

- *Qu'est-ce qui assure la convergence vers la « politique » optimale ?*
- *Comment régler les paramètres (e.g. α) ?*
- *Comment généraliser ?*
 - Ne pas stocker une valeur pour chaque situation possible
- *A partir de quelles expériences apprendre ?*
 - Machine contre machine ?
 - Machine contre expert ?
 - Machine contre joueur moyen ?
 - Machine contre plusieurs joueurs ?

Apprentissage de fonction d'évaluation

- **Attention :**
 - **Le système s'adapte à l'adversaire**
 - ↳ Peut devenir très mauvais sur un autre adversaire
 - **Il faut aussi des coups exploratoires (risqués) pour éviter de toujours jouer de manière myope ce qui semble le meilleur (optimum local)**
 - **Problèmes si trop de positions :**
 - ↳ Il faut généraliser les positions pour apprendre
On associe une évaluation à un type de position

État de l'art

- **Checkers** (10^{32}) : **Chinook** > H (1994 : *Chinook utilise une base de données sur toutes les fins de parties impliquant 8 pièces ou moins, soit 443 748 401 247 positions*) (jeu résolu en 2007)
- **Othello** (10^{58}) : **Logistello** > H (1997)
- **Échecs** (10^{123}) : **Deep Blue** >= H (1997. *DeepBlue explore 200 millions de positions/s., possède une fonction d'évaluation très sophistiquée, et peut parfois explorer plus de 40 coups en avant (fin de parties).*)
StockFish >> H
- **19x19 Go** (10^{172}) : **AlphaGo** > H (2016 : *Utilise de l'apprentissage par renforcement et des fonctions d'évaluation et de politique apprises avec des réseaux de neurones « profonds »*)
- **Backgammon** : **TD-Gammon** ~ H (1992 : *TD-Gammon, utilise de l'apprentissage par renforcement avec des réseaux de neurones.*)
- **Pocker** : beaucoup de travaux en cours. Niveau des champions du monde.
- **Bridge** : Meilleur pgm ~ H champion. **Nook** (de l'entreprise Nukkai) a battu le 8^{ème} champion du monde au jeu de cartes en mars 2022.
(en annonce, la machine est encore mauvaise)

<http://www.gameai.com/clagames.html>

Sources documentaires

- Ouvrages / articles

- Ginsberg M. : ***Essentials of Artificial Intelligence***. Morgan Kaufmann 1993.
- Nilsson N. : ***Artificial Intelligence : A new synthesis***. Morgan Kaufmann, 1998.
- Max Pumperla & Kevin Ferguson: ***Deep Learning and the Game of Go***. Manning, 2019.
- Russel S. & Norvig P. : ***Artificial Intelligence : A modern approach (4th. Ed.)***. Prentice Hall, 2021. (traduit en français)

- Sites web

- <http://www.gameai.com/clagames.html>