

Faire jouer l'ordinateur

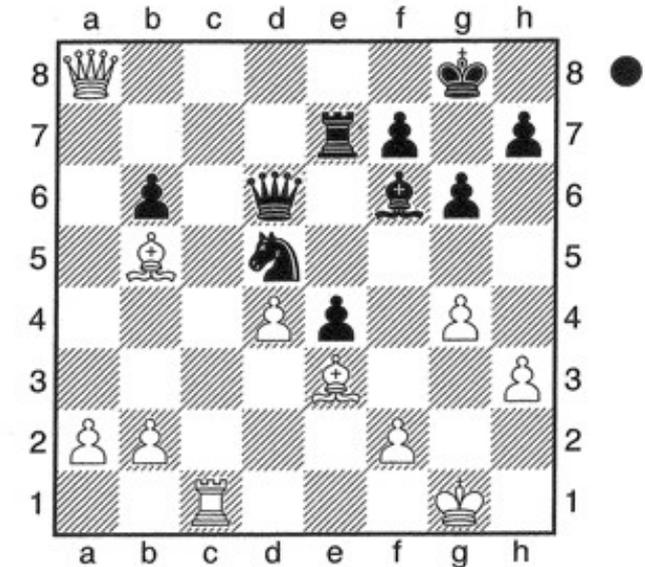
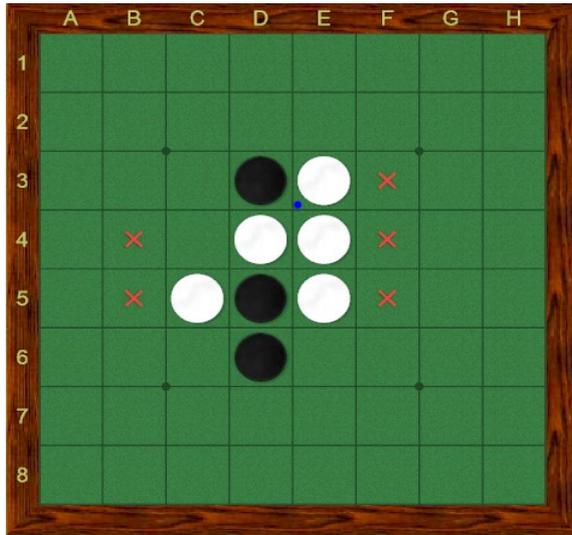
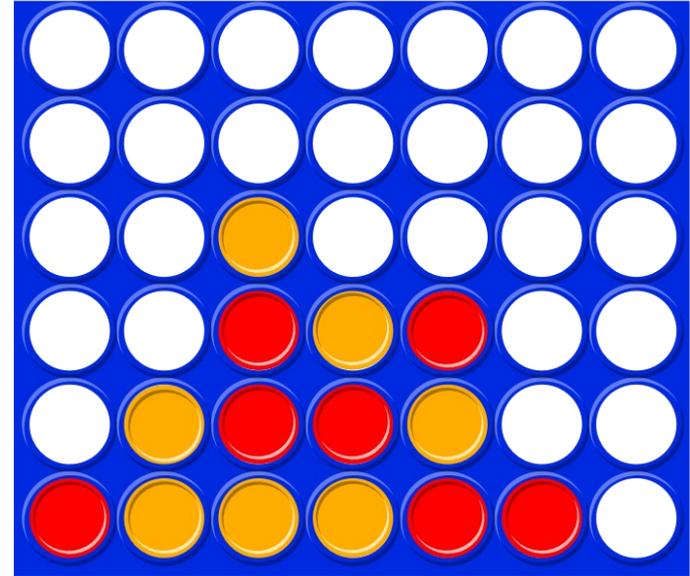
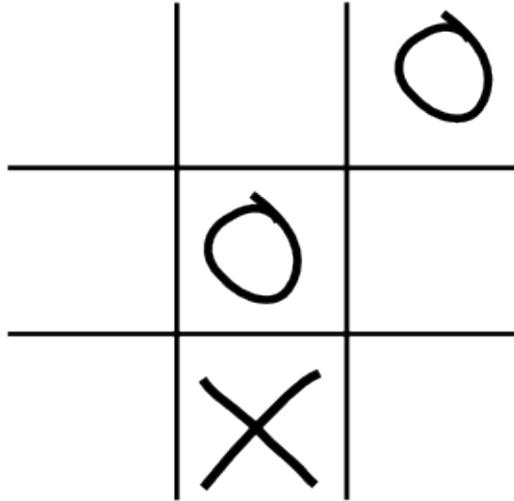
Jean-Christophe Routier
jean-christophe.routier@univ-lille1.fr



ufr d'**IEEA**
Formations en
Informatique de
Lille 1



que faut-il jouer ?



quels jeux ?

Jeux

à 2 joueurs jouant en alternance

à information complète

les joueurs ont toute l'information

les joueurs ont la même information

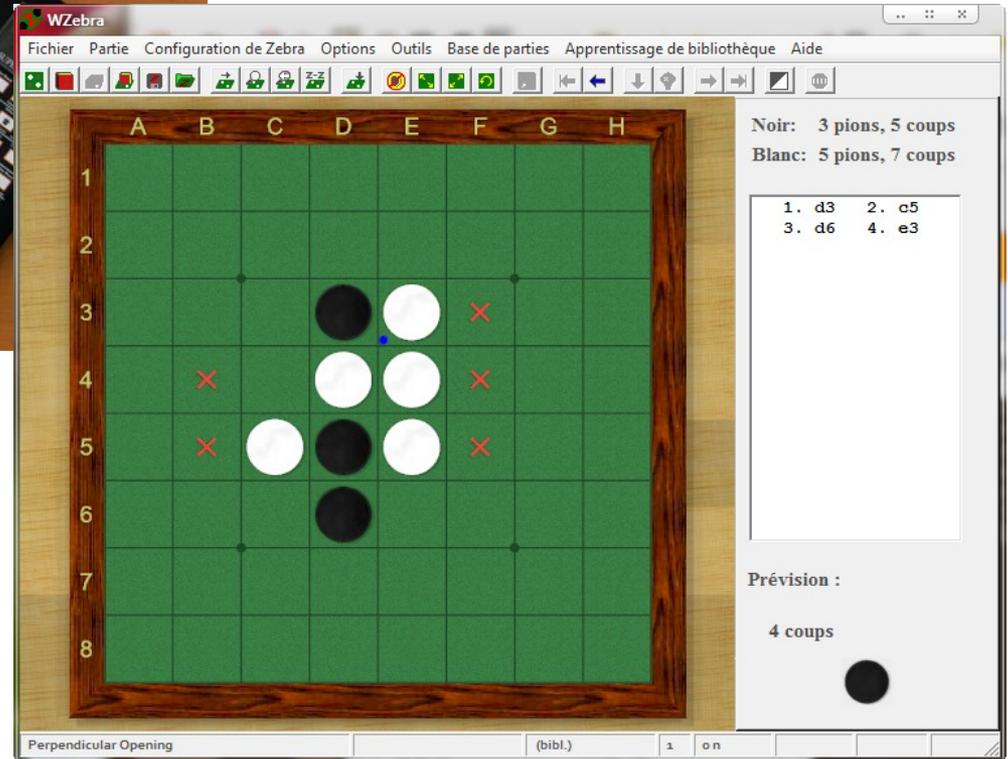
à somme nulle

les intérêts des joueurs sont opposés

la somme des gains est nulle :

ce qui est gagné par l'un est perdu par l'autre

les échecs, les dames, le go, awele, puissance 4, othello, tic-tac-toe, etc.



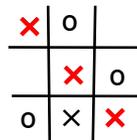
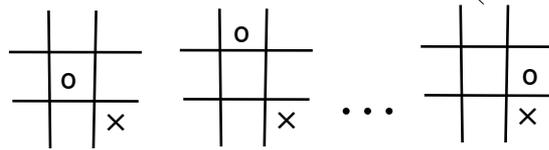
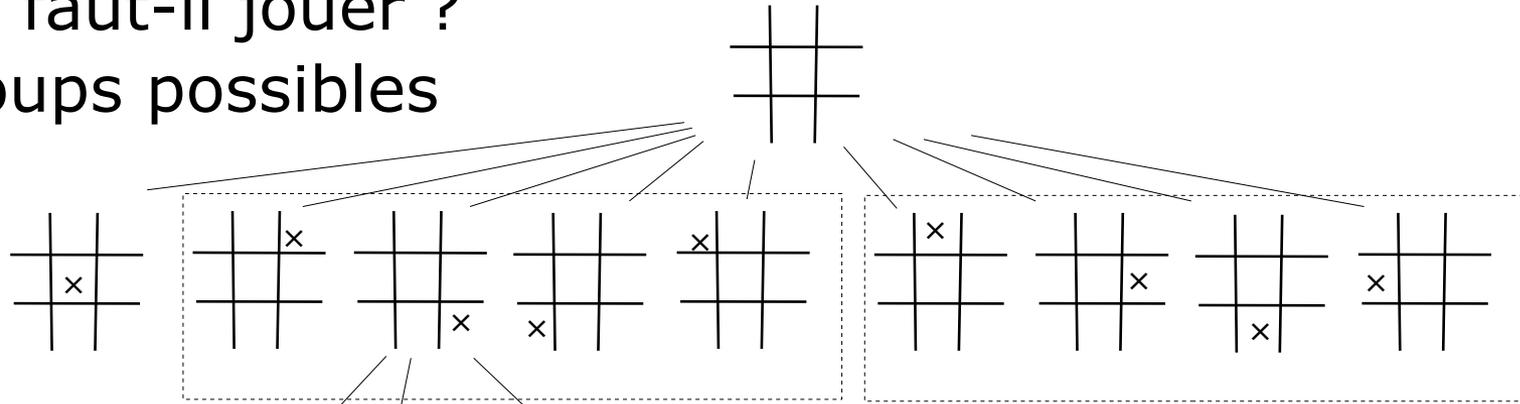
<http://www.jeu.fr/jeu/tic-tac-toe>

question

Comment faire jouer l'ordinateur ?

Comment programmer l'ordinateur pour jouer ?

que faut-il jouer ?
9 coups possibles



19 683 situations différentes

255 168 parties différentes

- 131 184 victoires de x
- 77 904 victoires de o
- 46 080 égalités

comment gérer toutes ces situations ?

aux échecs :

entre 10^{43} et 10^{50} situations possibles

10^{123} parties possibles...

10^{80} atomes dans l'univers

c'est à x de jouer

	o	
x	o	x

o		
o	x	o
		x

o		
x	x	o
		o

x peut-il gagner ?

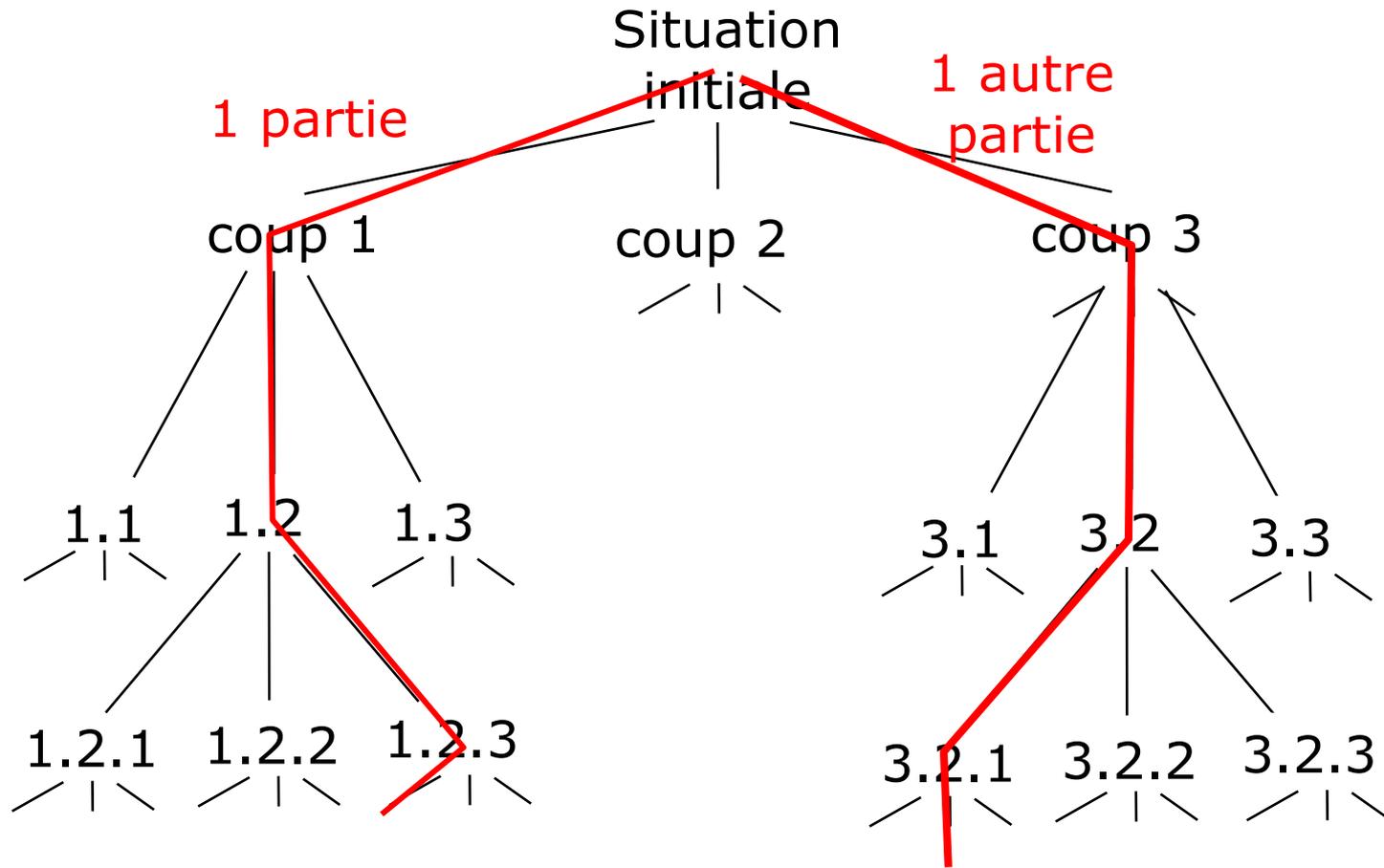
quel est le meilleur coup à jouer ?

activité

calculer/développer l'arbre de jeu pour

	o	
x	o	x
	x	

arbre de jeu



caractéristiques

une situation initiale

à chaque tour les règles du jeu déterminent les situations suivantes possibles/atteignables

jouer c'est choisir la prochaine situation de jeu

une partie

= suite de coups de chacun des joueurs

= suite des situations choisies par les joueurs

une branche
=
déroulement d' une partie

problème :

choisir une branche qui mène à une victoire
ou
choisir un coup **maintenant** pour gagner **plus tard** ?

	o	
x	o	x
	x	

solution :

calculer des coups « à l'avance »

1) calculer l'arbre de jeu

2) choisir la/les meilleures parmi les situations atteintes

problèmes :

- 1) comment comparer les situations atteintes ?
- 2) combien de « coups à l'avance » ?

principe :

à chaque **situation atteinte**
on attribue une
valeur numérique croissante
avec la qualité de la situation

activité

attribuer des valeurs aux situations atteintes
pour l'arbre de

	o	
x	o	x
	x	

quelles valeurs ?
 $P < N < G$

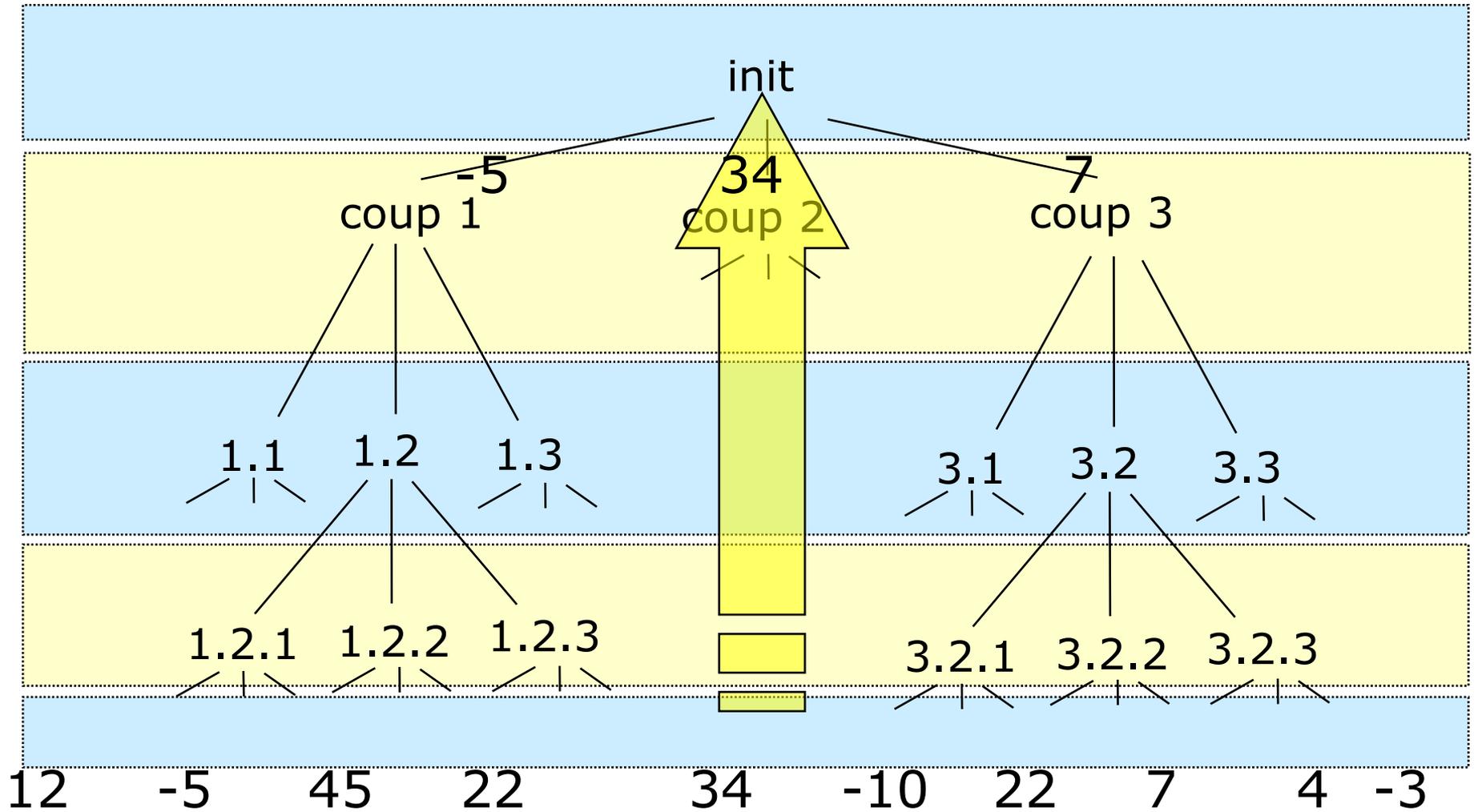
par exemple

$$P = -1 < N = 0 < G = +1$$

principe :

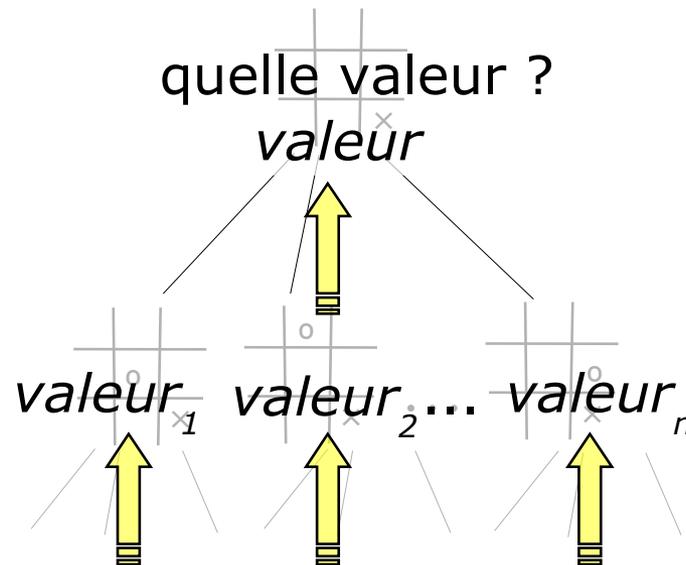
bien choisir c'est...

viser la situation avec la **plus grande valeur**
maximiser la valeur de la situation atteinte
propager les valeurs « vers le haut »



propager ?

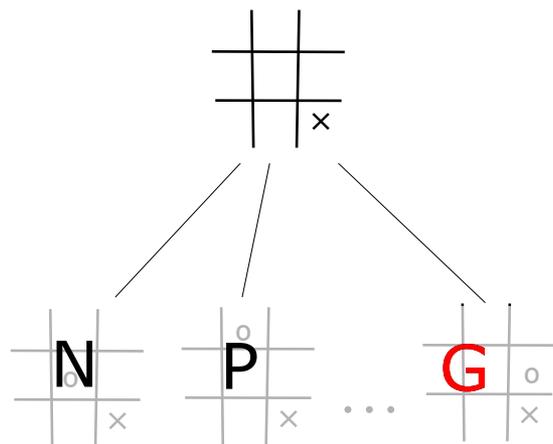
la valeur d'une situation
dépend des valeurs de ses
situations suivantes



maximiser

à chaque fois que l'on doit choisir

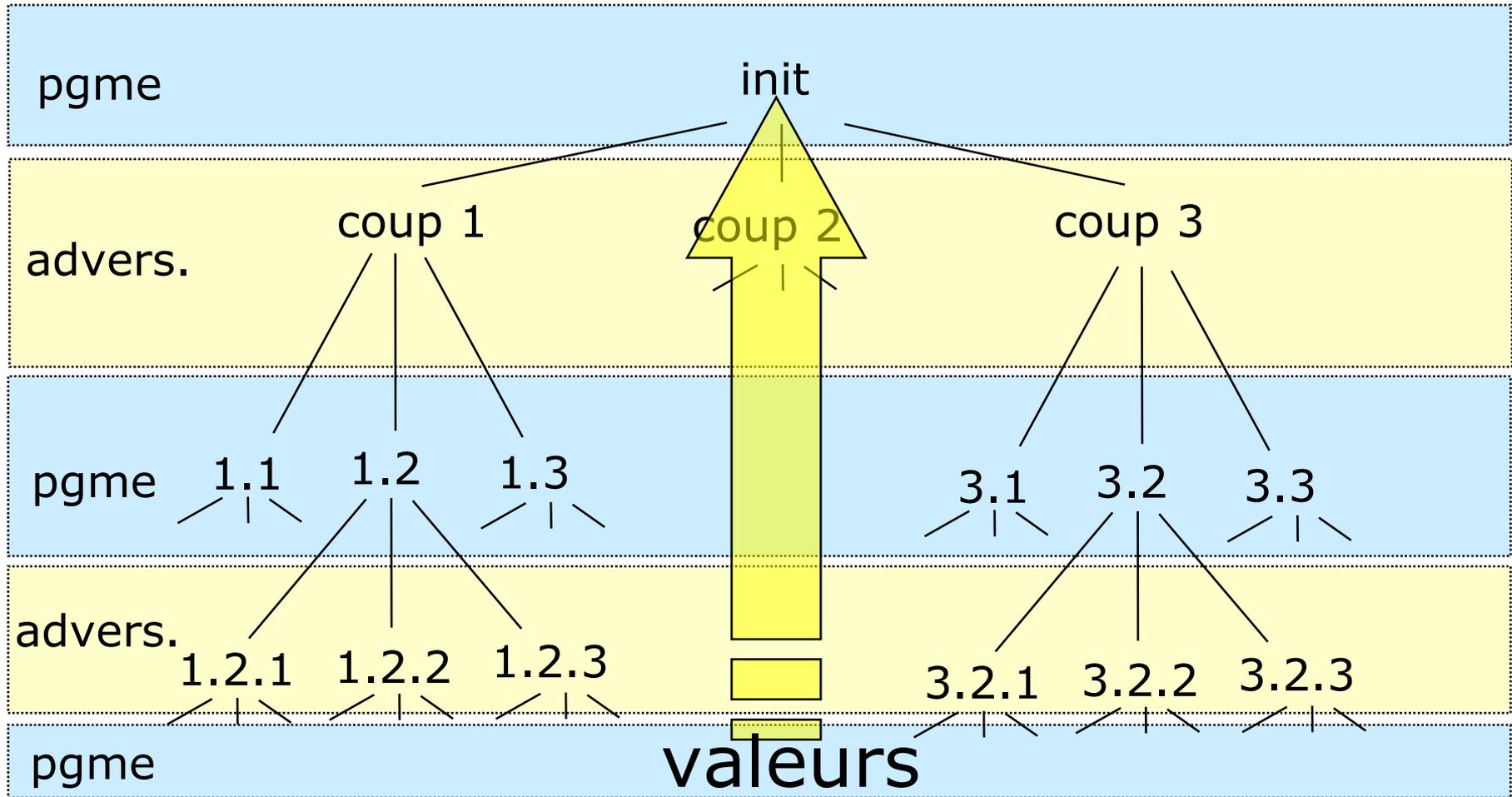
sélectionner parmi les situations suivantes
celle avec la **plus grande** valeur
attribuer sa valeur à la situation actuelle



$$P < N < G$$

$$G = \max(N, P, G)$$

comment propager ?



il faut prendre en compte l'adversaire...

simuler l'adversaire ?

hypothèse : l'adversaire joue le mieux possible
rappel : les intérêts de l'adversaire sont opposés

c'est quoi le « mieux possible » pour le programme ?

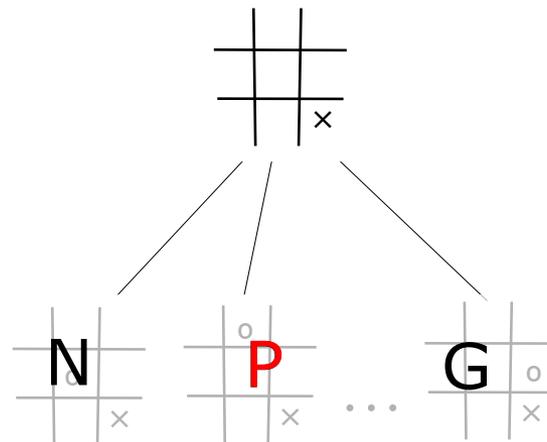
c'est lui-même...

mais l'adversaire a un objectif opposé, il doit donc...

minimiser

donc à chaque fois que l'on doit choisir
à la place de l'adversaire

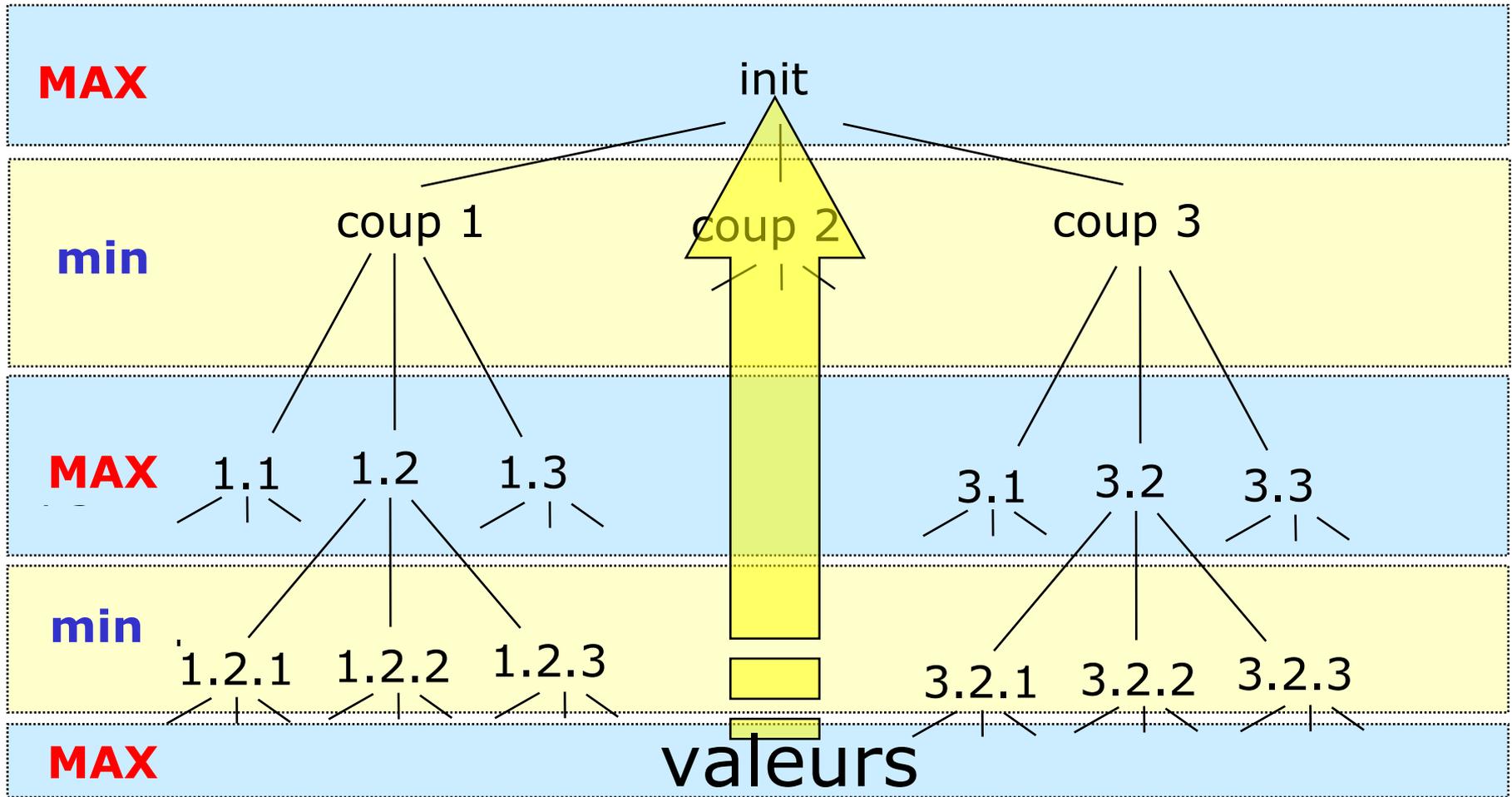
sélectionner parmi les situations suivantes
celle avec **la plus petite** valeur
attribuer sa valeur à la situation actuelle



$$P < N < G$$

$$P = \min(N, P, G)$$

l'algorithme min-MAX



activité

propager les valeurs
pour l'arbre de

	o	
x	o	x
	x	

quel est le meilleur coup à jouer?

que peut-on dire de cette situation ?

synthèse – algorithme min-MAX

- 1) développer l'arbre de jeu
- 2) attribuer des valeurs aux situations atteintes
- 3) propager les valeurs du bas vers le haut :
 - c'est au tour du programme de jouer :
 - on sélectionne la plus grande des valeurs suivantes
 - on attribue cette valeur à la situation actuelle
 - c'est au tour de l'adversaire de jouer
 - on le simule :
 - on sélectionne la plus petite des valeurs suivantes
 - on attribue cette valeur à la situation actuelle
 - si on n'a pas fini on recommence avec le niveau « au dessus »
- 4) arrivé « en haut » on choisit la situation qui a obtenu la plus grande valeur

algorithme min-MAX

appel initial :
`minmax(situation_initiale, profondeur, MAX)`

```

minmax(situation, prof, joueur) =
  si prof = 0 ou situation est une fin de partie
    retourner eval(situation)
  sinon
    situFilles = les situations filles légales de situation
    si joueur = MAX
      retourner  $\max_{fille \in \text{situFilles}} \{ \text{minmax}(fille, \text{prof}-1, \text{min}) \}$ 
    sinon // joueur = min
      retourner  $\min_{fille \in \text{situFilles}} \{ \text{minmax}(fille, \text{prof}-1, \text{MAX}) \}$ 
    fin si
  fin si

```

le plus souvent
il n'est pas possible de calculer
l'arbre de jeu complet

choisir un nombre de coups calculés à l'avance
et
définir une fonction d'évaluation

fonction d'évaluation

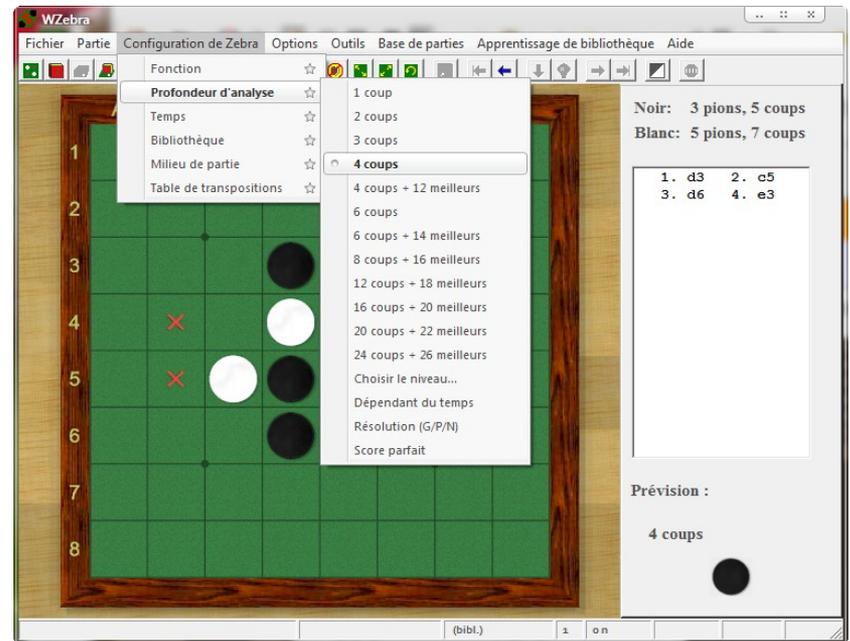
exemple : othello
(max = noir)

1) valeur = nb noirs - nb blancs

2)

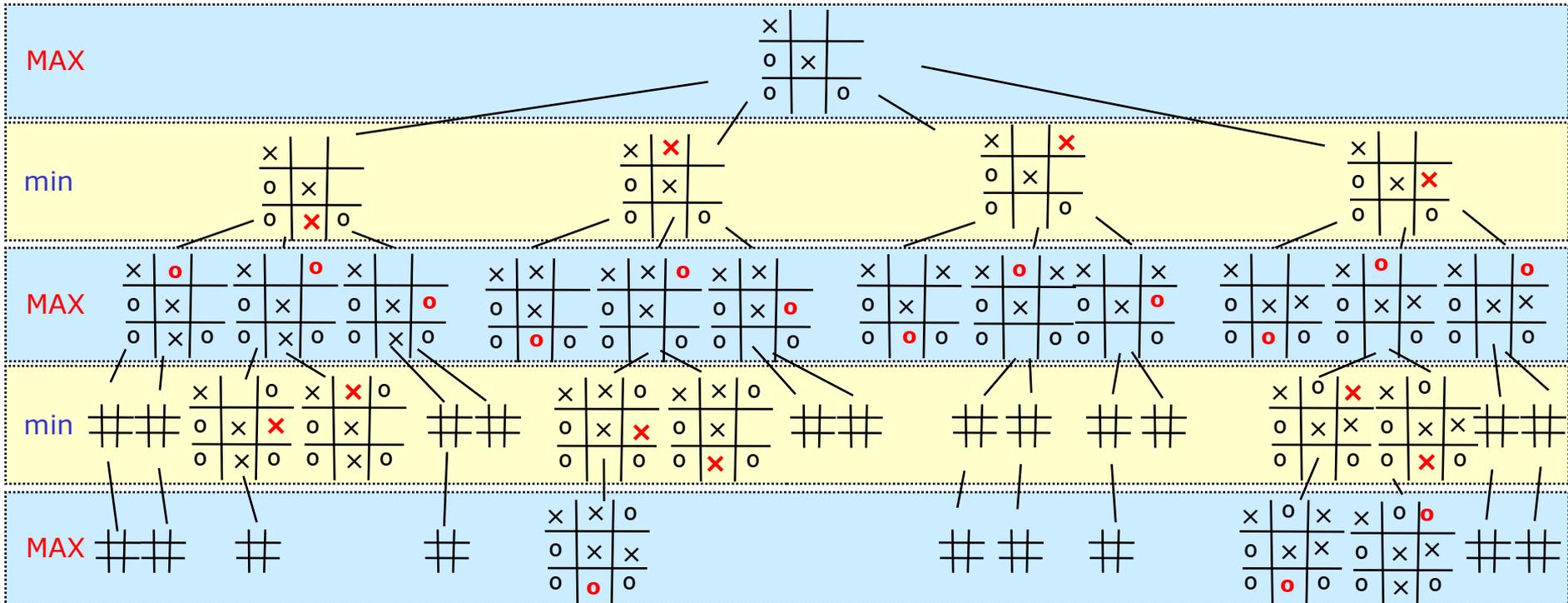
donnez une valeur aux cases
somme des valeurs des cases occupées par les noirs
- somme des valeurs des cases occupées par les
blancs

- qualité du jeu du programme :
- 1) nombre de coups calculés
 - 2) pertinence fonction d'évaluation

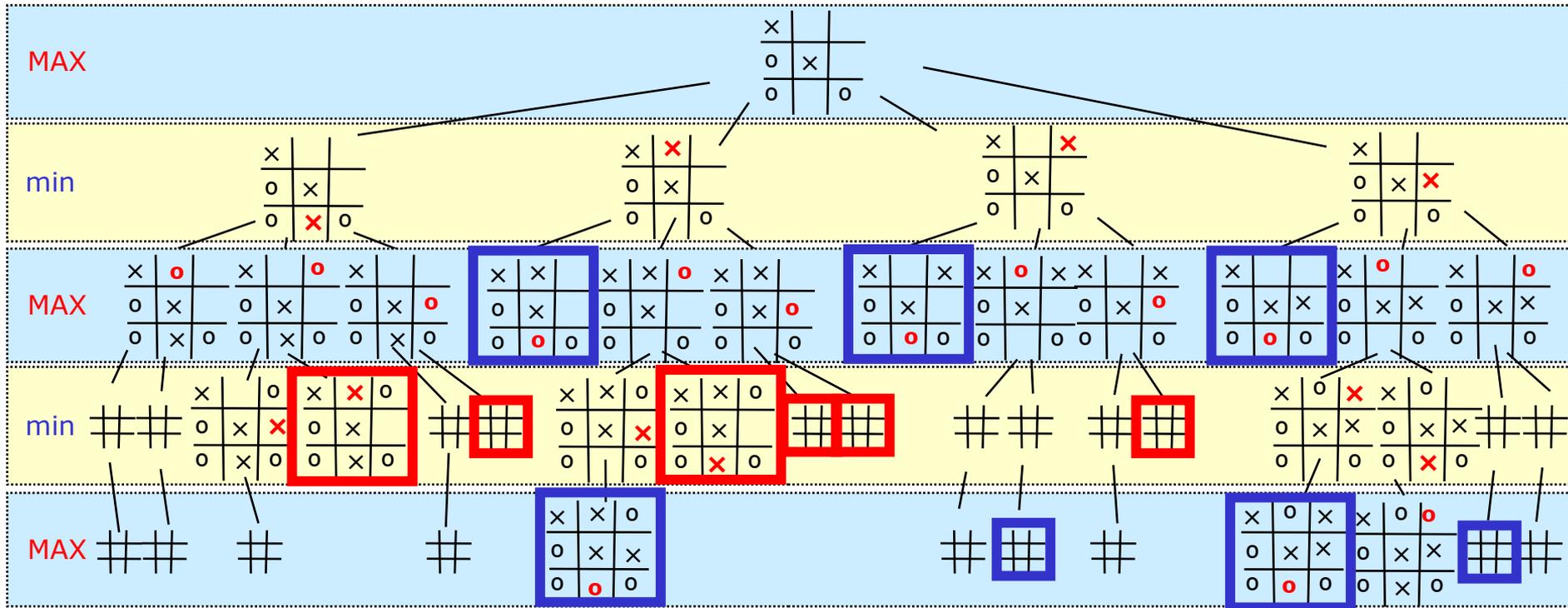


problème : l'effet d'horizon

exemple : tic-tac-toe



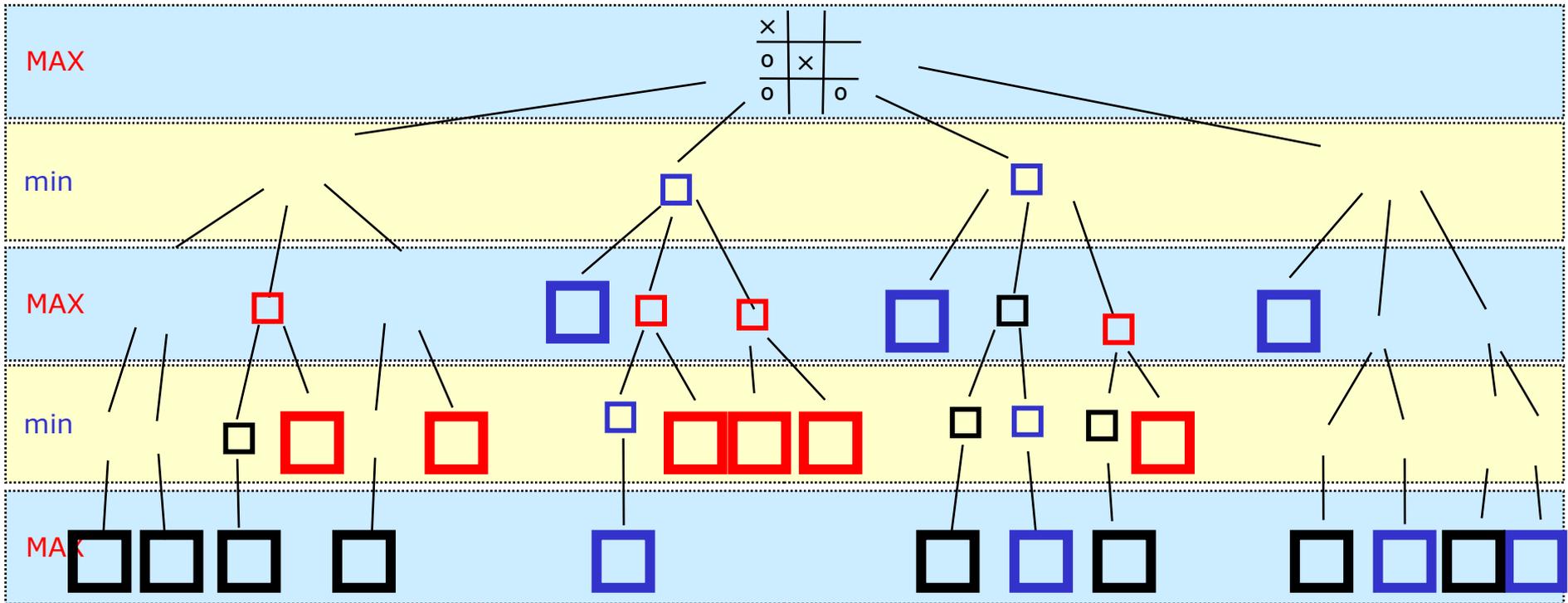
exemple : tic-tac-toe



 victoire de MAX

 victoire de min

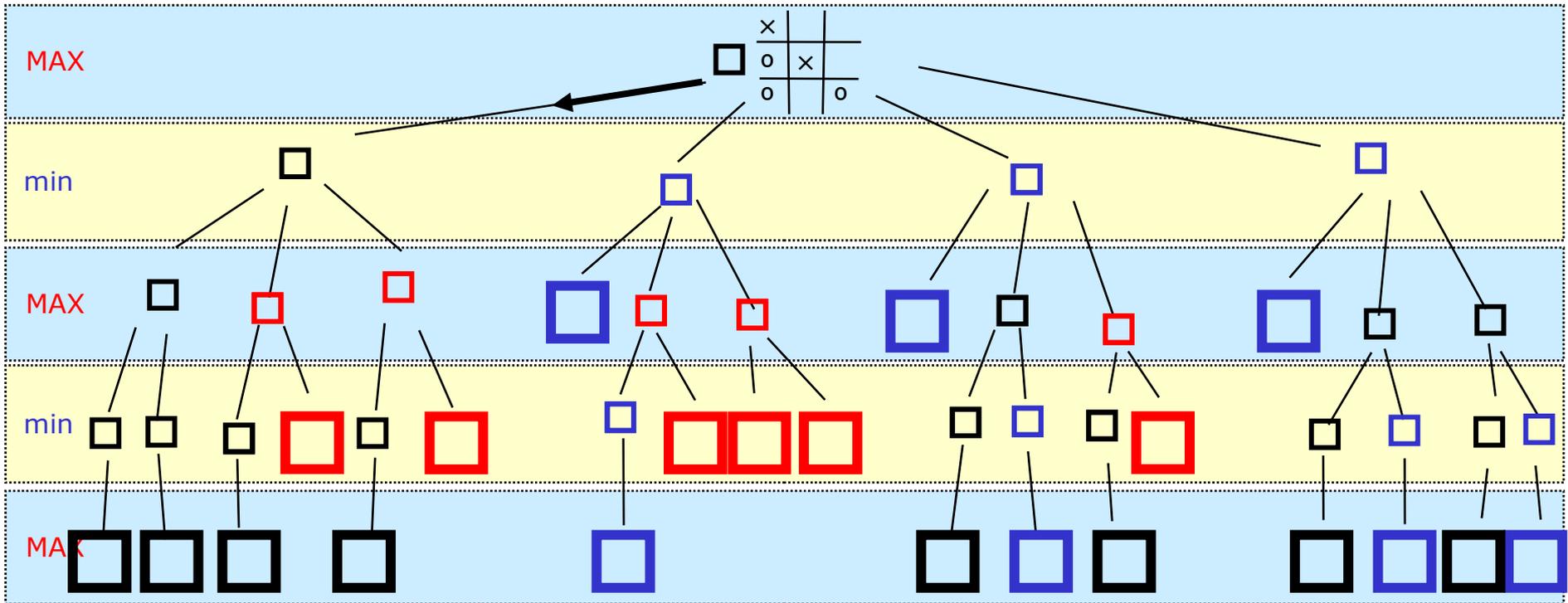
exemple : tic-tac-toe



□ = G victoire de MAX
 = N match nul
 □ = P perdu victoire de min

valeurs : □ < < □

exemple : tic-tac-toe



valeurs : $\square < \square < \square$

c'est à x de jouer

	o	
x	o	x

o		
o	x	o
		x

o		
x	x	o
		o

x peut-il gagner ?
 quel est le meilleur coup à jouer ?

que dit l'algorithme ?

à vous de jouer...