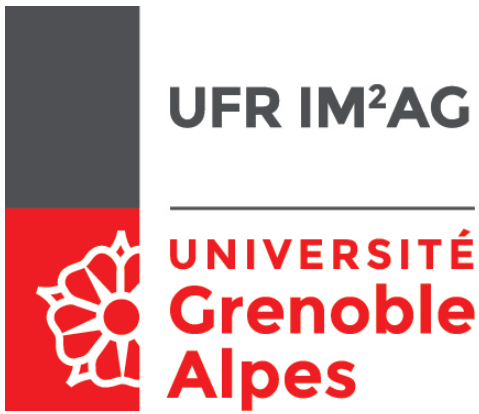
Jeux et arbres

Introduction aux stratégies

Jean-Marc.Vincent@imag.fr



Avril 2017

JEUX ET ARBRES

PROBLÈME

Jeux de plateau (et dérivés)

- Jeux à 2 joueurs (joueur A et joueur B), le joueur A commence
- Jeu à information complète : tout est connu par les 2 joueurs
  - pas de connaissance statistique (excepté l'historique de la partie en cours)
  - pas d'information non partagée (ex : poker)
  - information = configuration du jeu à un instant donné
- Jeu à somme nulle
  - le gain du joueur A est égal à la perte du joueur B (et réciproquement)
  - cas de partie nulle possible (3 valeurs de gain pour le joueur A $+1$, 0 ou -1
  - pas de "banque" (ex : roulette)

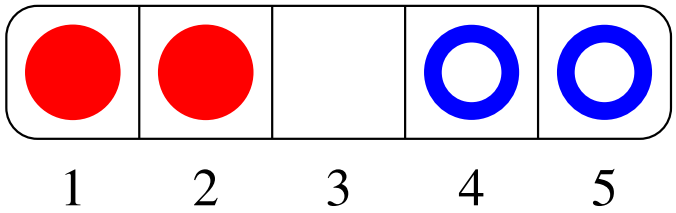
Exemples

- Jeux de stratégie classiques
  - Échecs, Dames, Marelles (moulin), Othello,...
  - Awele et variantes de jeux de semailles
  - Jeu de Go
- Jeux jouets
  - Tic, Tac, Toe, Marelle, jeu de Nim, Pipopipette, la gaufre,...

UNIVERSITÉ Grenoble Alpes

EXEMPLE : SAUTE-MOUTON

Description du jeu



Objectif pour le joueur rouge (resp. bleu) de placer ses 2 pions en case 4 et 5 (resp. 1 et 2).

Règles

- À chaque tour le joueur ne déplace qu'un seul pion.
- Les pions rouges (resp. bleus) se déplacent vers la droite (resp. gauche).
- Un pion se déplace en occupant la case libre si elle se trouve dans son sens de déplacement.
- Si aucun déplacement n'est possible le joueur passe son tour.

Exemple En début de partie le joueur rouge le pion en position 1 en position 3 ou le pion en position 2 en position 3. Si le joueur rouge place le pion 1 en position 3, le joueur bleu peut placer son pion en position 4 en position 1 (ou son pion 5...)

EXEMPLE : SAUTE-MOUTON

Analyse du jeu

Configuration

- la position des 4 pions
- le trait (c'est à dire le joueur devant jouer)

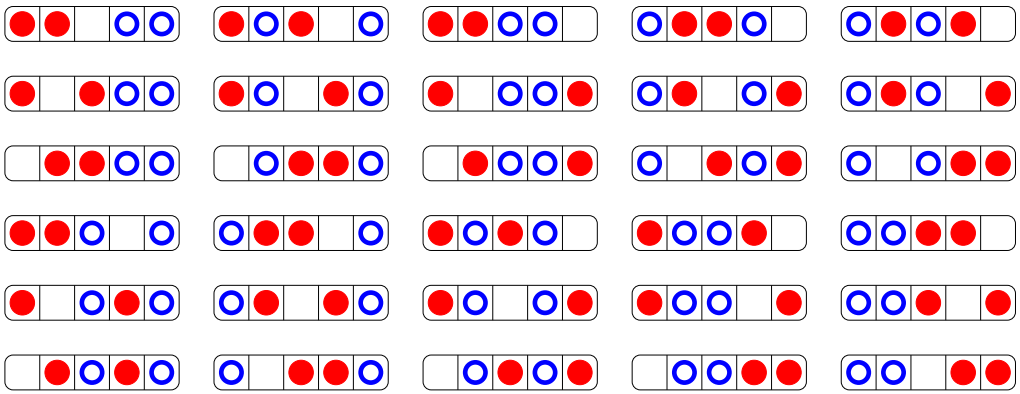
Complexité

- N taille de l'espace dans lequel évolue les configurations

$$N = \binom{5}{2}\binom{3}{2} \times 2 = 60$$

- degré de branchement : ici 2 possibilités au plus par joueur
- profondeur maximale (nombre maximum de $\frac{1}{2}$ coups) : ici $\leqslant 12$

Espace des configurations



UNIVERSITÉ Grenoble Alpes

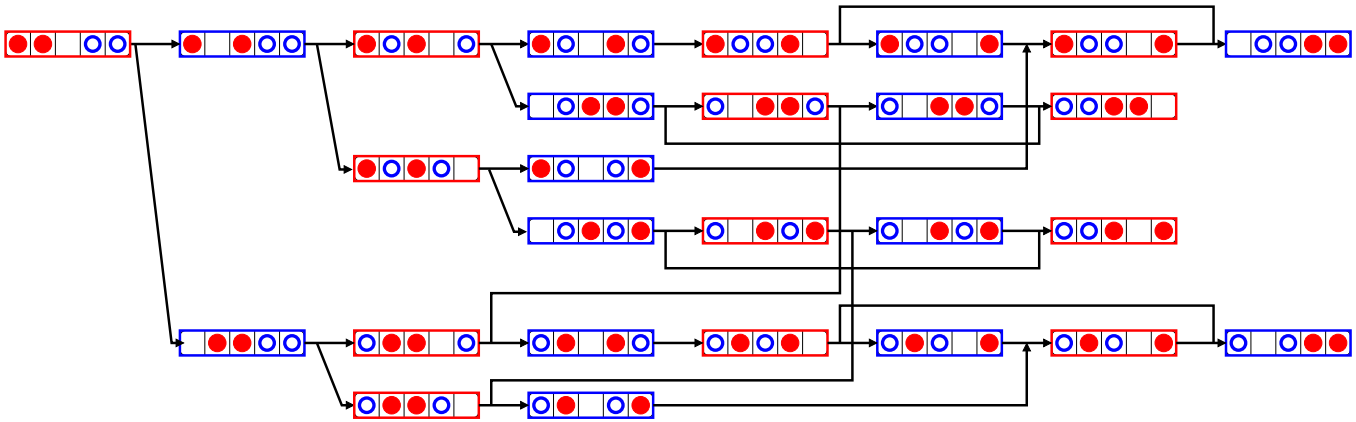
EXEMPLE : SAUTE-MOUTON

Graphe du jeu

Configuration initiale

Configurations terminales

Relation : $\frac{1}{2}$ coups admissibles

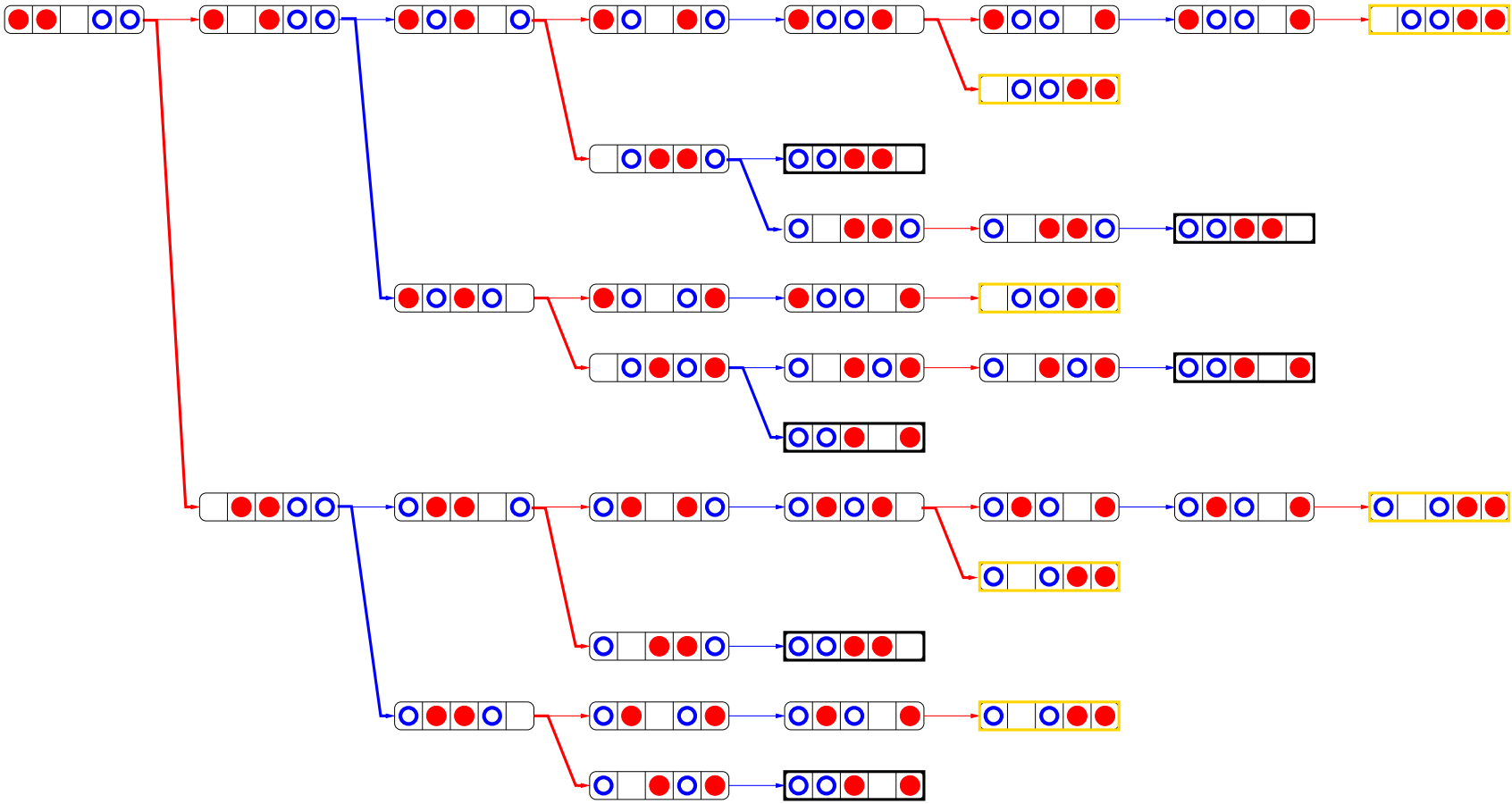


Propriétés du graphe

- graphe sans circuits (fonction décroissante stricte)
- graphe biparti (alternance des coups)
- toutes les configurations ne sont pas accessibles : 27 configurations
- nombre de nœuds terminaux : 4

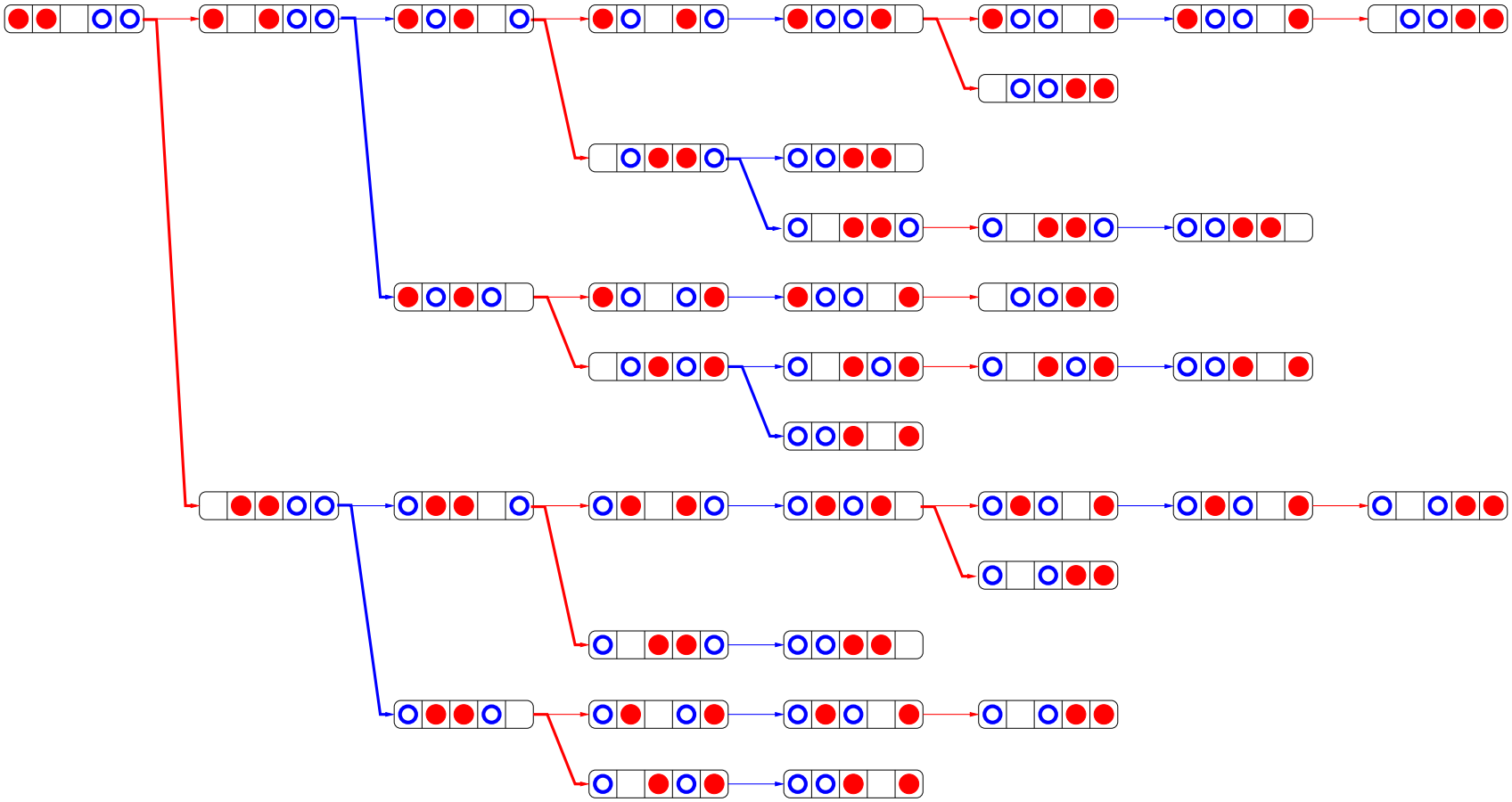
EXEMPLE : SAUTE-MOUTON

Analyse de l'arbre des chemins



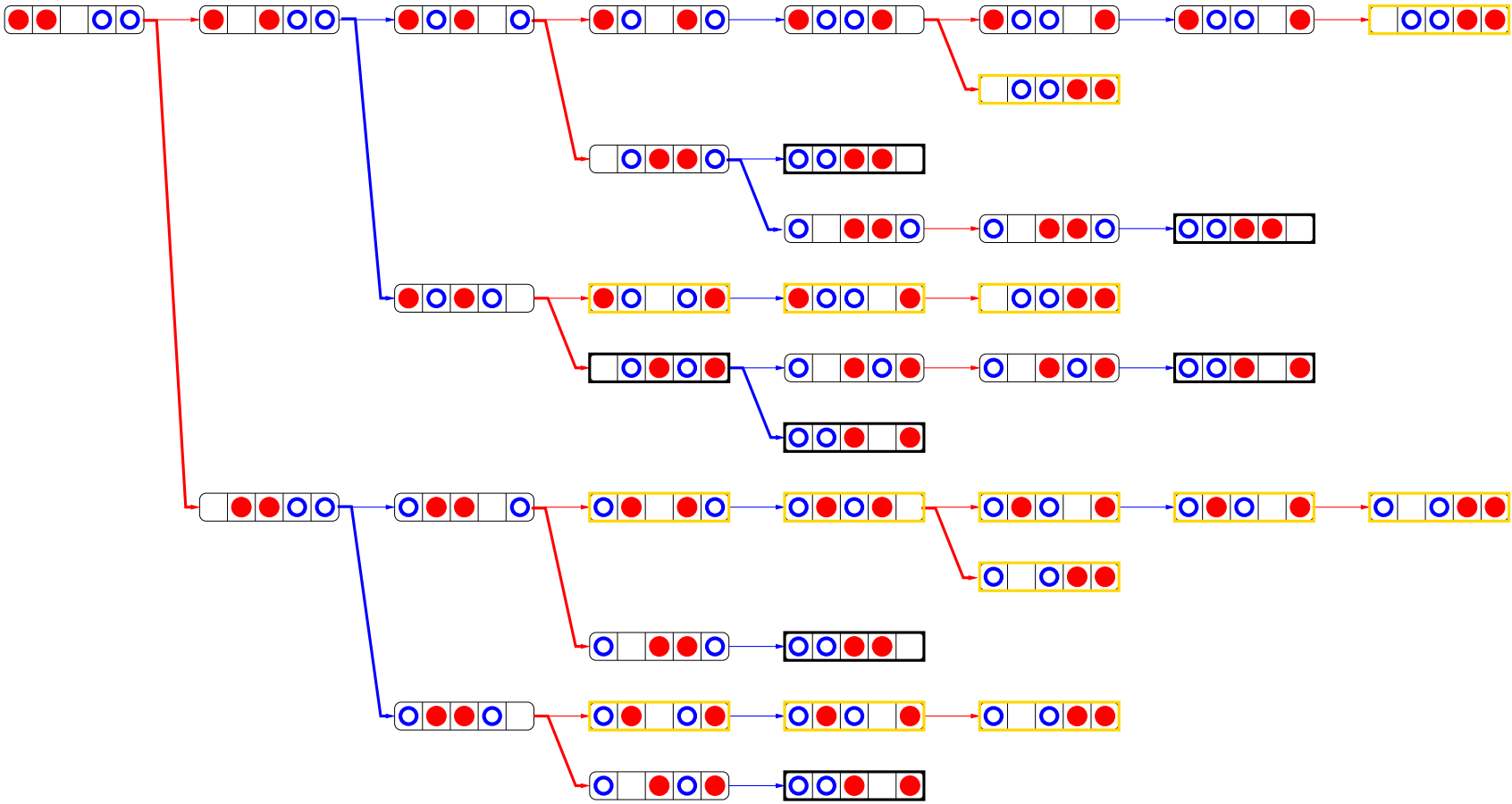
EXEMPLE : SAUTE-MOUTON

Valuation de l'arbre des chemins



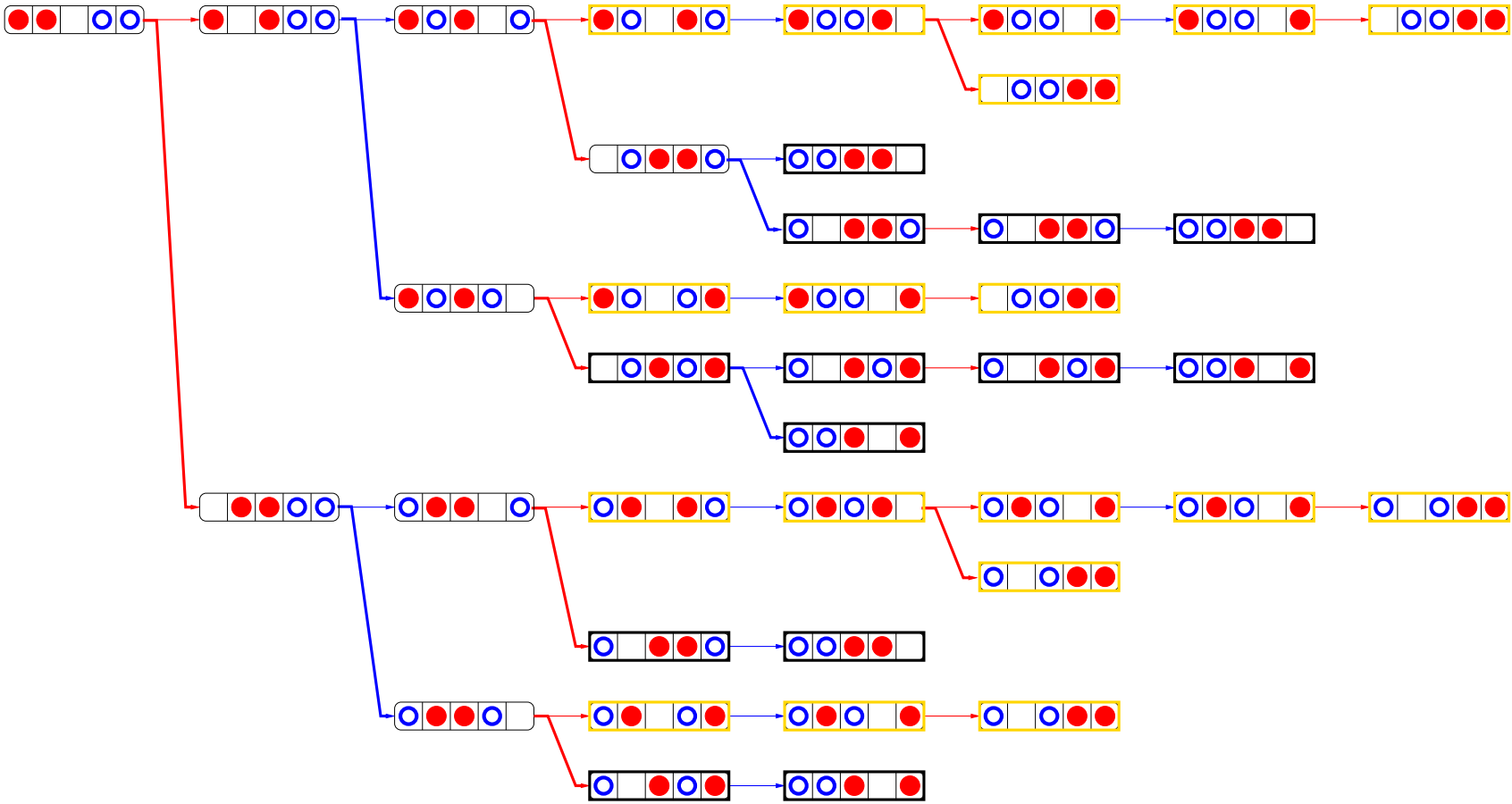
EXEMPLE : SAUTE-MOUTON

Valuation de l'arbre des chemins



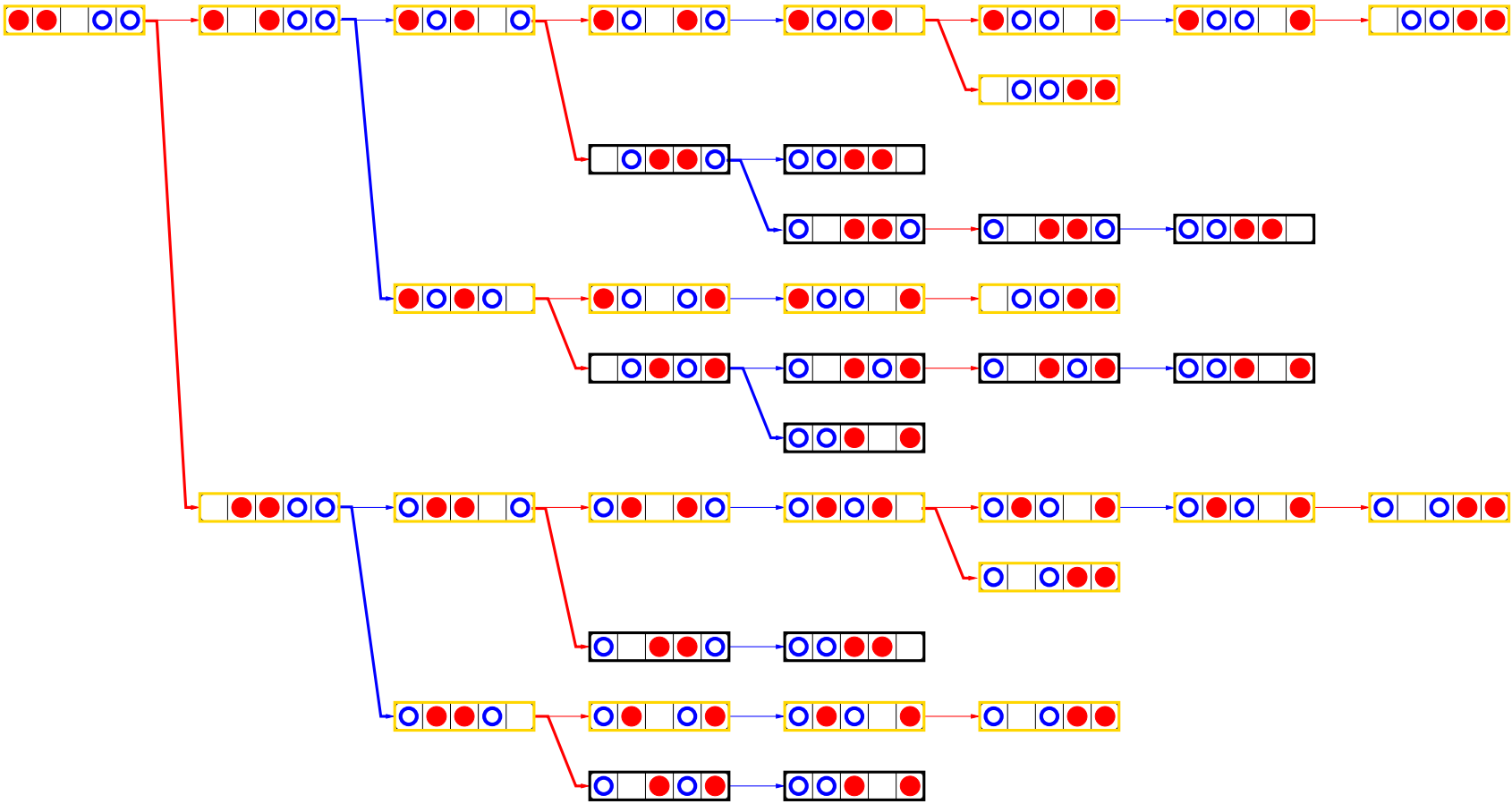
EXEMPLE : SAUTE-MOUTON

Valuation de l'arbre des chemins



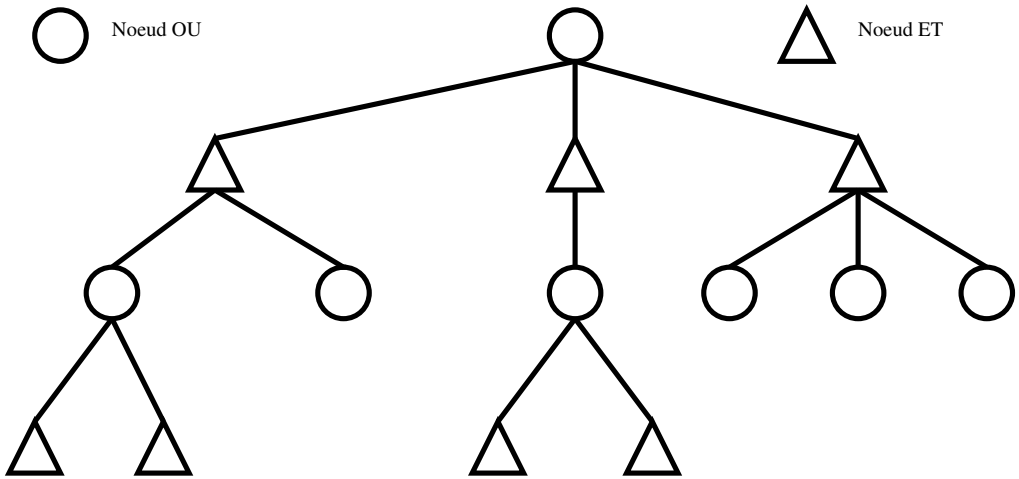
EXEMPLE : SAUTE-MOUTON

Valuation de l'arbre des chemins



JEUX ET ARBRES

ARBRE ET/OU



- Expression booléenne
- Valuation des feuilles : vrai/faux
- Évaluation de l'arbre : parcours de l'arbre
  - profondeur/largeur d'abord
  - post-fixé : la valeur d'un nœud est calculée après l'évaluation de ses fils
  - amélioration par idempotence

ÉVALUATION DE L'ARBRE ET/OU

```
Évaluation_Joueur_A (configuration)
  Données: Configuration de jeu, c'est au joueur A de jouer
  Résultat: vrai si la configuration est gagnante,  faux sinon (elle est perdante)
  if configuration = feuille
      // la configuration ne permet pas de jouer, elle est gagnante ou perdante pour le joueur A
      Return gagnante (configuration)
  else
      // Le joueur A doit jouer
      C=Coup (A,configuration)
      // C Ensemble des coups jouables par A
      Valeur = faux
      forall the c∈ C do
          Valeur = Valeur Ou Evaluation_Joueur_B (Jeu (c,configuration))
      Return Valeur
```

ÉVALUATION DE L'ARBRE ET/OU

Évaluation_Joueur_A (configuration)
Données: Configuration de jeu, c'est au joueur A de jouer
Résultat: vrai si la configuration est gagnante, faux sinon (elle est perdante)
if configuration = feuille
    `// la configuration ne permet pas de jouer, elle est gagnante ou perdante pour le joueur A`
    `Return gagnante` (configuration)
else
    `// Le joueur A doit jouer`
    C=`Coup` (A,configuration)
    `// C Ensemble des coups jouables par A`
    Valeur = faux
    forall the c∈ C do
        Valeur = Valeur `Ou` Evaluation_Joueur_B (`Jeu` (c,configuration))
    `Return` Valeur

Évaluation_Joueur_B (configuration)
Données: Configuration de jeu, c'est au **joueur B** de jouer
Résultat: vrai si la configuration est gagnante **pour A**, faux sinon (elle est perdante)
if configuration = feuille
    `// la configuration ne permet pas de jouer, elle est gagnante/perdante pour le joueur A`
    `Return gagnante` (configuration)
else
    `// Le joueur B doit jouer`
    C=`Coup` (**B**,configuration)
    `// C Ensemble des coups jouables par B`
    Valeur = **vrai**
    forall the c∈ C do
        Valeur = Valeur `et` Évaluation_Joueur_A (`Jeu` (c,configuration))
    `Return` Valeur

UNIVERSITÉ Grenoble Alpes

ÉVALUATION DE L'ARBRE ET/OU

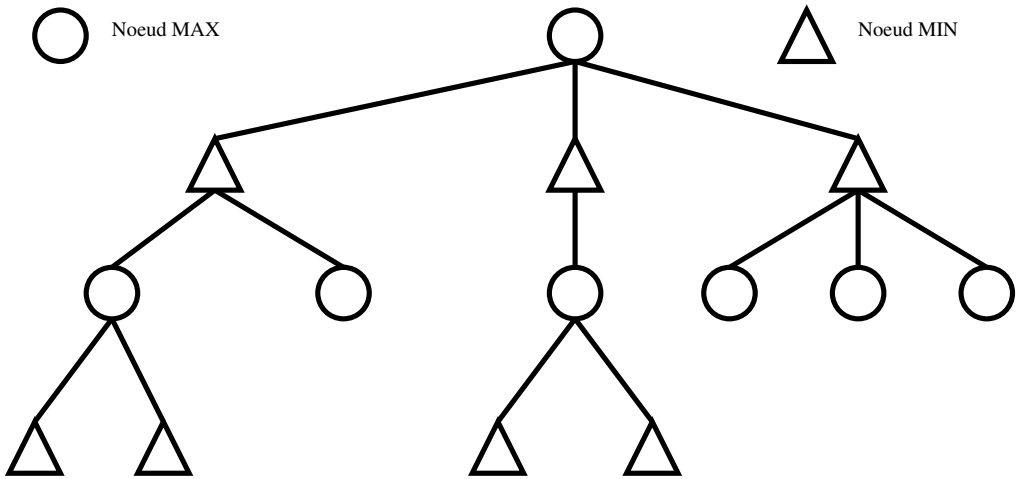
Commentaires

- le joueur A et le joueur B ne sont pas symétriques
- jeu avec un gagnant (pas de partie nulle)
- parcours de tout l'arbre des parties : coût en $\mathcal{O}(b^p)$ avec b facteur de branchement et p profondeur de l'arbre
- graphe sans circuit : assure la terminaison de l'algorithme

```
Évaluation_Joueur_A (configuration)
    Données: Configuration de jeu, c'est au joueur A de jouer
    Résultat: vrai si la configuration est gagnante,  faux sinon (elle est perdante)
    if configuration = feuille
        // la configuration ne permet pas de jouer, elle est gagnante ou perdante pour le joueur A
        Return gagnante (configuration)
    else
        // Le joueur A doit jouer
        C=Coup (A,configuration)
        // C Ensemble des coups jouables par A
        Valeur = faux
        coup_gagnant[configuration]= ∅
        forall the c∈ C do
            if Evaluation_Joueur_B (Jeu (c,configuration))
                Valeur = vrai
                ajouter (c, coup_gagnant[configuration])
        // si coup_gagnant[configuration]) est vide faire un choix arbitraire (randomisé ou non)
        Return Valeur
```

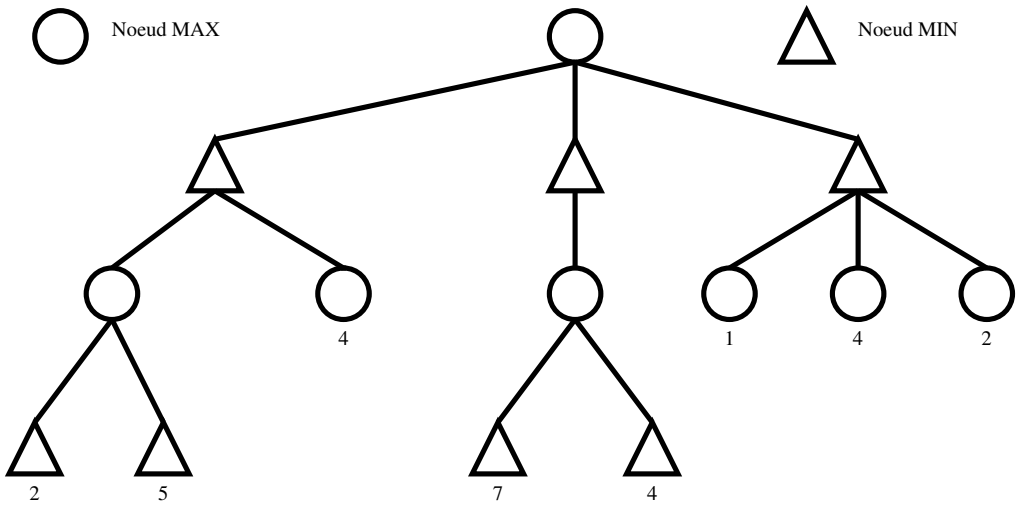
JEUX ET ARBRES

ARBRE MINIMAX



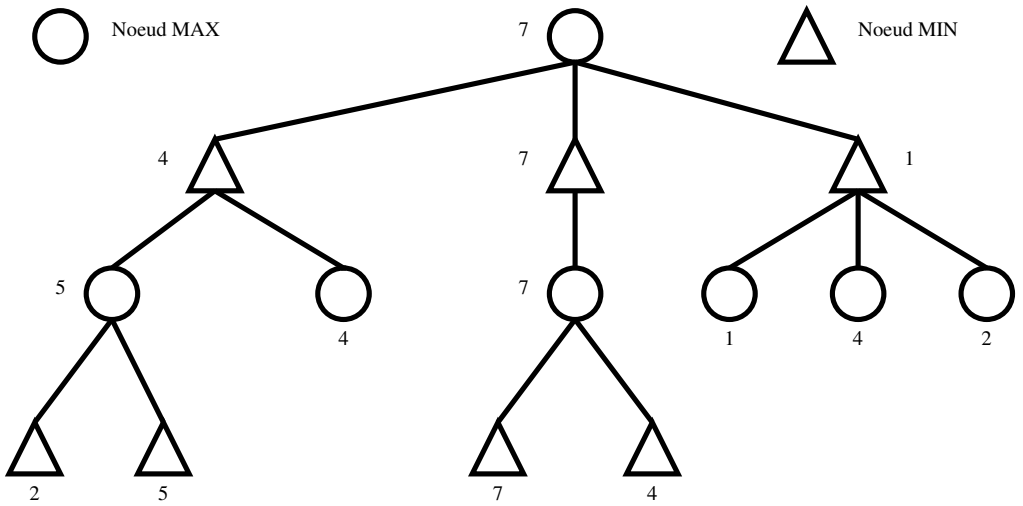
- Valuation des feuilles dans un ensemble ordonné
- Évaluation de l'arbre : parcours de l'arbre
  - profondeur/largeur d'abord
  - post-fixé : la valeur d'un nœud est calculée après l'évaluation de ses fils
  - amélioration par idempotence

ARBRE MINIMAX



- Valuation des feuilles dans un ensemble ordonné
- Évaluation de l'arbre : parcours de l'arbre
  - profondeur/largeur d'abord
  - post-fixé : la valeur d'un nœud est calculée après l'évaluation de ses fils
  - amélioration par idempotence

ARBRE MINIMAX



- Valuation des feuilles dans un ensemble ordonné
- Évaluation de l'arbre : parcours de l'arbre
  - profondeur/largeur d'abord
  - post-fixé : la valeur d'un nœud est calculée après l'évaluation de ses fils
  - amélioration par idempotence

ÉVALUATION DE L'ARBRE MINIMAX

Évaluation_Joueur_A (configuration)
Données: Configuration de jeu, c'est au joueur A de jouer
Résultat: meilleure valeur (si B joue parfaitement)
if configuration = feuille
    `// la configuration ne permet pas de jouer, on évalue sa valeur`
    `Return` Valeur (configuration)
else
    `// Le joueur A doit jouer`
    C=`Coup` (A,configuration)
    `// C Ensemble des coups jouables par A`
    Valeur = $-\infty$
    forall the c∈ C do
        Valeur = `max` (Valeur,Evaluation_Joueur_B (`Jeu` (c,configuration)))
    `Return` Valeur

Évaluation_Joueur_B (configuration)
Données: Configuration de jeu, c'est au joueur B de jouer
Résultat: valeur minimisant la perte de B
if configuration = feuille
    `// la configuration ne permet pas de jouer`
    `Return` Valeur (configuration)
else
    `// Le joueur B doit jouer`
    C=`Coup` (B,configuration)
    `// C Ensemble des coups jouables par B`
    Valeur = **vrai**
    forall the c∈ C do
        Valeur = `min` (Valeur, Évaluation_Joueur_A (`Jeu` (c,configuration))
    `Return` Valeur

JEUX ET ARBRES

ÉVALUATION DE L'ARBRE MINIMAX

Commentaires

- le joueur A et le joueur B ne sont pas symétriques
- maximisation du gain pour A (minimisation de la perte)
- parcours de tout l'arbre des parties : coût en $\mathcal{O}(b^p)$ avec b facteur de branchement et p profondeur de l'arbre
- arrêt à une profondeur d'exploration fixée : **évaluation heuristique**

Évaluation d'une configuration

- choix de la fonction heuristique : **LE PROBLÈME**
- arrêt garanti
- pas de preuve d'optimalité

UNIVERSITÉ Grenoble Alpes

Algorithme en horizon fini

```
Évaluation_Joueur_A (configuration,horizon)
    Données: Configuration de jeu, c'est au joueur A de jouer
    Résultat: meilleure valeur (si B joue parfaitement)
    if configuration = feuille ou profondeur = 0
        // la configuration ne permet pas de jouer, on évalue sa valeur
        Return(Valeur (configuration)) // fonction d'évaluation si ce n'est pas une feuille
    else
        // Le joueur A doit jouer
        C=Coup (A,configuration)
        // C Ensemble des coups jouables par A
        Valeur = −∞
        forall the c∈ C do
            Valeur = max (Valeur,Evaluation_Joueur_B (Jeu (c,configuration)),horizon -1)
        Return Valeur
```

Coût de l'algorithme *minimax* pour évaluer une configuration

- dépend du facteur de branchement b ;
- de la profondeur choisie p ;
- du coût du calcul de la valeur d'une configuration C.

$$\text{Coût} = C(b^{p+1} - 1)/(b - 1) \simeq Cb^p$$

ÉVALUATION DE LA VALEUR D'UNE CONFIGURATION : APPROCHE EMPIRIQUE

Fonction du **chemin** emprunté pour arriver à la configuration courante.
Les valeurs de configurations sont comparables.

- différence entre le gain cumulé de A et le gain cumulé de B
- estimation du meilleur gain de la configuration courante à l'objectif (cf A^*)
- valeur empirique d'une configuration (exemples)
  - nombre de pions / points / cases occupées-libres / tours ... comptage
  - séparation : somme de valeurs de sous-configuration + évaluation de la décomposition
  - configurations spécifiques (motifs gagnants ou forts)
  - ...
- Validation de la fonction valeur empirique
  - propriétés
  - test expérimentaux (faire jouer les stratégies l'une contre l'autre)
  - faire un plan d'expérience

$\Longrightarrow$ le moteur doit jouer des stratégies l'une contre l'autre : auto-apprentissage

ÉVALUATION DE LA VALEUR D'UNE CONFIGURATION : APPROCHE RANDOMISÉE

Valeurs de configurations entières $\Longrightarrow$ cas d'égalité :
choix randomisé du coup à jouer

avantages

- stratégie imprévisible pour l'adversaire
- en moyenne "devrait" être bon (empirique ! ! ! à tester)
- simple à mettre en œuvre

inconvénients

- reproductibilité (rejouer des parties pour analyser les choix)
- contrôle de l'aléatoire : l'uniformité est la garantie de non prévisibilté
  - générateur `Random` : qualité
  - générateur uniforme de coup (exemple)
  - méthode basée sur l'énumération
  - méthode basée sur le rejet
  - méthode basée sur le conditionnement
  - ...

Évaluation de la valeur d'une configuration : méthode de Monte-Carlo

Hypothèse : on dispose d'une stratégie S_0 (complexité de calcul simple)

```
Évaluation_Joueur_A (configuration,taille_echantillon)
    Données: Configuration de jeu, c'est au joueur B de jouer
    Résultat: probabilité de gain de A sous la stratégie S0 pour A et B
    Somme = 0
    for i=0 to taille_echantillon
        C=configuration
        Joueur = B
        while Fin de jeu
            C=Jouer(C,Joueur, S0)
            Joueur = adversaire(Joueur)
        Somme = Somme + Valeur(C)
    Return Somme / taille_echantillon
```

- la profondeur du jeu doit être faible (linéaire ou polynomiale)
- la précision dépend de la taille de l'échantillon (théorème central-limite)

$$\epsilon = \frac{\Phi\sigma}{\sqrt{\text{taille\_echantillon}}}$$

Φ associé au niveau de confiance, $\sigma = \sqrt{p(1-p)}$ l'écart-type

- voir $\alpha - \beta$, si l'estimation de la probabilité est proche de 0 ou 1 la taille de l'échantillon peut être réduite
- mélange minimax et méthode de Monte-Carlo

JEUX ET ARBRES

JEUX ET ARBRES