Programmation de Jeux

1 Principes des jeux

Les jeux considérés sont des jeux à deux joueurs qui jouent l'un contre l'autre selon des règles déterminées à l'avance. On peut étendre aisément à n joueurs, mais on se limitera à 2 joueurs pour simplifier. La position du jeu est l'état dans lequel se trouve la partie par exemple la position des pièces sur un échiquier. Nous ne considérons que des jeux dans lesquels les coups possibles à partir d'une seule position sont en nombre fini, plus précisément borné par une certaine valeur.

Arbre de Jeu : c'est la représentation des parties possibles sous forme d'un arbre : chaque position est un noeud et les fils de ce noeuds sont les positions atteignables à partir de cette position par le joueur qui a la main.

Si le nombre de coups d'une partie est fini (cas des échecs et de nombreux jeux) alors l'arbre de jeu est fini et on peut le construire explicitement : le jeu peut être complètement exploré (et n'a donc plus d'intéret). En pratique les arbres de jeux sont de taille très grande et il n'est pas possible de les construire effectivement.

Jeu à somme nulle : le gain du jeu par un joueur correspond à une perte de même valeur pour l'autre joueur. Une égalité -si ce cas est possible donne un gain de 0 à chaque joueur.

Les jeux ne sont pas tous à somme nulle : l'exemple du dilemme du prisonnier en est un.

A et B sont soupconnés d'avoir réalisé un vol et on propose à chacun d'eux le marché suivant :

- si tu dénonces ton complice tu seras libéré et lui condammé à 12 ans.
- si aucun ne dénonce l'autre, chacun est condamné à 1 ans.
- si chacun dénonce l'autre, chacun est condammé à 10 ans.

(A,B)	Dénonce	se Tait
Dénonce	(10,10)	(0,12)
se Tait	(12,0)	(1,1)

On peut noter qu'un participant a intérêt à dénoncer l'autre puisque dans tous les cas, il risque moins que s'il se tait (quel que soit ce que fait l'autre). Cela conduit à ce que chacun dénonce l'autre entrainant un coût supérieur (en nombre d'années à effectuer) pour l'ensemble des joueurs.

Stratégie de jeu : fonction calculable qui permet à un joueur de déterminer le coup à jouer à partir d'une position de jeu.

La stratégie peut également se calculer en prenant en compte l'historique ou partie de l'historique ayant mené à la position.

Stratégie gagnante : stratégie permettant d'assurer le gain de la partie (quelle que soit la manière de jouer de l'adversaire).

Fonction d'évaluation : la stratégie gagnante est souvent calculée avec une fonction qui permet d'évaluer une position de jeu indépendament de l'historique qui l'a construite.

Exemple : une fonction élémentaire au jeu d'echec est *la somme des valeurs des pièces blanches - somme des valeurs des pièces noires* avec des valeurs des pièces : pion : 1
cavalier : 3
fou : 4
tour : 6
reines : 10

Cette fonction ne prend pas en compte la position des pièces et elle n'est pas utilisable en pratique.

La théorie des jeux inventée par Von Neumann et Morgenstern a été très développée par la suite et est utilisée en économie et modélisation de comportements sociaux.

2 Programmation de jeu

2.1 Objectifs et difficultés

Objectif : programmer un joueur capable de rivaliser avec les meilleurs experts du domaine ou de les battre systématiquement.

Etat des lieux :

- Dame : aux dames anglaises, les programmes battent les experts et il n'y a plus de compétition entre machine et humains.
- Echec : en 1996, le champion du monde (Kasparov) a été battu par un programme implémenté sur une machine spéciale d'IBM (Deep Blue). Un autre concours a été organisé en 2006 entre le champion du monde (Kramnik) et une machine spécialisée (Deep Fritz) qui a remporté la confrontation.
- Go : les programmes sont d'un niveau assez bas. Un niveau grand maitre a été atteint récemment par un programme disposant de plusieurs coups d'avance.
- D'autres jeux sont résolus et d'autres sont très loin de l'être.

Les programmes explorent les arbres de jeu jusqu'à une certaine profondeur éventuellement en élaguant certaines branches (alpha-beta) et en combinant avec des règles d'expertise (par exemple pour les finales aux echecs).

2.2 Exemple

Le jeu de Nim (simplifié).

- On prend une rangée de n allumettes : $||||\dots|$
- Chaque joueur en prend une ou deux à chaque coup.
- Le perdant est celui qui ramasse la dernière[1].

Le gagnant gagne 1 euro et le perdant perd un euro.

Construction de l'arbre de Jeu :

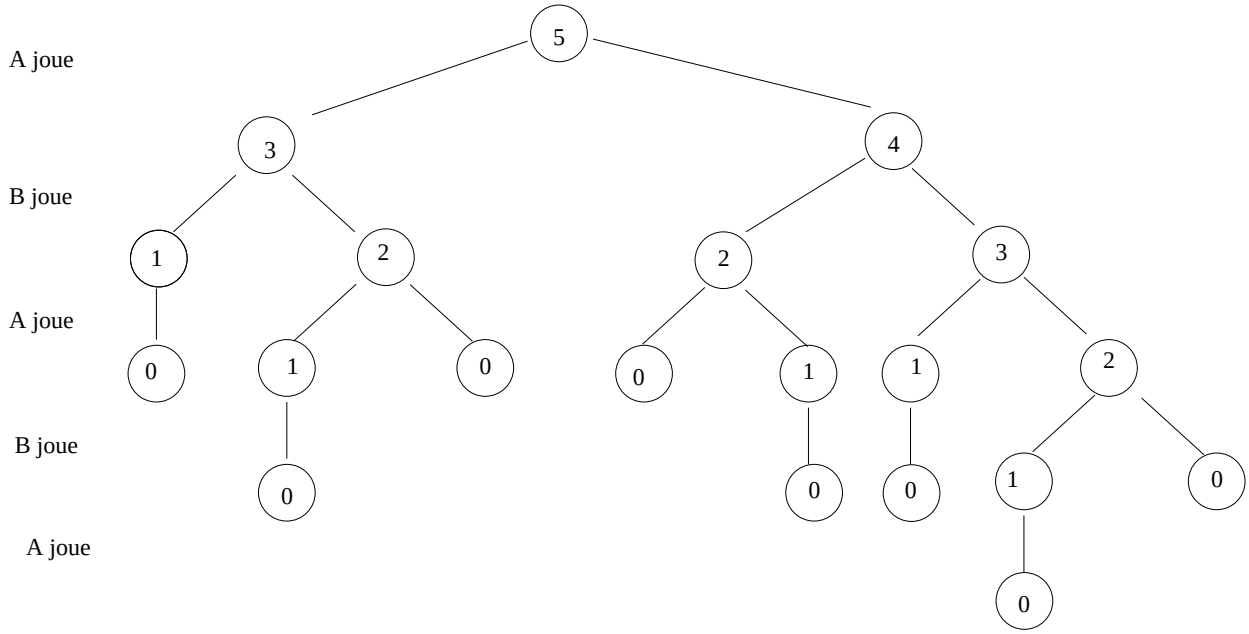

FIGURE 1 – Arbre de Jeu de Nim pour $n = 5$

1. dans d'autres versions le gagnant est celui qui ramasse la dernière allumette

Cette figure est un arbre, les ronds sont des noeuds et les traits sont des arêtes. Les noeuds accessibles directement depuis un noeud sont ses fils, et un noeud sans fils est une feuille.

2.3 Min-Max

L'arbre de jeu permet de donner une valeur aux positions de jeu depuis les feuilles de l'arbre de jeu.

Une position finale (0 allumette) se voie attribuer une valeur $+1$ si c'est le joueur B qui a joué, -1 sinon.

On remonte des feuilles à la racines ainsi :

1. n est un noeud dont tous les fils sont évalués à $+1$ ou -1.
2. Si le joueur qui joue depuis n est A, il cherche à choisir le coup l'amenant à obtenir le plus grand gain possible : la valeur de n est donc le maximum des valeurs des fils.
3. Si le joueur qui joue depuis n est B, il cherche à choisir le coup l'amenant à obtenir le plus grand gain possible pour lui, donc le plus petit gain possible pour A : la valeur de n est donc le minimum des valeurs des fils.

C'est un principe Min-Max (B minimise, A maximise) qui permet d'évaluer une position en prenant en compte le fait que l'adversaire joue du mieux possible. Le résultat sur le jeu de Nim est le suivant :

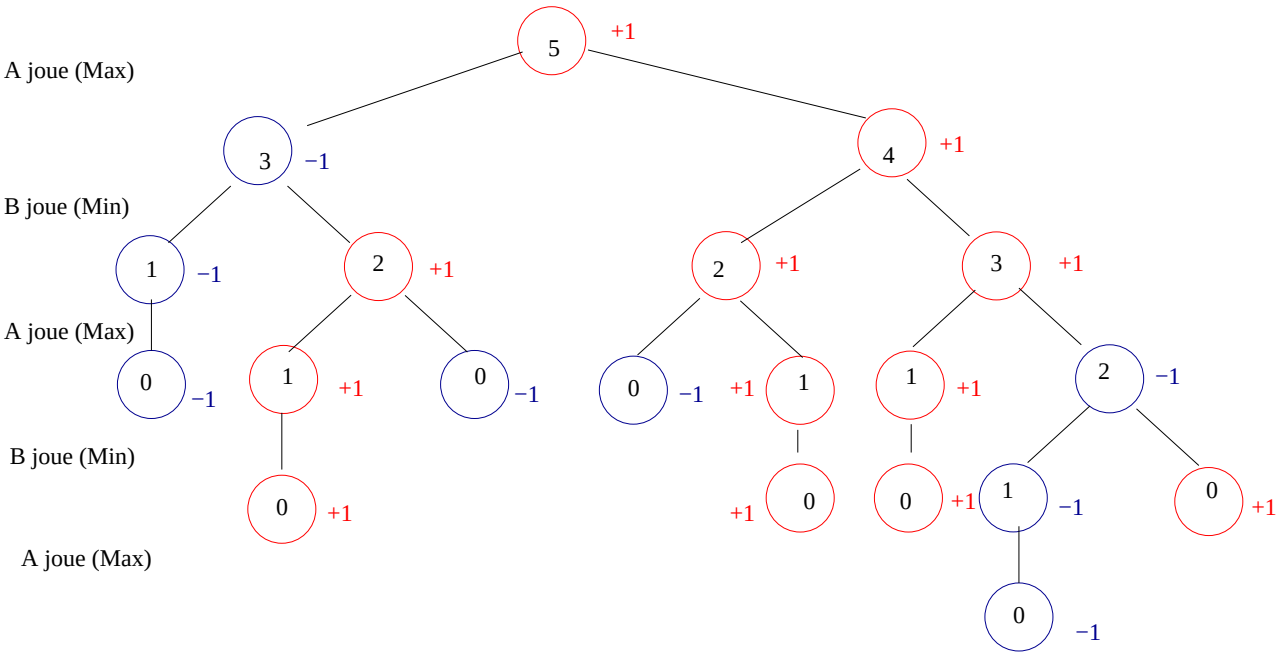

FIGURE 2 – Arbre de Jeu de Nim pour $n = 5$

Le principe de l'algorithme est donc de définir deux fonctions $eval_A(p)$ et $eval_B(p)$ qui évalue une position du point de vue du joueur A et du point de vue du joueur B.

$eval_A(p)$
si p est finale alors
renvoyer la valeur correspondante à p
(+1 si p est gagnante pour A, -1 sinon)
sinon
renvoyer $Max\{eval_B(p') \mid p' fils\ de\ p\}$
fsi

et la fonction $eval_B$ est la duale.

$eval_B(p)$
si p est finale alors
renvoyer la valeur correspondante à p
(-1 si p est gagnante pour B, +1 sinon)
sinon
renvoyer $Min\{eval_A(p') \mid p' \ fils \ de \ p\}$
fsi

Il est possible de combiner ces deux fonctions en une seule qui évalue la position p du point de vue du joueur qui joue en p et renvoie +1 si c'est une position gagnante et -1 sinon.

2.4 Programmation du jeu de Nim

Le programme suivant permet de jouer au jeu de Nim avec une stratégie optimale pour la machine basée sur l'algorithme Min-Max. Ce programme pourrait être amélioré pour prendre en compte certaines régularités de l'arbre de jeu.

```
def finale(p):
    return p==0

def nbsuc(p):
    if p==1:
        return 1
    else:
        return 2

def suc1(p):
    return p-1

def suc2(p):
    return p-2

def affiche(p):
    i=1
    ch=""
    while i<=p:
        ch=ch+'I'
        i=i+1
    print ch

#les fonctions clef d'évaluation par min/max
#evalA pour le joueur qui maximise
def evalA(p):
    "eval par joueur A de la position p"
    if finale(p):
        return 1
    elif nbsuc(p)==1:
        return evalB(suc1(p))
    else:
        return max(evalB(suc1(p)),evalB(suc2(p)))

#evalB pour le joueur qui minimise (l'adversaire)
def evalB(p):
    "eval par joueur B de la position p"
```

```
    if finale(p):
        return -1
    elif nbsuc(p)==1:
        return evalA(suc1(p))
    else:
        return min(evalA(suc1(p)),evalA(suc2(p)))

#la fonction de choix de la machine
def choix(p):
    "choix par la machine de la position à jouer"
    if nbsuc(p)==1:
        return 1
    elif evalB(suc1(p))> evalB(suc2(p)):
        return 1;
    else:
        return 2

#le jeu homme/machine
def jeu():
    print "nb allumettes"
    nball=input()
    while not finale(nball):
        if choix(nball)==1:
            nball=nball-1
        else:
            nball=nball-2
        print "je joue"
        affiche(nball)
        if finale(nball):
            print "perdu"
        else:
            couphumain=lecture(nball)
            nball=nball-couphumain
            if finale(nball):
                print "gagne"
            else:
                print "le jeu est "
                affiche (nball)
# fonction de service pour lire le coup
def lecture(n):
    "lire un nombre <=n à l'écran"
    couphumain=-1
    while couphumain<0 or couphumain >n:
        print "entrer coup 1 ou 2, inférieur ou égal à", n
        couphumain=input()
    return couphumain
```

3 Limites de l'analyse

La méthode précédente donne une méthode générique d'analyse de jeu fini. La seule limite est sa faisabilité qui est donnée par la taille de l'arbre de jeu.

Taille de l'arbre de jeu Estimation de la taille de l'arbre de jeu pour les échecs.

– Premier coup possible pour les blancs : avance 1 ou 2 cases pour les pions, 2 déplacements de chaque cavalier, soit 20 coups.
– Premier coup possible pour les noirs :idem.

L'arbre de jeu a donc $1 + 20 + 20 \times 20 = 421$ noeuds pour les deux coups initiaux.

Même si les possibilités de coups sont plus faibles ensuite, l'arbre de jeu est beaucoup trop important pour etre analysé en totalité. Si on suppose que le branchementest limité à 10 un arbre de hauteur 10 (qui correspond à 5 coups pour chaque participant) a $1+10+10^2+10^3+\ldots+10^{10} = (10^{11}-1)/(10-1)$ noeuds soit de l'ordre de 10 milliards de noeuds.

La conséquence est qu'il n'est pas possible d'analyser le jeu en partant des feuilles qui décrivent une position finale de gain ou perte comme dans l'algorithme Min-Max. L'arbre sera exploré jusqu'à une certaine profondeur et une feuille sera évaluée avec la fonction d'évaluation (sauf si elle correspond à un gain ou une perte). Cela implique que la fonction d'évaluation permet de bien exprimer si une position correspond à un avantage pour un joueur (comme le fait la fonction mentionnée plus haut).

L'intéret de l'algorithme est de pouvoir explorer le plus de coups possibles en avant et une manière de le faire est d'élaguer l'arbre de jeu en éliminant des positions qui ne sont pas utiles à examiner (et surtout d'éliminer les positions qui s'obtiennent de celles qu'on élimine).

Idée de l'élagage alpha-beta L'algorithme le plus utilisé est l'algorithme alpha-beta dont l'idée est expliquée sur l'exemple ci-dessous.

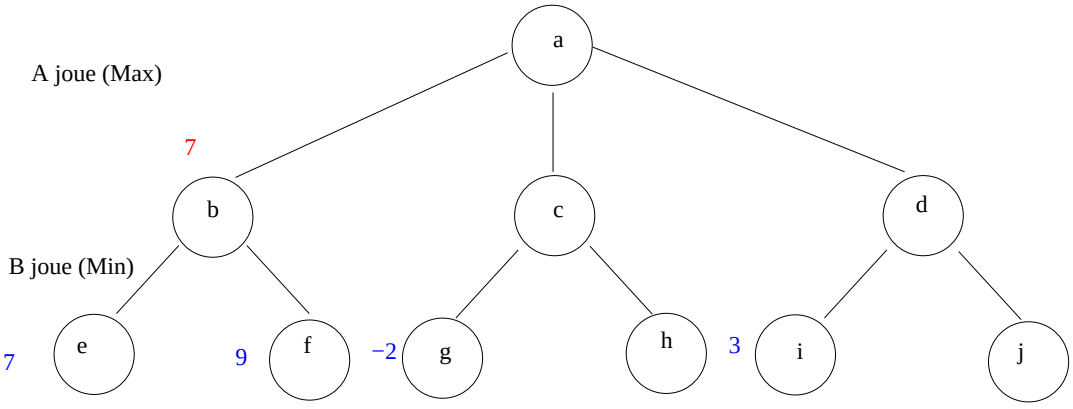

FIGURE 3 – Principe de l'alpha-beta

L'évaluation du noeud h est inutile : B minimise, donc la valeur du noeud père c sera inférieure à -2. Par contre A maximise et a déjà une valeur de 7 au noeud b, donc la valeur du noeud c ne sera pas utilisée car elle est ≤ -2. La valeur 7 du noeud b est utilisée comme valeur d'élagage α pour elaguer les noeuds dont la valeur ne pourra pas donner une valeur supérieur à α. De la même manière on peut utiliser une valeur β pour elaguer les noeuds dont la valeur ne contribuera pas à l'évaluation des positions par B.

Le principe est de remplacer $eval_A(p)$ par $evalA_{\alpha\beta}(p,\alpha,\beta)$ qui vérifie (I) :
$evalA_{\alpha\beta}(p,\alpha,\beta) = eval_A(p)$ si $\alpha \leq eval_A(p) \leq \beta$
$evalA_{\alpha\beta}(p,\alpha,\beta) = \alpha$ si $eval_A(p) < \alpha$
$evalA_{\alpha\beta}(p,\alpha,\beta) = \beta$ si $eval_A(p) > \beta$

(et symmétriquement pour $eval_B$ et une fonction $evalB_{\alpha\beta}(p,\alpha,\beta)$). L'appel initial se fera avec $evalA_{\alpha\beta}(p,-\infty,+\infty)$.

La fonction $evalA_{\alpha\beta}$ est donnée par :

$evalA(p, \alpha, \beta)$
si p finale alors la valeur de p selon le jeu
sinon $v = -\infty$
position atteignable depuis $p = [p_1, \ldots, p_n]$
i=1
Tant que $i \leq n$ faire :
$w = evalB_{\alpha\beta}(p_i, \alpha, \beta)$
$v = Max(v, w)$
si $v > \beta$ alors
retourner β
sinon $\alpha = Max(\alpha, v)$
$i = i + 1$
fsi
fait

la fonction $evalB_{\alpha\beta}$ est similaire (remplacer Max par Min et la mise à jour de α par la mise à jour de β). On montre que ces fonctions satisfont la relation (I) en montrant que si les appels récursifs satisfont (I) alors les appels principaux satisfont (I).

4 Stratégies

Stratégie gagnante au jeu de Nim (généralisé) Le jeu de Nim original consiste à avoir plusieurs rangées d'allumettes, le coup d'un joueur est de choisir une rangée et de prendre un nombre quelconque strictement positif d'allumettes sur cette rangée. Le perdant est celui qui ramasse la dernière allumette.

Il existe une stratégie gagnante pour le joueur qui commence si la position de départ satisfait une propriété C_I (comme configuration Impaire).

On écrit le nombre d'allumettes de chaque colonne en binaire et on considère le tableau obtenu en écrivant ces nombres l'un sous l'autre, en complétant à gauche par des 0 si nécessaire. Les colonnes sont numérotées $0, 1, \ldots$ en partant de la droite vers la gauche. Pour chaque colonne k on calcule $S(k)$ l'addition des chiffres de la colonne. La configuration est une configuration Paire dite C_P si tous les $S(k)$ valent 0 modulo 2, ou bien toutes les lignes contiennent une seule allumette et il y a un nombre impair de rangées (pour tous $k > 0$, $S(k) = 0$ et $S(0) = 2p+1$). Une configuration qui n'est pas C_P est dite C_I (Impaire).

Exemple :

nombre d'allumettes	représentation	Ce qui donne $S(2) = 1, S(1) = 1, S(0) = 0$ modulo 2
3	011	
5	101	
	112	

donc la configuration est C_I.

La propriété essentielle est la suivante : Partant d'une configuration C_I on peut amener à une configuration C_P, et tout coup joué d'une configuration C_P amène en C_I.

Sur la configuration précédente, en enlevant deux allumettes en ligne deux on obtient 3 et 3 allumettes, donnant 022 donc $S(2), S(1), S(0)$ sont pairs et la configuration est C_P.

C_P donne C_I : Si C_P n'a que des rangées d'une allumette, on supprime une rangée donc le nombre de rangées devient pair. Sinon dans la rangée dans laquelle on enlève des allumettes, un nombre est remplacé par un autre plus petit. Dans la représentation binaire, en lisant de gauche à droite, les chiffres sont inchangés jusqu'à ce que un 1 soit remplacé par un 0. Par conséquent la valeur $S(k)$ de cette colonne change de parité et on a une configuration C_I.

C_I donne C_P : Si toutes les rangées sauf une ont une allumette. Si le nombre de rangées est impair alors enlever toutes les allumettes de la rangée ayant plus d'une allumette sinon supprimer la rangée. Sinon il y a au moins deux rangées avec plus d'une allumette. Donc il y a une colonne k avec $k > 1$ et $S(k)$ impair. Prendre la colonne la plus à gauche qui donne un $S(k)$ impair. Prendre la rangée r qui correspond au plus grand nombre ayant un 1 en colonne k (il en existe

un nécessairement). Remplacer le nombre n correspondant à un nombre n' construit ainsi : en position $k' > k$, n' a le même chiffre que n, a 0 en colonne k, et pour $k' < k$, n' a le même chiffre que n en colonne k' si $S(k')$ est pair, et a le chiffre complémentaire si $S(k')$ est impair. Par construction, $n' < n$ et toutes les valeur $S(k)$ sont paires.

Le joueur qui commence avec une position C_I devra amener systématiquement son adversaire en position C_P ce qui est possible. L'adversaire sera finalement amené à une position avec une allumette sur une seule ligne, donc il aura perdu.

Jeu ayant une stratégie gagnante inconnue. Le jeu du mange savon généralisé consiste à jouer avec une tablette de chocolat dont le carreau supérieur gauche est un carré de savon. Un coup d'un joueur consiste à choisir un carreau et supprimer le quadrant SW de la tablette correspondant.

Comme le jeu de Nim, ce jeu est fini et le nombre de coups possibles à chaque étape est fini. Il n'y a pas de position d'égalité et un arbre de jeu permet de calculer pour chaque position si elle est perdante ou gagnante pour le joueur qui joue.

Le joueur qui commence a une stratégie gagnante mais elle n'est pas connue. Si ce joueur enlève le carreau inférieur droit :

1. Soit ce coup fait partie d'une stratégie gagnante pour lui.
2. Soit ce coup amène sur une position gagnante pour B. Mais alors le premier coup que B joue avec sa stratégie gagnante aurait pu être joué directement par A (tout coup contient le carreau inférieur droit). Par conséquent A a un coup qui l'amène dans une position gagnante.

Donc A a une stratégie gagnante mais ne sait pas laquelle. On peut noter que si on part d'une tablette carrée A a une stratégie gagnante simple à décrire :

1. Son premier coup est d'enlever le carré intérieur le plus grand possible ne contenant pas le savon.
2. Il reste alors une bande horizontale et une bande verticale de même longueur chacune d'épaisseur 1, dont l'intersection est le carreau de savon. La stratégie de A est alors de jouer le coup de B sur l'autre bande.

Chaque coup de B se fait dans une position avec deux bandes de même longueur et s'intersectant sur le savon. A un moment B joue en prenant le dernier carreau de chocolat d'une bande, A duplique ce coup et B est contraint de prendre le carreau de savon.