Université Paris-Dauphine

Mémoire d'initiation à la Recherche

Cycle Pluridisciplinaire d'Etudes Supérieures

Etude des Jeux Impartiaux

Auteur :
Auguste Lehuger

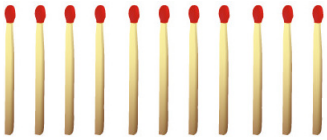
Sous la direction de :
M. Guillaume Vigeral

Table des matières

1 Introduction

Mon stage de recherche porte sur l'étude des jeux impartiaux qui est une vaste catégorie de jeu, allant d'un simple jeu de soustraction à des jeux bien plus complexes. Un jeu impartial se caractérise comme un jeu à 2 joueurs où les deux joueurs jouent à tour de rôle et dont les mouvements possibles ne dépendent pas du joueur qui joue. Le perdant est celui qui ne peut plus jouer.

Ainsi, le jeu d'échec n'est pas un jeu impartial car chaque joueur ne peut jouer que sa couleur. Le jeu impartial le plus élémentaire est le jeu de soustraction. On se dote de 11 allumettes et on peut en retirer 1, 2 ou 3. Il est facile de comprendre que ce jeu est **résoluble**, c'est à dire qu'à chaque état du jeu (il y a 11 allumettes dans l'état initial) il existe un unique joueur gagnant, qui possède une stratégie permettant de gagner à coup sûrs. Dans notre jeu de soustraction à 11 allumettes, le joueur 1 est gagnant : il doit faire en sorte que J_2 ait $n \equiv 0 \pmod 4$ allumettes à chaque coup et retire donc 3 allumettes au premier coup.



En fait, l'ensemble des jeux impartiaux est résoluble à l'aide d'outils théoriques puissants et possède dans de nombreux cas, une forme de périodicité toute particulière qu'il est intéressant d'analyser. Nous tenterons d'analyser les principaux jeux impartiaux en partant du jeu le plus simple, le jeu de soustraction, à un jeu plus complexe : le Dawson's Chess.

L'objectif de mon mémoire est d'assimiler la formalisation de ces jeux, de comprendre les théorèmes mathématiques qui peuvent s'appliquer et mettre en place, par programmation, des applications pour différents jeux impartiaux. Nous pourrons voir dans un dernier temps, les défis modernes que soulèvent les jeux impartiaux sur le plan mathématique et informatique.

2 Notions de bases, étude du jeu de soustraction

Dans cette section, nous allons établir une série de définitions formelles pour pouvoir étudier le jeu de soustraction et ainsi établir les bases de l'étude des jeux impartiaux.

2.1 Définition

Définition 1 (Jeu). *Un jeu* Γ *est un triplet* $\Gamma := (N, (S^i)_{i\in N}, (g^i)_{i\in N})$ *où* N *désigne l'ensemble des joueurs,* S^i *la stratégie du joueur* i *et* g^i *la fonction de paiement associée au joueur* i.

Définition 2 (Jeu Impartial Combinatoire). *Un jeu impartial combinatoire est un jeu à somme nulle, où l'ensemble des stratégies est fini et se termine en un temps fini (il n'y a pas de boucle possible), joué en pur avec condition normale de victoire (celui qui ne peut plus jouer perd). Ce sont des jeux sans hasard, en information parfaite, qui peuvent être décrit pas l'ensemble des successeurs légaux du jeu et un état de départ.*

Le Jeu de Soustraction est un jeu impartial combinatoire constitué d'un tas d'allumettes. Le Jeu de Soustraction se caractérise par un nombre N d'allumette et un n-uplet R de règle qui définit combien d'allumettes un joueur peut retirer à chaque tour.
Dans notre exemple, N=11 et R={1,2,3}

Définition 3 (Etat). *Un Etat du jeu* Γ *est une configuration du plateau de jeu et un joueur qui s'apprête à jouer.*

Pour le jeu de soustraction, le couple (n,J_1) désigne que J_1 doit jouer son tour alors qu'il reste n allumettes sur le plateau de jeu.

Définition 4 (Positions P et N). *A chaque état d'un jeu résolu, le joueur qui doit jouer peut-être dans un position* $\mathscr{P}$ *(perdante) ou un position* $\mathscr{N}$ *(gagnante).*

2.2 Analyse

Proposition 1. *Les états* $\mathscr{P}$ *et* $\mathscr{N}$ *forment une partition de l'ensemble des états d'un jeu.*

Démonstration. Pour déterminer dans quelle position est un état du jeu, il suffit de voir le jeu sous forme extensive (un arbre) et d'implémenter l'algorithme suivant :
1) Toutes positions terminales (on ne peut plus jouer) sont des positions $\mathscr{P}$.
2) Toutes positions qui atteignent une position $\mathscr{P}$ sont des positions $\mathscr{N}$.
3) Toutes positions qui ne vont que vers des positions $\mathscr{N}$ sont des positions $\mathscr{P}$.
4) Si aucune position $\mathscr{P}$ n'a été trouvée en 3) s'arrêter, sinon retourner en 2). □

Considérons le jeu de soustraction avec R ={1,3,4}. On cherche à obtenir les positions des différents états du jeu (caractérisé par le nombre n d'allumettes).
1) 0 est une position $\mathscr{P}$.
2) 1,3 et 4 atteignent une position $\mathscr{P}$ (0) donc sont des positions $\mathscr{N}$.
3) 2 ne va que vers des positions $\mathscr{N}$ (1) donc est une position $\mathscr{P}$.
2) 5 et 6 atteignent une position $\mathscr{P}$ (4 et 3) donc sont des positions $\mathscr{N}$.
3) 7 ne va que vers des positions $\mathscr{N}$ (3,4,6) donc est une position $\mathscr{P}$ ...
On peut continuer à l'infini par induction, on obtient finalement l'ensemble des nombres congrus à 0 ou 2 modulo 7.

Pour tout jeu de soustraction, on peut définir la suite O(n) qui à une configuration n, associe 1 si c'est une position N et 0 sinon. Pour le jeu de soustraction, cette suite peut-être définie récursivement : si un des successeurs i de l'état n est tel que $O(i) = 0$ alors $O(n) = 1$ sinon $O(n) = 0$ C'est au sens de cette suite que l'on peut définir une périodicité pour le jeu de soustraction.

Définition 5 (Période et prépériode en fonction de O(n))**.** *Soit* Γ *un jeu de soustraction,* Γ *est périodique de période* p *et de prépériode* n_0 *si,* $\forall n \geq n_0$*, on a :*

$$O(n+p) = O(n)$$

Proposition 2. *Tout jeu de soustraction est périodique en terme de la suite O(n).*

Démonstration. Soit $\Gamma = \{a_1, .., a_k\}$ un jeu de soustraction fini. Supposons les a_i triées dans l'ordre croissant.
$\exists\ 2^{a_k}$ suites différentes de a_k termes dans $\{0, 1\}$
Ainsi, parmis ces $2^{a_k} + 1$ suites :
- $(O(0), ..., O(a_k - 1))$
- $(O(1), ..., O(a_k))$
- ...
- $(O(2^{a_k}), ..., O(2^{a_k} + a_k - 1))$

au moins 2 sont identiques.
Supposons qu'une commence à p et l'autre à $p + T$.
Donc :
- $O(p) = O(p + T)$
- ...
- $O(p + a_k - 1) = O(p + T + a_k - 1)$

Regardons ce qu'il se passe au rang suivant, on notera F(y) les successeurs de l'état y : Comme la valeur suivante dépend au maximum des a_k dernières valeurs prises par la suite et que ces dernières sont identiques alors :

$$O(p + a_k) = O(p + T + a_k)$$

Ainsi, à partir du rang p, la suite $O(n)$ est périodique de période T. □

2.3 Application numérique

On a montré dans la partie analyse, l'existence d'une période et d'une prépériode pour tout jeu de soustraction fini. On a pour objectif dans cette partie de calculer les premiers termes de la suite O(n) et de constater la régularité concrètement. On a de plus, la périodicité des nombres de Sprague-Grundy pour un jeu de soustraction, notion qui est développée dans la suite de l'exposé pour plus de clarté et qui est démontrée en annexe car elle ressemble beaucoup à la preuve pour O(n). On a donc simulé les 100 premiers termes de la suite SG(n) pour le jeu de soustraction {1,3,7,8}

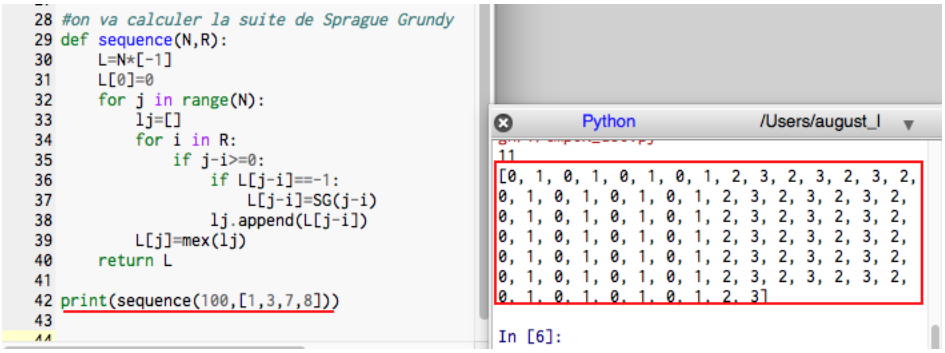


FIGURE 1 – Simulation de la suite SG(n) pour un jeu de soustraction. (cf. Code complet dans les annexes).

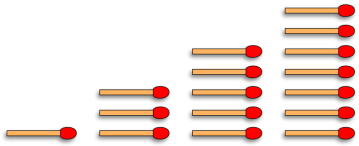
On constate effectivement une périodicité de la suite de SG(n) sans même de prépériode : [0, 1, 0, 1, 0, 1, 0, 1, 2, 3, 2, 3, 2, 3, 2]

3 Complexification, étude du jeu de Nim

Dans la première partie, nous avons étudié le jeu de soustraction qui est un jeu assez simple pour ne pas avoir à introduire de nouveaux outils théoriques pour sa résolution. Considérons dans un second temps, un jeu un peu plus complexe dont la résolution est moins triviale et nécessite une formalisation et une étude mathématique : le Jeu de Nim.

3.1 Définition

Le Jeu de Nim est un jeu impartial combinatoire constitué de plusieurs tas d'allumettes possédant chacun un nombre quelconque d'allumettes. Les joueurs jouent à tour de rôle en choisissant un tas et en retirant autant d'allumettes que souhaité dans le tas. C'est en cela que le jeu de Nim n'est pas une généralisation du jeu de soustraction. Un état dans un jeu de Nim à n tas est un n-uplet qui caractérise la taille de chaque tas.



Définition 6 (Somme de deux jeux). *Si l'on considère A et B, deux jeux impartiaux combinatoires, alors A+B est encore un jeu impartial combinatoire où, à chaque état, le joueur peut jouer dans A ou dans B. Formellement, Succ(A+B) := {A + x, x ∈ Succ(B)} U {B+y, y ∈ Succ(A)}*

Le Jeu de Nim à 2 tas peut être vu comme la somme de 2 jeux de soustraction triviaux où l'on peut retirer autant d'allumettes que l'on veut. Or si l'on sait que les deux tas sont dans une position $\mathscr{N}$, quand est-il de la somme des 2 jeux ?
Par exemple, si l'on a deux tas de une allumette, on est dans une position $\mathscr{P}$ et les deux tas sont dans des position $\mathscr{N}$. Alors que si l'on a un tas de une allumette et un tas de deux allumettes, on est dans un position $\mathscr{N}$ alors que les deux tas sont aussi dans un position $\mathscr{N}$. Par conséquent, la somme de deux positions n'est pas bien définie. Cet exemple, montre la limite de la notion de position $\mathscr{P}$ et $\mathscr{N}$ et légitime l'introduction d'outils théoriques.

Pour analyser le jeu de Nim à plusieurs tas, il faut le voir comme un jeu plus simple qui lui est équivalent. On a donc besoin d'une définition formelle d'équivalence de jeu impartiaux.

Définition 7 (Equivalence de deux jeux). *Si l'on considère A et B, deux jeux impartiaux combinatoires, on dit que* $A \sim B$ *ssi* $\forall$ *jeu impartial C, on a :* $O(A+C) = O(B+C)$

Pour manipuler cette notion d'équivalence, on va devoir définir une nouvelle addition qui va nous être très utile par la suite.

Définition 8 (Addition binaire modulo 2). *On définit une addition sur* $\mathbb{N}$ *qui consiste à sommer modulo 2 les écritures binaires de deux entiers. On note* $\oplus$ *cette addition. Par exemple,* $5 \oplus 3 = 6$

Propriétés : $\oplus$ sur $\mathbb{N}$ définit un groupe additif abélien car :
— 0 est l'élément neutre : $a \oplus 0 = a$

— Commutativité : $a \oplus b = b \oplus a$
— Assossiativité : $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
— Tout entier est inversible et est son propre inverse : $a \oplus a = 0$

3.2 Analyse

Lemme 3. *Si 2 jeux A et B sont dans des positions différentes alors leur somme est dans une position gagnante pour le joueur 1.*

Démonstration. Il suffit de jouer dans le jeu à position gagnante, alors le joueur suivant est obligé de jouer dans un jeu avec position perdante et donc remet ce même jeu en position gagnante. Le joueur a ainsi toujours un coup gagnant à jouer et finit donc par gagner. □

Théorème de Bouton Le théorème de Bouton démontre que tout jeu de Nim est équivalent à un jeu de Nim trivial à un tas. Ce tas est vide si l'on est dans un position perdante et non vide (de taille quelconque) si l'on est dans un position gagnante.

Lemme 4. $a \oplus x = 0 \Rightarrow a = x$

Démonstration. Cela revient à dire que a et x ont la même décomposition binaire qui est unique donc sont égaux. □

Théorème 5 (L.Bouton, 1902)**.** *Un état $(x_1, ..., x_n)$ dans le jeu de Nim est une position $\mathscr{P}$ si et seulement si $x_1 \oplus x_2 \oplus ... \oplus x_n = 0$, où $\oplus$ est l'addition binaire modulo 2.*

Démonstration. On désigne par P l'ensemble des positions de nim-somme nulle et N l'ensemble complémentaire. Montrons que P et N vérifient la caractérisation de $\mathscr{P}$ et $\mathscr{N}$.

— *Toutes les positions terminales sont dans P* :
(0,0,...,0) seule position terminale et $0 \oplus 0 \oplus ... \oplus 0 = 0$.

— *De toutes N-positions, il y a un mouvement vers une P-position* :
Soit $(x_1, ..., x_n) \in N$, on enlève x à n'importe quel tas qui le permet tel que $x_1 \oplus x_2 \oplus ... \oplus x_n \oplus x = 0$; en base 2, x aura un $1 * 2^k$ à chaque fois qu'il y aura un nombre impair de $1 * 2^k$ dans la décomposition des autres nombres $(x_1, ..., x_n)$ Si $x > x_i \, \forall i$ alors on considère x^* un maximum et on retire à x^* : $x \oplus x^* < x^*$ qui est donc un mouvement légal.

— *Tout mouvement d'une P-position amène à une N-position* : Soit $(x_1, ..., x_n) \in P$, on note $a := x_2 \oplus ... \oplus x_n$. On a : $x_1 \oplus a = 0$
On change x_1 en x_1'.On ne peut pas avoir $x_1' \oplus a = 0$ car on aurait $x_1 = a = x_1'$(cf. Lemme 5) et donc le mouvement ne serait pas légal.

□

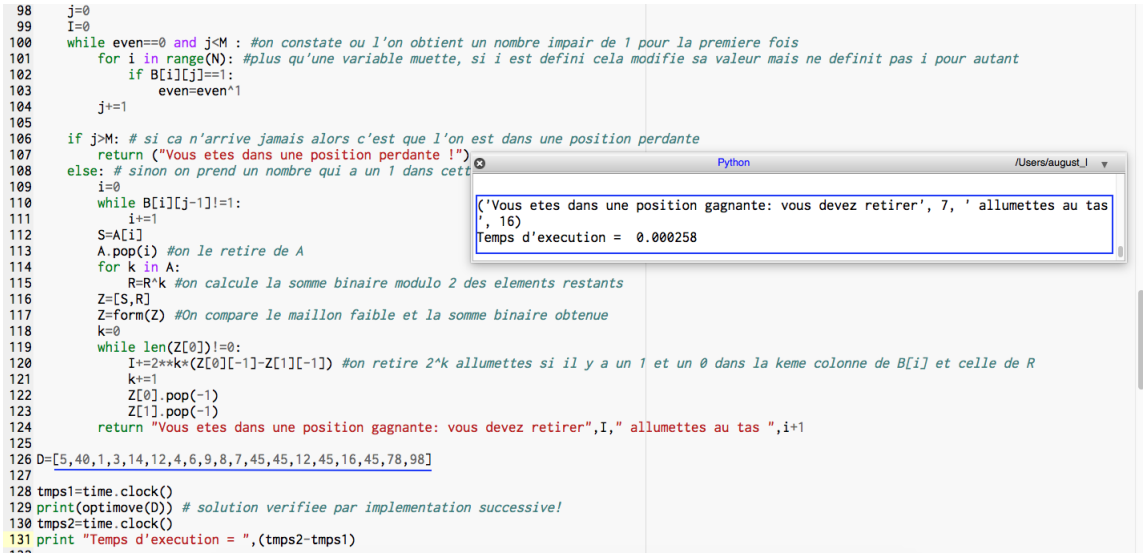
Grâce au théorème de Bouton, on a montré exactement que :

$$Nim(x_1) + ... + Nim(x_k) \sim Nim(x_1 \oplus ... \oplus x_k)$$

Ainsi, comme annoncé, on peut voir le jeu de Nim avec pour état un n+1-uplet (configuration, joueur) ou plus simplement comme le jeu équivalent dont l'état est juste un couple. La configuration de tout jeu de Nim est alors un entier, appelé **Nimber**. On peut définir de la même manière la suite O(n). Dans ce cas, la suite est triviale : elle vaut 0 pour un Nimber nul et 1 pour un Nimber non nul.

3.3 Application numérique

Si la suite O(n) n'a pas d'intérêt pour le jeu de Nim, la simulation numérique va néanmoins être très utile. Comme l'on sait que chaque position est perdante ou gagnante, cela signifie que si la position est gagnante, on peut trouver une stratégie qui nous permet de gagner à coup sûrs. Notre objectif est donc de programmer un algorithme qui renvoie rapidement dans quel position on est et le mouvement à effectuer si l'on est en position gagnant. Le programme ainsi présenté permet de calculer le coup optimal avec 19 tas en moins d'un millième de seconde. Une capture d'écran montre les résultats et une partie du code, le reste figurant dans les annexes.




```
    j=0
    I=0
    while even==0 and j<M : #on constate ou l'on obtient un nombre impair de 1 pour la premiere fois
        for i in range(N): #plus qu'une variable muette, si i est defini cela modifie sa valeur mais ne definit pas i pour autant
            if B[i][j]==1:
                even=even^1
        j+=1

    if j>M: # si ca n'arrive jamais alors c'est que l'on est dans une position perdante
        return ("Vous etes dans une position perdante !")
    else: # sinon on prend un nombre qui a un 1 dans cett
        i=0
        while B[i][j-1]!=1:
            i+=1
        S=A[i]
        A.pop(i) #on le retire de A
        for k in A:
            R=R^k #on calcule la somme binaire modulo 2 des elements restants
        Z=[S,R]
        Z=form(Z) #On compare le maillon faible et la somme binaire obtenue
        k=0
        while len(Z[0])!=0:
            I+=2**k*(Z[0][-1]-Z[1][-1]) #on retire 2^k allumettes si il y a un 1 et un 0 dans la keme colonne de B[i] et celle de R
            k+=1
            Z[0].pop(-1)
            Z[1].pop(-1)
        return "Vous etes dans une position gagnante: vous devez retirer",I," allumettes au tas ",i+1

D=[5,40,1,3,14,12,4,6,9,8,7,45,45,12,45,16,45,78,98]

tmps1=time.clock()
print(optimove(D)) # solution verifiee par implementation successive!
tmps2=time.clock()
print "Temps d'execution = ",(tmps2-tmps1)
```

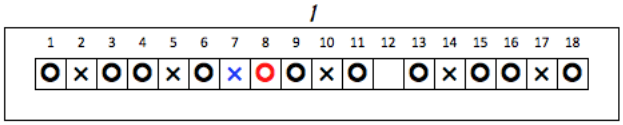
FIGURE 2 – Réponse optimale à un jeu de Nim à 19 tas.

4 Jeux Take-n-Break, étude du Dawson's chess

Le jeu de Nim est un jeu central dans la théorie des jeux impartiaux combinatoires mais est loin d'être le jeu le plus général. On va considérer un troisième jeu qui est un illustre représentant des jeux Take-n-Break. Un jeu Take-n-Break est une variante du jeu de Nim où l'on a la possibilité, en retirant des allumettes, de séparer un tas en 2 tas selon certaines règles.

4.1 Définition

Le Dawson's Chess est un jeu impartial combinatoire type Take-n-Break qui peut-être vu comme un morpion unidimensionnel. Il y a un nombre fini de cases (ici 18) où les joueurs peuvent tracer des croix. Une croix peut-être placée si la case et ses cases adjacentes sont vides. Les cases adjacentes sont aussi barrées (3 cases sont barrés par tour ou 2 si l'on place la croix dans une extrêmité).



La classe des jeux Take-n-Break est-elle encore résoluble, peut-on voir un état comme un simple couple de la même manière que pour le jeu de Nim ? Pour répondre à ces questions, il faut d'abord voir le Dawson's chess comme un Jeu de Nim où l'on peut séparer un tas en deux en retirant des allumettes. Il faut ensuite comprendre pourquoi nos outils actuels ne permettent pas de répondre à notre problème.

Un état est décrit de la même façon pour le Dawson's chess et pour le Jeu de Nim mais les règles sont différentes. Bouton ne peut pas s'appliquer car il ne prend pas en compte le changement de règles et la nouvelle variété de choix qui s'offre aux joueurs. On cherche alors une notion plus complexe qui s'intéresserait aux successeurs des différents états.

Définition 9 (Mex)**.** *Le mex est une opération sur une liste qui renvoie le plus petit entier qui n'est pas compris dans la liste. eg : $mex([0,3,4]) = 1$*

Définition 10 (Fonction de Sprague-Grundy)**.** *La fonction de Sprague-Grundy est une fonction g à valeur entière définie récursivement sur l'ensemble des états. Soit x un état :*

$$g(x) := mex\{g(y), y \in F(x)\} \quad , \text{ où } F \text{ est la fonction successeur.}$$

Le calcul de cette fonction semble très fastidieux dans la mesure où l'on doit parcourir un très grand nombre d'état pour calculer le Sprague-Grundy de l'état initial. Néanmoins,

l'utilité de cette fonction est assez claire, les états où g est nulle sont des positions $\mathscr{P}$ et les autres sont des positions $\mathscr{N}$. On peut ainsi savoir dans quel état on est pour n'importe quel jeu impartial.

Proposition 6. *Un état d'un jeu impartial est dans une position P ssi sa valeur de Sprague-Grundy est nulle.*

Démonstration. On peut vérifier aisément que :
- Pour tout état terminal, g est nulle.
- Si $g(x) = 0$ alors les successeurs y de x sont tels que $g(y) \neq 0$.
- Si $g(x) \neq 0$ alors il existe un successeur y de x tel que $g(y) = 0$.

□

4.2 Analyse

L'objectif ici est d'utiliser ces notions pour un établir des résultats forts sur l'ensemble des jeux impartiaux cette fois puisque la notion de mex et de fonction de Sprague-Grundy sont générales. Nous allons énoncer deux théorèmes, dits de Sprague-Grundy. Le premier théorème de Sprague-Grundy caractérise la fonction de Sprague-Grundy d'une somme de jeu impartiaux combinatoires.

Théorème 7 (Sprague-Grundy I, 1935-1939)**.** *Si g_i est la fonction de Sprague-Grundy de Γ_i, alors $\Gamma := \Gamma_1 + ... + \Gamma_n$ a pour fonction de Sprague-Grundy $g(x_1, ..., x_n) = g_1(x_1) \oplus ... \oplus g_n(x_n)$*

Démonstration. Soit $(x_1, ..., x_n) \in$ X. $b := g_1(x_1) \oplus ... \oplus g_n(x_n)$. Nous allons montrer deux choses pour la fonction g :
- $\forall a < b$, il existe un successeur de $(x_1, ..., x_n)$ dont l'image par g vaut a.
- Aucun successeur de $(x_1, ..., x_n)$ possède b pour image par g.

1) Soit a $< b$, $d := a \oplus b$ et k le nombre de bits dans la décomposition binaire de d. On a : $2^{k-1} \leq d < 2^k$. Comme a<b, b a un 1 en k-ième position alors que a non. Comme $b := g_1(x_1) \oplus ... \oplus g_n(x_n)$, il existe i tel que $g_i(x_i)$ a un 1 en k-ième position. Supposons que $i = 1$, alors $g_1(x_1) \oplus d < g_1(x_1)$ et donc il existe un état x'_1 tel que $g_1(x'_1) = g_1(x_1) \oplus d$ atteignable de x_1 d'après la définition de la fonction de Sprague-Grundy. On a alors :

$g_1(x'_1) \oplus g_2(x_2) \oplus ... \oplus g_n(x_n) = d \oplus g_1(x_1) \oplus ... \oplus g_n(x_n) = d \oplus b = a$(d'après les propriétés de $\oplus$)

2) Supposons qu'il existe un successeur $(x'_1, ..., x_n)$ dont l'image par g vaut b et qu'il implique un mouvement dans le 1er jeu. Alors $g_1(x'_1) \oplus ... \oplus g_n(x_n) = g_1(x_1) \oplus ... \oplus g_n(x_n)$ donc $g_1(x'_1) = g_1(x_1)$ d'après propriétés. Absurde, car un état ne peut pas avoir la même valeur qu'un de ses successeurs par définition du mex. □

Ce théorème réduit l'analyse de la somme de plusieurs jeux à l'étude plus simple des jeux qui compose la somme. Le Sprague-grundy de la somme de jeux est la somme ($\oplus$) des Sprague-grundy des jeux qui la compose.

Proposition 8. *La valeur de Sprague-Grundy d'un jeu de Nim à un tas de taille m vaut m pour tout entier m.*

Démonstration. Montrons le résultat par récurrence forte :
$m = 0$: Un jeu de Nim vide est perdant donc de SG-valeur nulle.
Supposons par récurrence forte l'hypothèse de récurrence.
Alors $g(m+1) := mex\{g(y), y \in F(x)\} = mex\{1, ..., m\} = m+1$ car dans un jeu de Nim de taille $m+1$ on peut retirer entre 1 et m+1 allumettes donc les états successeurs sont toutes les tailles entre 0 et m dont on connaît la SG-valeur d'après l'hypothèse de récurrence forte. □

Le deuxième théorème de Sprague-Grundy démontre que tout jeu impartial est équivalent à un jeu de Nim trivial à un tas. Ce tas est de taille n, appelé Nimber qui vaut la SG-valeur du jeu impartial. Il est vide si l'on est dans un position perdante et non vide si l'on est dans un position gagnante.

Lemme 9. *$A \sim B \Leftrightarrow A+B$ perdant.*

Démonstration. On va raisonner par double implication. Soit A, B des jeux impartiaux.
$\Rightarrow$: On a donc $O(A+B) = O(B+B)$ Or $B+B$ est perdant à coup sûrs car il suffit pour le second joueur d'imiter, à chaque coup, le mouvement du joueur 1. Ainsi $O(A+B)$ est la suite nulle donc $A+B$ est perdant.
$\Leftarrow$: Raisonnons par contraposée,
Il existe C, jeu impartial tel que $O(A+C) \neq (B+C)$. Sans perte de généralité, on suppose que $A+C$ est dans une position gagnante (N) et $B+C$ est dans une position perdante (P). Alors $(A+C)+(B+C)$ est dans une position gagnante car $P+N = N$ d'après lemme. Et comme, $A+B+C+C \sim A+B$ car si un joueur joue dans C, suffit de jouer le même mouvement dans l'autre jeu C, alors A+B est gagnant. □

Théorème 10 (Sprague-Grundy II, 1935-1939)**.** *Tout jeu $\Gamma \sim Nim(SG(\Gamma))$*

Démonstration. Soit n $\in \mathbb{N}$. On considère Γ, un jeu de SG-valeur n. On raisonne par rec et on suppose le resultat connu pour tous les successeurs de Gamma. On montre alors que le jeu $\Gamma + *n$ est dans une position perdante. Si le joueur 1 joue dans $*n$ et laisse m allumettes alors on joue dans Γ vers un jeu Γ' de SG-valeur m (qui existe par définition de la SG-valeur). Si le joueur 1 joue dans Γ vers un jeu Γ' de SG-valeur m<n alors on laisse m allumettes dans le jeu *n. S'il joue dans un jeu Γ' de SG-valeur m>n, alors on joue dans Γ pour obtenir un successeur de SG-valeur n (qui existe nécessairement par la définition du

mex). Dans tous les cas, le joueur 1 reste dans une position perdante. Donc $\Gamma + *n \sim 0$. Donc d'après le lemme, $G \sim *n$. □

On peut donc voir n'importe quel état d'un jeu impartial combinatoire comme un couple caractérisé par un joueur et un Nimber. C'est en cela que ce théorème est fondamental. On a réduit par équivalence une vaste classe de jeux complexe à un jeu complètement trivial caractérisé par un entier n. Ainsi par équivalence des jeux, on peut considérer que les états d'un jeu impartial sont indexés par un entier n, on peut alors considérer la suite de Sprague-Grundy des jeux impartiaux vu comme leur Nimber équivalent. (On appelle Nimber, par abus de langage, à la fois le jeu trivial équivalent et la taille de ce dernier.)

4.3 Application numérique

Dans les deux premières parties, on a pu étudier les suites de Sprague-Grundy des jeux considérés. Quant est-il du Dawson's Chess ? On va faire varier le nimber du Dawson's chess et essayer de deviner une périodicité en s'appuyant sur le théorème 8 pour le calcul du Sprague-Grundy du Dawson's chess, qui peut-être vu comme une somme de jeux. Dans ce but, nous allons calculer la suite SG(n) jusqu'à n=100 pour le Dawson's chess et tenter de déceler un motif.

En observant les 100 premiers termes, on a du mal à observer une période. Peut-être que la période ou la prépériode est trop grande ou qu'elle n'existe pas ? Wikipedia stipule que le Dawson's chess possède une période, qu'elle est de 34 et qu'elle est observable à partir du rang 68. Notre objectif est de l'observer sur 3 périodes !

```
Python                                                                    /Users/august_l
Vous etes dans une position perdante !


[[4], [1, 3], [1, 2], [1, 1], [1]]
[0, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2, 4, 0, 5, 2, 2, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0, 4, 5, 2, 7, 4, 0, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2, 4, 4,
5, 5, 2, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0, 4, 5, 3, 7, 4, 8, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2, 4, 4, 5, 5, 9, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0, 4, 5
3, 7, 4, 8, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2, 4, 4, 5, 5, 9, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0, 4, 5, 3, 7, 4, 8, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2
4, 4, 5, 5, 9, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0, 4, 5, 3, 7, 4, 8, 1, 1, 2, 0, 3, 1, 1, 0, 3, 3, 2, 2, 4, 4, 5, 5, 9, 3, 3, 0, 1, 1, 3, 0, 2, 1, 1, 0
4, 5, 3, 7, 4, 8, 1, 1, 2, 0, 3, 1]
True

In [8]:
```

```
174
175 HG=sequenceD(210)
176 print(HG)#Ok par comparaison sur Wiki
177
178 #D'apres Wikipedia, il y a une periodicite 34  partir du rang 68: prouvons le pour 3 periodes !
179 i=0
180 while HG[68+i]==HG[102+i]==HG[136+i]==HG[170+i] and i<34:
181     i+=1
182 if i==34:
183     print(True)
184 else:
185     print(False)
186 #Renvoi True !
187 #Par consequent, si periodicite il y a, on peut se contenter de HG=sequenceD(101)!
188
189
```

Les résultats sont concluants : on confirme la périodicité jusqu'à 171 ! La formalisation des jeux octaux évoquée dans la conclusion apporte une base théorique à nos résultats en

démontrant que, pour certains jeu dont fait partie le Dawson's chess, s'il existe une répétition alors elle se répète à l'infini et est donc une période.

Théorème 11 (Périodicité Octal (Admis))**.** *Soit* $\Gamma = d_0.d_1...d_k$ *un jeu octal de taille k. S'il existe* $n_0 \geq 1$ *et* $p \geq 1$ *tel que :* $SG(n+p) = SG(n), \forall n_0 \leq n \leq 2n_0 + p + k$,
Alors, $SG(n+p) = SG(n), \forall n_0 \leq n$

5 Conclusion

Les jeux impartiaux représentent une des motivations mathématiques les plus concrètes possibles puisqu'elles s'illustrent dans le domaine le plus ludique qu'on puisse trouver : les jeux. Ce domaine d'étude est passionnant dans la mesure où il fait des va-et-vient entre concret et abstrait en établissant un cadre mathématique pour l'étude de ces jeux. Fait rare en mathématique, la portée de cette formalisation peut-être illustré concrètement à n'importe qui. Si le jeu de soustraction et les premiers concepts sont assez simple, ils témoignent de la méthodologie et de la logique en place dans ce domaine. La complexification apporte ensuite de nouvelles contraintes et pousse le mathématicien à prendre du recul et introduire des concepts plus abstraits. Le théorème de Bouton introduit une nouvelle opération d'une grande richesse algébrique pour résoudre un jeu loin d'être trivial. Le théorème de Sprague-Grundy réunit une classe complexe de jeux pourtant différent à un jeu trivial grâce à une notion d'équivalence définie mathématiquement.

Malgré l'absence d'application réelle, les jeux impartiaux assez étudiés pour être formaliser scrupuleusement. Tous les jeux que nous avons considérés sont des jeux octaux, c'est à dire qu'ils peuvent être codés avec 3 bits ($2^3 = 8$). On note un jeu octal $\Gamma = d_0.d_1...d_k$ où $d_i \in \{0, ..., 8\}$ désigne la régle qu'on doit suivre pour retirer i allumettes au jeu considéré. Les règles sont représentées dans le tableau ci-dessous. On peut voir que les trois jeux que nous avons considérés sont des jeux octaux. Par exemple, le Dawson's chess est le jeu octal $\Gamma = 0.37$

Durant ces dernières décennies, certains mathématiciens ont cherché les jeux octaux avec les suites de Sprague-Grundy de plus grandes périodes ou prépériodes. Le grand gagnant est le jeu 0.106 proposé par A. Flammenkamp en 2000. La suite de Sprague-Grundy de ce jeu admet une prépériode et une période de l'ordre de la centaine de milliards. Ces dernières avancées témoignent de la passion de mathématiciens pour le défi mathématique et informatique purement récréatif.

6 Annexe

```
# -*- coding: utf-8 -*-
```

d_i		It is legal to remove i tokens ...
0	000	Never
1	001	Provided that no tokens remain after they are removed
2	010	Provided that at least one token remains (and any remaining tokens are left in a single heap)
3	011	Always (but the remaining tokens must be left in a single heap)
4	100	Provided that the remaining tokens are split into exactly two nonempty heaps
5	101	Provided that *if* any tokens remain, *then* they are split into exactly two nonempty heaps
6	110	Provided that at least one token remains; the remaining tokens may optionally be split into two heaps
7	111	Always; the remaining tokens may optionally be split into two heaps

```
import time

def mex(liste):
        b=0
        liste=sorted(liste)
        for i in liste:
            if b==i:
                b+=1
        return b

l=[1,7,0,3,5,6,7,8,10,10,9,12,2,4,9,27]
print(mex(l))

#On definit un jeu comme un n-uplet de soustraction, R=(a1, ..., ak)
R=[1,3,7,8]

def SG(n,R):
    ln=[]
    if n==0:
        return 0
    else :
        for i in R:
            if n-i>=0:
                a=SG(n-i)
                ln.append(a)
```

```
            return mex(ln)

#on va calculer la suite de Sprague Grundy
def sequence(N,R):
    L=N*[-1]
    L[0]=0
    for j in range(N):
        lj=[]
        for i in R:
            if j-i>=0:
                if L[j-i]==-1:
                    L[j-i]=SG(j-i)
                lj.append(L[j-i])
        L[j]=mex(lj)
    return L

sequence(25,[1,3,7,8])


# In[5]:

#La somme binaire modulo 2 est deja programmee: x^y
#On veut creer une fonction binaire qui renvoie les nombres sous forme de liste

def binaire(n):
    q = -1
    res = []
    while q != 0:
        q = n // 2
        r = n % 2
        res.insert(0,r)
        n = q
    return res

binaire(10)


# In[6]:

def form(A): #mettre les elements de A en binaire et uniformiser les tailles
    B=list(A) #Creer un nouvelle liste identique a A mais sans les memes referen
```

```
    N=len(B)
    m=0
    j=0
    for i in range(N):
        if m<B[i]:
            j=i
            m=B[i]
        B[i]=binaire(B[i])
    n=len(B[j])
    for i in range (N):
        while len(B[i]) !=n:
            B[i].insert(0,0)
    return B

A=[3,5,6,4,7]
B=form(A)
print(B)
print("\n")
print("\n")

# In[10]:

#On va programmer la reponse optimale d'un jeu de Nim a N tas de taille [a1,...,
#Donner que des nombres > 0
def optimove(A):
    R=0
    B=form(A) # A n'est pas modifie
    N=len(B)
    M=len(B[0])
    even=0
    j=0
    I=0
    while even==0 and j<M : #on constate ou l'on obtient un nombre impair de 1 p
        for i in range(N): #plus qu'une variable muette, si i est defini cela m
            if B[i][j]==1:
                even=even^1
        j+=1

    if j>M or even==0: # si ca n'arrive jamais alors c'est que l'on est dans une
        return ("Vous etes dans une position perdante !")
    else: # sinon on prend un nombre qui a un 1 dans cette colonne
```

```
        i=0
        while B[i][j-1]!=1:
            i+=1
        S=A[i]
        A.pop(i) #on le retire de A
        for k in A:
            R=R^k #on calcule la somme binaire modulo 2 des elements restants
        Z=[S,R]
        Z=form(Z) #On compare le maillon faible et la somme binaire obtenue
        k=0
        while len(Z[0])!=0:
            I+=2**k*(Z[0][-1]-Z[1][-1]) #on retire 2^k allumettes si il y a un 1
            k+=1
            Z[0].pop(-1)
            Z[1].pop(-1)
        return "Vous etes dans une position gagnante: vous devez retirer",I,"al

D=[5,40,1,3,14,12,4,6,9,8,7,45,45,12,45,16,45,78,98]

tmps1=time.clock()
print(optimove(D)) # solution verifiee par implementation successive!
tmps2=time.clock()
print "Temps d'execution =",(tmps2-tmps1)
print(optimove([2,3,1]))
#Surement du travail au niveau de la complexite!


print("\n")

# In[12]:

def SNim(A):
    S=[]
    for j in range (len(A)):
        for i in range(A[j]):
            B=list(A)
            B[j]=B[j]-(i+1)
            if B[j]==0:
                B.pop(j)
            S.append(B)
    return S
```

```
print(SNim([1,4]))



# On peut voir le Dawson chess (Morpion unidimensionnel) comme un jeu de Nim ou
# *spliter un tas en 2 en enlevant 3 d'allumettes a l'un des tas.

# Ainsi le Dawson chess se caracterise seulement par un entier n. On peut en den
# recurrence et remonter a SG(n) via mex{succ(n)}

def sequenceD(N):
    L=(N+1)*[0]
    L[1]=1
    L[2]=1
    for j in range(3,N+1):
        lj=[L[j-2],L[j-3]]#Succeseurs sans split
        for i in range(1,int((j+1)/2)-1):#Avec split
            a=0
            a=L[i]^L[(j-3-i)]
            lj.append(a)
        L[j]=mex(lj)
    return L


HG=sequenceD(210)
print(HG)#Ok par comparaison sur Wiki

#D'apres Wikipedia, il y a une periodicite 34  partir du rang 68: prouvons le po
i=0
while HG[68+i]==HG[102+i]==HG[136+i]==HG[170+i] and i<34:
    i+=1
if i==34:
    print(True)
else:
    print(False)
#Renvoi True !
#Par consequent, si periodicite il y a, on peut se contenter de HG=sequenceD(101
```

7 Bibliographie

Références

[1] Elwyn Berlekamp and John Conway. *Winning Ways for your Mathematical Plays.* Academic Press, 1982.

[2] Aaron Siegel. *Combinatorial Game theory*

[3] $ms.mcmaster.ca/mbays/teaching/3U03/notes/14-games.pdf$.

[4] $https://www.math.ucla.edu/tom/Game\_Theory/comb.pdf$

[5] Wikipédia. *Nimber, Sprague-Grundy, Mex.*