Rapport de TIPE

Salim Rostam

Résumé

La bataille est un jeu de cartes très simple. Cependant, une étude théorique de ce jeu se révèle être plus complexe ; pour simplifier les choses, j'ai étudié la bataille à une seule couleur. Même avec cette simplification, beaucoup de questions restent en suspens. Certaines parties peuvent-elles ne jamais se terminer ? Peut-on prévoir l'issue de la partie dès le début ?

Table des matières

Règles

Soit N un entier naturel pair non nul. On distribue initialement $\frac{N}{2}$ cartes à chaque joueur, chaque carte ayant un unique numéro dans $\{1, \ldots, N\}$ — il n'y a donc en fait pas de bataille ! Les deux joueurs disposent leurs cartes en tas devant eux ; à chaque tour, les deux joueurs comparent la carte du haut de leur tas : le joueur qui possède la carte la plus forte remporte le pli, et place les deux cartes gagnées en bas de son tas en plaçant *d'abord* la sienne *puis* celle de son adversaire. Pour ce qui est de la force des cartes, la carte qui porte le numéro i est bien sûr plus forte que celle qui porte le numéro $j < i$. Si au bout d'un certain nombre de coups un joueur n'a plus de cartes, la partie est terminée et ce joueur a perdu !

1 Premiers résultats

Compte-tenu de la règle du jeu, une partie est entièrement déterminée par sa position initiale : il y a donc au plus $N!$ parties possibles. De plus, cette règle offre une symétrie aux parties, au sens où si initialement on permute les deux jeux, la partie que l'on obtient sera la même à cette permutation près. On peut donc se contenter de n'étudier que $\frac{N!}{2}$ parties bien choisies ; c'est en fait exactement le nombre de parties possibles. On voit à la taille de ce nombre qu'il est impossible de mener des études exhaustives dès que N est assez grand !

Le fait d'avoir mis une borne sur le nombre de parties possibles nous offre le théorème suivant.

Théorème 1. *Soit une partie se termine avec le gain de l'un des deux joueurs, soit elle est périodique à partir d'un certain rang.*

2 Simulation de parties

J'ai écrit un programme en Scilab qui permet de simuler des parties : il est décrit dans l'annexe A.1.

2.1 Statistiques

Avant toute chose, on doit se faire une idée d'une bonne limite de coups à imposer : les graphes de l'annexe B.1 donnent une idée de la répartition de la longueur des parties (les temps de calculs sont indiqués entre parenthèses). De plus, on ne compte pas les longueurs qui sont égales à la limite de coups que l'on a imposée.

On peut maintenant simuler un grand nombre de parties pour avoir une idée de la réponse à la question suivante : existe-t-il des parties infinies, et si oui dans quelles proportions ? Pour les petites valeurs de N (c'est-à-dire 2,4,6 et 8) on peut vérifier explicitement que toutes les parties se finissent : il suffit d'engendrer toutes les permutations de $\mathfrak{S}_N$ puis de simuler les parties à partir de chacune d'entre elles. Dès que N est plus grand, simuler chaque partie prendrait beaucoup trop de temps : on est donc contraint de faire des statistiques sur un grand nombre de parties. Ainsi, on obtient les résultats du tableau de l'annexe B.2. On constate alors trois choses :

- les gains sont équilibrés entre les deux joueurs ;
- il semble y avoir des parties infinies quand N n'est pas un multiple de 4 (valeurs grisées) ;

– il ne semble pas y avoir de parties infinies pour certaines valeurs de N multiples de 4 ;
– le nombre de parties infinies semble tendre vers 0 quand N augmente.

Le dernier point soulève un problème délicat : il est de plus en plus dur de trouver au hasard des parties infinies quand N est grand.

Il faut remarquer que la troisième conjecture peut être affinée en : « il ne semble pas y avoir de parties infinies quand N est un multiple de 8 ». Compte-tenu de la remarque précédente, si cette conjecture s'avérait être fausse il faudrait beaucoup de chance pour trouver un contre-exemple en simulant au hasard des parties.

2.2 Des exemples de parties infinies

Il est maintenant nécessaire d'introduire une notation pour les positions : voici un exemple de partie pour $N = 4$. La première ligne est le tas du joueur 1 et le haut du tas est à gauche.

$$\begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix} \to \begin{pmatrix} 1 & 4 & 2 \\ 3 & & \end{pmatrix} \to \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix} \to \begin{pmatrix} 2 & 4 & 3 \\ 1 & & \end{pmatrix} \to \begin{pmatrix} 4 & 3 & 2 & 1 \\ & & & \end{pmatrix}.$$

Les premières parties infinies arrivent pour $N = 10$. Voici quelques exemples :

- $N = 10 = 4 \times 2 + 2$. Si on simule la partie de position initiale $\left(\begin{smallmatrix} 2 & 10 & 8 & 6 & 3 \\ 5 & 4 & 1 & 9 & 7 \end{smallmatrix}\right)$, on remarque que l'on retrouve cette position au début du 60e coup. De plus, après le premier coup on a une périodicité $+2/-2$ (le joueur 1 gagne deux fois, le joueur 2 gagne deux fois,...).
- $N = 14 = 4 \times 3 + 2$. Si on simule la partie de position initiale $\left(\begin{smallmatrix} 4 & 9 & 3 & 12 & 8 & 11 & 2 \\ 7 & 14 & 10 & 6 & 1 & 13 & 5 \end{smallmatrix}\right)$, on remarque qu'à partir du 4e coup cette partie est périodique de période 112 et de type $+2/-2$.
- $N = 20 = 4 \times 5$ (première valeur de N multiple de 4 pour laquelle il existe des parties infinies). Si on simule la partie de position initiale $\left(\begin{smallmatrix} 17 & 4 & 16 & 8 & 6 & 2 & 19 & 14 & 16 & 11 \\ 5 & 1 & 20 & 12 & 15 & 9 & 18 & 10 & 7 & 3 \end{smallmatrix}\right)$, alors après le deuxième coup cette partie est périodique, de période 670 et de type $+4/-4$.

3 Modèle de partie infinies

Les quelques parties simulées nous amènent au théorème suivant, qui va confirmer l'une des conjectures.

Théorème 2. *Soit N un entier de la forme $4k + 2$ où k est un entier supérieur ou égal à 2. Alors il existe des parties infinies pour la bataille à N cartes et une couleur.*

Idée de la démonstration. La démonstration se base sur ce que l'on observe expérimentalement : on va rechercher une périodicité de type $+2/-2$ à partir d'une position de la forme $\left(\begin{smallmatrix} N & & \cdots \\ & N-1 & \cdots \end{smallmatrix}\right)$. Soit $N = 4k + 2$ un entier avec $k \geqslant 2$. On définit trois sous-ensembles de $\{1, \ldots, N\}$:
– $\{g_1, \ldots, g_k\} = \{N - k + 1, \ldots, N\}$;
– $\{p_1, \ldots, p_{k+1}\} = \{1, \ldots, k + 1\}$;
– $\{x_1, \ldots, x_{2k+1}\} = \{k + 2, \ldots, N - k\}$.

On considère maintenant la position suivante :

$$P = \left(\begin{array}{cccc|cccc|c|cccc|cc} g_1 & x_1 & x_2 & p_1 & & & & & & & & & & & \\ x_3 & p_2 & g_2 & x_4 & x_5 & p_3 & g_3 & x_6 & \dots & x_{2k-1} & p_k & g_k & x_{2k} & x_{2k+1} & p_{k+1} \end{array} \right).$$

Le principe est que les g_i et les p_j vont affronter uniquement les x_k : ainsi, les g_i vont gagner tous leurs duels alors que les p_j vont tous les perdre. Il faut remarquer que cette position P est issue de la position initiale suivante :

$$P' = \begin{pmatrix} p_{k+1} & x_{2k} & p_k & \dots & x_6 & p_3 & g_1 & x_1 & x_2 & p_1 \\ x_{2k+1} & g_k & x_{2k-1} & \dots & g_3 & x_5 & x_3 & p_2 & g_2 & x_4 \end{pmatrix},$$

après que $2k-3$ coups ont été joués. En fait, à partir de P' le joueur 2 gagne tous les duels avant d'arriver à P.

Il suffit ensuite de remarquer que l'on retrouve cette position $4 \times \mathrm{ppcm}(k-1, k+1, 2k+1)$ coups après, d'où la périodicité. $\square$

4 Profils de parties

Pour tenter de mieux comprendre les longueurs de parties, l'existence de parties infinies,... on peut s'intéresser aux *profils* des parties. Plus précisément, en notant a le gain du joueur 1 et b le gain du joueur 2, le profil est le mot en a et b qui décrit la partie. Par exemple, pour la position initiale $\left(\begin{smallmatrix} 4 & 1 \\ 2 & 3 \end{smallmatrix}\right)$ le profil de la partie est aba^2.

Encore une fois, pour les petites valeurs de N on peut mener des études exhaustives. Par exemple, l'ensemble des mots qui décrivent les différentes parties possibles pour $N=2$ est $\{a, b\}$ et pour $N=4$ est $\left\{a^2, b^2, aba^2, bab^2, (ba^2)^2, (ab^2)^2\right\}$.

Le problème auquel je me suis intéressé est le suivant : à partir d'un mot en a et b, peut-on déterminer si ce mot décrit une partie pour une bataille à N cartes et une couleur ? Si tel est le cas, on dira que ce mot est *réalisable*. Pour répondre à cette question, on va s'aider d'un graphe.

4.1 Graphe associé à un mot

Soit $N = 2n$ un entier naturel pair et m un mot en a et b. L'idée est la suivante : on écrit un début de partie $\left(\begin{smallmatrix} x_1 & \dots & x_n \\ x_{n+1} & \dots & x_N \end{smallmatrix}\right)$ où $\{x_1, \dots, x_N\} = \{1, \dots, N\}$ puis on le fait évoluer selon ce qu'exige le mot m. Bien sûr, une condition nécessaire pour que le mot m soit réalisable est que l'un des tas soit vide à la fin, et donc que $|m|_a - |m|_b$ soit égal à $\pm n$. De plus, à la fin du processus on obtient un système d'inégalités (S) sur les x_i : m sera réalisable si et seulement si ce système admet (au moins) une solution.

Définition 3. On garde les notations précédentes. Le *graphe associé au mot m* est le graphe orienté défini comme suit :

– l'ensemble de ses sommets est $\{x_1, \dots, x_N\}$ où $N = 2\big||m|_a - |m|_b\big|$;
– (x_i, x_j) est un arc orienté si et seulement si $x_i < x_j$ est une inéquation de (S).

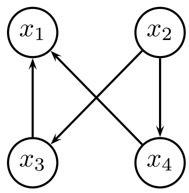
On va illustrer tout cela avec un exemple : le mot aba^2 est-il réalisable ? Bien sûr, on sait déjà que la partie qui commence par $\left(\begin{smallmatrix} 4 & 1 \\ 2 & 3 \end{smallmatrix}\right)$ admet ce mot comme profil, mais on va essayer de retrouver ce résultat.

Tout d'abord, si aba^2 est réalisable alors le nombre de cartes qu'il faut prendre est $N = 2\,|3-1| = 4$. On écrit donc $\left(\begin{smallmatrix} x_1 & x_2 \\ x_3 & x_4 \end{smallmatrix}\right)$ puis on fait évoluer ; la lettre que l'on doit respecter est indiquée au-dessus de la flèche et la condition que l'on obtient est indiquée en-dessous.

$$\begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \xrightarrow[x_3<x_1]{a} \begin{pmatrix} x_2 & x_1 & x_3 \\ x_4 & & \end{pmatrix} \xrightarrow[x_2<x_4]{b} \begin{pmatrix} x_1 & x_3 \\ x_4 & x_2 \end{pmatrix}$$

$$\xrightarrow[x_4<x_1]{a} \begin{pmatrix} x_3 & x_1 & x_4 \\ x_2 & & \end{pmatrix} \xrightarrow[x_2<x_3]{a} \begin{pmatrix} x_1 & x_4 & x_3 & x_2 \\ & & & \end{pmatrix}.$$

Le mot aba^2 est donc *a priori* réalisable (le deuxième joueur n'a plus de carte à la fin). Son graphe associé est :



On a alors nécessairement $x_1 = 4, x_2 = 1$ et $\{x_3, x_4\} = \{2, 3\}$. Le mot aba^2 est donc réalisable ; il y a deux positions initiales qui réalisent ce mot, qui sont $\left(\begin{smallmatrix} 4 & 1 \\ 2 & 3 \end{smallmatrix}\right)$ et $\left(\begin{smallmatrix} 4 & 1 \\ 3 & 2 \end{smallmatrix}\right)$.

On remarque le résultat suivant : si m est un mot en a et b tel que son graphe associé est cyclique, alors m n'est pas réalisable. Ce serait plutôt la réciproque qui nous intéresserait ! Pour la démontrer, on va introduire un intermédiaire.

Définition 4. Soit G un graphe orienté fini ; $S = \{x_1, \dots, x_N\}$ est l'ensemble des sommets. On dira que G est *résoluble* s'il existe $\sigma \in \mathfrak{S}_N$ telle que :

1. $x_{\sigma(N)}$ n'a pas de successeur ;
2. $\forall i \in \{1, \dots, N-1\}$, $x_{\sigma(i)}$ n'a pas de successeur si on ne tient pas compte des éventuels arcs $(x_{\sigma(i)}, x_{\sigma(j)})$ pour $j > i$.

Remarque. On peut également définir un graphe résoluble par récurrence, en posant que le graphe vide est résoluble et en remplaçant la deuxième condition par « G' est résoluble » où G' est le graphe construit à partir de G en enlevant le sommet $\sigma(N)$ et tous les arcs qui y mènent

On peut alors énoncer le théorème suivant.

Théorème 5. *Soit m un mot en a et b, soit G son graphe associé. Les trois propositions suivantes sont équivalentes :*

(i) *m est réalisable ;*

(ii) *G est acyclique ;*

(iii) *G est résoluble.*

La seule chose qu'il y a vraiment à justifier est $(ii) \Rightarrow (iii)$: on peut démontrer ce résultat en toute généralité, par récurrence sur le nombre de sommets du graphe.

J'ai écrit un programme en OCaml qui utilise l'équivalence $(i) \Leftrightarrow (iii)$ pour déterminer si un mot est réalisable : plus de détails sont donnés dans l'annexe A.2.

Conclusion

Pour finir, je vais rapidement rappeler les résultats établis ainsi que les conjectures :

- il existe des parties infinies quand N est de la forme $2(2k+1)$ avec $k \geqslant 2$;
- il semble exister des parties infinies quand N est de la forme $4(2k+1)$ avec $k \geqslant 2$;
- il semble ne pas exister de parties infinies quand N est de forme $8k$ avec $k \geqslant 1$.

On pourrait essayer de démontrer la première conjecture de la même façon que le premier résultat ; quant à la deuxième conjecture, je n'en ai aucune idée. On peut remarquer que si cette deuxième conjecture est vraie, alors on peut prévoir dès le début de la partie quel joueur sera le gagnant : c'est celui qui possède la carte N.

À propos de la bibliographie

Le seul ouvrage que j'ai trouvé en rapport avec le sujet est *Jeux mathématiques et mathématiques des jeux* (Belin, 1997). Les problèmes traités dans le chapitre 3 sont légèrement différents de ceux que j'ai abordés ici puisqu'il s'agit des batailles « ordinaires », c'est-à-dire avec des jeux de 32 et de 52 cartes standards. Cependant, les résultats sont semblables en ce qui concerne la répartition des parties infinies ; comme dans ce rapport, des preuves sont apportées pour démontrer l'existence de parties infinies dans certains cas, mais pas pour démontrer qu'il n'y en a pas. J'ai pris contact avec l'un des auteurs de l'article en question, Jean-Paul Delahaye : il m'a dit qu'à sa connaissance aucune avancée n'avait été faite depuis.

A Programmes

A.1 Simulation de parties

Le programme n'a rien de très compliqué :

- le mélange est généré en prenant une permutation de l'ensemble $\{1, \ldots, N\}$ (fonction Scilab) ;
- on fait évoluer les deux tas comme nous l'impose la règle et on ajuste leur longueurs, selon le schéma suivant :

```
c1=jeu1(1);c2=jeu2(1)
if c1>c2 then
  jeu1=[jeu1(2:tailles(1)),c1,c2]
  jeu2=jeu2(2:tailles(2))
  tailles=tailles+[1,-1]
```

Il est nécessaire de s'arrêter après un certain nombre de coups (donné en argument) car on ne sait pas *a priori* si la partie se termine ; le programme s'arrête avant cette limite si l'une des composantes du tableau des tailles s'annule.

A.2 Déterminer si un mot est réalisable

La construction du graphe associé est un peu plus simple si le mot est sous forme développée. Comme il est plus simple de fournir les mots sous forme réduite, on aimerait avoir un type qui nous permette cette transformation. On définit alors les trois types suivant :

```
type lettre = A | B;;
type terme = {caractere: lettre; mutable nombre: int};;
type mot = terme Queue.t;;
```

Ces types nous permettent de réduire ou de développer un mot en le modifiant directement, sans créer de nouveau mot.

Reste encore la question de l'encodage du graphe : j'ai choisi de le représenter par sa matrice d'adjacence. Si la condition $x_i < x_j$ arrive pendant une partie, on affecte la valeur 1 à la case (i, j) ; en outre, cela évite de stocker les répétitions.

Une fois le graphe construit, il faut déterminer s'il est résoluble. La structure de matrice d'adjacence se prêtant mal à la récursivité, on utilise la définition « itérative » d'un graphe résoluble. Les nombres de successeurs de chaque sommet sont stockés dans un tableau : une fois que l'on a trouvé un élément qui n'a pas de successeur (i.e. un 0 dans ce tableau), on efface tous les arcs qui lui sont incidents :

```
for i = 0 to n - 1 do
 if g.(i).(!sans_successeur) = 1 then
  (g.(i).(!sans_successeur) <- 0;
   successeurs.(i) <- successeurs.(i) - 1)
done;
```

Pour éviter de choisir plusieurs fois le même élément sans successeur, une fois qu'il a été choisi on affecte la valeur -1 à sa case dans le tableau du nombre de successeurs. Bien sûr, le programme s'arrête si l'on ne trouve pas d'élément qui n'a pas de successeurs.

Reste finalement à retourner un début de partie qui réalise m : lorsque l'on choisit le $i^{\text{ème}}$ élément, $x_{\sigma(N+1-i)}$, qui n'a pas (ou plus) de successeur, on lui associe le numéro $N+1-i$. Le programme retourne alors la position initiale $\left(\begin{smallmatrix} x_1 & \dots & x_n \\ x_{n+1} & \dots & x_N \end{smallmatrix}\right)$.

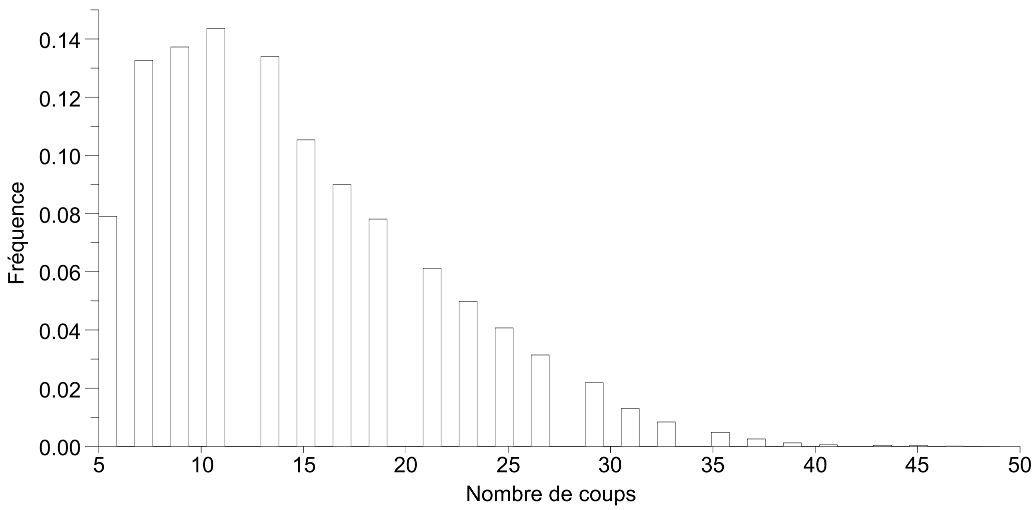
Voici trois exemples d'appels à la fonction finale :

```
# trouve_debut [|(A,2);(B,1)|];;
Exception: Failure "Le mot n'est pas réalisable".
# trouve_debut [|(A,1);(B,2);(A,3)|];;
Exception: Failure "Le mot ne code pas le profil d'une partie entière".
# trouve_debut [|(A,1);(B,1);(A,2)|];;
- : int array array = [|[|4;1|];[|3;2|]|]
```

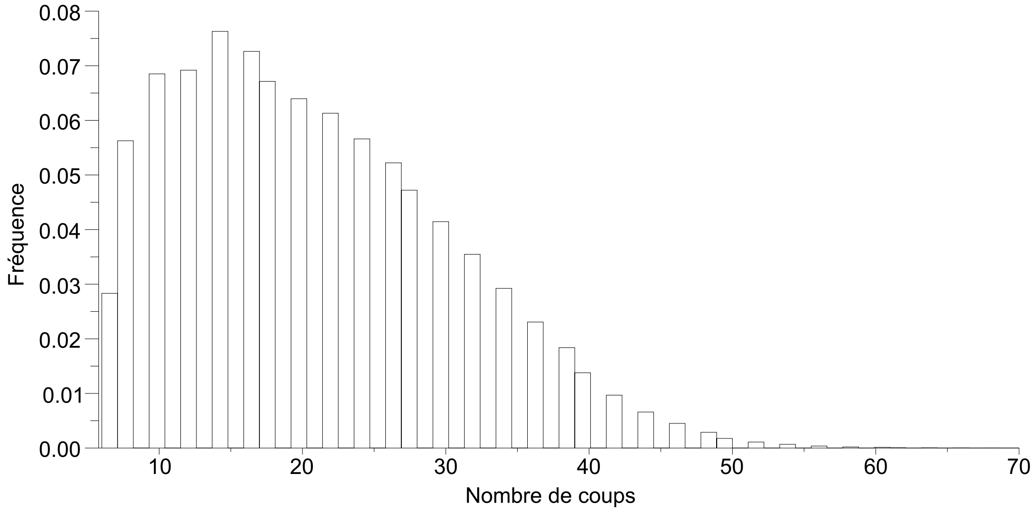
B Statistiques

Dans toute cette annexe, les échantillons considérés représentent 10^6 parties.

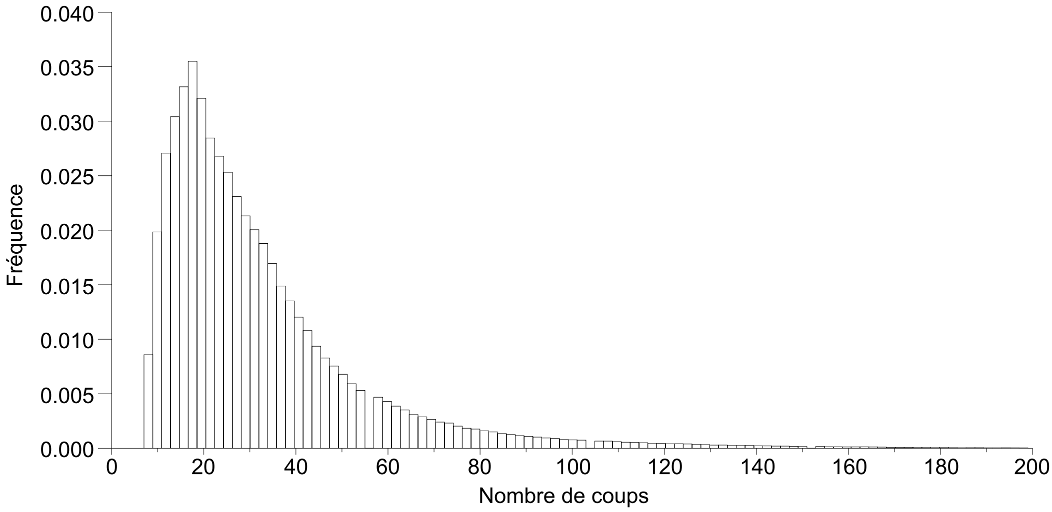
B.1 Répartition de la longueur des parties



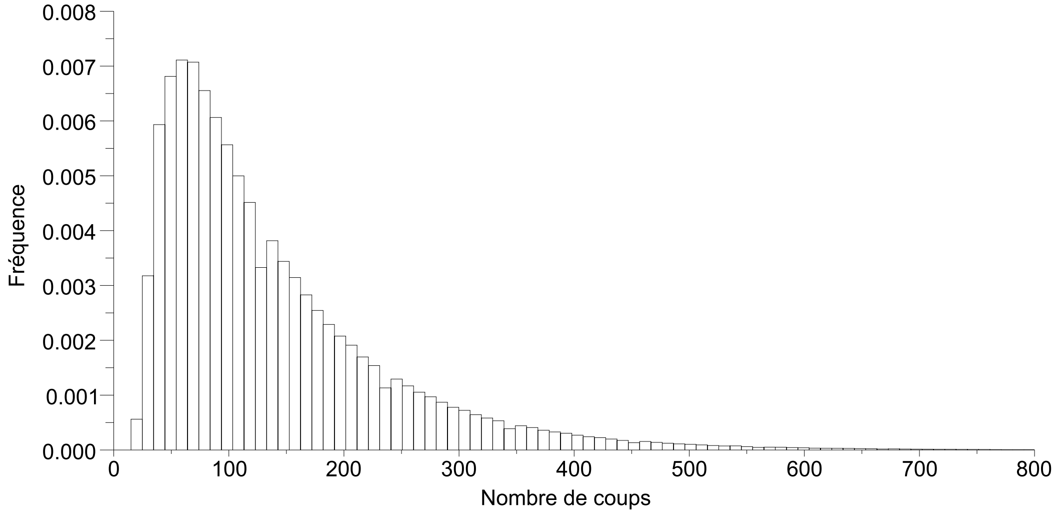
$N = 10$ (7 minutes).



$N = 12$ (4 minutes).



$N = 14$ (8 minutes).



$N = 30$ (26 minutes).

B.2 Répartition des gains

N	Limite	Joueur 1	Joueur 2	Inconnu	Temps de calcul
10	200	0.445	0.445	0.110	7 minutes
12	200	0.500	0.500	0	4 minutes
14	300	0.478	0.479	0.043	8 minutes
16	500	0.500	0.500	0	7 minutes
18	800	0.483	0.485	0.032	13 minutes
20	800	0.4988	0.4997	0.0015	10 minutes
22	800	0.496	0.496	0.008	15 minutes
24	800	0.500	0.500	0	13 minutes
26	800	0.498	0.498	0.004	20 minutes
28	800	0.500	0.500	$< 10^{-3}$	20 minutes
30	1500	0.4984	0.4997	0.0019	26 minutes
32	2000	0.501	0.499	0	25 minutes
34	2000	0.500	0.500	$< 10^{-3}$	31 minutes

Fréquences de gains.