Mélanges de cartes parfaits : du tour de magie à la théorie des groupes.
Rapport de TIPE

Perrine Jouteur

Résumé

Le tour de magie du mélange parfait repose sur deux permutations agissant sur les positions des cartes du paquet appelées In et Out. Grâce à ces deux permutations, un habile prestidigitateur peut déplacer une carte d'une position quelconque vers n'importe quelle autre place dans le paquet. Un algorithme constructif donne ainsi une séquence de In et de Out à effectuer pour transporter une carte de la position i à la position j. Enfin, le groupe engendré par In et Out peut être étudié en lui-même comme sous groupe du groupe des permutations, notamment par l'établissement d'isomorphismes vers d'autres groupes mieux connus.

Table des matières

1 Préliminaires

1.1 Position du problème

Le tour de magie qui fonde ce TIPE consiste à déplacer sur demande du public une carte au sein du paquet en se contentant de le mélanger plusieurs fois. Le truc réside dans la façon de battre les cartes : le magicien exécute en effet une séquence de mélanges entièrement déterministes, dits "mélanges parfaits".

Ces mélanges sont de deux sortes, et fonctionnent de la même manière : on coupe en deux moitiés le paquet de cartes (on travaille avec un nombre pair de carte, soit 2n), et on entrelace les cartes de chaque moitié, une sur deux exactement. Si l'on commence par la moitié du bas, il s'agit d'un Out-shuffle, tandis que commencer par la moitié du haut revient à effectuer un In-shuffle.

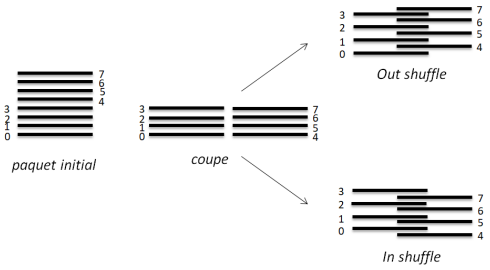

Figure 1

L'objectif de ce TIPE est de comprendre les mécanismes de ces deux mélanges et du tour de magie qui en résulte, et d'étudier par transport de structure le groupe qu'ils engendrent.

1.2 Notations et définitions

Par convention, les cartes seront numérotées de 0 à $2n-1$. De plus, on identifie les mélanges parfaits à des permutations des positions des cartes dans le paquet, donc à des permutations de $[\![0, 2n-1]\!]$. Le groupe engendré par In et Out sera noté $< I, O >$.
On peut alors donner une définition explicite de In et Out :

$$I(i) = \begin{cases} 2i+1 & si\ 0 \leqslant i \leqslant n-1 \\ 2i-2n & si\ n \leqslant i \leqslant 2n-1 \end{cases} \qquad O(i) = \begin{cases} 2i & si\ 0 \leqslant i \leqslant n-1 \\ 2i+1-2n & si\ n \leqslant i \leqslant 2n-1 \end{cases}$$

1.3 Premières propriétés

♣ I et O sont des permutations à symétrie centrale, c'est-à-dire vérifiant la propriété suivante :

$$\forall i \in [\![0, 2n-1]\!], \quad g(i) + g(2n-1-i) = 2n-1$$

où $g \in \mathfrak{S}_{2n}$.

♣ L'ordre de O est l'ordre de 2 modulo $2n-1$ et celui de I est l'ordre de 2 modulo $2n+1$.

2 Un algorithme constructif

2.1 Principe

L'exécution du tour de magie nécessite de connaître une succession de In et de Out qui transportera une carte d'une position i vers une position j dans le paquet. On divise le problème en deux étapes : d'abord, amener la carte en position 0, puis, de la position 0, amener la carte en position j.

♣ Première étape : déplacer la ième carte en position 0
L'astuce consiste à renverser le problème en cherchant à déplacer la carte 0 vers la position i, mais à l'aide des inverses des mélanges parfaits, dont la définition est :

$$I^{-1}(x) = \begin{cases} \left\lfloor \frac{x}{2} \right\rfloor + n & si\ x\ est\ pair \\ \left\lfloor \frac{x}{2} \right\rfloor & sinon \end{cases} \qquad O^{-1}(x) = \begin{cases} \left\lfloor \frac{x}{2} \right\rfloor & si\ x\ est\ pair \\ \left\lfloor \frac{x}{2} \right\rfloor + n & sinon \end{cases}$$

Partant donc de la position 0, on construit un arbre résumant toutes les possibilités de déplacement grâce à O^{-1} et I^{-1}.

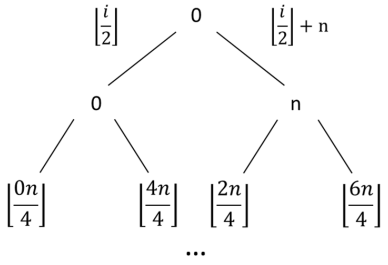

Figure 2

Si le nœud père est pair, son fils gauche sera son image par O^{-1} et son fils droit son image par I^{-1}, et inversement s'il est impair. On définit l'entier r comme suit : $2^{r-1} < 2n \leqslant 2^r$.
Chaque nœud de l'arbre est de la forme $\left\lfloor \frac{2nt}{2^{k-1}} \right\rfloor$, avec k la profondeur du nœud et t un entier entre 0 et 2^{k-1}.
Le niveau r contient alors tous les entiers entre 0 et $2n-1$, ce qui signifie que toutes les positions peuvent être atteintes à partir de la position 0 grâce à une combinaison de I^{-1} et O^{-1}, d'où la validité de l'algorithme.
Pour un nœud de la forme $\left\lfloor \frac{2nt}{2^{k-1}} \right\rfloor$, on retrouve le chemin parcouru pour l'atteindre en lisant la décomposition binaire de t, du bit de poids le plus faible vers le bit de poids le plus fort.
Afin de tenir compte de la parité des nœuds rencontrés, on introduit un terme correctif s défini par : $s = 2nt - 2^r i$.
En additionnant deux à deux les bits des décompositions de t et ceux de s, on obtient une séquence de mélanges parfaits à réaliser, un 1 correspondant à un In et un 0 à un Out. [4]

♣ Deuxième étape : déplacer la carte du haut du paquet (position 0) vers la position j
Il suffit de décomposer j en binaire, et d'effectuer un Out pour un 0 et un In pour un 1, en lisant la décomposition du bit de poids le plus fort vers le bit de poids le plus faible. [2]

2.2 Implémentation en python

En pratique, les mélanges parfaits requièrent beaucoup de dextérité. La modélisation informatique permet de manipuler des paquets de grandes tailles (quelques dizaines de cartes) sans risque d'erreur due à un défaut d'agilité. On représente un paquet de $2n$ cartes par une liste python d'entiers où la $i$ème valeur correspond au numéro de la carte située en $i$ème position dans le paquet réel.
In et Out sont codés par des fonctions qui modifient un paquet donné en argument.
Des fonctions auxiliaires (*encadrement* et *binaire*) permettent respectivement de calculer l'entier r défini précédemment et d'obtenir la décomposition binaire d'un entier.
La fonction *top* donne une succession de mélanges à effectuer pour amener la $i$ème carte en position 0.

```
def top(i,n):
    r = encadrement(2*n) + 1
    if i == 0:
        t = 0
    elif i == (2*n) - 1:
        t = (2**r) - 1
    else:
        t = ((i+1)*(2**r))//(2*n)
    if t == (i+1)*(2**r)*1./(2*n):
        t = t-1
    T = binaire(t,r)
    S = binaire(2*n*t - i*2**r,r)
    return [(T[k] + S[k])%2 for k in range(r)]
```

Exemple : on veut transporter la 10ème carte en position 0 dans un paquet de taille 56 :

```
>>> execfile('perfect shuffle.py')
>>> top(10,28)
[1, 0, 1, 1, 0, 0]
>>> p = range(56)
>>> In(p)
>>> Out(p)
>>> In(p)
>>> In(p)
>>> p
[10, 35, 3, 28, 24, 49, 17, 42, 39, 7, 32, 0, 53, 21, 46, 1
4, 11, 36, 4, 29, 25, 50, 18, 43, 40, 8, 33, 1, 54, 22, 47,
15, 12, 37, 5, 30, 26, 51, 19, 44, 41, 9, 34, 2, 55, 23, 48
, 16, 13, 38, 6, 31, 27, 52, 20, 45]
>>>
```

Enfin, la fonction *tour* synthétise le tour de magie en modifiant un paquet donné en argument pour déplacer la $i$eme carte en position j (i et j donnés en arguments). Par exemple, sur un paquet de taille 32, en souhaitant déplacer la 9ème carte en position 26 :

```
>>> execfile('perfect shuffle.py')
>>> p = range(32)
>>> tour(p,9,26)
[4, 20, 5, 21, 6, 22, 7, 23, 0, 16, 1, 17, 2, 18, 3, 19, 12
, 28, 13, 29, 14, 30, 15, 31, 8, 24, 9, 25, 10, 26, 11, 27]
>>> p[26]
9
```

3 Étude théorique

L'ensemble des séquences de In et de Out forme un groupe dont la structure dépend de la taille du paquet sur lequel ces mélanges s'appliquent. En étudiant ce groupe par transport de structure (utilisation d'isomorphismes vers d'autres groupes plus maîtrisés), on espère mettre en évidence des propriétés des mélanges parfaits qui pourraient compléter ou améliorer le tour de magie.

3.1 Cas des puissances de 2

On se place dans le cas $2n = 2^k$, où $k \in \mathbb{N}$.

Théorème: $< I, O >$ est isomorphe à $(\mathbb{Z}/_{2\mathbb{Z}})^k \ltimes \mathbb{Z}/_{k\mathbb{Z}}$.

Démonstration :
On pose :

$$\begin{array}{rccc} f: & \left(\mathbb{Z}/_{2\mathbb{Z}}\right)^k, + & \longrightarrow & (< O, I >, .) \\ & (a_1, a_2, ..., a_k) & \longmapsto & \prod_{j=1}^{k} B_j^{a_j} \end{array}$$

où $B_j = O^{j-1} I O^{-j}$
On montre que f est un morphisme injectif. (annexe 2)

Construction de l'opération sur le produit semi-direct :

$$\begin{array}{rccc} \phi: & \mathbb{Z}/_{k\mathbb{Z}} & \longrightarrow & Aut\left((\mathbb{Z}/_{2\mathbb{Z}})^k\right) \\ & \overline{p} & \longmapsto & \left(\begin{array}{rcc} \phi_p: & (\mathbb{Z}/_{2\mathbb{Z}})^k & \to & (\mathbb{Z}/_{2\mathbb{Z}})^k \\ & (a_1, ..., a_k) & \mapsto & (a_{p+1}, ..., a_k, a_1, ..., a_p) \end{array} \right) \end{array}$$

On vérifie aisément que ϕ est bien définie et qu'il s'agit d'un morphisme de groupes. On construit à partir de ϕ une loi de composition interne sur $(\mathbb{Z}/_{2\mathbb{Z}})^k \ltimes \mathbb{Z}/_{k\mathbb{Z}}$:

$$(a, \overline{p}) \star (a', \overline{p'}) = (a + \phi_p(a'), \overline{p + p'})$$

On pose enfin :

$$\begin{array}{rccc} \psi: & (\mathbb{Z}/_{2\mathbb{Z}})^k \ltimes \mathbb{Z}/_{k\mathbb{Z}} & \longrightarrow & < I, O > \\ & (a, \overline{p}) & \longmapsto & f(a) O^{-p} \end{array}$$

ψ est un isomorphisme. ■

3.2 Quelques lemmes

Notations et définitions :
On note sgn la signature de $\mathfrak{S}_{2n}$.
On note B_n l'ensemble des permutations de $\mathfrak{S}_{2n}$ à symétrie centrale, et D_n le groupe des permutations avec un nombre pair de changements de signe des vecteurs de la base canonique de $\mathbb{R}^n$ (groupe de Weyl de type D, d'ordre n) [5].
$\forall x \in [\![0, n-1]\!]$, on note $\overline{x}$ le doublet $\{x\,,\, 2n-1-x\}$.
On pose alors :

$$\begin{array}{rccc} {}^{-}: & B_n & \longrightarrow & \mathfrak{S}\left(\{\ \overline{x}, x \in [\![0, n-1]\!]\ \}\right) \\ & \sigma & \longmapsto & \overline{\sigma}: \ \overline{x} \ \mapsto \ \{\sigma(x)\,,\, \sigma(2n-1-x)\} \end{array}$$

Il s'agit d'un morphisme. On note K le noyau de sa restriction à $< I, O >$.
On pose :

$$\begin{array}{rccc} \overline{sgn}: & B_n & \longrightarrow & \{-1, 1\} \\ & \sigma & \longmapsto & \epsilon(\overline{\sigma}) \end{array}$$

où ϵ désigne la signature sur $\mathfrak{S}_n$, en identifiant $\mathfrak{S}_n$ et $\mathfrak{S}(\{\overline{x}\ , x \in [\![0, n-1]\!]\})$.

Propriété 1 :
Les images par sgn et $\overline{sgn}$ de I et O sont données par la figure suivante :

$n\%4$	0	1	2	3
$sgn(I)$	1	-1	-1	1
$\overline{sgn(I)}$	1	1	-1	1
$sgn(O)$	1	1	-1	-1
$\overline{sgn(O)}$	1	1	1	-1

Figure 3

Démonstration : Les calculs sont effectués en annexe 3.

Propriété 2 (admise) : Soit $< I, O >^*$ l'ensemble des éléments de $< I, O >$ qui envoient $[\![0, n-1]\!]$ sur lui-même. $< \overline{I}, O >^*$ peut donc être vu comme un sous-ensemble de $\mathfrak{S}_n$.
Si $A_n \subset < I, O >^*$, alors K contient tous les produits d'un nombre pair de transpositions de la forme $(x\ \ 2n–1–x)$ et $Card(K) \geqslant 2^{n-1}$ [3].

3.3 Cas n impair

Théorème :
Si $n \equiv 1$ [4], alors $< I, O > \ = \ Ker(\overline{sgn})$.
Si $n \equiv 3$ [4], alors $< I, O > \ \simeq \ D_n$.

Démonstration :

♣ Cas 1 : $n \equiv 1$ [4] :
L'objectif est de montrer que $Ker(\overline{sgn}) \subset < I, O >$.

Montrons que $A_n \subset < I, O >^*$:
On pose

$$c = O^{-1}(O^{-1}IOI^{-1})^2O \qquad w = I^{-1}Oc^{-1}I^{-1}Oc^2O^{-1}Ic^{-1}$$

Tous calculs faits,

$$c = (0\ 1\ 2...n{-}1)(2n{-}1\ 2n{-}2...n) \qquad w = (n{-}1\ n{-}3\ 1)(n\ n+2\ 2n{-}2)$$

On remarque que $w \in < I, O >^*$, donc en le restreignant à $[\![0, n-1]\!]$, il s'agit d'un 3-cycle. $(c^iwc^{-i})_{i\in\mathbb{N}}$ est une famille de 3-cycles vérifiant :

(i) $\bigcup_{i=0}^{+\infty} c^iwc^{-i} = [\![0, n-1]\!]$ (il s'agit en fait d'une réunion finie).

(ii) $\forall i \in \mathbb{N},\ \exists j \in \mathbb{N} \setminus \{i\},\ Supp(c^iwc{-}i) \cap Supp(c^jwc^{-j}) \neq \emptyset$.

Ainsi cette famille engendre A_n, qui est donc inclus dans $< I, O >^*$.

Montrons que $\overline{< I, O >} = A_n$ (on identifie $\mathfrak{S}(\{\overline{x},\ x \in [\![0, n-1]\!]\})$ et $\mathfrak{S}_n$).
Soit $\sigma \in A_n$.
Comme $A_n \subset < I, O >^*$, il existe $M \in < I, O >^*$ tel que la restriction de M à $[\![0, n-1]\!]$ soit égale à σ. On a alors, par symétrie centrale de M :

$$\overline{M} = \begin{pmatrix} \overline{0} & ... & \overline{n-1} \\ \overline{\sigma(0)} & ... & \overline{\sigma(n-1)} \end{pmatrix}$$

Ce qui équivaut, en identifiant $\overline{x}$ et x, à $\overline{M} = \sigma$.
Puis, d'après la figure 3, $\overline{sgn}(< I, O >) \subset \{1\}$ donc $\overline{< I, O >} \subset A_n$, ce qui fournit l'inclusion réciproque. On a égalité.

Comme $\overline{I} \in \overline{< I, O >}$, il existe $g \in < I, O >^*$, telle que $\overline{g} = \overline{I}$. $g \in < I, O >^*$ donc $sgn(g) = 1$. De plus d'après la figure 3, $sgn(I) = -1$.
Donc $sgn(I^{-1}g) = -1$. Et cette permutation est dans K.
Ainsi, et grâce à la propriété 2, Kcontient toutes les permutations de la forme $(a_1\ \ 2n{-}1{-}a_1)...(a_v\ \ 2n{-}1{-}a_v)$

sans restriction sur la parité de v.

On est à présent en mesure de montrer que $Ker(\overline{sgn}) \subset < I, O >$:
Soit $\sigma \in Ker(\overline{sgn})$.
$\overline{\sigma} \in A_n = \overline{< I, O >}$ donc il existe $M \in < I, O >$ tel que $\overline{\sigma} = \overline{M}$.
Ainsi $\overline{M^{-1}\sigma} = id$ donc on peut décomposer cette permutation en produit de transpositions de la forme $(x \;\; 2n{-}1{-}x)$. Or ces produits sont des éléments de K, donc de $< I, O >$. Finalement, $\sigma \in < I, O >$.

♣ Cas 2 : $n \equiv 3$ [4] :
On a toujours $A_n \subset \overline{< I, O >}$.
Cette fois, $\overline{O}$ est de signature -1, et il s'agit d'un élément de $\overline{< I, O >}$, donc $\overline{< I, O >} = \mathfrak{S}_n$.
D'autre part, $< I, O > \subset Ker(sgn \cdot \overline{sgn})$ donc en particulier, d'après la propriété 2, K ne contient que les produits d'un nombre pair de transpositions de la forme $(x \;\; 2n{-}1{-}x)$.
On pose :

$$\begin{array}{rccc} \text{g} : & < I, O > & \longrightarrow & \text{K} \\ & \text{M} & \longmapsto & \begin{cases} \displaystyle\prod_{\substack{x \in [\![0, n-1]\!], \\ M(x) \in [\![n, 2n-1]\!]}} (M(x) \;;\; 2n-1-M(x)) & \text{si } sgn(M) = 1 \\ \displaystyle\prod_{\substack{x \in [\![0, n-1]\!], \\ M(x) \in [\![0, n-1]\!]}} (M(x) \;;\; 2n-1-M(x)) & \text{si } sgn(M) = -1 \end{cases} \end{array}$$

Montrons que g est bien définie :
Soit $M \in < I, O >$.
On remarque que

$$M = \prod_{\substack{x \in [\![0, n-1]\!], \\ M(x) \in [\![n, 2n-1]\!]}} \big(M(x) \;\; 2n{-}1{-}M(x)\big) \circ e(\overline{M}) \qquad (*)$$

où e est un morphisme qui permet d'étendre une permutation de $\mathfrak{S}_n$ en une permutation à symétrie centrale de $\mathfrak{S}_{2n}$:

$$\begin{array}{rccccccc} \text{e} : & \mathfrak{S}_n & \longrightarrow & & & B_n & & \\ & & & e(\sigma) : & [\![0, 2n-1]\!] & \longrightarrow & [\![0, 2n-1]\!] & \\ & \sigma & \longmapsto & & \text{x} & \longmapsto & \begin{cases} \sigma(x) & si \; x \leqslant n-1 \\ 2n-1-\sigma(2n-1-x) & sinon \end{cases} \end{array}$$

On vérifie que e est bien défini, et qu'il s'agit d'un morphisme injectif.

La relation $(*)$ se démontre point par point en distinguant les cas selon que $x \leqslant n-1$ ou non.
D'après cette relation,

$$sgn(M) = sgn\Big(\prod_{\substack{x \in [\![0, n-1]\!], \\ M(x) \in [\![n, 2n-1]\!]}} \big(M(x) \;\; 2n{-}1{-}M(x)\big)\Big) \cdot sgn(e(\overline{M}))$$

Or comme les images de e conservent $[\![0, n-1]\!]$, elles sont paires, donc la parité du nombre de transpositions dans le produit ci-dessus est celle de M. L'imparité de n finit de prouver que g est bien définie.

On construit enfin la fonction ψ :

$$\begin{array}{rccc} \psi : & < I, O > & \longrightarrow & \mathfrak{S}_n \ltimes K \\ & \text{M} & \longmapsto & (\overline{M} \;,\; g(M)) \end{array}$$

La loi de composition interne associée au produit semi-direct est donnée par :
$\forall \; \sigma_1, \sigma_2 \in \mathfrak{S}_n, \; \forall \; k_1, k_2 \in K,$

$$(\sigma_1, k_1) \star (\sigma_2, k_2) = \Big(\sigma_1\sigma_2, \;\; k_1 e(\sigma_1) k_2 e\big(\sigma_1^{-1}\big)\Big)$$

On montre que ψ est un isomorphisme.

$< I, O >$ est donc isomorphe à $\overline{< I, O >} \ltimes K$. Or on a vu que $\overline{< I, O >} = \mathfrak{S}_n$.

Il ne reste plus qu'à étudier K :
On note Z l'ensemble des éléments de $(\mathbb{Z}/_{2\mathbb{Z}})^n$ de somme des coordonnées nulle. Il s'agit d'un sous-groupe, de cardinal 2^{n-1}.

On pose :

$$\begin{array}{rccc} h_1 : & K & \longrightarrow & Z \\ & (a_1 \;\; 2n-1-a_1)...(a_v \;\; 2n-1-a_v) & \longmapsto & (x_1,...,x_n) \quad \text{avec } x_k = 1 \;\Leftrightarrow\; \exists j \in [\![1,v]\!],\; a_j = k \end{array}$$

h_1 est bien définie car K ne contient que des produits d'un nombre pair de transpositions. h_1 est injective, Z et K ont le même cardinal, donc elle est bijective. Et on vérifie que c'est un morphisme de groupes.

On pose ensuite :

$$\begin{array}{rccc} h_2 : & Z & \longrightarrow & (\mathbb{Z}/_{2\mathbb{Z}})^{n-1} \\ & (a_1,...,a_{n-1},a_n) & \longmapsto & (a_1,...,a_{n-1}) \end{array}$$

h_2 est bien défini, et il s'agit d'un isomorphisme, de réciproque :

$$\begin{array}{rccc} h_2^{-1} : & (\mathbb{Z}/_{2\mathbb{Z}})^{n-1} & \longrightarrow & Z \\ & (a_1,a_2,...,a_{n-1}) & \longmapsto & \left(a_1,a_2,...,a_{n-1},\; \displaystyle\sum_{i=1}^{n-1} a_i\right) \end{array}$$

Finalement, par transitivité de la relation d'isomorphisme, $K \simeq (\mathbb{Z}/_{2\mathbb{Z}})^{n-1}$.
Donc

$$< I, O > \simeq\; \mathfrak{S}_n \ltimes (\mathbb{Z}/_{2\mathbb{Z}})^{n-1}$$

Le groupe de Weyl D_n est isomorphe à ce produit direct [5], ce qui achève la démonstration. ■

3.4 Cas n pair

Théorème :
Si $n \equiv 0$ [4], alors $< I, O > = \; Ker(sgn) \cap Ker(\overline{sgn})$.
Si $n \equiv 2$ [4], alors $< I, O > \simeq B_n$.

Démonstration :
En considérant des combinaisons judicieuses de In et de Out, on parvient à montrer que $A_n \subset < I, O >^*$, puis avec le même raisonnement qu'à la sous-section précédente, on a $A_n \subset \overline{< I, O >}$.

♣ Cas 1 : $n \equiv 0$ [4] :
Ce cas se traite de manière analogue au cas $n \equiv 1$ [4], par double inclusion.

♣ Cas 2 : $n \equiv 2$ [4]:
Comme $\overline{< I, O >}$ contient A_n et une permutation impaire, en l'occurrence I (figure 3), on a $\overline{< I, O >} = \mathfrak{S}_n$.

D'autre part, $\overline{O} \in A_n$ et $A_n \subset < I, O >^*$, donc il existe $g \in < I, O >^*$ tel que $gO^{-1} \in K$.
$g \in < I, O >^*$, $sgn(g) = 1$, donc gO^{-1} est impaire. Ainsi K contient une permutation impaire, donc d'après la propriété 2, il est composé exactement des produits de transpositions de la forme $(x \;\; 2n–1–x)$, et $Card(K) = 2^n$.
Le reste de la démonstration est similaire au cas $n \equiv 3$ [4]. ■

4 Étude pratique : cas 2n = 32

4.1 Généralités

On se place dans le cas d'un paquet de cartes de taille 32, soit 2^5. On a alors :

$$< I, O > \;\; \simeq \; (\mathbb{Z}/_{2\mathbb{Z}})^5 \ltimes \mathbb{Z}/_{5\mathbb{Z}}$$

L'ordre de I vaut 10, l'ordre de O vaut 5 et le cardinal de $< I, O >$ vaut 160.

L'isomorphisme ψ qui permet de passer de $< I, O >$ à $(\mathbb{Z}/_{2\mathbb{Z}})^5 \ltimes \mathbb{Z}/_{5\mathbb{Z}}$ permet également d'expliciter les inverses des mélanges parfaits.

$$I = \psi\big((1,0,0,0,0),\overline{4}\big) \quad \text{et} \quad O = \psi\big((0,0,0,0,0),\overline{4}\big).$$

Donc :

$$I^{-1} = \psi\big(((1,0,0,0,0),\overline{4})^{-1}\big) = \psi((0,0,0,0,1),-\overline{4}) = O^4IO^4 \qquad O^{-1} = \psi((0,0,0,0,0),-\overline{4}) = O^4$$

On savait déjà que $O^4 = O^{-1}$. L'expression de I^{-1} présente un intérêt lorsque l'on s'intéresse à la carte en position 0. En effet, le Out la laisse invariante, donc il suffit d'effectuer un In puis quatre Out pour obtenir l'image de 0 par I^{-1}, au lieu de neuf In consécutifs.

4.2 Calcul du centre

On montre que le centre de $< I, O >$ est trivial :

Soit $(a,\overline{p}) \in Z\left((\mathbb{Z}/_{2\mathbb{Z}})^5 \ltimes \mathbb{Z}/_{5\mathbb{Z}}\right)$. En particulier, $(a,\overline{p})$ commute avec $(a',\overline{0})$, pour $a' \in (\mathbb{Z}/_{2\mathbb{Z}})^5$, c'est-à-dire :
$(a,\overline{p}) \star (a',\overline{0}) = (a',\overline{0}) \star (a,\overline{p})$ donc $\phi_p(a') = a'$.
Ainsi $\phi_p = id$, puis $p = 0$, nécessairement.

D'autre part, $(a,\overline{p})$ commute avec $(0,\overline{p'})$ pour tout $\overline{p'} \in \mathbb{Z}/_{5\mathbb{Z}}$, donc : $a + \phi_p(0) = 0 + \phi_{p'}(a)$, puis $a = \phi_{p'}(a)$.
Donc pour $p' = 0$, on obtient $a = 0$, nécessairement.

Le centre de $< I, O >$ est réduit au singleton $\{Id\}$.

5 Conclusion

Assimiler les mélanges parfaits à des permutations a permis d'expliquer les mécanismes qui sous-tendent le tour de magie. Une fois ce tour compris, il est intéressant d'explorer de manière plus fine le groupe engendré par les In et Out shuffle, notamment pour en dégager des propriétés utiles au tour de magie.
Le cas d'un paquet de cartes de taille impaire n'a pas été évoqué car la définition même des mélanges parfaits repose sur l'hypothèse d'un paquet de taille paire. Cette contrainte provient du fait que l'on n'envisage qu'une seule coupe du paquet. En généralisant le problème à plusieurs coupes, on pourrait considérer des paquets de tailles quelconques.

Bibliographie

[1] PERRIN : Cours d'Algèbre : Chapitre I, Collection de l'Ecole Normale Supérieure de Jeunes Filles, 1988

[2] Aimé LACHAL : Quadrature, Mélanges parfaits de cartes (I) – In-shuffles et out-shuffles : numéro 76, 2010, pp. 13-25

[3] DIACONIS, GRAHAM, KANTOR : Advances in Applied Mathematics, The Mathematics of Perfect Shuffles : volume 4, 1983, pp. 175-196.

[4] DIACONIS, GRAHAM : Math Horizons, The solutions to Elmsley's Problem : volume 14, 2007, pp. 22-27.

[5] PAULIN : Groupes et géométrie : Cours de seconde année de mastère, 2013-2014

[6] Bruno BELHOSTE : Images des Mathématiques, Mélanges de cartes et mathématiques : CNRS, 2013 : $http://images.math.cnrs.fr/Melanges-de-cartes-et-mathematiques$

Annexes

Annexe 1

Complément concernant le terme correctif de l'algorithme :

On note $t = \sum_{i=0}^{r-1} t_i 2^i$.

On a :

$$\left\lfloor \frac{2nt}{2^k} \right\rfloor = \left\lfloor \frac{2n}{2^k} \sum_{i=0}^{k-1} t_i 2^i + 2n \sum_{i=k}^{r-1} t_i 2^{i-k} \right\rfloor$$

$$\left\lfloor \frac{2nt}{2^k} \right\rfloor = \left\lfloor \frac{2nq}{2^k} \right\rfloor + 2nx$$

Avec x entier et $q = \sum_{i=0}^{k-1} t_i 2^i$.

Donc la parité de $\left\lfloor \frac{2nt}{2^k} \right\rfloor$ est celle de $\left\lfloor \frac{2nq}{2^k} \right\rfloor$, ce qui correspond au bit de poids le plus faible dans la décomposition binaire de $\left\lfloor \frac{2nq}{2^k} \right\rfloor$, c'est-à-dire au coefficient numéro k de la décomposition binaire de $2nt$. L'entier $s = 2nt - i2^{r-1}$ donne donc, par sa décomposition binaire, la parité de chaque noeud rencontré. Le soustraction par $i2^r$ permet d'éviter de dépasser r bits. ■

Annexe 2

Complément de démonstration du cas des puissances de 2 :

Montrons que f est un morphisme injectif :
Soient $a = (a_1, a_2, ..., a_k)$ et $b = (b_1, b_2, ..., b_k)$, éléments de $(\mathbb{Z}/_{2\mathbb{Z}})^k$. Montrons que $f(a+b) = f(a)f(b)$:

$$\begin{aligned} f(a+b) &= \prod_{j=1}^{k} B_j^{a_j+b_j} \\ &= \prod_{j=1}^{k} (B_j)^{a_j} \prod_{j=1}^{k} (B_j)^{b_j} \text{ car les } B_j \text{ commutent deux à deux} \\ &= f(a)f(b) \end{aligned}$$

Donc f est un morphisme de groupe. Montrons qu'il est injectif :
Soit $a \in Ker(f)$.

$$f(a) = id$$

$$\prod_{j=1}^{k} B_j^{a_j} = id$$

Action de B_j sur un nombre en binaire :
$\forall (x_1, ..., x_n) \in \{0,1\}^n$,

$$B_j(x_1, ..., x_k) = (x_1, ..., 1 - x_j, ..., x_k)$$

D'où : $\forall (x_1, ..., x_n) \in \{0,1\}^n$,

$$\prod_{j=1}^{k} B_j^{a_j}(x_1, ...x_n) = \begin{pmatrix} x_1 \text{ si } a_1 \text{ est pair} & ... & x_n \text{ si } a_k \text{ est pair} \\ 1 - x_1 \text{ si } a_1 \text{ est impair} & ... & 1 - x_k \text{ si } a_n \text{ est impair} \end{pmatrix}$$

Ainsi :

$$(x_1, ..., x_n) = \begin{pmatrix} x_1 \text{ si } a_1 \text{ est pair} & ... & x_n \text{ si } a_k \text{ est pair} \\ 1 - x_1 \text{ si } a_1 \text{ est impair} & ... & 1 - x_k \text{ si } a_n \text{ est impair} \end{pmatrix}$$

Donc tous les a_j sont pairs, c'est-à-dire nuls (on est dans $\mathbb{Z}/_{2\mathbb{Z}}$). Donc f est injectif.

Annexe 3

Démonstration de la propriété 1 :

Comme le In shuffle correspond à un Out dans un paquet contenant $2n + 2$ cartes, il suffit de démontrer le résultat pour O et de remplacer n par $n + 1$ pour avoir le résultat sur I.

♣ Afin de déterminer la signature de O, on dénombre les inversions de O, c'est-à-dire les couples $(x, y) \in [\![0, 2n-1]\!]^2$ tels que $x < y$ et $O(x) > O(y)$.
Soit $(x, y) \in [\![0, 2n-1]\!]^2$ tel que $x < y$.
Par définition, $O(x) = \begin{cases} 2x \text{ si } x \leqslant n-1 \\ 2x - 2n + 1 \text{ sinon} \end{cases}$.
Donc

$$O(x) > O(y) \quad \Leftrightarrow \quad x \in [\![0, n-1]\!] \text{ et } y \in [\![n, x+n-1]\!]$$

Il y a $\sum_{x=0}^{n-1}(n + x - 1 - n + 1) = \frac{n(n+1)}{2}$ inversions.
Si $n \equiv 0$ [4], alors $\frac{n(n+1)}{2}$ est pair donc $sgn(O) = 1$.
Si $n \equiv 1$ [4], $sgn(O) = 1$.
Si $n \equiv 2$ [4], $sgn(O) = -1$.
Si $n \equiv 3$ [4], $sgn(O) = -1$.

♣ Pour la signature de $\overline{O}$, on dénombre les couples $(x, y) \in [\![0, n-1]\!]^2$ tels que $x < y$ et $\overline{O}(x) > \overline{O}(y)$, en identifiant les doublets $\{x, 2n-1-x\}$ aux entiers $x \in [\![0, n-1]\!]$.
Soit $(x, y) \in [\![0, n-1]\!]^2$ tel que $x < y$ et $\overline{O}(x) > \overline{O}(y)$.

Si $x \geqslant \frac{n}{2}$, les doublets $\{2x, 2n-1-2x\}$ et $\{2y, 2n-1-2y\}$ s'identifient à $2n-1-2x$ et $2n-1-2y$, donc la condition $\overline{O}(x) > \overline{O}(y)$ se réécrit :

$$2(n-x) - 1 > 2(n-y) - 1$$

On note $m = \lfloor \frac{n+1}{2} \rfloor$.
Le nombre de couples qui vérifient cela est $\sum_{x=m}^{n-1}(n - 1 - (x+1)) = \frac{(n-m)(n-m-1)}{2}$, et ce sont tous des inversions.

Si $x < \frac{n}{2}$, comme (x, y) est une inversion, nécessairement $y \geqslant \frac{n}{2}$ (sinon on aurait $2x > 2y$ et $x < y$, ce qui est absurde). Donc $2x > 2n - 1 - 2y$, ce qui équivaut à :

$$x + y \geqslant n$$

Le nombre de couples qui vérifient cela est $\sum_{x=0}^{\lfloor \frac{n-1}{2} \rfloor} (n-1-(n-x)+1) = \frac{m(m-1)}{2}$.

Le nombre total d'inversion vaut $\frac{(n-m)(n-m-1)}{2} + \frac{m(m-1)}{2} = \begin{cases} m(m-1) \text{ si n est pair} \\ \frac{(n-1)^2}{4} \text{ si n est impair} \end{cases}$. ■

Script python

```
import numpy.random as rd

"""p est une liste python representant le paquet de cartes."""

"""Melanges parfaits : """

def In(p):
    n = (len(p)/2)
    q = [k for k in p]
    for i in range(n):
        p[2*i+1] = q[i]
    for i in range(n,2*n):
        p[2*i-2*n] = q[i]

def Out(p):
    n = (len(p))/2
    q = [k for k in p]
    for i in range(n):
        p[2*i] = q[i]
    for i in range(n,2*n):
        p[2*i+1-2*n] = q[i]

"""Inverses des melanges parfaits :"""

def rIn(p):
    n = (len(p))/2
    q = [k for k in p]
    for i in range(n):
        p[i] = q[2*i+1]
    for i in range(n,2*n):
        p[i] = q[2*i-2*n]

def rOut(p):
    n = (len(p))/2
    q = [k for k in p]
    for i in range(n):
        p[i] = q[2*i]
    for i in range(n,2*n):
        p[i] = q[2*i+1-2*n]

"""Melanges Bj qui servent dans la demonstration du cas des puissances de 2 :"""

def B(j,p):
    for i in range(j-1):
        Out(p)
    In(p)
    for i in range(j):
        rOut(p)

"""Implementation de l'algorithme du tour de magie :"""

def encadrement(n): #calcul de l'entier r
```

```
    q = n
    r = 0
    while q > 1:
        r += 1
        q = q//2
    return r


def binaire(t,r): #decomposition en binaire comprenant r bits d'un entier t
    T = r*[0]
    u = t
    while u != 0:
      c = encadrement(u)
      if r != c:
          T[r-c-1] = 1
      u = u-(2**c)
    return T

"""La fonction top renvoie une sequence de melanges qui transporte
la ieme carte en position 0."""

def top(i,n):
    r = encadrement(2*n) + 1
    if i == 0:
        t = 0
    elif i == (2*n) - 1:
        t = (2**r) - 1
    else:
        t = ((i+1)*(2**r))//(2*n)
    if t == (i+1)*(2**r)*1./(2*n):
        t = t-1
    T = binaire(t,r)
    S = binaire(2*n*t - i*2**r,r)
    return [(T[k] + S[k])%2 for k in range(r)]


"""La fonction sequence renvoie la liste des melanges a effectuer pour deplacer
la ieme carte en position j dans un paquet de taille n :"""

def sequence(i,j,n):
    A = top(i,n)
    if j == 0:
        B = []
    else:
        r = encadrement(j) + 1
        B = binaire(j,r)
    return A+B

"""La fonction tour effectue elle-meme le tour, par effets de bords, sur un
paquet p :"""

def tour(p,i,j):
    n = len(p)//2
    for k in sequence(i,j,n):
        if k == 0:
            Out(p)
        else:
            In(p)
    return p

"""Generation d'un paquet aleatoire :"""
```

```
def paquet(n):
    p = []
    q = range(n)
    while len(q) >= 2 :
        c = rd.randint(0,len(q)-1)
        p.append(q[c])
        del q[c]
    p.append(q[0])
    return p

"""Out et In shuffle en notations permutation de S_2n : """

def I(n):
    s = [0]*(2*n)
    for i in range(n):
        s[i] = 2*i+1
    for i in range(n,2*n):
        s[i] = 2*i-2*n
    return s

def O(n):
    s = [0]*(2*n)
    for i in range(n):
        s[i] = 2*i
    for i in range(n,2*n):
        s[i] = 2*i+1-2*n
    return s

"""Fonction s -> sbarre :
Les permutations en arguments sont a symetrie centrale."""

def permutation_barre(s):
    n = len(s)/2
    sbarre = [0]*n
    for i in range(n):
        sbarre[i] = (s[i],s[2*n-1-i])
    return sbarre

"""Fonctions de base sur les permutations : """

def compose(f,g):
    n = len(f)
    h = [0]*n
    for i in range(n):
        h[i] = f[g[i]]
    return h

def itere(f,k):
    g = range(len(f))
    for i in range(k):
        g = compose(f,g)
    return g

def inverse(f):
    n = len(f)
    g = [0]*n
    for i in range(n):
        g[f[i]] = i
    return g
```