N° d'ordre : 31/1999 **Université de Limoges**

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : **Mathématiques et Applications**
Spécialité : **Mathématiques et Informatique**

présentée par

Gintaras SKERSYS

Sujet de la thèse :

CALCUL DU GROUPE D'AUTOMORPHISMES DES CODES. DÉTERMINATION DE L'ÉQUIVALENCE DES CODES

Soutenue le 21 octobre 1999 devant le jury composé de :

Président

M.	**Guy**	**ROBIN**	Professeur Émérite, Université de Limoges

Directeur

M.	**Thierry**	**BERGER**	Professeur, Université de Limoges

Codirecteur

M.	**Nicolas**	**SENDRIER**	Chargé de Recherche, INRIA-Rocquencourt

Rapporteurs

Mme	**Pascale**	**CHARPIN**	Directeur de Recherche, INRIA-Rocquencourt
M.	**Patrick**	**FITZPATRICK**	Professeur, Université de Cork (Irlande)

Examinateurs

M.	**François**	**LAUBIE**	Professeur, Université de Limoges
M.	**François**	**MORAIN**	Ingénieur de l'Armement, École Polytechnique

N° d'ordre : 31/1999 **Université de Limoges**

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : **Mathématiques et Applications**
Spécialité : **Mathématiques et Informatique**

présentée par

Gintaras SKERSYS

Sujet de la thèse :

CALCUL DU GROUPE D'AUTOMORPHISMES DES CODES. DÉTERMINATION DE L'ÉQUIVALENCE DES CODES

Soutenue le 21 octobre 1999 devant le jury composé de :

Président			
M.	**Guy**	**ROBIN**	Professeur Émérite, Université de Limoges
Directeur			
M.	**Thierry**	**BERGER**	Professeur, Université de Limoges
Codirecteur			
M.	**Nicolas**	**SENDRIER**	Chargé de Recherche, INRIA-Rocquencourt
Rapporteurs			
Mme	**Pascale**	**CHARPIN**	Directeur de Recherche, INRIA-Rocquencourt
M.	**Patrick**	**FITZPATRICK**	Professeur, Université de Cork (Irlande)
Examinateurs			
M.	**François**	**LAUBIE**	Professeur, Université de Limoges
M.	**François**	**MORAIN**	Ingénieur de l'Armement, École Polytechnique

Remerciements

Je tiens à adresser mes remerciements à

Thierry Berger pour l'attention qu'il a portée sur ce travail durant ces trois années, pour sa grande disponibilité et son contact facile, ses nombreux et toujours précieux conseils. Il a su me donner le goût de la recherche.

Nicolas Sendrier pour le suivi régulier qu'il a apporté à cette thèse ainsi que pour toutes les enrichissantes discussions que nous avons eues.

Stanislovas Norgėla, dont les efforts m'ont permis d'effectuer ce séjour à Limoges, pour son soutien constant.

Guy Robin pour la gentillesse qu'il a toujours témoignée à mon égard et pour avoir accepté d'être le président du jury.

Pascale Charpin et Patrick Fitzpatrick qui m'ont fait l'honneur d'accepter la tâche d'être rapporteurs.

François Laubie et François Morain pour leur aimable participation au jury.

François Arnault pour sa disponibilité pour toutes les questions d'ordre mathématique ou informatique.

Tous les collègues du Département de mathématiques de l'Université de Limoges. Ils ont su instaurer l'ambiance sympathique et d'agréables conditions de travail au sein du département.

L'équipe du Projet Codes de l'INRIA pour son accueil chaleureux qui me fut réservé lors de mes passages à Rocquencourt.

Asta Radikienė qui était la première à me montrer la beauté des mathématiques.

Violeta, mon épouse, pour son amour.

Tous ceux dont j'ai reçu, sur le plan humain et mathématique, soutien tout au long de ce travail.

Table des matières

Notations

$\mathbb{N}$	$=$	$\{0, 1, 2, 3, \ldots\}$ l'ensemble des entiers rationnels[1] positifs,
$\mathbb{N}^*$	$=$	$\mathbb{N} \setminus \{0\} = \{1, 2, 3, \ldots\}$ l'ensemble des entiers positifs non nuls,
$\mathbb{Z}$	$=$	$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ l'ensemble des entiers,
$\mathbb{R}$	$=$	l'ensemble des nombres réels.

Soient A et B des ensembles, E un ensemble fini muni d'un ordre total.

$\|A\|$	$=$	le cardinal de A,
$A \times B$	$=$	$\{(a, b) \mid a \in A, b \in B\}$ l'ensemble des couples ordonnés des éléments de A et B,
$\min(E)$	$=$	le plus petit élément de E,
$\max(E)$	$=$	le plus grand élément de E.

Soit n un entier positif.

I_n	$=$	$\{1, \ldots, n\}$,
Z_n	$=$	$\{0, 1, \ldots, n-1\}$.

Soient a et b des entiers non nuls ou des polynômes à une indéterminée non nuls.

$a \mid b$	$=$	si a divise b,
(a, b)	$=$	le plus grand commun diviseur de a et b,
$\operatorname{ppcm}(a, b)$	$=$	le plus petit commun multiple de a et b.

Soit m un entier, α un entier positif et p un nombre premier.

$p^\alpha \parallel m$	$=$	si p^α est la plus grande puissance de p divisant m, c'est-à-dire $p^\alpha \mid m$ et $p^{\alpha+1} \nmid m$.

Soit n un entier positif non nul.

$n\mathbb{Z}$	$=$	$\{nm \mid m \in \mathbb{Z}\}$,
$\mathbb{Z}/n\mathbb{Z}$	$=$	$\{0, 1, 2, 3, \ldots, n-1\}$ l'ensemble des restes de la division des éléments de $\mathbb{Z}$ par n,
$(\mathbb{Z}/n\mathbb{Z})^*$	$=$	$\{m \in \mathbb{Z}/n\mathbb{Z} \mid m > 0,\ (m, n) = 1\}$.

Soit h un polynôme à une indéterminée.

$\deg h$	$=$	le degré de h.

Soient f et g des fonctions de $\mathbb{R}$ dans $\mathbb{R}$ ou de $\mathbb{N}$ dans $\mathbb{R}$.

1. Par la suite, nous dirons simplement "un entier" pour désigner un entier rationnel.

$$
\begin{array}{lcl}
f = o(g) & = & \text{si } \frac{f(x)}{g(x)} \to 0 \text{ quand } x \to \infty, \\
f \sim g & = & \text{si } \frac{f(x)}{g(x)} \to 1 \text{ quand } x \to \infty.
\end{array}
$$

Introduction générale

Présentation du problème

Il est possible de définir la notion de groupe d'automorphismes d'un code de plusieurs manières :

- comme groupe de permutations,
- comme groupe de matrices monomiales,
- comme groupe de matrices monomiales et d'automorphismes de corps de base,

chaque définition étant plus générale que les précédentes. Dans ce travail nous nous intéresserons à deux premiers cas. S'il s'agit des permutations, nous parlerons de groupe de permutations, sinon de groupe d'automorphismes.

L'étude des groupes d'automorphismes et de permutations d'un code est un problème important en théorie des codes correcteurs d'erreurs :

- certains algorithmes de décodage utilisent les permutations,
- la connaissance des groupes d'automorphismes et de permutations aide à déterminer si deux codes sont équivalents,
- la plupart des codes classiques ont des groupes d'automorphismes et de permutations assez particuliers, par exemple, les codes de Golay et les groupes de Mathieu, les codes de Reed et Muller et les groupes linéaires, les codes de Reed-Solomon et le groupe affine, les codes résidus quadratiques et le groupe spécial projectif, etc...

Un grand travail a été fait pour déterminer de façon théorique les groupes de permutations et d'automorphismes des certaines familles de codes. Par exemple, T. Berger et P. Charpin ont déterminé les groupes de permutations et d'automorphismes des codes BCH primitifs au sens strict et des codes affine-invariants cycliques étendus [BC96], W.C. Huffman l'a fait pour les codes résidus quadratiques généralisés [Huf95], voir aussi [Huf98] pour une synthèse de la recherche théorique dans ce domaine. Nous nous intéressons au calcul effectif des groupes de permutations et d'automorphismes des codes.

La notion d'équivalence entre deux codes est liée à la notion de groupe d'automorphismes. Elle aussi peut être définie en plusieurs niveaux comme ci-dessus. Supposons que nous ayons deux codes. Il s'agit de trouver une permutation (ou une matrice monomiale,...) telle que l'image du premier code par cette permutation est le deuxième code. S'il s'agit des permutations, nous dirons que les codes sont équivalents par permutation, sinon nous dirons tout simplement équivalents.

Toute une classe de cryptosystèmes fondés sur la théorie des codes correcteurs d'erreurs a émergé dès l'apparition des premiers systèmes à clef publique. Les systèmes de chiffrement proposés par R.J. McEliece [McE78] et H. Niederreiter [Nie86] utilisent comme clef secrète un code linéaire très structuré, pour lequel on dispose d'un algorithme de décodage rapide, et fournissent comme clef publique un code équivalent, construit de façon à dissimuler la structure algébrique initiale et donc pour qu'il se comporte comme un code aléatoire.

D'ailleurs, l'apparition d'un algorithme efficace pour trouver une permutation entre deux codes équivalents par permutation (dans le cas où cette permutation est unique, cf. ci-dessous et [Sen97b, Sen99a, Sen99b]) a permis de "casser" dans certains cas les systèmes de chiffrement de McEliece et de Niederreiter (cf. [LS98]).

Outre son importance en cryptographie, l'équivalence aide à classifier les codes, en particulier, les codes auto-duaux. Deux codes équivalents ont la même structure : même distance minimale, même distribution des poids, leurs groupes d'automorphismes sont isomorphes.

Tout cela nous a incité à nous intéresser à la détermination effective de l'équivalence et de l'équivalence par permutation des codes et au calcul effectif des groupes d'automorphismes et de permutations des codes.

Point de vue algorithmique

Avec le développement de l'informatique dû à l'apparition des ordinateurs toujours plus rapides et pourvus de plus de mémoire, la théorie des algorithmes a connu un essor spectaculaire pendant les trois dernières décennies. La théorie algorithmique des groupes est une de ses branches les plus avancées, aussi bien du point de vue de la gamme et de la sophistication des algorithmes que de leur efficacité. Les algorithmes de la théorie algorithmique des groupes trouvent une application non seulement dans l'étude directe des groupes, mais aussi dans beaucoup d'autres branches de mathématiques, par exemple, dans la théorie des codes correcteurs d'erreurs, la théorie des designs, les équations différentielles, la géométrie finie, la théorie des graphes, l'analyse harmonique, la cristallographie mathématique, la théorie des nombres et la topologie (voir [CH92] pour un survol de la théorie algorithmique des groupes).

Base et ensemble fort de générateurs

Soit G un groupe de permutations d'un ensemble fini Ω. Un problème important de la théorie algorithmique des groupes est de déterminer efficacement les propriétés du groupe G. Le plus souvent G est donné par un ensemble de générateurs. Mais cette spécification ne permet pas, par exemple, de trouver facilement l'ordre de G. Si nous voulons manipuler efficacement de grands groupes de permutations sur un ordinateur, nous avons besoin d'une méthode pour les représenter telle que cette représentation satisfasse les caractéristiques suivantes:

- elle doit refléter les aspects clés de la structure du groupe,
- elle doit posséder la propriété d'héritage, c'est-à-dire un sous-groupe de G doit hériter directement sa représentation de celle de G,
- elle doit être efficacement calculable.

Charles C. Sims [Sim70, Sim71a, Sim71b] a introduit les notions d'une base et d'un ensemble fort de générateurs (BSGS, "base and strong generating set") comme une représentation appropriée (cf. section 1.7 [2]). Une *base* de G est une suite $B = (\alpha_1, \ldots, \alpha_s)$ d'éléments distincts de Ω telle que l'identité est l'unique élément de G qui fixe tous les points de B. B définit la chaîne de stabilisateurs de G

$$G \supseteq G_{\alpha_1} \supseteq G_{\alpha_1,\alpha_2} \supseteq \ldots \supseteq G_{\alpha_1,\alpha_2,\ldots,\alpha_{s-1}} \supseteq G_{\alpha_1,\alpha_2,\ldots,\alpha_{s-1},\alpha_s} = \{id\}.$$

Un *ensemble fort de générateurs* S de G associé à la base B est un ensemble de générateurs de G contenant des générateurs pour tout stabilisateur de cette chaîne. Étant donnés B et S, il est facile à calculer l'orbite $\Delta^{(i)} = {\alpha_i}^{G_{\alpha_1,\ldots,\alpha_{i-1}}}$ de α_i sous l'action de $G_{\alpha_1,\ldots,\alpha_{i-1}}$ et le transversal $U^{(i)}$ de $G_{\alpha_1,\ldots,\alpha_i}$ dans $G_{\alpha_1,\ldots,\alpha_{i-1}}$ pour tout $i = 1, \ldots, s$ (cf. section 1.7.5 et algorithme 1.7.10). La connaissance de $\Delta^{(i)}$ et $U^{(i)}$ pour tout $i = 1, \ldots, s$ nous donne immédiatement l'ordre de G, le test d'appartenance à G, la possibilité de parcourir tous les éléments de G sans répétition, et la possibilité de choisir un élément de G au hasard (cf. section 1.7.5).

L'hypothèse principale dans les calculs avec des groupes de permutations est la supposition que le groupe G est représenté par une BSGS (cf. section 1.7.6 pour des remarques sur le calcul d'une BSGS de G). Sims a montré que la disponibilité d'une BSGS nous permet de créer des algorithmes efficaces pour construire des classes importantes de sous-groupes et de groupes quotients de G. Partout où cela est possible, cette construction est organisée de façon que le nouveau groupe hérite une BSGS de la BSGS connue de G. Par exemple, les algorithmes de recherche arrière pour construire des sous-groupes, décrits ci-dessous, produisent automatiquement une BSGS du sous-groupe.

La recherche arrière

Il existe une grande classe de problèmes intéressants de la théorie des groupes pour lesquels on ne connaît pas d'algorithmes fiables lorsque les groupes sont de degré élevé, par exemple, les problèmes du calcul des centralisateurs, des normalisateurs, des intersections

2. Nous utilisons la numérotation suivante : la section $p.q.r$ est la sous-section r de la section q du chapitre p. Pareil pour la section $p.q$ ou $p.q.r.s$. D'ailleurs, la proposition (le lemme, le théorème,. . .) $p.q.r$ est le r-ième énoncé de la section $p.q$.

des groupes, des stabilisateurs d'ensembles, etc... De plus, il est connu (cf. [Hof82]) que tous ces problèmes sont au moins aussi difficiles que le célèbre problème d'isomorphisme des graphes, pour lequel on ne connaît pas, à ce jour, d'algorithme polynomial. D'ailleurs, le problème d'équivalence des codes est aussi connu pour être au moins aussi difficile que le problème d'isomorphisme des graphes (cf. [PR97]). Cela nous incite à chercher des algorithmes qui seraient efficaces pour la plupart des groupes de degré élevé (ou pour la plupart des codes de grande longueur) qui apparaissent dans la pratique, même si les performances des mauvais cas de ces algorithmes restent faibles. Dans ce travail nous présentons une technique pour les problèmes liés aux codes (le calcul des groupes d'automorphismes et de permutations, et la détermination d'équivalence et d'équivalence par permutation) qui est, à notre avis, un pas considérable dans cette direction.

La plupart des algorithmes les plus efficaces de la théorie algorithmique des groupes (par exemple, pour les problèmes mentionnés ci-dessus) effectuent presque une recherche exhaustive de toutes les possibilités. Une telle recherche peut être faite en utilisant une technique appelée *recherche arrière* ("backtrack search" ou "backtracking", cf. section 2.3.1 pour sa description). Elle permet de diminuer fortement l'espace de la recherche en éliminant par paquets les possibilités qui ne conviennent pas. Nous appelons *algorithmes de recherche arrière* des algorithmes utilisant cette technique.

De l'autre coté, ces algorithmes utilisent la notion de BSGS pour tailler l'arbre associé à la recherche arrière. En général, si on veut trouver un sous-groupe H de G composé de tous les éléments de G satisfaisant une certaine propriété, on construit H en commençant par l'autre bout de sa chaîne de stabilisateurs. D'abord on trouve une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ de H, on pose $H^{(s+1)} \stackrel{\text{déf}}{=} H_{\alpha_1,\alpha_2,\ldots,\alpha_s} = \{id\}$, et on remonte de $H^{(i)}$ à $H^{(i-1)}$ en utilisant la recherche arrière. Comme ça on trouve H et, en même temps, sa BSGS. Le calcul du groupe de permutations d'un code C entre aussi dans cette catégorie d'algorithmes. En effet, on peut considérer qu'on cherche le sous-groupe $Perm(C)$ de $Sym(\Omega)$, $\Omega = \{1, \ldots, n\}$, où n est la longueur de C, composé de tous les éléments g de $Sym(\Omega)$ tels que C reste inchangé sous l'action de g.

La méthode des partitions

L'article de J. Leon [Leo91] représente un pas majeur dans l'évolution des algorithmes de recherche arrière en théorie algorithmique des groupes. La *méthode des partitions* présentée dans cet article, en plus de concepts de BSGS et de recherche arrière, utilise les raffinements consécutifs de partitions ordonnées, introduits pour la première fois par B. McKay [McK78, McK81] comme une partie de son très efficace algorithme pour le problème d'isomorphisme des graphes. Ces raffinements facilitent le choix d'une base plus courte et plus optimale et permettent une taille plus importante dans l'arbre associé à la recherche arrière.

Un autre avantage de la méthode des partitions est qu'elle permet le traitement uniforme d'une grande classe de problèmes, par exemple, ceux liés aux codes. Le même algorithme peut être appliqué à tous les problèmes de cette classe ; ce n'est que son entrée, un ensemble d'applications appelées *raffinements*, qui varie selon le problème particulier.

Soit Ω un ensemble fini. Nous voulons trouver l'ensemble H des permutations de Ω satisfaisant une certaine propriété $\mathcal{P}$ (par exemple, la propriété d'appartenance au groupe de permutations d'un code C).

Soit $PartOrd(\Omega)$ l'ensemble des partitions ordonnées de Ω. Une partition est *discrète* si toutes ses cellules (ses blocs) sont de cardinal 1. Une partition $\mathbf{\Pi}$ est *plus fine* qu'une autre partition $\mathbf{\Sigma}$ si toutes les cellules de $\mathbf{\Pi}$ sont contenues dans les cellules de $\mathbf{\Sigma}$. Un *raffinement* est un couple d'applications de $PartOrd(\Omega)$ dans $PartOrd(\Omega)$ qui raffinent les partitions et qui préservent l'action des permutations de H.

La recherche des permutations de H peut être représentée par un arbre[3]. Les noeuds de cet arbre sont étiquetés par les couples de partitions ordonnées. La racine de l'arbre est étiquetée par $((\Omega), (\Omega))$, où (Ω) est la partition d'une unique cellule. À mesure que nous descendons dans l'arbre, les partitions deviennent de plus en plus fines, en particulier à cause de l'utilisation des raffinements, jusqu'à ce que nous atteignions les feuilles étiquetées par $(\mathbf{\Pi}, \mathbf{\Pi}^g)$ avec $\mathbf{\Pi}$ (et donc $\mathbf{\Pi}^g$) discrète et $g \in H$. Il existe exactement une feuille pour toute permutation de H, et nous pouvons trouver g à partir de $(\mathbf{\Pi}, \mathbf{\Pi}^g)$.

L'arbre est parcouru à l'aide de la méthode de recherche arrière, et il est taillé, en particulier, à l'aide de la BSGS.

L'arbre peut contenir des noeuds inutiles, cela dépend du "pouvoir" des raffinements.

La méthode des partitions nous semble très technique, nous avons donc essayé de la clarifier en la présentant (chapitre 2) d'une manière différente de celle de [Leo91].

Nous présentons d'abord la méthode de recherche arrière (section 2.3.1), et ensuite nous introduisons la méthode des partitions (section 2.3.3) en la comparant avec cette dernière. Nous espérons que c'est le moyen le plus compréhensible pour faire connaissance avec la méthode des partitions. Puis nous donnons les définitions nécessaires (section 2.4) et décrivons la méthode des partitions en détail (section 2.5–2.10).

Les raffinements pour les problèmes liés aux codes linéaires

Dans son article [Leo91] Leon propose un ensemble de raffinements pour, entre autres, les problèmes liés aux codes linéaires. Mais ces raffinements ne sont pas propres aux codes. Ce sont des raffinements pour les problèmes liés aux matrices. L'algorithme de Leon pour les problèmes liés aux codes linéaires demande donc un ensemble W de vecteurs invariant sous l'action du groupe d'automorphismes (ou, respectivement, de permutations) d'un code linéaire donné C, et le considère comme une matrice. W peut être, par exemple, l'ensemble de mots de poids minimal (ou fixé) de C, de son dual $C^{\perp}$ ou de son hull $\mathcal{H}(C)$ (l'intersection de C avec son dual), ou une réunion de tels ensembles. Mais W doit être assez grand pour "bien représenter" le code, et assez petit pour qu'on puisse le manipuler facilement. Il est difficile de trouver un tel ensemble W quand C et $C^{\perp}$ sont trop grands et $\mathcal{H}(C)$ est trop petit, ce qui est souvent le cas, par exemple, les codes linéaires aléatoires ont, en moyenne,

3. Nous l'imaginons renversé, c'est-à-dire la racine est en haut et les feuilles sont en bas.

le hull très petit (de dimension rarement supérieure à 3, cf. [Sen97a, Sen99a]). Donc dans le cas binaire, par exemple, l'algorithme de Leon ne marche plus pour les codes linéaires de longueur supérieure à une centaine. Dans ce travail nous proposons une solution pour les grands codes linéaires ayant un petit hull.

N. Sendrier a proposé [Sen96, Sen97b, Sen99a, Sen99b] un algorithme pour trouver une permutation entre deux codes équivalents par permutation dans le cas où elle est unique. Il a introduit la notion de *signature* comme une propriété d'une position d'un code telle que l'ensemble des signatures pour un code donné est globalement invariant par permutation. Si une permutation qui envoie un code sur un autre est unique (ce qui est valable si et seulement si les groupes de permutations de ces codes sont triviaux, i.e. ne contiennent que l'identité), pour la déterminer il suffit de trouver une signature à la fois discriminante et facile à calculer. Sendrier a proposé d'utiliser l'énumérateur des poids du hull d'un code linéaire, qui permet de construire une telle signature pour les codes de petit hull et de groupe de permutations trivial.

Nous généralisons la notion de signature (section 3.3) pour pouvoir en construire des raffinements efficaces pour la méthode des partitions (section 3.4). Si nous avons une partition $\mathbf{\Pi}$ du support du code C, nous poinçonnons C dans une réunion de cellules de $\mathbf{\Pi}$, et nous appliquons la signature S de Sendrier aux positions restantes. L'ensemble des valeurs de S sur ces positions induit une partition du support de C, dont l'intersection avec $\mathbf{\Pi}$ donnera une partition plus fine.

C'est, sans entrer dans les détails, le fonctionnement de nos raffinements pour les problèmes d'équivalence par permutation des codes linéaires et de groupe de permutations d'un code linéaire. La signature de Sendrier n'est pas applicable directement pour les problèmes de groupe d'automorphismes d'un code linéaire et d'équivalence des codes linéaires. Leon a proposé [Leo91, §9(g)] une idée concernant le remplacement du calcul du groupe d'*automorphismes* d'une matrice par celui du groupe de *permutations* de sa matrice étalée. Nous avons pu adapter cette idée pour les codes (section 1.6.3). La méthode des partitions avec les raffinements construits à partir des signatures généralisées marche donc aussi pour les problèmes de groupe d'automorphismes d'un code linéaire et d'équivalence des codes linéaires.

Comme cette méthode utilise la signature de Sendrier, elle ne marche en pratique que si les hulls des codes linéaires donnés sont petits.

Implantation

Nous avons implanté en langage C et testé la méthode des partitions, avec les raffinements construits à partir des signatures généralisées, pour le problème du groupe de permutations des codes linéaires binaires. Nous avons choisi C pour l'efficacité, mais cela a rendu la programmation ardue. La mise au point a été complexe (à peu près 3800 lignes du code source), a pris du temps et a abouti, entre autres, aux heuristiques que nous décrivons dans la section 3.8.

Nous avons étudié les heuristiques possibles et les performances par rapport à divers paramètres (sections 3.8 et 3.10). Cette méthode nous semble très prometteuse, car, par exemple, pour les codes résidus quadratiques binaires de la forme $[8k+1, 4k]$ (leurs hulls sont réduits à zéro), notre programme prend plusieurs secondes pour la longueur 1000, plusieurs minutes pour la longueur 2000, jusqu'à une heure pour la longueur jusqu'à 5000.

Étude de dimension moyenne du hull des codes cycliques

La méthode des partitions avec nos raffinements ne marche que si le hull d'un code linéaire donné est petit (dans le cas binaire, par exemple, de dimension inférieure en général à vingt, cela dépend en fait de la longueur du code, plus la longueur est grande, plus la dimension du hull doit être petite). Les hulls de la plupart des codes linéaires sont petits, mais dans certaines classes importantes de codes il existe beaucoup de codes ayant un grand hull. Par exemple, P. Charpin et F. Levy-dit-Vehel [CL94] ont trouvé une famille infinie de codes affine-invariants auto-duaux. De même, nous avons remarqué que beaucoup de codes cycliques possèdent un grand hull. Nous nous sommes posés la question suivante : combien y-a-t-il de codes cycliques pour lesquels notre méthode n'est pas applicable? Nous avons fait des essais (cf. figure 4.1, p. 142) qui nous ont conduit à la conjecture suivante :

Le hull de la "plupart" de codes cycliques est "grand".

Nous présentons dans le chapitre 4 quelques résultats obtenus au cours de notre étude pour démontrer cette conjecture, que nous n'avons malheureusement pas pu établir.

Nous définissons $Edimh_q(n)$ comme la dimension moyenne du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ (section 4.6.1). En utilisant les techniques de la théorie des nombres, en particulier, l'étude de l'ensemble $\mathcal{N}_q$ des diviseurs des entiers de la forme q^m+1, $m>0$, nous avons pu montrer que $Edimh_q(n) = 0$ si et seulement si $n \in \mathcal{N}_q$ (sections 4.6.2 et 4.7.1), et si ce n'est pas le cas, $Edimh_q(n)$ est supérieur à $\frac{n}{c}$ où c est une constante (section 4.8.4). Donc soit $Edimh_q(n)$ est nul (si et seulement si $n \in \mathcal{N}_q$), soit il est "grand" (de l'ordre de n).

Nous avons étudié le comportement asymptotique de $Edimh_q(n)$ quand n tend vers l'infini (sections 4.8.2.2–4.8.2.3 et 4.8.3.1).

Il reste la question de savoir si $\mathcal{N}_q$ est grand par rapport à $\mathbb{N}$, c'est-à-dire s'il y a beaucoup de n tels que $Edimh_q(n) = 0$. Nous conjecturons (section 4.9) que

$\mathcal{N}_q$ est négligeable par rapport à $\mathbb{N}$, autrement dit, presque tous les entiers positifs non nuls n'appartiennent pas à $\mathcal{N}_q$.

Si cette conjecture est vraie, il existe "peu" d'entiers n tels que $Edimh_q(n)$ soit "petit", i.e. tels que notre méthode marche pour la plupart de codes cycliques de longueur n sur $\mathbf{F}_q$.

Chapitre 1

Définitions et propriétés élémentaires

1.1 Introduction

Ce chapitre est un chapitre de préliminaires. Il s'agit ici de présenter la terminologie et les principales notations, tout en ciblant les objets étudiés. Les définitions et résultats énoncés constituent la base pour explorer ces objets. D'autres éléments viendront les compléter au cours des différents chapitres. Par exemple, les notions et les rappels, qui ne seront nécessaires que pour l'étude de la dimension du hull des codes cycliques dans le chapitre 4, seront présentés dans la section 4.2.

D'abord nous rappelons très brièvement la notion d'un corps fini (section 1.2), et introduisons les automorphismes d'un corps fini (section 1.2.1). Ensuite nous définissons les codes linéaires (section 1.3.1) et cycliques (section 1.3.4) et donnons les premières propriétés.

Comme nous traitons les groupes de permutations dans le chapitre 2, nous aurons besoin de certaines notions de la théorie des groupes de permutations. Nous les présentons dans la section 1.4 : nous rappelons les définitions et les propriétés des permutations (section 1.4.1), des groupes de permutations (section 1.4.2), des classes à droite (section 1.4.4), des orbites (section 1.4.5) et des stabilisateurs (section 1.4.6). De même, pour présenter la méthode des partitions de J. Leon (chapitre 2), nous aurons besoin de beaucoup de notations concernant les partitions et de deux ordres sur les partitions. Nous les présentons dans la section 1.5.

Dans la section 1.6 nous définissons les groupes de permutations et d'automorphismes des codes et introduisons les notions d'équivalence et d'équivalence par permutation des codes (sections 1.6.1–1.6.2). D'ailleurs, nous montrons qu'il est possible de remplacer le calcul du groupe d'automorphismes d'un code par celui du groupe de permutations de son code étalé, et de la même manière il est possible de remplacer la détermination d'équivalence de deux codes par celle d'équivalence par permutation de leurs codes étalés (section 1.6.3).

Enfin, dans la section 1.7 nous introduisons la notion d'une base et d'un ensemble fort de générateurs (BSGS) pour un groupe de permutations, et nous montrons comment ces concepts permettent de répondre efficacement à certaines questions sur ce groupe, par exemple, de trouver l'ordre du groupe ou d'énumérer tous les éléments du groupe sans répétition. Ces concepts sont utilisés dans la méthode des partitions de J. Leon (chapitre 2).

1.2 Corps finis

Nous définissons dans cette section les corps finis. Nous le faisons très brièvement parce que dans la description de notre algorithme (cf. chapitres 2 et 3) les corps finis n'interviennent presque pas. Pour étudier la dimension du hull des codes cycliques (cf. chapitre 4), nous aurons besoin de plus de détails sur les polynômes minimaux et les classes cyclotomiques, nous les donnerons au début du chapitre 4, cf. section 4.2.4. Pour la description plus détaillée des corps finis voir [LN83] ou un des ouvrages sur les codes correcteurs d'erreurs, par exemple, [MS77], [PW72] ou [PW95].

Théorème 1.2.1 *Soit F un corps fini de cardinal $q > 1$. Alors*

1. *$q = p^m$, où p est un nombre premier et m un entier positif non nul.*
2. *F est de caractéristique p.*
3. *F est unique à isomorphisme près.*

Notation 1.2.2 Nous notons $\mathbf{F}_q$ le corps fini de cardinal $q = p^m$.

Théorème 1.2.3 $\mathbf{F}_q^* \stackrel{\text{déf}}{=} \mathbf{F}_q \setminus \{0\}$ *est un groupe multiplicatif cyclique.*

Définition 1.2.4 *Un générateur de $\mathbf{F}_q^*$ est appelé* élément primitif *de $\mathbf{F}_q$.*

Théorème 1.2.5 *Soient m et m' deux entiers positifs non nuls. $\mathbf{F}_{q^{m'}}$ est isomorphe à un sous-corps de $\mathbf{F}_{q^m}$ si et seulement si $m' \mid m$. Si c'est le cas, $\mathbf{F}_{q^{m'}}$ est isomorphe à $\{\beta \in \mathbf{F}_{q^m} \mid \beta^{q^{m'}} = \beta\}$.*

1.2.1 Automorphismes d'un corps fini

Quand nous définirons le groupe d'automorphismes d'un code (cf. section 1.6.2), nous mentionnerons les automorphismes d'un corps fini (cf. remarque 1.6.19). Nous définissons brièvement cette notion.

Soit $q = p^m$, où p est un nombre premier et m un entier positif non nul.

Définition 1.2.6 *Un* automorphisme *de $\mathbf{F}_q$ est un isomorphisme de corps de $\mathbf{F}_q$ dans $\mathbf{F}_q$.*

Théorème 1.2.7 *Pour tout automorphisme η de $\mathbf{F}_q$, il existe $i \in \mathbb{N}$ tel que $\eta(\alpha) = \alpha^{p^i}$ pour tout $\alpha \in \mathbf{F}_q$.*

Théorème 1.2.8 *L'ensemble des automorphismes de $\mathbf{F}_q$ est un groupe cyclique d'ordre m engendré par $\psi : \alpha \mapsto \alpha^p$.*

1.3 Codes correcteurs d'erreurs — définitions et propriétés

Nous rappelons dans cette section les principales notions de la théorie des codes. Tous les résultats exposés dans cette section sont extraits de [MS77].

1.3.1 Définitions générales

Soit $n \in \mathbb{N}^*$, et q une puissance d'un nombre premier p.

Définition 1.3.1 *Soient $\mathbf{F}_q^n$ l'espace vectoriel de dimension n sur $\mathbf{F}_q$ (l'ensemble des mots de longueur n sur $\mathbf{F}_q$), $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbf{F}_q^n$ et $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbf{F}_q^n$.*

- *La* distance de Hamming *entre $\mathbf{x}$ et $\mathbf{y}$ est*

$$d_H(\mathbf{x}, \mathbf{y}) = \left|\{i \mid 0 \leq i \leq n-1,\ x_i \neq y_i\}\right|.$$

- *Le* poids de Hamming $w_H(\mathbf{x})$ *de $\mathbf{x}$ est le nombre de ses coordonnées non nulles*

$$w_H(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0}),$$

  où $\mathbf{0}$ est le mot de $\mathbf{F}_q^n$ ayant toutes ses coordonnées égales à 0.

- *Un* code *C sur $\mathbf{F}_q$ est une partie de $\mathbf{F}_q^n$, n est appelé* longueur *du code, les éléments de C sont appelés* mots de code.

Définition 1.3.2 *On appelle* distance minimale *d'un code C sur $\mathbf{F}_q$, l'entier*

$$d = min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

On appelle poids minimal *d'un code C, l'entier*

$$min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\}.$$

Définition 1.3.3 *Un code C est dit* linéaire *sur $\mathbf{F}_q$, si C est un sous-espace vectoriel de $\mathbf{F}_q^n$. La dimension de C sur $\mathbf{F}_q$ est appelée la* dimension *du code C et est noté* $\dim C$.

Notation 1.3.4 Un code linéaire de longueur n, dimension k, et distance minimale d sera noté un code $[n, k, d]$ (ou $[n, k]$ si d n'est pas connue ou elle n'est pas importante).

Remarque 1.3.5 Pour un code linéaire le poids minimal est égal à la distance minimale.

Définition 1.3.6 *Une* matrice génératrice *G d'un code linéaire C est une matrice $k \times n$ dont les lignes forment une base de C.*

Définition 1.3.7 *Soit C un code de longueur n et soit A_i le nombre de mots de code de C de poids i. On appelle* polynôme énumérateur des poids *de C, le polynôme*

$$\mathcal{W}_C(X) = \sum_{i=0}^{n} A_i X^i.$$

Nous le noterons aussi $\mathcal{W}(C)$.

Définition 1.3.8 *Soit C un code de longueur n sur $\mathbf{F}_q$. Le* code dual *de C, noté $C^\perp$, est l'orthogonal de C pour le produit scalaire usuel $\mathbf{x} \cdot \mathbf{y} = \sum_{i=0}^{n-1} x_i y_i$, i.e. l'ensemble*

$$C^\perp = \{\mathbf{x} \in \mathbf{F}_q^n \mid \forall\, \mathbf{y} \in C,\ \mathbf{x} \cdot \mathbf{y} = 0\}.$$

Notez qu'habituellement on ne définit le code dual que pour les codes linéaires. Nous le faisons aussi pour les codes non linéaires. Cela nous permettra d'introduire un invariant pour les codes non linéaires (cf. section 3.5).

Propriété 1.3.9 *Soit C un code de longueur n sur $\mathbf{F}_q$. Soit $\bar{C}[n,k]$ le plus petit code linéaire contenant C. Alors $C^\perp$ est égal à $\bar{C}^\perp$ et $\left(C^\perp\right)^\perp = \bar{C}$. Le code $C^\perp$ est un code linéaire $[n, n-k]$.*

Évidemment, si C est linéaire, $C = \bar{C}$.

Définition 1.3.10 *Le* hull *d'un code C est défini comme son intersection avec son dual. Nous noterons $\mathcal{H}(C) = C \cap C^\perp$.*

1.3.2 Code poinçonné

Soit Ω un ensemble de cardinal n. Nous indexons les coordonnées des mots de $\mathbf{F}_q^n$ par cet ensemble, par exemple, $\mathbf{x} = (x_\gamma)_{\gamma\in\Omega}$.

Définition 1.3.11 *Soit C un code de longueur n sur $\mathbf{F}_q$, et soit J un sous-ensemble de Ω. Le code C* poinçonné en J, *noté C_J, est composé de tous les mots de code de C avec les coordonnées indexées par J remplacées par zéros, c'est-à-dire*

$$C_J = \{(x'_\gamma)_{\gamma\in\Omega} \mid (x_\gamma)_{\gamma\in\Omega} \in C \text{ où } x'_\gamma = x_\gamma \text{ si } \gamma \notin J \text{ et } x'_\gamma = 0 \text{ sinon}\}.$$

Nous notons $C_\gamma \stackrel{\text{déf}}{=} C_{\{\gamma\}}$.

Notez que cette définition est différente de la définition usuelle des codes poinçonnés ("punctured code", cf. [MS77, Ch.1§9]). Dans notre cas C_J reste dans le même espace $\mathbf{F}_q^n$ que C.

1.3.3 Concaténation

Notation 1.3.12 – Soient $\mathbf{x}^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)})$ et $\mathbf{x}^{(2)} = (x_1^{(2)}, \ldots, x_n^{(2)})$ deux mots de $\mathbf{F}_q^n$. Nous notons $(\mathbf{x}^{(1)}|\mathbf{x}^{(2)})$ la concaténation de $\mathbf{x}^{(1)}$ et $\mathbf{x}^{(2)}$, i.e.

$$(\mathbf{x}^{(1)}|\mathbf{x}^{(2)}) = (x_1^{(1)}, \ldots, x_n^{(1)}, x_1^{(2)}, \ldots, x_n^{(2)}).$$

De la même façon il est possible de définir $(\mathbf{x}^{(1)}|\mathbf{x}^{(2)}|\cdots|\mathbf{x}^{(r)})$ pour un entier positif non nul r.

– Si C est un code de longueur n sur $\mathbf{F}_q$, et si $\alpha_1, \ldots, \alpha_r \in \mathbf{F}_q$, $r \geq 1$, alors nous notons $(\alpha_1 C|\alpha_2 C|\cdots|\alpha_r C)$ le code suivant :

$$(\alpha_1 C|\alpha_2 C|\cdots|\alpha_r C) = \{(\alpha_1 \mathbf{c}|\cdots|\alpha_r \mathbf{c}) \mid \mathbf{c} \in C\}.$$

Remarque 1.3.13 Si C est un code linéaire, $(\alpha_1 C|\cdots|\alpha_r C)$ l'est aussi. Si G est une matrice génératrice de C, $(\alpha_1 G|\cdots|\alpha_r G)$ est une matrice génératrice de $(\alpha_1 C|\alpha_2 C|\cdots|\alpha_r C)$.

Définition 1.3.14 *Un code C est appelé* dégénéré *s'il est obtenu par répétition d'un code de plus petite longueur, c'est-à-dire s'il existe un code C' tel que $C = (C'|\cdots|C')$.*

1.3.4 Codes cycliques

Nous ne donnons ici qu'une définition des codes cycliques. Voir section 4.3 pour leur description plus complète.

Soit $n \in \mathbb{N}^*$, et q une puissance d'un nombre premier p, et soit $\mathbf{F}_q$ le corps fini à q éléments.

Définition 1.3.15 *Un code linéaire C de longueur n sur $\mathbf{F}_q$ est* cyclique *si pour tout mot de code $\mathbf{c} = (c_0, c_1, \ldots, c_{n-2}, c_{n-1})$ de C, $(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$ appartient aussi à C.*

L'espace $\mathbf{F}_q^n$ peut être identifié à l'algèbre quotient de polynômes $R = \mathbf{F}_q[X]/(X^n - 1)$ comme suit : à chaque vecteur $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ de $\mathbf{F}_q^n$ correspond $c(X) = \sum_{i=0}^{n-1} c_i X^i$.

Théorème 1.3.16 *Un code linéaire de longueur n sur $\mathbf{F}_q$ est cyclique si et seulement si c'est un idéal de R.*

Théorème 1.3.17 *Soit C un code cyclique de longueur n sur $\mathbf{F}_q$. Alors :*

(i) Il existe un unique polynôme unitaire $g(X)$ de degré minimal dans C.

(ii) C est l'idéal engendré par $g(X)$, c'est-à-dire

$$C = \{g(X)h(X) \mid h(X) \in R\}$$

(la multiplication se fait dans R).

(iii) La dimension de C est $n-r$ où $r = \deg g(X)$.

(iv) $g(X)$ est un diviseur de $X^n - 1$ (dans $\mathbf{F}_q[X]$).

(v) Tout $c(X) \in C$ s'écrit de façon unique $c(X) = g(X)f(X)$ dans $\mathbf{F}_q[X]$.

Définition 1.3.18 *Un* polynôme générateur *d'un code cyclique C est un générateur de l'idéal C. Par abus de langage nous appellerons* le polynôme générateur *le polynôme générateur unitaire de plus petit degré.*

Nous notons $C =< g >$ le code cyclique C engendré par un polynôme générateur g.

1.4 Groupes de permutations

Dans cette section nous rappelons les définitions et les notations usuelles de la théorie des groupes de permutations (cf., par exemple, [Hal59, BR74]).

Soit Ω un ensemble fini.

1.4.1 Permutations

Définition 1.4.1 *Une* permutation *g de Ω est une bijection de Ω sur Ω. Nous noterons α^g l'image de $\alpha \in \Omega$ sous l'action de la permutation g.*

Si $\alpha^g = \alpha$, nous disons que g fixe *α (α est* fixé *par g). Nous notons fix (g) l'ensemble de points de Ω fixés par g.*

Le produit *gh des permutations g et h est défini par $\alpha^{gh} = (\alpha^g)^h$ pour tout $\alpha \in \Omega$. Muni de ce produit, l'ensemble $Sym(\Omega)$ de toutes les permutations de Ω est un groupe appelé le* groupe symétrique *de Ω. Nous noterons id l'élément neutre (l'identité) de ce groupe, et g^{-1} l'inverse de g (i.e. g^{-1} est un élément de $Sym(\Omega)$ tel que $gg^{-1} = id$).*

Propriété 1.4.2 *Si $g, h \in Sym(\Omega)$, alors $(gh)^{-1} = h^{-1}g^{-1}$.*

Définition 1.4.3 *Une permutation g de Ω est un* cycle *s'il existe $\gamma_1, \gamma_2, \ldots, \gamma_l \in \Omega$ tels que $g(\gamma_i) = \gamma_{i+1}$ pour tout i, $1 \leq i \leq l-1$, $g(\gamma_l) = \gamma_1$ et $g(\gamma) = \gamma$ pour $\gamma \in \Omega \setminus \{\gamma_1, \ldots, \gamma_l\}$.*

Deux cycles g_1 et g_2 sont appelés disjoints *si fix $(g_1) \cup$ fix $(g_2) = \Omega$, c'est-à-dire tout point de Ω est fixé par au moins un cycle g_1 ou g_2.*

Lemme 1.4.4 *Toute permutation g de Ω se décompose en produit (commutatif) $g_1 \cdots g_r$ de cycles disjoints. Cette décomposition induit une partition de Ω de taille r dont les cellules sont $\Omega \setminus$ fix (g_i) pour tout g_i, $1 \leq i \leq r$ (cf. section 1.5.1 pour la définition des partitions).*

1.4.2 Groupes de permutations

Définition 1.4.5 *Soit G un sous-groupe de $Sym(\Omega)$. On l'appelle un* groupe de permutations *de Ω.*

Le cardinal $|\Omega|$ de Ω est le degré *de G.*

*Le cardinal $|G|$ de G est l'*ordre *de G.*

Nous notons fix (G) l'ensemble de points de Ω fixés par tous les éléments de G, i.e.

$$\mathit{fix}\,(G) = \{\alpha \in \Omega \mid \forall g \in G,\ \alpha^g = \alpha\}.$$

Définition 1.4.6 *Nous dirons que G est* engendré *par un sous-ensemble de permutations $X \subset G$ et nous noterons $G =< X >$ s'il est possible d'exprimer chaque permutation g de G comme un produit d'éléments de $X \cup X^{-1}$ où X^{-1} est l'ensemble des inverses des éléments de X, c'est-à-dire si pour tout $g \in G$, il existe un entier $m \geq 1$ et des permutations $s_1, s_2, \ldots, s_m \in X \cup X^{-1}$ tels que $g = s_1 s_2 \cdots s_m$. Alors X est appelé un* ensemble de générateurs *de G.*

Si $X = \{g_1, \ldots, g_l\}$, nous noterons $G =< g_1, \ldots, g_l >$. Si Y est un ensemble de permutations ou un groupe de permutations, $< Y, g >$ signifie $< Y \cup \{g\} >$. Nous conviendrons que l'identité n'appartient pas à l'ensemble de générateurs. Donc si le groupe G n'a qu'un élément — l'identité, — nous considérerons qu'il est engendré par l'ensemble vide $\emptyset$, et nous noterons $< \emptyset >=<>= \{id\}$.

1.4.3 Notations et propriétés

Si g est une permutation, un ensemble de permutations ou un groupe de permutations, x^g désignera l'action de g sur x. Plus précisément :

Notation 1.4.7 Soit G un groupe de permutations de Ω.

- Si H est un sous-ensemble de G et Γ est un sous-ensemble de Ω, alors

  $$\Gamma^H \overset{\text{déf}}{=} \{\alpha^h \mid h \in H, \alpha \in \Gamma\}.$$

  En particulier, nous noterons $\Gamma^h \overset{\text{déf}}{=} \Gamma^{\{h\}}$ et $\alpha^H \overset{\text{déf}}{=} \{\alpha\}^H$.

- Si $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ est un s-vecteur d'éléments de Ω (autrement dit, une suite d'éléments de Ω de taille s), $g \in G$, H est un sous-ensemble de G, alors

  $$\Gamma^g \overset{\text{déf}}{=} ({\alpha_1}^g, {\alpha_2}^g, \ldots, {\alpha_s}^g) \quad \text{et} \quad \Gamma^H \overset{\text{déf}}{=} \{\Gamma^h \mid h \in H\}.$$

- Il est possible de définir l'action d'une permutation sur les permutations comme suit : si g et h sont deux permutations, h^g désigne $g^{-1}hg$. Si H est un sous-ensemble de G,

  $$H^g \overset{\text{déf}}{=} \{h^g \mid h \in H\}.$$

Nous utiliserons par la suite les notations suivantes.

Notation 1.4.8 – Si g est une permutation d'un ensemble fini Ω et $A = (\alpha_1, \ldots, \alpha_s)$ est une suite d'éléments de Ω, $\mathcal{L}(A, g)$ ou, plus simplement, $\mathcal{L}(g)$ représentera le plus petit entier i, $1 \leq i \leq s$, tel que ${\alpha_i}^g \neq \alpha_i$, ou $s + 1$ si $A^g = A$, i.e.

$$\mathcal{L}(A, g) = \begin{cases} s+1 & \text{si } A^g = A, \\ \min\{i \mid 1 \leq i \leq s,\ {\alpha_i}^g \neq \alpha_i\} & \text{sinon.} \end{cases}$$

– Si $H_1, H_2, \ldots, H_s$ sont des sous-ensembles d'un groupe de permutations G, alors le produit de ces sous-ensembles est

$$H_1 H_2 \cdots H_s = \{h_1 h_2 \cdots h_s \mid h_1 \in H_1, h_2 \in H_2, \ldots, h_s \in H_s\}.$$

Nous utiliserons la propriété suivante.

Propriété 1.4.9 *Si $\{A_i\}$ est un ensemble de sous-ensembles de Ω, alors*

$$\textstyle\bigcup_i (A_i)^g = (\bigcup_i A_i)^g \qquad \textit{pour tout } g \in \mathit{Sym}(\Omega).$$

1.4.4 Classes à droite

Définition 1.4.10 *Soit G un groupe de permutations de Ω. Soit H un sous-groupe de G. Soit $g \in G$. Nous notons Hg l'ensemble $\{hg \mid h \in H\}$. Nous appelons cet ensemble la* classe à droite de g modulo H.

De même, nous appelons $gH \stackrel{\text{déf}}{=} \{gh \mid h \in H\}$ la classe à gauche de g modulo H.

On montre facilement que

$$\text{si } g_1, g_2 \in G, \text{ alors } Hg_1 = Hg_2 \text{ si } g_1 g_2^{-1} \in H, \text{ et } Hg_1 \cap Hg_2 = \emptyset \text{ sinon.}$$

Cela produit une relation d'équivalence dans G ($g_1 \sim g_2$ si et seulement si $g_1 g_2^{-1} \in H$), et les classes à droite Hg sont les classes de cette équivalence. De cette façon nous obtenons une partition (cf. section 1.5.1 pour la définition des partitions) de G,

$$G = Hg_1 \cup Hg_2 \cup \cdots \cup Hg_r, \tag{1.1}$$

et une formule exprimant le cardinal de H :

$$|H| = \frac{|G|}{r}, \tag{1.2}$$

où r est le nombre des classes à droite modulo H dans G.

Nous avons, en particulier,

$$\text{si } g \in H, \text{ alors } Hg = H \text{ et } gH = H. \tag{1.3}$$

Définition 1.4.11 *Si $\{g_1, \ldots, g_r\}$ vérifie (1.1), nous l'appelons un* ensemble de représentants des classes à droite modulo H dans G *ou, plus brièvement, un* transversal de H dans G.

La classe à droite de id modulo H est $H\,id = H$, il est donc toujours possible de choisir id comme le représentant de sa classe à droite.

1.4.5 Orbites

Définition 1.4.12 *Soit G un groupe de permutations de Ω. Soit $\alpha \in \Omega$. L'*orbite *α^G de α sous l'action de G (ou la G-orbite de α) est l'ensemble des images α^g de α sous l'action de g, où g parcourt G, c'est-à-dire*

$$\alpha^G = \{\alpha^g \mid g \in G\}.$$

Lemme 1.4.13 *Si $\beta \in \alpha^G$, alors $\beta^G = \alpha^G$.*

Preuve. Comme $\beta \in \alpha^G$, nous avons $\beta = \alpha^h$ pour un $h \in G$.

$$\alpha^G = \{\alpha^g \mid g \in G\} = \{\alpha^{hh^{-1}g} \mid g \in G\} = \{\beta^{h^{-1}g} \mid g \in G\} = \{\beta^f \mid f = h^{-1}g,\ g \in G\}.$$

Quand g parcourt G, $f = h^{-1}g$ parcourt G aussi (cf. (1.3)), donc

$$\alpha^G = \{\beta^f \mid f = h^{-1}g,\ g \in G\} = \{\beta^f \mid f \in G\} = \beta^G. \qquad \square$$

Donc si O est une G-orbite, alors $O = \alpha^G$ pour tout $\alpha \in O$.

Lemme 1.4.14 *Si O est une G-orbite, alors $O^g = O$ pour toute permutation g de G.*

Preuve. Soit $\alpha \in O$. Compte tenu du lemme 1.4.13, $O = \alpha^G = \{\alpha^h \mid h \in G\}$. Donc

$$O^g = \{\alpha^{hg} \mid h \in G\} = \{\alpha^f \mid f = hg,\ h \in G\}.$$

Quand h parcourt G, $f = hg$ parcourt G aussi (cf. (1.3)), donc

$$O^g = \{\alpha^f \mid f = hg,\ h \in G\} = \{\alpha^f \mid f \in G\} = \alpha^G = O. \qquad \square$$

Notation 1.4.15 [Leo91, p.545] et [Leo97, Déf.1] Si G est un groupe de permutations de Ω et $<$ est un ordre arbitraire[1] de Ω, alors $\mathbf{\Theta}(G)$ est la partition ordonnée dont les cellules sont les G-orbites, une orbite Λ_1 précédant une autre orbite Λ_2 si $\min(\Lambda_1) < \min(\Lambda_2)$, où nous calculons min pour l'ordre $<$. Autrement dit, si $\{\Lambda_1, \Lambda_2, \ldots, \Lambda_l\}$ est l'ensemble des G-orbites, alors $\mathbf{\Theta}(G) = (\Lambda_{i_1}, \Lambda_{i_2}, \ldots, \Lambda_{i_l})$, où $i_1, \ldots, i_l$ sont tels que

$$\min(\Lambda_{i_1}) < \min(\Lambda_{i_2}) < \cdots < \min(\Lambda_{i_l}).$$

1. $<$ n'a pas de rapport avec l'ordre important $\prec$ de la notation 2.4.12. Si $\Omega = \{1, \ldots, n\}$, le choix naturel pour $<$ est $1 < 2 \cdots < n$.

1.4.6 Stabilisateurs

Définition 1.4.16 *Soit G un groupe de permutations de Ω. Si Y est un point de Ω, une suite de points de Ω, un sous-ensemble (ordonné ou non) de Ω, ou une partition (ordonnée ou non) de Ω, on définit le* stabilisateur *G_Y de Y dans G comme l'ensemble des éléments de G qui fixent Y, c'est-à-dire*

$$G_Y = \{g \in G \mid Y^g = Y\}.$$

Si $Y = (Y_1, \ldots, Y_s)$ est une suite de points de Ω, un sous-ensemble ordonné de Ω, ou une partition ordonnée de Ω, nous pouvons écrire $G_{Y_1,\ldots,Y_s}$ au lieu de $G_{(Y_1,\ldots,Y_s)}$.

On montre facilement que G_Y est un sous-groupe de G.

Soit $\alpha \in \Omega$. Notez que si l'image de α sous une permutation $g \in G$ est β, alors l'image de α sous h est β pour tout $h \in G_\alpha g$, car $h = fg$, $f \in G_\alpha$, et $\alpha^h = \alpha^{fg} = \left(\alpha^f\right)^g = \alpha^g = \beta$. Il existe donc une bijection entre α^G et l'ensemble des classes à droite modulo G_α : tout $\beta \in \alpha^G$ correspond à la classe $G_\alpha g$ de g tel que $\alpha^g = \beta$, et, inversement, la classe $G_\alpha g$ correspond à $\beta = \alpha^g \in \alpha^G$. Si nous notons U un ensemble de représentants des classes à droite modulo G_α dans G, nous avons de la même façon une bijection entre α^G et U, et donc $\left|\alpha^G\right| = |U|$. De plus, à partir de l'égalité (1.2) nous obtenons que

$$|G_\alpha| = \frac{|G|}{|\alpha^G|},$$

et, par induction, si $s \geq 1$ et $(\alpha_1, \ldots, \alpha_s)$ est un sous-ensemble ordonné de Ω,

$$|G_{\alpha_1,\alpha_2}| = \frac{|G_{\alpha_1}|}{|{\alpha_2}^{G_{\alpha_1}}|} = \frac{|G|}{|{\alpha_1}^G|\,|{\alpha_2}^{G_{\alpha_1}}|}, \text{ etc.}\ldots,$$

$$|G_{\alpha_1,\ldots,\alpha_s}| = \frac{|G|}{|{\alpha_1}^G|\,|{\alpha_2}^{G_{\alpha_1}}|\,|{\alpha_3}^{G_{\alpha_1,\alpha_2}}|\cdots\left|{\alpha_s}^{G_{\alpha_1,\ldots,\alpha_{s-1}}}\right|}. \tag{1.4}$$

1.4.7 Ordre sur les permutations

Supposons que Ω soit ordonné suivant un ordre total $\prec$. Cet ordre induit un ordre lexicographique (noté aussi $\prec$) sur les s-vecteurs $(\gamma_1, \ldots, \gamma_s)$ d'éléments de Ω, c'est-à-dire

$$(\gamma_1, \ldots, \gamma_s) \prec (\delta_1, \ldots, \delta_s) \quad \text{si et seulement si}$$

$$\text{il existe } i,\ 1 \leq i \leq s, \text{ tel que} \quad (\gamma_1, \ldots, \gamma_{i-1}) = (\delta_1, \ldots, \delta_{i-1}) \quad \text{et} \quad \gamma_i \prec \delta_i.$$

Et cela induit un ordre sur les permutations de Ω : si

$$\Omega = \{\omega_1, \ldots, \omega_n\}, \quad \omega_1 \prec \omega_2 \prec \cdots \prec \omega_n,$$

$$g_1,\, g_2 \in Sym(\Omega), \quad (\gamma_1, \ldots, \gamma_n) \stackrel{\text{déf}}{=} (\omega_1, \ldots, \omega_n)^{g_1}, \quad (\delta_1, \ldots, \delta_n) \stackrel{\text{déf}}{=} (\omega_1, \ldots, \omega_n)^{g_2},$$

alors

$$g_1 \prec g_2 \quad \text{si et seulement si} \quad (\gamma_1, \ldots, \gamma_n) \prec (\delta_1, \ldots, \delta_n).$$

1.5 Partitions

Les définitions de cette section sont en majeure partie prises dans [McK78, Leo91, Leo97].

Soit Ω un ensemble fini, $|\Omega| = n$.

1.5.1 Définitions et notations

Nous rappelons la définition d'une partition.

Définition 1.5.1 *Une* partition $\mathbf{\Pi}$ *de* Ω *est un ensemble de sous-ensembles non vides de* Ω*, disjoints deux à deux et dont la réunion est* Ω*, i.e.* $\mathbf{\Pi} = \{\Pi_1, \ldots, \Pi_l\}$ *est une partition de* Ω *si*

1. $\emptyset \subsetneq \Pi_i \subseteq \Omega \quad \forall\, i,\ 1 \leq i \leq l,$
2. $\Pi_i \cap \Pi_j = \emptyset \quad \forall\, i,\ i \neq j,$
3. $\bigcup_{i=1}^{l} = \Omega.$

Nous donnons plusieurs définitions liées aux partitions.

Définition 1.5.2 – *Une* partition ordonnée *est une suite* $(\Pi_1, \ldots, \Pi_l)$ *telle que l'ensemble* $\{\Pi_1, \ldots, \Pi_l\}$ *est une partition. Nous appellerons* $\{\Pi_1, \ldots, \Pi_l\}$ partition non ordonnée*.*

- *Les éléments* Π_i *d'une partition (ordonnée ou non)* $\mathbf{\Pi}$ *sont appelés* cellules *de* $\mathbf{\Pi}$*.*
- *Le nombre des cellules de* $\mathbf{\Pi}$ *est appelé* taille *de* $\mathbf{\Pi}$*. Nous la notons* $|\mathbf{\Pi}|$*.*
- *Nous dirons que* $\mathbf{\Pi}$ *est* discrète *si* $|\mathbf{\Pi}| = |\Omega|$*.*
- *Nous dirons que* $\mathbf{\Pi}$ *est* triviale *si elle ne possède qu'une cellule.*

Soit $\mathbf{\Pi}$ une partition (ordonnée ou non). Il est évident que $\mathbf{\Pi}$ est discrète si et seulement si chaque cellule de $\mathbf{\Pi}$ contient exactement un élément de Ω. De même, il est évident que $\mathbf{\Pi}$ est triviale si et seulement si la première cellule Π_1 de $\mathbf{\Pi}$ est égale à Ω. Nous avons donc alors $\mathbf{\Pi} = \{\Omega\}$ (partition non ordonnée) ou $\mathbf{\Pi} = (\Omega)$ (partition ordonnée).

Notation 1.5.3 Soit g une permutation de Ω, $\mathbf{\Pi}$ et $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_m)$ deux partitions ordonnées de Ω, $(\mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_r)$ une suite de partitions ordonnées de Ω, $\mathbf{\Phi} = \{\Phi_1, \ldots, \Phi_l\}$ une partition non ordonnée de Ω. Nous notons :

$PartOrd(\Omega)$	=	l'ensemble de toutes les partitions ordonnées de Ω.
Π_i	=	la i-ième cellule de $\mathbf{\Pi}$ si $1 \leq i \leq \|\mathbf{\Pi}\|$, $\emptyset$ si $i > \|\mathbf{\Pi}\|$.
$\Psi_{j,i}$	=	la i-ième cellule de $\mathbf{\Psi}_j$ si $1 \leq i \leq \|\mathbf{\Psi}_j\|$, $\emptyset$ si $i > \|\mathbf{\Psi}_j\|$, où $1 \leq j \leq r$.
Π_∞	=	la dernière cellule de $\mathbf{\Pi}$, i.e. $\Pi_{\|\mathbf{\Pi}\|}$ (cette notation est utile quand on n'a pas de symbole explicite pour $\|\mathbf{\Pi}\|$ et l'expression de $\mathbf{\Pi}$ est compliquée).
$\\|\mathbf{\Sigma}\\|$	=	$(\|\Sigma_1\|, \ldots, \|\Sigma_m\|)$. Notez que $\\|\mathbf{\Sigma}\\|$ est une suite d'entiers.
$\mathit{fix}\,(\mathbf{\Sigma})$	=	l'ensemble ordonné $(\gamma_1, \ldots, \gamma_t)$, où $\{\gamma_1\}, \ldots, \{\gamma_t\}$ sont toutes les cellules de $\mathbf{\Sigma}$ contenant exactement un élément, prises dans l'ordre d'apparition dans $\mathbf{\Sigma}$. Plus précisément, si $i_1 < \cdots < i_t$ sont des entiers tels que $\left\|\Sigma_{i_j}\right\| = 1$ (soit $\Sigma_{i_j} = \{\gamma_j\}$) $\forall j$, $1 \leq j \leq t$, et $\|\Sigma_j\| > 1$ $\forall j$, $1 \leq j \leq m$, $j \notin \{i_1, \ldots, i_t\}$, alors $\mathit{fix}\,(\mathbf{\Sigma}) = (\gamma_1, \ldots, \gamma_t)$.
$\mathbf{\Phi}^g$	=	$\{{\Phi_1}^g, \ldots, {\Phi_l}^g\}$.
$\mathbf{\Sigma}^g$	=	$({\Sigma_1}^g, \ldots, {\Sigma_m}^g)$.

Nous utiliserons la propriété suivante :

Propriété 1.5.4 $(\mathbf{\Pi}^g)_j = (\Pi_j)^g$ *pour tous les* $\mathbf{\Pi} \in PartOrd(\Omega)$, $j \in I_n$, $g \in Sym(\Omega)$.

Nous écrivons les partitions constantes en utilisant la notation simplifiée, présentée sur un exemple : si $\Omega = \{1, \ldots, 10\}$ et $\mathbf{\Pi} = (\Pi_1, \Pi_2, \Pi_3)$, où $\Pi_1 = \{5, 8, 10\}$, $\Pi_2 = \{1, 7\}$ et $\Pi_3 = \{2, 3, 4, 6, 9\}$, alors nous écrirons $\mathbf{\Pi} = (5, 8, 10|1, 7|2, 3, 4, 6, 9)$. Pour les partitions non ordonnées, nous remplaçons les parenthèses par les accolades, par exemple, $\{5, 8, 10|1, 7|2, 3, 4, 6, 9\}$.

1.5.2 Ordres sur les partitions

Nous aurons besoin de comparer les partitions.

Définition 1.5.5 *[Leo97, Déf.2] et [Leo91, p.536] Si* $\mathbf{\Pi}$ *et* $\mathbf{\Sigma}$ *sont des partitions ordonnées,* $\mathbf{\Pi} \leq \mathbf{\Sigma}$ *signifie que :*

1. $|\mathbf{\Pi}| \geq |\mathbf{\Sigma}|$,
2. $\Pi_i \subseteq \Sigma_i$ *pour* $1 \leq i \leq |\mathbf{\Sigma}|$, *et*
3. Π_i, $|\mathbf{\Sigma}| < i \leq |\mathbf{\Pi}|$, *est contenu dans une cellule de* $\mathbf{\Sigma}$ *(i.e.* $\Pi_i \subseteq \Sigma_{r_i}$, *où* r_i *est un entier,* $1 \leq r_i \leq |\mathbf{\Sigma}|$*).*

Nous dirons alors que $\mathbf{\Pi}$ *est* plus fine *que* $\mathbf{\Sigma}$, *ou que* $\mathbf{\Sigma}$ *est* plus grossière *que* $\mathbf{\Pi}$.

Si $\mathbf{\Pi} \leq \mathbf{\Sigma}$ *et* $\mathbf{\Pi} \neq \mathbf{\Sigma}$, *nous écrivons* $\mathbf{\Pi} < \mathbf{\Sigma}$ *et nous disons que* $\mathbf{\Pi}$ *est* strictement plus fine *que* $\mathbf{\Sigma}$, *ou que* $\mathbf{\Sigma}$ *est* strictement plus grossière *que* $\mathbf{\Pi}$.

Raffiner *une partition signifie la rendre strictement plus fine par un procédé quelconque.*

Parfois nous aurons besoin d'un critère plus faible, ayant un sens pour les partitions ordonnées ou non.

Notation 1.5.6 [Leo97, Déf.3] et [Leo91, p.536] Si $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ sont des partitions (ordonnées ou non), nous écrivons $\mathbf{\Pi} \lesssim \mathbf{\Sigma}$ si chaque cellule de $\mathbf{\Pi}$ est contenue dans une cellule de $\mathbf{\Sigma}$. Si $\mathbf{\Pi} \lesssim \mathbf{\Sigma}$ et $\mathbf{\Sigma} \lesssim \mathbf{\Pi}$, nous écrivons $\mathbf{\Pi} \simeq \mathbf{\Sigma}$.

1.5.3 Union de partitions

Soient $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ deux partitions (non ordonnées) de Ω. Considérons $E(\mathbf{\Pi}, \mathbf{\Sigma})$ l'ensemble des partitions $\mathbf{\Phi}$ telles que $\mathbf{\Pi} \lesssim \mathbf{\Phi}$ et $\mathbf{\Sigma} \lesssim \mathbf{\Phi}$. Cet ensemble est non vide, car $\mathbf{\Pi} \lesssim \{\Omega\}$ et $\mathbf{\Sigma} \lesssim \{\Omega\}$. On peut montrer que $E(\mathbf{\Pi}, \mathbf{\Sigma})$ admet un unique élément minimal, i.e. il existe $\mathbf{\Phi} \in E(\mathbf{\Pi}, \mathbf{\Sigma})$ tel que $\mathbf{\Phi} \lesssim \mathbf{\Psi}$ pour tout $\mathbf{\Psi} \in E(\mathbf{\Pi}, \mathbf{\Sigma})$.

Définition 1.5.7 *L'*union $\mathbf{\Pi} \cup \mathbf{\Sigma}$ *de* $\mathbf{\Pi}$ *et* $\mathbf{\Sigma}$ *est l'élément minimal de* $E(\mathbf{\Pi}, \mathbf{\Sigma})$.

Par exemple, soient G_1 et G_2 des groupes de permutations, soit $\mathbf{\Pi}_i$, $i \in \{1, 2\}$, une partition composée des G_i-orbites. Alors $\mathbf{\Pi}_1 \cup \mathbf{\Pi}_2$ est une partition composée des $< G_1 \cup G_2 >$-orbites.

1.5.4 Une notation importante

Le moyen le plus simple pour raffiner une partition donnée $\mathbf{\Pi}$ est de diviser une cellule Π_i en deux parties et d'en mettre une à la fin de la partition. On peut exprimer cela en précisant i et un sous-ensemble Γ de Ω tels que Π_i est divisé en $\Pi_i \setminus \Gamma$ et $\Pi_i \cap \Gamma$; on met la dernière partie à la fin. Plus précisément :

Notation 1.5.8 [Leo91, Déf.15] et [Leo97, Déf.6] Si $\mathbf{\Pi} = (\Pi_1, \ldots, \Pi_k)$ est une partition ordonnée de Ω, $i \in I_n$, et Γ est un sous-ensemble de Ω, alors $\mathcal{F}_{i,\Gamma}(\mathbf{\Pi})$ est défini par

$$\mathcal{F}_{i,\Gamma}(\mathbf{\Pi}) = \begin{cases} (\Pi_1, \ldots, \Pi_{i-1}, \Pi_i \setminus \Gamma, \Pi_{i+1}, \ldots, \Pi_k, \Pi_i \cap \Gamma) & \text{si } 1 \leq i \leq k \text{ et } \emptyset \subsetneq \Pi_i \cap \Gamma \subsetneq \Pi_i, \\ \mathbf{\Pi} & \text{sinon.} \end{cases}$$

Si $|\Gamma| = 1$, par exemple $\Gamma = \{\gamma\}$, nous écrirons $\mathcal{F}_{i,\gamma}$ au lieu de $\mathcal{F}_{i,\{\gamma\}}$. □

Nous voyons facilement que $\mathcal{F}_{i,\Gamma}$ satisfait les propriétés suivantes :

Propriété 1.5.9 *Pour tous les* $\mathbf{\Pi} \in PartOrd(\Omega)$, $\Gamma \subset \Omega$ *et* $i \in I_n$, *nous avons :*

(a) $\mathcal{F}_{i,\Gamma}(\mathbf{\Pi}) \leq \mathbf{\Pi}$,

(b) $|\mathcal{F}_{i,\Gamma}(\mathbf{\Pi})| \leq |\mathbf{\Pi}| + 1$,

(c) si $\mathcal{F}_{i,\Gamma}(\mathbf{\Pi}) < \mathbf{\Pi}$, *alors* $\mathcal{F}_{i,\Gamma}(\mathbf{\Pi})_\infty \subset \Pi_i$.

$\mathcal{F}_{i,\Gamma}$ ne rend pas seulement les partitions plus fines, mais aussi, sous certaines conditions, préserve les images des partitions. Plus précisément :

Propriété 1.5.10 *[Leo97, p.129]*

$$\textit{Si } g \in Sym(\Omega),\ \mathbf{\Pi}^g = \mathbf{\Sigma} \textit{ et } \Gamma^g = \Delta,\ \textit{alors } \mathcal{F}_{i,\Gamma}(\mathbf{\Pi})^g = \mathcal{F}_{i,\Delta}(\mathbf{\Sigma}).$$

Autrement dit,

$$\mathcal{F}_{i,\Gamma^g}(\mathbf{\Pi}^g) = \mathcal{F}_{i,\Gamma}(\mathbf{\Pi})^g \textit{ pour tous les } \mathbf{\Pi} \in PartOrd(\Omega), \Gamma \subseteq \Omega, i \in I_n \textit{ et } g \in Sym(\Omega).$$

C'est une propriété critique de $\mathcal{F}_{i,\Gamma}$ qui permet de l'utiliser pour construire les raffinements élémentaires (cf. définition 2.4.1 et proposition 3.4.1).

1.6 Groupes de permutations et d'automorphismes des codes

Dans cette section nous définissons les groupes de permutations et d'automorphismes d'un code. Ces groupes, ainsi définis, ont la propriété de préserver la distance de Hamming : ce sont des groupes d'isométries.

1.6.1 Définition du groupe de permutations d'un code

Soient n un entier positif non nul et q une puissance d'un nombre premier. Soit Ω un ensemble ordonné de cardinal n utilisé pour indexer les coordonnées des mots de $\mathbf{F}_q^n$ (par exemple, $\Omega = \{1, \ldots, n\}$).

Une permutation g de Ω agit sur les mots de $\mathbf{F}_q^n$ comme suit : si $\mathbf{c} = (c_i)_{i\in\Omega}$ est un mot de $\mathbf{F}_q^n$, alors l'image de $\mathbf{c}$ par g est

$$\mathbf{c}^g = \left(c_{i^{g^{-1}}}\right)_{i\in\Omega}. \tag{1.5}$$

Nous voyons facilement que

$$\text{si} \quad g, h \in Sym(\Omega), \quad \text{alors} \quad \mathbf{c}^{gh} = (\mathbf{c}^g)^h.$$

En effet, $(\mathbf{c}^g)^h = \left(\left(c_{i^{g^{-1}}}\right)_{i\in\Omega}\right)^h$. Notons $d_i = c_{i^{g^{-1}}}$, $i \in \Omega$. Alors

$$\begin{aligned}(\mathbf{c}^g)^h &= \left(\left(c_{i^{g^{-1}}}\right)_{i\in\Omega}\right)^h = \left((d_i)_{i\in\Omega}\right)^h = \left(d_{i^{h^{-1}}}\right)_{i\in\Omega} = \left(c_{\left(i^{h^{-1}}\right)^{g^{-1}}}\right)_{i\in\Omega} \\ &= \left(c_{i^{h^{-1}g^{-1}}}\right)_{i\in\Omega} = \left(c_{i^{(gh)^{-1}}}\right)_{i\in\Omega} = \mathbf{c}^{gh}.\end{aligned}$$

Soit C un code de longueur n sur $\mathbf{F}_q$. La permutation g agit sur C comme suit :

$$C^g = \{\mathbf{c}^g \mid \mathbf{c} \in C\}.$$

Comme ci-dessus, nous obtenons que

$$C^{gh} = (C^g)^h,$$

car

$$(C^g)^h = (\{\mathbf{c}^g \mid \mathbf{c} \in C\})^h = \{(\mathbf{c}^g)^h \mid \mathbf{c} \in C\} = \{\mathbf{c}^{gh} \mid \mathbf{c} \in C\} = C^{gh}.$$

Définition 1.6.1 *Soient C et C' deux codes de longueur n sur $\mathbf{F}_q$. Nous dirons qu'ils sont* équivalents par permutation *s'il existe une permutation g de Ω telle que $C' = C^g$.*

Définition 1.6.2 *Soit C un code de longueur n sur $\mathbf{F}_q$. Le* groupe de permutations *$Perm(C)$ de C est le sous-ensemble de tous les éléments g de $Sym(\Omega)$ tels que $C^g = C$, i.e.*

$$Perm(C) = \{g \in Sym(\Omega) \mid C^g = C\}.$$

Proposition 1.6.3 *L'ensemble $Perm(C)$, muni du produit usuel des permutations (cf. définition 1.4.1, p. 22), est un groupe.*

Preuve. D'après la définition d'un sous-groupe, il suffit de montrer que $uv \in Perm(C)$ et $u^{-1} \in Perm(C)$ si $u, v \in Perm(C)$.

Comme $C^{uv} = (C^u)^v = C^v = C$, $uv \in Perm(C)$.

Si $C^u = C$, alors $(C^u)^{u^{-1}} = C^{u^{-1}}$. Mais $(C^u)^{u^{-1}} = C^{uu^{-1}} = C$, donc $C^{u^{-1}} = C$ et $u^{-1} \in Perm(C)$. □

On démontre facilement les propriétés suivantes :

Propriété 1.6.4 *Un code linéaire et son dual ont le même groupe de permutations.*

Propriété 1.6.5 *Deux codes équivalents par permutation ont le même énumérateur des poids.*

Propriété 1.6.6 *Pour tout code C, tout sous-ensemble J de Ω et toute permutation g de Ω nous avons $(C^g)_{J^g} = (C_J)^g$, où C_J est le code C poinçonné en J (cf. définition 1.3.11).*

1.6.2 Définition du groupe d'automorphismes d'un code

Dans cette section nous définissons les permutations monomiales, donnons leurs propriétés, et définissons le groupe d'automorphismes d'un code C.

1.6.2.1 Permutations monomiales

Définition 1.6.7 *Une* matrice de permutation *est une matrice $n \times n$ inversible à coefficients dans $\{0, 1\} \subset \mathbf{F}_q$ ayant un et un seul élément non nul par ligne et par colonne.*

Si $\mathbf{c} \in \mathbf{F}_q^n$ et P est une matrice de permutation, alors le produit $\mathbf{c}P$ donne un mot de $\mathbf{F}_q^n$ qui est égal en fait à $\mathbf{c}$ avec des coordonnées permutées. C'est la raison pour laquelle on appelle les matrices de cette sorte matrices de permutation.

Définition 1.6.8 *Une* matrice monomiale *est une matrice $n \times n$ inversible à coefficients dans $\mathbf{F}_q$ ayant un et un seul élément non nul par ligne et par colonne.*

Si $q = 2$, toutes les matrices monomiales sont des matrices de permutation. C'est plus intéressant dans le cas $q \neq 2$. On montre facilement qu'une matrice monomiale M est le produit d'une matrice diagonale inversible D et une matrice de permutation P, où P est obtenue à partir de M en remplaçant les scalaires non nuls par des 1, et le i-ième terme sur la diagonale de D est égal au terme non nul de la i-ième ligne de M. Cela se voit tout de suite si on remarque que si $D = (d_{ij}), P = (p_{ij})$, alors $DP = (\sum_{k=1}^{n} d_{ik}p_{kj}) = (d_{ii}p_{ij})$, car $d_{ij} = 0$ si $i \neq j$.

Soit $\mathbf{c} \in \mathbf{F}_q^n$. Multiplier $\mathbf{c}$ par M (à droite) revient à multiplier $\mathbf{c}$ par D et par P, i.e. multiplier d'abord chaque composante c_i de $\mathbf{c}$ par un élément de $\mathbf{F}_q^*$ (plus précisément, par le i-ième terme d_{ii} sur la diagonale de D), et permuter les composantes du mot de $\mathbf{F}_q^n$ résultant par la permutation correspondant à P.

Il n'est pas difficile de montrer que le produit de deux matrices monomiales est aussi une matrice monomiale, et que l'ensemble des matrices monomiales muni du produit de matrices forme un groupe appelé *groupe monomial*.

Nous pouvons alors définir le groupe d'automorphismes d'un code C comme étant le sous-groupe de tous les éléments M du groupe monomial laissant C globalement invariant, i.e. tels que $CM = C$, où M agit sur C comme suit : $CM = \{\mathbf{c}M \mid \mathbf{c} \in C\}$.

Mais traiter les matrices est plus difficile que les permutations, c'est pourquoi nous définirons le groupe d'automorphismes de C de façon différente, comme le sous-groupe d'un certain groupe de permutations, même si au fond c'est la même chose exprimée en langage mathématique de deux manières différentes.

Comment décrire l'action de M sur $\mathbf{c} \in \mathbf{F}_q^n$ comme l'action d'une permutation ? L'idée est la suivante. D'abord nous définirons les permutations monomiales, nous montrerons qu'elles forment un groupe, nous les "décomposerons" d'une certaine manière (proposition 1.6.11), nous donnerons quelques propriétés et nous montrerons comment on peut définir leur action sur les mots de $\mathbf{F}_q^n$.

Définition 1.6.9 *Notons* $\Omega_q = \mathbf{F}_q^* \times \Omega$.

Une permutation monomiale de Ω *est une permutation* g *de* Ω_q *(qui agit donc sur les couples* (a, γ), $a \in \mathbf{F}_q^*$, $\gamma \in \Omega$*) telle que*

$$(ab, \gamma)^g = a((b, \gamma)^g) \text{ pour tous les } a, b \in \mathbf{F}_q^* \text{ et } \gamma \in \Omega,$$

où le produit $c(d, \alpha)$ *est défini par*

$$c(d, \alpha) = (cd, \alpha) \text{ pour tous les } c, d \in \mathbf{F}_q^*, \alpha \in \Omega.$$

Notons $Sym_q(\Omega)$ *l'ensemble des permutations monomiales de* Ω *(c'est un sous-ensemble de* $Sym(\Omega_q)$*).*

Proposition 1.6.10 $Sym_q(\Omega)$ *avec le produit usuel des permutations est un groupe.*

Preuve. Comme $Sym_q(\Omega)$ contient l'identité (il n'est pas vide), il suffit de montrer que $uv \in Sym_q(\Omega)$ et $u^{-1} \in Sym_q(\Omega)$ si $u, v \in Sym_q(\Omega)$.

Nous avons

$$(ab, \gamma)^{uv} = ((ab, \gamma)^u)^v = (a(b, \gamma)^u)^v.$$

Soit $(d, \delta) = (b, \gamma)^u$. Alors

$$(ab, \gamma)^{uv} = (a(b, \gamma)^u)^v = (a(d, \delta))^v = (ad, \delta)^v = a(d, \delta)^v = a((b, \gamma)^u)^v = a(b, \gamma)^{uv},$$

et donc $uv \in Sym_q(\Omega)$.

Continuons : soit $(ab, \gamma)^{u^{-1}} = (d, \delta)$. Alors

$$(ab, \gamma) = (d, \delta)^u \quad \text{et} \quad (d, \delta)^u = (aa^{-1}d, \delta)^u = a(a^{-1}d, \delta)^u,$$

d'où

$$(ab,\gamma) = a(a^{-1}d,\delta)^u \quad \text{et} \quad a^{-1}(ab,\gamma) = (a^{-1}d,\delta)^u.$$

Comme

$$a^{-1}(ab,\gamma) = a^{-1}a(b,\gamma) = (b,\gamma),$$

nous obtenons

$$(b,\gamma) = (a^{-1}d,\delta)^u \quad \text{et donc} \quad (b,\gamma)^{u^{-1}} = (a^{-1}d,\delta) = a^{-1}(d,\delta),$$

d'où

$$(d,\delta) = a(b,\gamma)^{u^{-1}} \quad \text{et donc} \quad (ab,\gamma)^{u^{-1}} = a(b,\gamma)^{u^{-1}},$$

ce qui montre que $u^{-1} \in Sym_q(\Omega)$. □

Proposition 1.6.11 *Soit g une permutation monomiale de Ω. Alors il existe une fonction $j : \Omega \to \mathbf{F}_q^*$ et une permutation z de Ω telles que $(1,\gamma)^g = (j(\gamma),\gamma^z)$ pour tout $\gamma \in \Omega$.*

Preuve. Comme g est une permutation de $\Omega_q = \mathbf{F}_q^* \times \Omega$, à chaque couple $(1,\gamma)$ g fait correspondre une autre couple (d,δ) (i.e. $(1,\gamma)^g = (d,\delta)$). C'est comme cela que nous définissons j : l'image de $\gamma \in \Omega$ par j est le d correspondant.

Soient maintenant $(d_1,\delta_1) = (1,\gamma_1)^g$ et $(d_2,\delta_2) = (1,\gamma_2)^g$, où $\gamma_1 \neq \gamma_2$. Supposons que $\delta_1 = \delta_2$. Alors

$$(d_1^{-1},\gamma_1)^g = d_1^{-1}(1,\gamma_1)^g = d_1^{-1}(d_1,\delta_1) = (1,\delta_1)$$

et

$$(d_2^{-1},\gamma_2)^g = d_2^{-1}(1,\gamma_2)^g = d_2^{-1}(d_2,\delta_2) = (1,\delta_2).$$

Mais $\delta_1 = \delta_2$, donc

$$(1,\delta_1) = (1,\delta_2) \quad \text{et} \quad (d_1^{-1},\gamma_1)^g = (d_2^{-1},\gamma_2)^g.$$

Comme g est une permutation de Ω_q, elle est injective, d'où $(d_1^{-1},\gamma_1) = (d_2^{-1},\gamma_2)$, ce qui contredit $\gamma_1 \neq \gamma_2$. Donc pour tous les γ_1 et γ_2 différents nous obtenons δ_1 et δ_2 différents. L'application $z : \Omega \to \Omega$ qui à γ associe δ comme ci-dessus est donc injective, donc c'est une permutation de Ω. □

Corollaire 1.6.12 *Soient g une permutation monomiale de Ω, (a_1,α_1) et (a_2,α_2) deux points de Ω_q. Notons $(b_1,\beta_1) \overset{\text{déf}}{=} (a_1,\alpha_1)^g$ et $(b_2,\beta_2) \overset{\text{déf}}{=} (a_2,\alpha_2)^g$. Alors $\alpha_1 = \alpha_2$ si et seulement si $\beta_1 = \beta_2$. De plus, si $\alpha_1 = \alpha_2$, alors $b_2 = a_2a_1^{-1}b_1$.*

Preuve. Nous avons que $(1,\alpha_1)^g = (a_1^{-1}b_1,\beta_1)$ et $(1,\alpha_2)^g = (a_2^{-1}b_2,\beta_2)$. Si $\alpha_1 = \alpha_2$, alors $(1,\alpha_1)^g = (1,\alpha_2)^g$, d'où $(a_1^{-1}b_1,\beta_1) = (a_2^{-1}b_2,\beta_2)$, donc $\beta_1 = \beta_2$ et $b_2 = a_2a_1^{-1}b_1$.

Si $\beta_1 = \beta_2$, alors comme dans la démonstration de la proposition 1.6.11 nous montrons que $\alpha_1 = \alpha_2$. □

Corollaire 1.6.13 *Une permutation monomiale g de Ω est définie complètement par les images de $(1,\gamma)$, $\gamma \in \Omega$, c'est-à-dire par j et z de la proposition 1.6.11.*

Preuve. Supposons que nous connaissions les images de $(1, \gamma)$, $\gamma \in \Omega$. Il faut montrer que nous pouvons déduire les images des autres points par g. Mais comme $(d, \gamma)^g = d(1, \gamma)^g$ et $(1, \gamma)^g = (j(\gamma), \gamma^z)$ est connu, $(d, \gamma)^g = d(j(\gamma), \gamma^z) = (dj(\gamma), \gamma^z)$ est déterminé de manière unique. □

Lemme 1.6.14 *Soient g une permutation monomiale, j et z comme dans la proposition 1.6.11. Alors l'action de l'inverse de g est donnée par*

$$(1, \gamma)^{g^{-1}} = \left(j \left(\gamma^{z^{-1}} \right)^{-1}, \gamma^{z^{-1}} \right)$$

pour tout $\gamma \in \Omega$.

Preuve. Nous savons que $(1, \delta)^g = (j(\delta), \delta^z)$ pour tout $\delta \in \Omega$ (cf. proposition 1.6.11). Alors, pour tout $\delta \in \Omega$,

$$\begin{aligned} & (j(\delta), \delta^z)^{g^{-1}} = (1, \delta) \\ \Leftrightarrow\ & j(\delta)(1, \delta^z)^{g^{-1}} = (1, \delta) \\ \Leftrightarrow\ & (1, \delta^z)^{g^{-1}} = j(\delta)^{-1}(1, \delta) \\ \Leftrightarrow\ & (1, \delta^z)^{g^{-1}} = (j(\delta)^{-1}, \delta) \\ \Leftrightarrow\ & (1, \delta^z)^{g^{-1}} = \left(j \left((\delta^z)^{z^{-1}} \right)^{-1}, (\delta^z)^{z^{-1}} \right). \end{aligned}$$

Notons $\gamma = \delta^z$. Quand δ parcourt Ω, γ aussi parcourt Ω. Donc

$$(1, \gamma)^{g^{-1}} = \left(j \left(\gamma^{z^{-1}} \right)^{-1}, \gamma^{z^{-1}} \right)$$

pour tout $\gamma \in \Omega$. □

Nous aurons besoin des orbites sous l'action des stabilisateurs dans $Sym_q(\Omega)$. Le lemme suivant montre qu'elles sont faciles à calculer.

Lemme 1.6.15 *Soit $(a_1, \alpha_1), \ldots, (a_h, \alpha_h)$ une suite de points de Ω_q. Soit*

$$H = Sym_q(\Omega)_{(a_1,\alpha_1),\ldots,(a_h,\alpha_h)}.$$

La partition $\mathbf{\Theta}(H)$, représentant les H-orbites, est composée des cellules de cardinal 1 $\{(b, \alpha_i)\}$ pour tout $b \in \mathbf{F}_q^*$ et tout i, $1 \leq i \leq h$, et d'une cellule regroupant les points restants de Ω_q.

Preuve. Si $b \in \mathbf{F}_q^*$ et $1 \leq i \leq h$, alors

$$(b, \alpha_i)^g = b(1, \alpha_i)^g = ba_i^{-1}a_i(1, \alpha_i)^g = ba_i^{-1}(a_i, \alpha_i)^g = ba_i^{-1}(a_i, \alpha_i) = (b, \alpha_i)$$

pour tous les $g \in H$, donc (b, α_i) est fixé par tous les éléments de H.

Si $a, b \in \mathbf{F}_q^*$ et $\alpha, \beta \in \Omega \setminus \{\alpha_1, \ldots, \alpha_h\}$, alors il existe $g \in H$ tel que $(a, \alpha)^g = (b, \beta)$, donc (a, α) et (b, β) sont dans la même H-orbite. En effet, g défini par

$$(1, \alpha)^g = (a^{-1}b, \beta), \quad (1, \beta)^g = (a^{-1}b, \alpha) \quad \text{et} \quad (1, \gamma)^g = (1, \gamma)\ \forall \gamma \in \Omega \setminus \{\alpha, \beta\},$$

appartient à H et est tel que $(a, \alpha)^g = (b, \beta)$. □

1.6.2.2 Équivalence et groupe d'automorphismes

Notation 1.6.16 Nous définissons l'action d'une permutation monomiale g de Ω sur un mot $\mathbf{c} = (c_\gamma)_{\gamma\in\Omega}$ de $\mathbf{F}_q^n$ comme suit :

$$\mathbf{c}^g = (j(\delta)c_\delta)_{\gamma\in\Omega}, \quad \text{où} \quad \delta = \gamma^{z^{-1}}.$$

Concrètement, nous multiplions chaque coordonnée c_γ de $\mathbf{c}$ par $j(\gamma)$, en obtenant ainsi un vecteur $(j(\gamma)c_\gamma)_{\gamma\in\Omega}$, et nous permutons les coordonnées de ce vecteur par z.

Nous voyons que l'action d'une permutation monomiale coïncide totalement avec l'action d'une matrice monomiale : d'abord nous multiplions les composantes de $\mathbf{c}$ par les éléments de $\mathbf{F}_q^*$, et ensuite nous permutons les coordonnées. D'ailleurs, à toute fonction $j : \Omega \to \mathbf{F}_q^*$ on peut associer une matrice diagonale inversible, et inversement ; à chaque permutation z de Ω on peut associer une matrice de permutation, et inversement. Donc, les ensembles des matrices monomiales et des permutations monomiales correspondent.

Définition 1.6.17 *Soient C et C' deux codes de longueur n sur $\mathbf{F}_q$. Nous dirons qu'ils sont* équivalents *(ou* isomorphes*) s'il existe une permutation monomiale g de Ω telle que $C' = C^g$, où $C^g = \{\mathbf{c}^g \mid \mathbf{c} \in C\}$.*

Définition 1.6.18 *Le* groupe d'automorphismes *$Aut(C)$ d'un code C est le sous-ensemble de toutes les permutations monomiales g de Ω telles que $C^g = C$, i.e.*

$$Aut(C) = \{g \in Sym_q(\Omega) \mid C^g = C\}.$$

La démonstration du fait que $Aut(C)$ forme un groupe est analogue à celle de la proposition 1.6.3.

Notez que les propriétés 1.6.4 et 1.6.5 restent valables pour le groupe d'automorphismes et pour deux codes équivalents.

Remarque 1.6.19 Notez que [MS77, p.40 et p.238] définit l'équivalence et le groupe d'automorphismes des codes au sens plus général, à savoir on autorise en plus l'action d'un automorphisme du corps $\mathbf{F}_q$ sur les composantes des mots de code. Nous ne connaissons pas d'algorithme "praticable" pour calculer ce groupe général tandis que pour calculer le groupe monomial on pourra appliquer l'algorithme de Leon décrit dans le chapitre 2. C'est pourquoi nous ne nous intéresserons qu'au groupe monomial.

1.6.3 Expression du groupe d'automorphismes au moyen d'un groupe de permutations

Nous avons élaboré certaines fonctions (appelées *raffinements*, cf. définitions 2.4.1, 2.4.7 et section 3.4) permettant de bien accélérer le calcul du groupe de permutations $Perm(C)$

d'un code C en utilisant l'algorithme de Leon. Par contre, pour trouver le groupe d'automorphismes $Aut(C)$ directement elles ne sont pas applicables. Si nous pouvions exprimer $Aut(C)$ comme le groupe de permutations $Perm(C^*)$ pour un certain code C^*, nous pourrions appliquer ces raffinements pour calculer efficacement $Perm(C^*)$ et trouver ainsi $Aut(C)$. Dans cette section nous allons construire un code C^* satisfaisant cette condition. Cette construction repose sur une construction similaire pour trouver un groupe d'automorphismes d'une matrice, proposée par Leon [Leo91, §9(g)]. Nous avons pu l'adapter pour les codes.

Rappelons que $Aut(C)$ est défini comme un groupe de permutations de Ω_q, et que Ω_q est l'ensemble des couples (a, γ), $a \in \mathbf{F}_q^*$, $\gamma \in \Omega$. Nous définissons le groupe de permutations d'un code C comme un sous-groupe de Ω, où Ω est un ensemble qui indexe les coordonnées des mots de C. Donc en fait le groupe de permutations de C est un groupe de permutations de Ω. Nous voulons trouver un code C^* tel que son groupe de permutations contienne $Aut(C)$, c'est-à-dire contienne un groupe de permutations de Ω_q. Donc les coordonnées des mots de C^* doivent être indexées par Ω_q.

Nous construisons un tel code C^* de manière suivante :

Définition 1.6.20 *Soit n un entier positif non nul, q une puissance d'un nombre premier, Ω un ensemble de cardinal n utilisé pour indexer les coordonnées des mots de $\mathbf{F}_q^n$, $\Omega_q = \Omega \times \mathbf{F}_q^*$ et C un code de longueur n sur $\mathbf{F}_q$. Le* code étalé *C^* de C est le code défini à partir de C comme suit :*

$$C^* = \{ \left(a^{-1}c_\delta\right)_{(a,\delta)\in\Omega_q} \mid \mathbf{c} = (c_\gamma)_{\gamma\in\Omega} \in C \}. \tag{1.6}$$

Alors nous avons le résultat suivant :

Proposition 1.6.21 *Soit n un entier positif non nul, q une puissance d'un nombre premier, Ω un ensemble de cardinal n utilisé pour indexer les coordonnées des mots de $\mathbf{F}_q^n$, $\Omega_q = \Omega \times \mathbf{F}_q^*$, C un code de longueur n sur $\mathbf{F}_q$ et C^* le code étalé de C. Alors*

$$Aut(C) = Perm(C^*) \cap Sym_q(\Omega) = \{g \in Sym_q(\Omega) \mid g \in Perm(C^*)\}.$$

Preuve. Montrons d'abord que $Aut(C) \subset Perm(C^*) \cap Sym_q(\Omega)$.

Nous avons par définition de $Aut(C)$ que $Aut(C) \subset Sym_q(\Omega)$. Il faut donc montrer que $Aut(C) \subset Perm(C^*)$.

Soit $g \in Aut(C)$, c'est-à-dire

$$\text{pour tout } \mathbf{c} = (c_\gamma)_{\gamma\in\Omega} \in C, \quad \mathbf{c}^g = \left(j\left(\gamma^{z^{-1}}\right) c_{\gamma^{z^{-1}}}\right)_{\gamma\in\Omega} \in C$$

(cf. notation 1.6.16). Il faut montrer que $g \in Perm(C^*)$, c'est-à-dire pour tout $\mathbf{c}^* \in C^*$, $(\mathbf{c}^*)^g \in C^*$, où g agit sur C^* comme une permutation (cf. formule (1.5), p. 30) et non comme une permutation monomiale.

Soit $\mathbf{c}^* \in C^*$. Par construction de C^*, il existe $\mathbf{c} \in C$ tel que

$$\mathbf{c}^* = \left(a^{-1}c_\gamma\right)_{(a,\gamma)\in\Omega_q}.$$

Notons $c^*_{(a,\gamma)}$ la composante de $\mathbf{c}^*$ indexée par $(a,\gamma) \in \Omega_q$. Par définition

$$c^*_{(a,\gamma)} = a^{-1} c_\gamma. \tag{1.7}$$

Alors

$$(\mathbf{c}^*)^g = \left(\left(c^*_{(a,\gamma)} \right)_{(a,\gamma)\in\Omega_q} \right)^g \overset{(1.5)}{=} \left(c^*_{(a,\gamma)^{g^{-1}}} \right)_{(a,\gamma)\in\Omega_q}.$$

Nous utilisons le lemme 1.6.14 pour exprimer l'action de g^{-1}. En continuant les égalités nous obtenons

$$\begin{aligned}
(\mathbf{c}^*)^g &= \left(c^*_{(a,\gamma)^{g^{-1}}} \right)_{(a,\gamma)\in\Omega_q} \\
&= \left(c^*_{a(1,\gamma)^{g^{-1}}} \right)_{(a,\gamma)\in\Omega_q} \\
&= \left(c^*_{a\left(j\left(\gamma^{z^{-1}}\right)^{-1}, \gamma^{z^{-1}}\right)} \right)_{(a,\gamma)\in\Omega_q} \\
&= \left(c^*_{\left(aj\left(\gamma^{z^{-1}}\right)^{-1}, \gamma^{z^{-1}}\right)} \right)_{(a,\gamma)\in\Omega_q} \\
&\overset{(1.7)}{=} \left(a^{-1} j\left(\gamma^{z^{-1}}\right) c_{\gamma^{z^{-1}}} \right)_{a\in\mathbf{F}_q^*, \gamma\in\Omega} \\
&= \left(a^{-1} \left(j\left(\gamma^{z^{-1}}\right) c_{\gamma^{z^{-1}}} \right)_{\gamma\in\Omega} \right)_{a\in\mathbf{F}_q^*} \\
&= \left(a^{-1} \mathbf{c}^g \right)_{a\in\mathbf{F}_q^*}.
\end{aligned}$$

Notons $\mathbf{c}^g = \mathbf{d} = (d_\gamma)_{\gamma\in\Omega}$. Nous avons donc

$$(\mathbf{c}^*)^g = \left(a^{-1} \mathbf{c}^g \right)_{a\in\mathbf{F}_q^*} = \left(a^{-1} (d_\gamma)_{\gamma\in\Omega} \right)_{a\in\mathbf{F}_q^*} = \left(a^{-1} d_\gamma \right)_{(a,\gamma)\in\Omega_q} \in C^*,$$

car $\mathbf{d} \in C$. Donc $g \in Perm(C^*)$ et $Aut(C) \subset Perm(C^*)$.

Maintenant nous allons montrer que $Perm(C^*) \cap Sym_q(\Omega) \subset Aut(C)$. Supposons que $g \in Perm(C^*) \cap Sym_q(\Omega)$. Il faut montrer que $g \in Aut(C)$, c'est-à-dire que pour tout $\mathbf{c} \in C$, $\mathbf{c}^g \in C$.

Soit $\mathbf{c} \in C$. Montrons que $\mathbf{c}^g \in C$. Soit $\mathbf{c}^*$ le mot de C^* associé, c'est-à-dire

$$\mathbf{c}^* = \left(a^{-1} c_\gamma \right)_{(a,\gamma)\in\Omega_q}.$$

Exactement comme dans la première partie de la démonstration nous montrons que

$$(\mathbf{c}^*)^g = \left(a^{-1} \mathbf{c}^g \right)_{a\in\mathbf{F}_q^*}.$$

D'ailleurs, $(\mathbf{c}^*)^g \in C^*$, car $g \in Perm(C^*)$. Donc

$$\left(a^{-1} \mathbf{c}^g \right)_{a\in\mathbf{F}_q^*} = \left(a^{-1} d_\gamma \right)_{(a,\gamma)\in\Omega_q} \in C^*.$$

Par construction de C^* nous avons que $\mathbf{c}^g = (d_\gamma)_{\gamma\in\Omega} \in C$. □

D'ailleurs, de la même façon il est possible d'exprimer l'équivalence entre deux codes C et C' par l'équivalence par permutation entre C^* et C'^* :

Proposition 1.6.22 *Soit n un entier positif non nul, q une puissance d'un nombre premier et Ω un ensemble de cardinal n utilisé pour indexer les coordonnées des mots de $\mathbf{F}_q^n$. Soient C et C' deux codes de longueur n sur $\mathbf{F}_q$. Les codes C et C' sont équivalents si et seulement si leurs codes étalés C^* et C'^* sont équivalents par une permutation de $Sym_q(\Omega)$. Et même plus : $C^g = C'$ pour un $g \in Sym_q(\Omega)$ si et seulement si $(C^*)^g = C'^*$.*

Preuve. La démonstration est analogue à la démonstration de la proposition 1.6.21. □

1.6.3.1 En pratique

Quand on implante les permutations monomiales de Ω, i.e. les permutations de $\Omega_q = \mathbf{F}_q^* \times \Omega$, on les représente habituellement par les permutations de $\{1, 2, \ldots, |\Omega_q|\}$. On fixe donc un ordre quelconque sur les éléments $b_1, \ldots, b_{|\Omega_q|}$ de Ω_q et on représente b_i par i. Comme les coordonnées des mots de code de C^* sont indexées par Ω_q, cela fixe aussi ces coordonnées, c'est-à-dire cela fixe l'ordre des colonnes de C^*. Soit $\mathbf{F}_q^* = \{a_1, \ldots, a_{q-1}\}$, $a_1 < a_2 < \cdots < a_{q-1}$, et soit $\Omega = \{\omega_1, \ldots, \omega_n\}$, $\omega_1 < \omega_2 < \cdots < \omega_n$. Ordonnons, par exemple, Ω_q selon l'ordre lexicographique, c'est-à-dire si (a, γ) et (d, δ) appartiennent à Ω_q, $(a, \gamma) < (d, \delta)$ si et seulement si $a < d$ ou $a = d$ et $\gamma < \delta$. Plus précisément,

$$(a_1, \omega_1) < (a_1, \omega_2) < \cdots < (a_1, \omega_n) < (a_2, \omega_1) < \cdots$$

$$< (a_2, \omega_n) < \cdots < (a_{q-1}, \omega_1) < \cdots < (a_{q-1}, \omega_n).$$

Alors le mot de code $\mathbf{c}^* = \left(c^*_{(a,\gamma)}\right)_{(a,\gamma)\in\Omega_q}$ de C^* correspondant à un mot de code $\mathbf{c} = (c_\delta)_{\delta\in\Omega}$ de C est

$$\begin{aligned}
\mathbf{c}^* &= \left(c^*_{(a_1,\omega_1)}, \ldots, c^*_{(a_1,\omega_n)}, c^*_{(a_2,\omega_1)}, \ldots, c^*_{(a_2,\omega_n)}, \ldots, c^*_{(a_{q-1},\omega_1)}, \ldots, c^*_{(a_{q-1},\omega_n)}\right) \\
&\overset{(1.6)}{=} \left(a_1^{-1} c_{\omega_1}, \ldots, a_1^{-1} c_{\omega_n}, a_2^{-1} c_{\omega_1}, \ldots, a_2^{-1} c_{\omega_n}, \ldots, a_{q-1}^{-1} c_{\omega_1}, \ldots, a_{q-1}^{-1} c_{\omega_n}\right) \\
&= \left(a_1^{-1}\left(c_{\omega_1}, \ldots, c_{\omega_n}\right) | a_2^{-1}\left(c_{\omega_1}, \ldots, c_{\omega_n}\right) | \cdots | a_{q-1}^{-1}\left(c_{\omega_1}, \ldots, c_{\omega_n}\right)\right) \\
&= \left(a_1^{-1}\mathbf{c} | a_2^{-1}\mathbf{c} | \cdots | a_{q-1}^{-1}\mathbf{c}\right),
\end{aligned}$$

ce qui signifie que

$$C^* = (a_1^{-1} C | a_2^{-1} C | \cdots | a_{q-1}^{-1} C)$$

(cf. notation 1.3.12).

Un autre exemple. Supposons que nous représentions les permutations de Ω_q en utilisant l'ordre suivant sur les éléments de Ω_q :

$$(a_1, \omega_1) < (a_2, \omega_1) < \cdots < (a_{q-1}, \omega_1) < (a_1, \omega_2) < \cdots$$

$$< (a_{q-1}, \omega_2) < \cdots < (a_1, \omega_n) < \cdots < (a_{q-1}, \omega_n).$$

Alors

$$\begin{aligned}
\mathbf{c}^* &= \left(c^*_{(a_1,\omega_1)}, \ldots, c^*_{(a_{q-1},\omega_1)}, c^*_{(a_1,\omega_2)}, \ldots, c^*_{(a_{q-1},\omega_2)}, \ldots, c^*_{(a_1,\omega_n)}, \ldots, c^*_{(a_{q-1},\omega_n)}\right) \\
&\overset{(1.6)}{=} \left(a_1^{-1}c_{\omega_1}, \ldots, a_{q-1}^{-1}c_{\omega_1}, a_1^{-1}c_{\omega_2}, \ldots, a_{q-1}^{-1}c_{\omega_2}, \ldots, a_1^{-1}c_{\omega_n}, \ldots, a_{q-1}^{-1}c_{\omega_n}\right) \\
&= \left(c_{\omega_1}\left(a_1^{-1}, \ldots, a_{q-1}^{-1}\right) \mid c_{\omega_2}\left(a_1^{-1}, \ldots, a_{q-1}^{-1}\right) \mid \cdots \mid c_{\omega_n}\left(a_1^{-1}, \ldots, a_{q-1}^{-1}\right)\right).
\end{aligned}$$

1.7 Représentation des groupes de permutations

1.7.1 Introduction

Soit G un groupe de permutations d'un ensemble fini Ω. Un problème important de la théorie algorithmique des groupes ("computational group theory") est de déterminer efficacement les propriétés du groupe G. Même pour Ω relativement petit G peut être assez grand, il est donc rarement possible de produire une liste explicite des éléments de G. Le plus souvent G est donné par un ensemble de générateurs. Mais cette spécification ne permet pas, par exemple, de trouver facilement l'ordre de G. Si nous voulons manipuler efficacement de grands groupes de permutations sur un ordinateur, nous avons besoin d'une méthode pour les représenter telle que cette représentation ne demande pas beaucoup de mémoire et qu'elle permette au moins de répondre efficacement aux questions suivantes :

1. Quel est l'ordre de G?
2. Étant donnée une permutation g de Ω, est-ce que $g \in G$?
3. Comment choisir un élément de G au hasard?
4. Comment parcourir tous les éléments de G sans répétition?

Pour résoudre ces problèmes Charles C. Sims [Sim70, Sim71a, Sim71b] a introduit les notions de *chaîne de stabilisateurs*, de *base* et d'*ensemble fort de générateurs* ("chain of stabilizers, base and strong generating set"). Considérons une chaîne de stabilisateurs de G

$$G \supseteq G_{\alpha_1} \supseteq G_{\alpha_1,\alpha_2} \supseteq \ldots \supseteq G_{\alpha_1,\alpha_2,\ldots,\alpha_{s-1}} \supseteq G_{\alpha_1,\alpha_2,\ldots,\alpha_{s-1},\alpha_s} = \{id\},$$

où α_i, $1 \leq i \leq s$, sont des éléments de Ω. Soit $U^{(i)}$ un ensemble de représentants des classes à droite modulo $G_{\alpha_1,\alpha_2,\ldots,\alpha_{i-1},\alpha_i}$ dans $G_{\alpha_1,\alpha_2,\ldots,\alpha_{i-1}}$, $1 \leq i \leq s$. Alors toute permutation $g \in G$ a une représentation unique de la forme $g = u_s u_{s-1} \cdots u_2 u_1$, où $u_i \in U^{(i)}$ pour tout i, $1 \leq i \leq s$ (cf. section 1.7.2). Cela va permettre de répondre aux questions posées (cf. sections 1.7.2 et 1.7.5). Il n'est même pas nécessaire de garder $U^{(i)}$ en mémoire, parce que nous pouvons calculer facilement $U^{(i)}$ en entier (cf. algorithme 1.7.10) ou un de ses éléments (cf. algorithme 1.7.8) si nous avons un ensemble de générateurs de $G_{\alpha_1,\alpha_2,\ldots,\alpha_{i-1}}$. Donc si nous avons un ensemble X de générateurs de G tel que X contient un ensemble de générateurs de $G_{\alpha_1,\alpha_2,\ldots,\alpha_{i-1},\alpha_i}$ pour tout i, $1 \leq i < s$ (X est appelé *ensemble fort de générateurs* de G associé à la *base* $(\alpha_1, \alpha_2, \ldots, \alpha_{s-1}, \alpha_s)$), nous pouvons trouver facilement, au besoin, les éléments de $U^{(i)}$ pour tout i, $1 \leq i \leq s$. C'est pourquoi les notions de base et d'ensemble fort de générateurs sont si importantes. Ce sont les concepts sous-jacents dans la quasi-totalité des algorithmes polynomiaux de la théorie algorithmique des groupes (par exemple, les algorithmes pour calculer les centralisateurs, les normalisateurs, les intersections, les classes de conjugaison, etc...). Nous allons décrire ces concepts dans les sections 1.7.2–1.7.5. Ces sections sont basées essentiellement sur les articles de Charles C. Sims [Sim71b], Jeffrey S. Leon [Leo80], Gregory Butler et John J. Cannon [BC82]. À la section 1.7.6 nous donnons les références d'autres travaux liés à ces notions importantes.

1.7.2 Chaîne de stabilisateurs

Dans cette section nous présentons l'approche originale de Sims [Sim70] pour répondre aux questions posées dans l'introduction (section 1.7.1).

Soit G un groupe de permutations d'un ensemble fini Ω.

Définition 1.7.1 *Soit $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ un sous-ensemble ordonné de Ω. Nous appelons B* base[2] *de G, si l'unique élément de G qui fixe tous les points de B est l'identité, c'est-à-dire $G_{\alpha_1,\alpha_2,\ldots,\alpha_s} = \{id\}$.*

Évidemment, chaque groupe de permutations a une base (par exemple, Ω ordonné est une base), mais deux bases pour un groupe donné peuvent être de tailles différentes.

Proposition 1.7.2 *Un élément g de G est déterminé de manière unique par l'image de la base B sous g, c'est-à-dire si $g_1, g_2 \in G$, $B^{g_1} = B^{g_2}$, alors $g_1 = g_2$.*

Preuve. En effet, si $g_1, g_2 \in G$ et $B^{g_1} = B^{g_2}$, alors

$$B^{g_1 g_2^{-1}} = (B^{g_1})^{g_2^{-1}} = (B^{g_2})^{g_2^{-1}} = B^{g_2 g_2^{-1}} = B^{id} = B,$$

et d'après la définition de la base $g_1 g_2^{-1} = id$, d'où $g_1 = g_2$. □

La réciproque est aussi vrai, c'est-à-dire :

Proposition 1.7.3 *Si $B = (\alpha_1, \ldots, \alpha_s)$ est une suite de points de Ω, satisfaisant :*

$$\textit{pour toute paire de permutations } g_1, g_2 \textit{ de } G, \textit{ si } B^{g_1} = B^{g_2}, \textit{ alors } g_1 = g_2, \tag{1.8}$$

alors B est une base de G.

2. Ici l'utilisation du terme "base" diffère de son utilisation habituelle. D'habitude le terme "base" désigne un sous-ensemble Y de la structure Z considérée, qui engendre cette dernière de manière unique (c'est-à-dire chaque élément de Z s'exprime de façon unique par les éléments de Y, par exemple, prenez une base d'un espace vectoriel). Dans notre cas une base de G sera composée des points de l'ensemble sur lequel G agit. Il vaut mieux peut-être utiliser ici le terme "base-point", mais nous utilisons le terme "base" qui est plus court et qui est utilisé dans ce contexte dans les publications en anglais.

La condition (1.8) s'écrit autrement comme suit :

$$\begin{aligned}&\textit{si } M = (\beta_1, \ldots, \beta_s) \textit{ est une suite de points de } \Omega,\\ &\textit{alors il existe au plus une permutation } g \in G \textit{ telle que } B^g = M.\end{aligned} \tag{1.9}$$

Preuve. Montrons d'abord que la condition (1.8) est vraiment équivalente à la condition (1.9).

Supposons que la dernière condition soit satisfaite. Soient $g_1, g_2 \in G$ et $M \overset{\text{déf}}{=} B^{g_1} = B^{g_2}$. Il existe au plus une permutation $g \in G$ telle que $B^g = M$, donc $g_1 = g_2$.

De l'autre coté, si (1.8) est satisfait, soit $M = (\beta_1, \ldots, \beta_s)$ une suite de points de Ω. Supposons qu'il existe deux permutations différentes $g_1, g_2 \in G$ telles que $B^{g_1} = M$ et $B^{g_2} = M$. Mais alors $B^{g_1} = B^{g_2}$ et d'après (1.8) $g_1 = g_2$, ce qui contredit l'hypothèse sur g_1 et g_2.

Montrons maintenant que B est une base si (1.8) est satisfait. Supposons qu'il existe $g \in G$, $g \neq id$, tel que g fixe tous les points de B, c'est-à-dire $B^g = B$. Mais $B^{id} = B$ aussi, donc $B^g = B^{id}$ et d'après (1.8) $g = id$, ce qui contredit l'hypothèse sur g. □

Notons $G^{(1)} = G$ et $G^{(i)} = G_{\alpha_1, \alpha_2, \ldots, \alpha_{i-1}}$ pour tout i, $2 \leq i \leq s+1$. Considérons la chaîne de stabilisateurs

$$G = G^{(1)} \supseteq G^{(2)} \supseteq \ldots \supseteq G^{(s)} \supseteq G^{(s+1)} = \{id\}.$$

Définition 1.7.4 *Soit $B = (\alpha_1, \ldots, \alpha_s)$ une base de G. Pour tout i, $1 \leq i \leq s$, nous notons $\Delta^{(i)}$ l'orbite de α_i sous l'action de $G^{(i)}$, c'est-à-dire $\Delta^{(i)} = {\alpha_i}^{G^{(i)}}$, et nous l'appelons* i-ième orbite de base *("i^{th} basic orbit") de G, et nous notons $U^{(i)}$ un ensemble de représentants des classes à droite modulo $G^{(i+1)}$ dans $G^{(i)}$, et nous l'appelons* i-ième transversal de base *("i^{th} basic transversal") de G.*

Comme nous avons vu dans la section 1.4.6, l'application $u \in U^{(i)} \mapsto {\alpha_i}^u \in \Delta^{(i)}$ est une bijection entre $U^{(i)}$ et $\Delta^{(i)}$. Notons $n_i = \left|U^{(i)}\right| = \left|\Delta^{(i)}\right|$.

Tout élément g de G s'exprime de manière unique sous la forme $g = u_s u_{s-1} \cdots u_2 u_1$, où $u_i \in U^{(i)}$ pour tout i, $1 \leq i \leq s$. En effet, comme les classes à droite modulo $G^{(2)}$ forment une partition de $G = G^{(1)}$, g appartient à une de ces classes. Soit u_1 le représentant de cette classe dans $U^{(1)}$. Alors $g \in G^{(2)} u_1$ et donc $g = g_1 u_1$, $g_1 \in G^{(2)}$. De la même façon nous montrons que $g_1 = g_2 u_2$, $g_2 \in G^{(3)}$, $u_2 \in U^{(2)}$, et donc $g = g_2 u_2 u_1$, et ainsi de suite. Nous continuons jusqu'à ce que nous obtenions $g = g_s u_s u_{s-1} \cdots u_2 u_1$, $u_i \in U^{(i)}$ $\forall\, i, 1 \leq i \leq s$ et $g_s \in G^{(s+1)} = \{id\}$, d'où $g_s = id$ et $g = u_s u_{s-1} \cdots u_2 u_1$. Comme g n'appartient qu'à une classe à droite modulo $G^{(2)}$, u_1 est unique, de la même façon les autres u_i sont uniques aussi d'où l'unicité de cette représentation de g.

Nous avons en même temps démontré que $G \subset U^{(s)} U^{(s-1)} \cdots U^{(2)} U^{(1)}$. Évidemment, $U^{(s)} U^{(s-1)} \cdots U^{(2)} U^{(1)} \subset G$, donc $G = U^{(s)} U^{(s-1)} \cdots U^{(2)} U^{(1)}$. De même, à cause de l'unicité

$$|G| = \prod_{i=1}^{s} \left|U^{(i)}\right| = \prod_{i=1}^{s} \left|\Delta^{(i)}\right| = \prod_{i=1}^{s} n_i. \tag{1.10}$$

Nous avons pu montrer (1.10) en utilisant la formule (1.4) (voir p. 26), il suffit de remplacer ${\alpha_i}^{G_{\alpha_1,\ldots,\alpha_{i-1}}}$ par $\Delta^{(i)}$, $2 \leq i \leq s$, et ${\alpha_1}^G$ par $\Delta^{(1)}$, et utiliser le fait que, comme $(\alpha_1, \ldots, \alpha_s)$ est une base de G, $|G_{\alpha_1,\ldots,\alpha_s}| = 1$. D'ailleurs, la formule (1.4) induit une autre égalité :

$$|G_{\alpha_1,\ldots,\alpha_i}| = \frac{|G|}{|\Delta^{(1)}|\,|\Delta^{(2)}|\cdots|\Delta^{(i)}|} \text{ pour tout } i,\ 1 \leq i \leq s. \tag{1.11}$$

Le raisonnement ci-dessus nous permet de tester si une permutation g de Ω donnée appartient à G. Si nous arrivons à l'exprimer sous la forme $g = u_s \cdots u_1$, où $u_i \in U^{(i)}$ pour tout i, $1 \leq i \leq s$, alors $g \in G$, sinon $g \notin G$. Plus précisément, nous faisons comme suit. Si ${\alpha_1}^g$ n'appartient pas à l'orbite $\Delta^{(1)} = {\alpha_1}^{G^{(1)}} = {\alpha_1}^G$, g n'appartient pas à G. Si ${\alpha_1}^g$ appartient à $\Delta^{(1)}$, soit $u_1 \in U^{(1)}$ l'unique représentant de la classe à droite de g modulo $G^{(2)}$ (c'est-à-dire l'élément de $U^{(1)}$ qui envoie α_1 sur ${\alpha_1}^g$). Alors ${\alpha_1}^{u_1} = {\alpha_1}^g$ et ${\alpha_1}^{gu_1^{-1}} = ({\alpha_1}^g)^{u_1^{-1}} = ({\alpha_1}^{u_1})^{u_1^{-1}} = \alpha_1$ et le problème devient de vérifier si gu_1^{-1} appartient à $G^{(2)} = G_{\alpha_1}$. En continuant ainsi nous construisons les $u_2 \in U^{(2)}$, $u_3 \in U^{(3)}$, etc. . . , jusqu'à ce que soit pour un certain i, $1 \leq i \leq s$, ${\alpha_i}^{g(u_{i-1}u_{i-2}\cdots u_2u_1)^{-1}}$ n'appartient plus à $\Delta^{(i)} = {\alpha_i}^{G^{(i)}}$ (dans ce cas-là $g(u_{i-1}u_{i-2}\cdots u_2u_1)^{-1}$ n'appartient pas à $G^{(i)}$ et donc g n'appartient pas à G), soit nous obtenons $g(u_su_{s-1}\cdots u_2u_1)^{-1}$ qui fixe $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$. Dans le dernier cas g appartient à G si et seulement si $g(u_su_{s-1}\cdots u_2u_1)^{-1} \in G^{(s+1)} = \{id\}$, c'est-à-dire si et seulement si $g = u_su_{s-1}\cdots u_2u_1$.

Nous présentons cet algorithme de façon plus formelle. Rappelons que l'application $u \in U^{(i)} \mapsto {\alpha_i}^u \in \Delta^{(i)}$ est une bijection entre $U^{(i)}$ et $\Delta^{(i)}$, nous pouvons donc indexer les éléments de $U^{(i)}$ par les éléments de $\Delta^{(i)}$, c'est-à-dire $U^{(i)}(\beta)$ sera la permutation de $U^{(i)}$ qui envoie α_i sur β.

Algorithme 1.7.5 `Appartenance`$(g, B, \mathbf{\Delta}, \mathbf{U})$

Entrée : g une permutation de Ω,
$B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ une base de G,
$\mathbf{\Delta} = (\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(s)})$ les orbites de base de G associées à B,
$\mathbf{U} = (U^{(1)}, U^{(2)}, \ldots, U^{(s)})$ les transversaux de base de G associés à B.

Sortie : $vrai$ si $g \in G$ et $faux$ sinon.

Début

$h := g$;
`pour` i `de` 1 `à` s `faire`
  `si` ${\alpha_i}^h \notin \Delta^{(i)}$ `alors`
    `retourner` $faux$;
  `sinon`
    $u := U^{(i)}({\alpha_i}^h)$; [$u \in U^{(i)}$ tel que ${\alpha_i}^u = {\alpha_i}^h$]
    $h := hu^{-1}$;
  `fin si`;
`fin pour`;
`si` $h = id$ `alors`
  `retourner` $vrai$;

```
sinon
   retourner faux;
fin si;
```
Fin.

Notez l'utilisation de l'instruction `retourner` *expr*. Dans les algorithmes présentés dans ce travail, cette instruction a un double effet : arrêter l'exécution de l'algorithme et sortir la valeur de l'expression *expr*.

Dans l'introduction (section 1.7.1) plusieurs questions ont été posées sur le groupe G. Nous pouvons maintenant répondre à ces questions assez facilement si nous connaissons une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ de G, les orbites $\mathbf{\Delta} = (\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(s)})$ de base et des transversaux $\mathbf{U} = (U^{(1)}, U^{(2)}, \ldots, U^{(s)})$ de base associés à B. L'ordre $|G|$ de G est égal à $\sum_{i=1}^{s} n_i$, où $n_i = \left|\Delta^{(i)}\right| = \left|U^{(i)}\right|$ (cf. formule (1.10)). Nous vérifions si une permutation g de Ω appartient à G en appliquant l'algorithme 1.7.5 ***Appartenance***$(g, B, \mathbf{\Delta}, \mathbf{U})$. Pour choisir un élément $g \in G$ au hasard, nous choisissons au hasard u_i dans chaque $U^{(i)}$, $1 \leq i \leq s$, et nous posons $g = u_s u_{s-1} \cdots u_2 u_1$. Si nous choisissons les $u_i \in U^{(i)}$ avec des probabilités égales, la probabilité d'obtenir $g \in G$ est la même pour tout élément de G, nous obtenons donc une loi uniforme de probabilités. Et, enfin, il est possible de parcourir tous les éléments g de G sans répétition en considérant tous les ensembles $\{u_1, u_2, \ldots, u_s\}$, où $u_i \in U^{(i)}$, $1 \leq i \leq s$, et en posant $g = u_s u_{s-1} \cdots u_2 u_1$.

Le problème, c'est que garder tous les $U^{(i)}$, $1 \leq i \leq s$, prend beaucoup de mémoire. Dans les sections suivantes nous montrerons que cela n'est pas nécessaire, et comment il est possible de construire $U^{(i)}$ en entier ou un de ses éléments facilement au besoin.

1.7.3 Orbites et vecteurs de Schreier

Soit G un groupe de permutations de Ω engendré par un ensemble de générateurs X. Soit $\alpha \in \Omega$. Il n'est nécessaire de garder en mémoire ni l'orbite de α sous l'action de G ni des représentants des classes à droite modulo G_α dans G si nous avons une liste v appelée *vecteur de Schreier*, car nous pouvons alors les calculer facilement à partir de X et v. Nous allons décrire dans les sections 1.7.3–1.7.5 comment le faire, donner un algorithme pour calculer v à partir de X, et montrer ainsi l'importance de l'*ensemble fort de générateurs* d'un groupe.

Nous allons essayer d'expliquer le fonctionnement d'un vecteur de Schreier. Soit Δ l'orbite de α sous l'action de G. Soit $\beta \in \Delta$. Alors il existe $g \in G$ tel que $\alpha^g = \beta$, et, comme $G =< X >$, g est un produit de permutations de $X = \{s_1, \ldots, s_m\}$, soit $g = s_{j_r} s_{j_{r-1}} \cdots s_{j_1}$. Trouver un tel g revient à trouver un représentant de la classe à droite modulo G_α correspondant à β (c'est-à-dire de la classe dont chaque élément α envoie sur β). Si nous ne voulons pas garder le représentant g en mémoire, que faut-il garder pour pouvoir le recalculer facilement ? Nous pouvons, par exemple, pour tout $\beta \in \Delta$ mettre en mémoire une suite d'entiers $\{j_r, j_{r-1}, \ldots, j_1\}$ telle que $\alpha^{s_{j_r} s_{j_{r-1}} \cdots s_{j_1}} = \beta$, et si nous avons besoin de g qui à α associe β, nous calculons $g = s_{j_r} s_{j_{r-1}} \cdots s_{j_1}$.

Mais si à $\beta \in \Delta$ nous associons $\{j_r, j_{r-1}, \ldots, j_1\}$, alors à $\gamma \overset{\text{déf}}{=} \beta^{s_{j_1}^{-1}}$ nous pouvons associer $\{j_r, j_{r-1}, \ldots, j_2\}$, car $\alpha^{s_{j_r} s_{j_{r-1}} \cdots s_{j_2}} = (\alpha^{s_{j_r} s_{j_{r-1}} \cdots s_{j_2} s_{j_1}})^{s_{j_1}^{-1}} = \beta^{s_{j_1}^{-1}} = \gamma$. De même, à $\gamma^{s_{j_2}^{-1}}$ nous pouvons associer $\{j_r, j_{r-1}, \ldots, j_3\}$, etc... Nous obtenons une suite d'ensembles $\{j_r, j_{r-1}, \ldots, j_i\}$, $1 \leq i \leq r$, et, bien sûr, ce n'est pas efficace de garder tous ces ensembles en mémoire quand nous pouvons les obtenir à partir d'un unique ensemble $\{j_r, j_{r-1}, \ldots, j_1\}$. Pour éviter cela, nous faisons comme suit : pour tout $\beta \in \Delta$ nous ne mettons en mémoire qu'un entier. Dans l'exemple ci-dessus, à β nous associons j_1, à $\gamma = \beta^{s_{j_1}^{-1}}$ nous associons j_2, à $\gamma^{s_{j_2}^{-1}}$ nous associons j_3, etc... Si nous voulons trouver $g \in G$ tel que $\alpha^g = \beta$, nous faisons comme suit : à β est associé l'entier j_1, nous calculons $\gamma \overset{\text{déf}}{=} \beta^{s_{j_1}^{-1}}$, nous trouvons dans la mémoire l'entier j_2 associé à γ, nous calculons $\gamma^{s_{j_2}^{-1}} (= \beta^{s_{j_1}^{-1} s_{j_2}^{-1}})$, etc..., jusqu'à ce que nous obtenions α. Alors nous posons $g = s_{j_r} s_{j_{r-1}} \cdots s_{j_1}$, et nous avons $\alpha^g = \beta$, car $\beta^{g^{-1}} = \beta^{s_{j_1}^{-1} s_{j_2}^{-1} \cdots s_{j_r}^{-1}} = \gamma^{s_{j_2}^{-1} \cdots s_{j_r}^{-1}} = \cdots = \alpha$. L'algorithme 1.7.8 ***Rep*** décrit cette procédure de façon formelle. Une liste qui à tout $\beta \in \Delta$ associe un entier (l'indice d'une permutation de X) comme décrit ci-dessus est appelé *vecteur de Schreier*. Voici sa définition formelle.

Notez que, pour simplifier les choses, dans cette définition à tout $\beta \in \Delta$ nous associons une permutation de X au lieu de son indice, nous définissons donc un vecteur de Schreier comme une liste de permutations de X, mais en pratique c'est une liste de leurs indices.

Définition 1.7.6 *Soit $G = < X >$ un groupe de permutations de Ω. Soit $\alpha \in \Omega$. Un* vecteur de Schreier *de α associé à X est une fonction $v : \Omega \to X \cup \{-1, 0\}$ telle que*

(a) $v(\beta) = 0$ *si* $\beta \notin \alpha^G$,

(b) $v(\alpha) = -1$,

(c) $v(\beta) \in X$ *si* $\beta \in \alpha^G$, $\beta \neq \alpha$,

(d) pour tout $\beta \in \alpha^G$, la suite définie par $\beta_1 = \beta$, $\beta_{i+1} = \beta_i^{v(\beta_i)^{-1}}$, s'arrête avec $\beta_{r+1} = \alpha$ pour un certain r.

Nous pouvons calculer facilement un vecteur de Schreier de α à partir d'un ensemble de générateurs X, c'est décrit dans la section 1.7.5.

Le vecteur de Schreier de α associé à X décrit l'orbite de α sous l'action de $G = < X >$. Pour chaque point β de Ω, le vecteur de Schreier v ainsi que X contiennent toute l'information nécessaire pour déterminer si $\beta \in \alpha^G$ ($v(\beta) = 0$ si et seulement si $\beta \notin \alpha^G$), et si c'est le cas, pour trouver $g \in G$ tel que $\alpha^g = \beta$. Nous pouvons poser $g = v(\beta_r) v(\beta_{r-1}) \cdots v(\beta_1)$, où les β_i, $1 \leq i \leq r$, sont définis dans la définition 1.7.6*(d)*. L'algorithme ci-dessous décrit cette procédure.

À partir de la définition 1.7.6*(d)*, nous voyons qu'au cours du calcul il faudra calculer les inverses de $v(\beta_i)$. Pour éviter de le faire à chaque appel de cette procédure, nous pouvons précalculer les $\beta_i^{v(\beta_i)^{-1}}$. Pour cela nous introduisons encore un vecteur.

Définition 1.7.7 *Le* vecteur des pointeurs arrières *("vector of backward pointers") de v est une fonction $w : \alpha^G \to \alpha^G \cup \{0\}$ définie par $w(\alpha) = 0$ et $w(\beta) = \beta^{v(\beta)^{-1}}$.*

Algorithme 1.7.8 $\texttt{Rep}(X, v, w, \beta)$
Entrée : X un ensemble de générateurs de G, où G est un groupe de permutations qui agit sur un ensemble fini Ω,
v un vecteur de Schreier de $\alpha \in \Omega$ associé à X,
w le vecteur des pointeurs arrières de v,
$\beta \in \Omega$.
Sortie : une permutation $g \in G$ telle que $\alpha^g = \beta$ (autrement dit, un représentant de la classe à droite modulo G_α dans G correspondant à β) si $\beta \in \alpha^G$, sinon 0.

Début
  `si` $v(\beta) = 0$ `alors retourner` 0; `fin si`;
  $g := id$; $\gamma := \beta$;
  `tant que` $v(\gamma) \neq -1$ `faire`
    $g := v(\gamma)g$;
    $\gamma := w(\gamma)$; [c'est-à-dire $\gamma := \gamma^{v(\gamma)^{-1}}$]
  `fin tant que`;
  `retourner` g;
Fin.

Notez que $\{\texttt{Rep}(X, v, w, \beta) \mid \beta \in \alpha^G\}$ est un ensemble de représentants des classes à droite modulo G_α dans G.

Quand on programme un vecteur de Schreier v de α associé à $X = \{s_1, \ldots, s_m\}$, on ordonne les permutations de X et on représente v par une fonction (un vecteur) de Ω dans $\{-1, 0, 1, 2, \ldots, m\}$, au lieu des éléments de X on ne met en mémoire que leurs indices.

On peut représenter un vecteur de Schreier de α associé à X par un graphe dont l'ensemble des sommets est α^G et l'ensemble des arêtes est $\{(\beta^{v(\beta)^{-1}}, \beta) \mid \beta \in \alpha^G \setminus \{\alpha\}\}$ où chaque arête est étiquetée par $v(\beta)$. Pour plus de détails cf. [CF92b, CF92c, CF93].

1.7.4 Ensemble fort de générateurs

Nous voyons maintenant que si nous connaissons une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ de G, un ensemble de générateurs $X^{(i)}$ pour chaque $G^{(i)} = G_{\alpha_1, \alpha_2, \ldots, \alpha_{i-1}}$ dans la chaîne de stabilisateurs $G = G^{(1)} \supseteq G^{(2)} \supseteq \ldots \supseteq G^{(s)} \supseteq G^{(s+1)} = \{id\}$, un vecteur de Schreier $v^{(i)}$ de α_i associé à $X^{(i)}$ et le vecteur $w^{(i)}$ des pointeurs arrières de $v^{(i)}$ pour tout i, $1 \leq i \leq s$, au besoin nous pouvons calculer rapidement un représentant de toute classe à droite modulo $G^{(i+1)}$ dans $G^{(i)}$, $1 \leq i \leq s$, en appliquant l'algorithme 1.7.8 $\texttt{Rep}$.

Cela montre qu'il est important d'avoir $X^{(i)}$ pour chaque $G^{(i)}$, $1 \leq i \leq s$, ce qui motive l'introduction du concept de l'ensemble fort de générateurs.

Définition 1.7.9 *Soit G un groupe de permutations de Ω engendré par X et soit $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ une base de G. Nous dirons que X est un* ensemble fort de générateurs *("strong generating set") de G associé à la base B si X contient un ensemble de générateurs pour tout $G^{(i)} = G_{\alpha_1, \alpha_2, \ldots, \alpha_{i-1}}$, $1 \leq i \leq s$. Autrement dit, $G^{(i)} = < X \cap G^{(i)} >$ pour tout i, $1 \leq i \leq s$. Nous notons $X^{(i)} = X \cap G^{(i)}$, $1 \leq i \leq s$.*

Nous désignons un ensemble fort de générateurs par l'abréviation SGS ("strong generating set"). Comme il est toujours associé à une base, nous écrivons BSGS pour une base et un ensemble fort de générateurs associé.

1.7.5 Calcul des vecteurs de Schreier

Nous pouvons calculer facilement l'orbite α^G, un vecteur de Schreier de α, le vecteur w des pointeurs arrières de v et des représentants des classes à droite modulo G_α dans G si nous connaissons X un ensemble de générateurs de G. Nous appliquons successivement les éléments de X à α et aux nouveaux points obtenus et ainsi de suite. Chaque fois que nous obtenons un nouveau point γ de α^G en appliquant x de X à $\beta \in \alpha^G$, nous posons $v(\gamma) = x$, $w(\gamma) = \beta$, et nous prenons ux comme un représentant de la classe à droite qui à α fait correspondre γ, où u est le représentant déjà obtenu de la classe qui à α associe β. Voici l'algorithme formel.

Algorithme 1.7.10 *`Orbite`*(Ω, X, α)
Entrée : Ω un ensemble fini,
  $X = \{s_1, \ldots, s_m\}$ un ensemble de générateurs
    d'un groupe de permutations G de Ω,
  $\alpha \in \Omega$.
Sortie : Δ l'orbite α^G de α sous l'action de G,
  v un vecteur de Schreier de α associé à X,
  w le vecteur des pointeurs arrières de v,
  U un ensemble de représentants des classes à droite modulo G_α dans G.
    Nous notons $U(\beta)$ un représentant de la classe correspondant à β,
    c'est-à-dire de la classe dont chaque permutation fait correspondre β à α.

Début
$r := 1$; [le nombre de points trouvés de Δ]
$\beta_r := \alpha$; [β_i, $1 \leq i \leq r$, seront les points de Δ]
$j := 0$; [l'indice du point courant de Δ]
$v(\beta) := 0$ pour tout $\beta \in \Omega$;
$v(\beta_r) := -1$; $w(\beta_r) := 0$;
$U(\beta_r) := id$;
`tant que` $j < r$ `faire`
  $j := j + 1$; [on prend le point suivant β_j de Δ]
  `pour tout` x `dans` X `faire` [appliquer chaque permutation x de X à β_j]
    $\gamma := {\beta_j}^x$; [on calcule l'image du point courant β_j sous x]
    `si` $v(\gamma) = 0$ `alors` [on a trouvé un nouveau point de Δ]
      $r := r + 1$; $\beta_r := \gamma$;
      $v(\beta_r) := x$; $w(\beta_r) := \beta_j$;
      $U(\beta_r) := U(\beta_j)x$;
    `fin si`;
  `fin pour`;

17. **fin tant que**;
18. **retourner** $\{\beta_1, \beta_2, \ldots, \beta_r\}, v, w, U$;

Fin.

En général, on ne calcule que Δ, v et w, c'est-à-dire on n'exécute pas les lignes 6 et 14 de cet algorithme. Nous les avons données pour montrer qu'au besoin on peut calculer un ensemble de représentants des classes à droite modulo G_α dans G aussi facilement que l'orbite de α sous l'action de G.

Nous voyons maintenant qu'il suffit d'avoir une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ et un ensemble fort de générateurs X de G pour pouvoir répondre (de nouveau) aux questions posées dans l'introduction (cf. sections 1.7.1 et 1.7.2). D'abord nous appliquons s fois l'algorithme 1.7.10 ***Orbite*** $(\Omega, X^{(i)}, \alpha_i)$ (cf. définition 1.7.9 pour la définition de $X^{(i)}$), cela nous fournit un vecteur de Schreier $v^{(i)}$ de α_i et le vecteur des pointeurs arrières $w^{(i)}$ de $v^{(i)}$ pour tout i, $1 \leq i \leq s$ (en pratique, nous trouvons et gardons en mémoire aussi les orbites de base $\Delta^{(i)} = {\alpha_i}^{G^{(i)}}$ parce qu'en général elles ne prennent pas beaucoup de place, mais ce n'est pas nécessaire). Nous calculons l'ordre de G selon la même formule $G = \sum_{i=1}^{s} n_i$ (cf. formule (1.10), p. 41), où $n_i = \left|\Delta^{(i)}\right| = \left|\{\gamma \in \Omega \mid v^{(i)}(\gamma) \neq 0\}\right|$. Il suffit de modifier un peu l'algorithme 1.7.5 ***Appartenance*** pour pouvoir tester si une permutation g de Ω appartient à G. Au lieu de tester si ${\alpha_i}^h \notin \Delta^{(i)}$ (3-ième ligne de l'algorithme 1.7.5), nous testons si $v^{(i)}({\alpha_i}^h) = 0$, et au lieu de prendre u dans $U^{(i)}$ (6-ième ligne), nous devons appliquer l'algorithme 1.7.8 ***Rep*** $(X^{(i)}, v^{(i)}, w^{(i)}, {\alpha_i}^h)$ pour le trouver. Voici cette nouvelle version de l'***Appartenance***.

Algorithme 1.7.11 *Appartenance2*$(g, B, X, \mathbf{v}, \mathbf{w})$

Entrée : g une permutation de Ω,
$B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ une base de G,
X un ensemble fort de générateurs de G associé à B,
$\mathbf{v} = (v^{(1)}, v^{(2)}, \ldots, v^{(s)})$ un ensemble de vecteurs de Schreier,
où $v^{(i)}$ est un vecteur de Schreier de α_i associé à $X^{(i)}$, $1 \leq i \leq s$
(cf. définition 1.7.9 pour la définition de $X^{(i)}$),
$\mathbf{w} = (w^{(1)}, w^{(2)}, \ldots, w^{(s)})$ un ensemble de vecteurs des pointeurs arrières,
où chaque $w^{(i)}$ correspond à $v^{(i)}$, $1 \leq i \leq s$.

Sortie : $vrai$ si $g \in G$ et $faux$ sinon.

Début

$h := g$;
pour i **de** 1 **à** s **faire**
  si $v^{(i)}({\alpha_i}^h) = 0$ **alors**
    retourner $faux$;
  sinon
    $u :=$ ***Rep*** $(X^{(i)}, v^{(i)}, w^{(i)}, {\alpha_i}^h)$; $[u \in U^{(i)}$ tel que ${\alpha_i}^u = {\alpha_i}^h]$
    $h := hu^{-1}$;
  fin si;
fin pour;

```
si h = id alors
  retourner vrai;
sinon
  retourner faux;
fin si;
Fin.
```

Pour choisir $g \in G$ au hasard, nous choisissons au hasard dans chaque orbite de base $\Delta^{(i)} = \{\gamma \in \Omega \mid v^{(i)} \neq 0\}$ un élément β_i et nous posons

$$g = \mathit{Rep}(X^{(s)}, v^{(s)}, w^{(s)}, \beta_s)\mathit{Rep}(X^{(s-1)}, v^{(s-1)}, w^{(s-1)}, \beta_{s-1}) \cdots \mathit{Rep}(X^{(1)}, v^{(1)}, w^{(1)}, \beta_1). \tag{1.12}$$

Et nous pouvons parcourir tous les éléments g de G sans répétition en parcourant tous les points β_i des orbites de base $\Delta^{(i)}$ et en formant les produits (1.12).

1.7.6 Remarques et références

Si on garde en mémoire les vecteurs de Schreier et les vecteurs des pointeurs arrières au lieu des ensembles de représentants des classes à droite, on gagne en mémoire, mais on perd en temps de calcul. Il existe une structure de données intermédiaire appelée *arbre de bifurcation étiqueté* (“labelled branching tree”), introduite dans [Jer86], dont la performance en mémoire n'est pas aussi bonne que celle du vecteur de Schreier, mais l'efficacité du calcul de représentants des classes à droite est meilleure (cela demande moins de multiplications de permutations). Pour plus de détails cf. aussi [CF94].

Il existe aussi diverses modifications du vecteur de Schreier, cf. [CF92b, CF92c, CF93].

Si on se donne une base et un ensemble fort de générateurs d'un groupe de permutations G, on peut faire facilement certaines choses avec G, comme c'est montré ci-dessus. Par contre, calculer une BSGS à partir d'un ensemble quelconque de générateurs de G est plus difficile. La première méthode a été présentée par Sims dans [Sim70]. Depuis elle a été améliorée [Sim71b, Sim78b, Leo80, BC82, CFP89, BLS97], les versions probabilistes sont apparues [Leo80, BCFLS91, BCFS91, CF92c, CF93, CF94, BLS97].

Si nous avons déjà une BSGS, nous pouvons avoir besoin d'un SGS associé à une autre base. La procédure pour le construire s'appelle *changement de la base.* Il existe des algorithmes déterministes [Sim71a, Sim71b, BL85, BFP89, CF92b, CF92c] et probabilistes [CF92b, CF92c, CF93] pour le faire.

Si un SGS devient trop grand au cours du calcul, il existe des procédés pour le réduire, cf. [Sim71b, BCFLS91, CF92b].

On trouve des applications des BSGS dans d'autres domaines : la théorie des groupes de matrices [But76, BC82, But82, BCFLS91, CF92c, CF93], les problèmes d'arrangement et de conception de réseaux interconnectés pour les CPU simultanés [KMS84, CFS91, CF92a, CF93], etc… Cf. aussi [CH92] pour plus de références.

Associé à la technique de *recherche arrière* (“backtrack search”), le concept de BSGS permet de faire bien plus : il permet de concevoir de puissants algorithmes de la théorie algorithmique des groupes ainsi que des algorithmes pour trouver les groupes d'automorphismes de certains objets combinatoires, parmi eux des codes, et ainsi de suite. En général, si on veut trouver le sous-groupe H de G composé de tous les éléments de G satisfaisant une certaine propriété, on construit H en commençant par l'autre bout de sa chaîne de stabilisateurs. D'abord on trouve une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ de H, on pose $H^{(s+1)} \stackrel{\text{déf}}{=} H_{\alpha_1,\alpha_2,\ldots,\alpha_s} = \{id\}$, et on remonte de $H^{(i)}$ à $H^{(i-1)}$ en utilisant la recherche arrière. Comme ça on trouve H et, en même temps, une BSGS de H. Le calcul du groupe de permutations d'un code C entre aussi dans cette catégorie d'algorithmes. En effet, on peut considérer qu'on cherche le sous-groupe $Perm(C)$ de $Sym(\Omega)$, $\Omega = \{1, \ldots, n\}$, où n est la longueur de C, composé de tous les éléments g de $Sym(\Omega)$ tels que C reste inchangé sous l'action de g. C'est pourquoi dans le chapitre suivant nous nous intéresserons à la recherche arrière.

Chapitre 2

Méthode des partitions

2.1 Introduction

La *recherche arrière* ("backtrack search") a une grande importance dans la théorie des algorithmes. Pour résoudre certains problèmes, il n'existe pas de méthode plus efficace que la recherche exhaustive. La recherche arrière permet de bien accélérer cette dernière. C'est depuis longtemps une méthode standard lorsque les autres échouent.

Considérons le problème suivant. Soit Ω un ensemble fini. Nous voulons trouver le sous-ensemble H de $Sym(\Omega)$ composé de toutes les permutations de Ω satisfaisant une certaine propriété $\mathcal{P}$, i.e. $H = \{g \in Sym(\Omega) \mid \mathcal{P}(g)\}$ (par la suite, nous noterons cet ensemble $Sym(\Omega)_{\mathcal{P}}$). Pour de nombreux tels problèmes, il existe des algorithmes utilisant la recherche arrière, et la plupart d'entre eux sont les plus efficaces pour leurs problèmes respectifs. Par exemple, la recherche arrière est utilisé dans les algorithmes pour calculer

- les centralisateurs [Sim71a, But82, Leo91],
- les normalisateurs [But83, Leo91, Leo97],
- les stabilisateurs d'ensembles [But82, Leo91],
- les stabilisateurs de partitions ordonnées [Leo91] et non ordonnées [Leo91, Leo97],
- les intersections de sous-groupes [Sim71b, But82, Leo91, CF93] et de classes à droite [Leo91],
- les classes de conjugaison [Sim71a],
- les groupes d'automorphismes (et tester l'isomorphisme)
  - des graphes [McK78, McK81, BL85, Leo91],
  - des carrés latins [BL85],
  - des matrices de Hadamard [Leo79, McK79, McK81],

– des matrices quelconques [Leo91],

– des designs [MS78, McK81, Leo91],

– des codes [Leo82, Leo91],

– etc...

Pour mieux tailler l'arbre associé à la recherche arrière dans le contexte du calcul de l'isomorphisme des graphes, Brendan D. McKay [McK78, McK81] a introduit les raffinements successifs des partitions ordonnées. Jeffrey S. Leon [Leo91, Leo97] a utilisé cette idée pour élaborer de nouveaux algorithmes plus efficaces pour certains problèmes mentionnés ci-dessus. Dans [Leo91] il décrit ces algorithmes de manière uniforme, pour chaque problème particulier il suffit de donner un ensemble de raffinements. Nous suivrons sa méthode et décrirons dans ce chapitre son algorithme général (appelé la *méthode des partitions*), en différant la description de nos raffinements spécifiques jusqu'au chapitre suivant.

Ce chapitre est basé essentiellement sur les articles de Leon [Leo91, Leo97]. Comme la méthode des partitions est assez technique, nous nous sommes efforcés ici de la reformuler d'une manière différente (que nous espérons plus accessible).

La section 2.2 introduit les propriétés et les classifie. La section 2.3 présente l'algorithme classique de recherche arrière (n'utilisant pas de partitions; sections 2.3.1–2.3.2), et introduit la méthode des partitions de Leon en la comparant avec ce dernier (section 2.3.3). Nous espérons que c'est le moyen le plus compréhensible pour faire connaissance avec la méthode des partitions. La section 2.4 donne les définitions nécessaires : nous y introduisons les raffinements élémentaires (sections 2.4.1–2.4.2), les raffinements (section 2.4.3) et **R**-base (section 2.4.4). Nous décrivons leurs propriétés. De plus, nous donnons une proposition (cf. proposition 2.4.13) sur laquelle la méthode des partitions est basée (section 2.4.5). La section 2.5 décrit la méthode des partitions pour trouver $H = \{g \in Sym(\Omega) \mid \mathcal{P}(g)\}$ et introduit l'arbre de la méthode des partitions. La section 2.6 (respectivement 2.7) décrit les modifications de cet algorithme si H est un sous-groupe de $Sym(\Omega)$ (respectivement une classe à droite modulo un sous-groupe de $Sym(\Omega)$). La section 2.8 présente les raffinements pour la propriété d'appartenance à un groupe de permutations donné. La section 2.9 décrit le calcul de **R**-base. La section 2.10 donne plus de détails sur l'implantation des algorithmes. Et enfin la section 2.11 donne quelques remarques sur la complexité de l'algorithme de recherche arrière et de la méthode des partitions.

2.2 Propriétés

Dans cette section nous définissons les propriétés de façon formelle et introduisons les problèmes généraux pour lesquels nous présenterons des algorithmes par la suite.

Soit Ω un ensemble fini de cardinal n.

Définition 2.2.1 *Une* propriété[1] $\mathcal{P}$ *sur* Ω *est une fonction de* $Sym(\Omega)$ *dans* $\{vrai, faux\}$. *Nous disons que* $g \in Sym(\Omega)$ satisfait *la propriété* $\mathcal{P}$ *si* $\mathcal{P}(g) = vrai$.

Notez que souvent au lieu de "$\mathcal{P}(g) = vrai$" nous écrirons simplement "$\mathcal{P}(g)$", car au point de vue de la logique c'est la même chose.

Nous ne nous intéresserons qu'aux propriétés $\mathcal{P}$ qui sont facilement calculables, c'est-à-dire telles qu'on peut évaluer $\mathcal{P}(g)$ en temps polynomial en n (de préférence en temps proche d'un temps linéaire).

Nous étudions le problème suivant :

$$\text{trouver } Sym(\Omega)_{\mathcal{P}} \stackrel{\text{déf}}{=} \{g \in Sym(\Omega) \mid \mathcal{P}(g)\}. \tag{2.1}$$

Les problèmes de la forme (2.1) peuvent être divisées en 3 catégories :

1. *Les problèmes de type sous-groupe ("subgroup-type problems").* $Sym(\Omega)_{\mathcal{P}}$ est un sous-groupe de $Sym(\Omega)$. Notre but est de trouver une BSGS de $Sym(\Omega)_{\mathcal{P}}$. Le calcul du groupe de permutations (ou du groupe d'automorphismes) d'un code est de ce type-là. Nous dirons que la propriété $\mathcal{P}$ est *de type sous-groupe*.

2. *Les problèmes de type classe ("coset-type problems").* $Sym(\Omega)_{\mathcal{P}}$ n'est pas un sous-groupe de $Sym(\Omega)$, mais il est une classe à droite modulo un sous-groupe F de $Sym(\Omega)$ (ou il est vide). Dans ce cas habituellement on veut déterminer si $Sym(\Omega)_{\mathcal{P}}$ n'est pas vide, et s'il ne l'est pas, trouver un élément h de $Sym(\Omega)_{\mathcal{P}}$. Si on veut trouver $Sym(\Omega)_{\mathcal{P}}$ en entier, on calcule F (c'est un problème de type sous-groupe) et on obtient $Sym(\Omega)_{\mathcal{P}} = Fh$. Le calcul d'une permutation (ou d'un isomorphisme) entre deux codes entre dans cette catégorie. Remarquez que si $Sym(\Omega)_{\mathcal{P}}$ est une classe à gauche d'un sous-groupe T de $Sym(\Omega)$, par exemple $Sym(\Omega)_{\mathcal{P}} = gT$, alors $Sym(\Omega)_{\mathcal{P}}$ est une classe à droite modulo le sous-groupe $F = gTg^{-1}$ de $Sym(\Omega)$, car $Sym(\Omega)_{\mathcal{P}} = gT = gTg^{-1}g = Fg$, il suffit donc de considérer les classes à droite. Nous dirons que la propriété $\mathcal{P}$ est *de type classe*.

3. *Les autres problèmes.* $Sym(\Omega)_{\mathcal{P}}$ n'est ni un sous-groupe de $Sym(\Omega)$ ni une classe à droite modulo un sous-groupe de $Sym(\Omega)$.

Nous nous intéresserons surtout aux problèmes des deux premiers types.

Proposition 2.2.2 *[Leo97, p.124] Soit* $\mathcal{P}$ *une propriété telle que* $Sym(\Omega)_{\mathcal{P}}$ *soit non vide et soit une classe à droite d'un sous-groupe de* $Sym(\Omega)$. *Les propriétés* $\mathcal{P}_{\mathsf{G}}$ *et* $\mathcal{P}_{\mathsf{D}}$*, définies*

1. Notez qu'en même temps nous continuerons d'utiliser le terme "propriété" en son sens général (intuitif). Cela ne causera pas d'ambiguïté, car la définition 2.2.1 ne change pas le sens de ce terme, elle ne fait que le préciser dans le contexte mathématique et limiter l'espace de son application : la propriété au sens strict ne s'applique qu'aux permutations. Donc quand nous disons "soit $\mathcal{P}$ une propriété" ou s'il s'agit d'une propriété d'une permutation, nous sous-entendons le sens strict de ce terme, sinon c'est le sens général qui doit être sous-entendu.

par

$$\mathcal{P}_{\mathsf{G}}(x) \;=\; \begin{cases} vrai & s\text{'}il\ existe\ u,v \in Sym(\Omega)_{\mathcal{P}}\ tels\ que\ u = xv, \\ faux & sinon, \end{cases} \tag{2.2}$$

$$\mathcal{P}_{\mathsf{D}}(x) \;=\; \begin{cases} vrai & s\text{'}il\ existe\ u,v \in Sym(\Omega)_{\mathcal{P}}\ tels\ que\ u = vx, \\ faux & sinon, \end{cases} \tag{2.3}$$

*sont des propriétés de type sous-groupe (les indices "*G*" et "*D*" sont des symboles formels choisis pour indiquer respectivement gauche et droite). D'ailleurs,*

1. $Sym(\Omega)_{\mathcal{P}}$ *est*
   (a) *une classe à droite modulo* $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ *et*
   (b) *une classe à gauche modulo* $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$.
2. *Pour tout* $x \in Sym(\Omega)_{\mathcal{P}}$ *nous avons :*
   (a) $(Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}})^x = Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$,
   (b) $\mathcal{P}_{\mathsf{G}}(s) = \mathcal{P}_{\mathsf{D}}(s^x)$ *pour tout* $s \in Sym(\Omega)$,

   où l'action des permutations sur les permutations est définie dans la notation 1.4.7.

Et que faire si $Sym(\Omega)_{\mathcal{P}} = \emptyset$? Quand on se donne une propriété de type classe, on espère que $Sym(\Omega)_{\mathcal{P}}$ est non vide. On définit alors les propriétés $\mathcal{P}_{\mathsf{G}}$ et $\mathcal{P}_{\mathsf{D}}$ de telle façon que si $Sym(\Omega)_{\mathcal{P}}$ s'avérait être non vide, $Sym(\Omega)_{\mathcal{P}}$ serait une classe à droite modulo $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et une classe à gauche modulo $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$. Par exemple, soient C et C' deux codes, et soit $\mathcal{P}(g)$ satisfaite si et seulement si $C' = C^g$. Alors soit $Sym(\Omega)_{\mathcal{P}}$ est vide (si C et C' ne sont pas équivalents par permutation), soit c'est une classe à droite modulo $Perm(C)$ et une classe à gauche modulo $Perm(C')$ (si C et C' sont équivalents par permutation). Donc $\mathcal{P}_{\mathsf{G}}$ est telle que $\mathcal{P}_{\mathsf{G}}(g)$ est satisfaite si et seulement si $C = C^g$, et $\mathcal{P}_{\mathsf{D}}$ est telle que $\mathcal{P}_{\mathsf{D}}(g)$ est satisfaite si et seulement si $C' = C'^g$. En effet, on a $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}} = Perm(C)$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}} = Perm(C')$. Il est clair que $\mathcal{P}_{\mathsf{G}}$ et $\mathcal{P}_{\mathsf{D}}$ ainsi définies satisfont les équations (2.2) et (2.3) quand $Sym(\Omega)_{\mathcal{P}} \neq \emptyset$.

D'habitude la propriété $\mathcal{P}$ est donnée comme une conjonction $\mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_l$ de propriétés plus simples, par exemple, une conjonction de propriétés définies dans la notation 2.2.4 ($\mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_l(g)$ est satisfaite si et seulement si $\mathcal{P}_i(g)$ est satisfaite pour tout $i,\ 1 \leq i \leq l$). Nous dirons alors que la propriété $\mathcal{P}$ *contient* les propriétés $\mathcal{P}_1, \ldots, \mathcal{P}_l$.

Nous considérerons parfois le problème suivant :

$$\text{trouver } G_{\mathcal{Q}} \stackrel{\text{déf}}{=} \{g \in G \mid \mathcal{Q}(g)\}, \tag{2.4}$$

où G est un groupe de permutations de Ω, $\mathcal{Q}$ est une propriété. Dans ce cas, afin de convertir le problème à la forme (2.1), nous posons $\mathcal{P} = \mathcal{E}_G \wedge \mathcal{Q}$, où $\mathcal{E}_G$ est la propriété d'appartenance à G, i.e. $\mathcal{E}_G(g)$ est satisfaite si et seulement si $g \in G$. En effet,

$$\begin{aligned} G_{\mathcal{Q}} &= \{g \in G \mid \mathcal{Q}(g)\} \\ &= \{g \in Sym(\Omega) \mid g \in G \text{ et } \mathcal{Q}(g)\} \\ &= \{g \in Sym(\Omega) \mid \mathcal{E}_G(g) \text{ et } \mathcal{Q}(g)\} \\ &= \{g \in Sym(\Omega) \mid (\mathcal{E}_G \wedge \mathcal{Q})(g)\} \\ &= Sym(\Omega)_{\mathcal{E}_G \wedge \mathcal{Q}}, \end{aligned} \tag{2.5}$$

et le problème devient de la forme (2.1) avec la propriété $\mathcal{P} = \mathcal{E}_G \wedge \mathcal{Q}$.

Notez que la propriété $\mathcal{E}_G(g)$ est facilement calculable, étant donnés une base et un ensemble fort de générateurs de G (cf. algorithme 1.7.11 ***Appartenance2***).

Remarque 2.2.3 Dans ce chapitre nous supposerons toujours qu'une propriété $\mathcal{P}$ est donnée comme une conjonction de propriétés plus simples (par exemple, une conjonction de propriétés de la notation 2.2.4), parmi lesquelles se trouve au moins une propriété d'appartenance à un groupe G, i.e. nous considérerons en fait que nous avons un problème de la forme (2.4). Si ce n'est pas le cas, nous pouvons toujours considérer que $\mathcal{P}$ contient $\mathcal{E}_{Sym(\Omega)}$. En résumé, nous supposerons toujours que $\mathcal{P}$ contient $\mathcal{E}_G$ pour un certain groupe G (qui peut être égal à $Sym(\Omega)$).

De plus, $\mathcal{P}$ peut contenir plusieurs $\mathcal{E}_G$, par exemple, si nous voulons trouver tous les éléments appartenant à l'intersection des groupes $G_1, \ldots, G_l$ et satisfaisant une propriété $\mathcal{Q}$, le problème est décrit par la propriété $\mathcal{E}_{G_1} \wedge \mathcal{E}_{G_2} \wedge \cdots \wedge \mathcal{E}_{G_l} \wedge \mathcal{Q}$.

Notez qu'une conjonction de propriétés de type sous-groupe est aussi une propriété de type sous-groupe, et qu'une conjonction de propriétés de type classe (et, éventuellement, de quelques propriétés de type sous-groupe) est une propriété de type classe.

Nous définissons maintenant certaines propriétés qui s'avéreront utiles par la suite.

Notation 2.2.4 Soit Ω un ensemble fini.

- $\mathcal{E}_G$ — *La propriété d'appartenance à un groupe.* Si G est un groupe de permutations de Ω, $\mathcal{E}_G(g)$, $g \in Sym(\Omega)$, est satisfaite si et seulement si $g \in G$. La propriété $\mathcal{E}_G$ est de type sous-groupe.
- $\mathcal{P}_C$ — *La propriété de permutation laissant un code invariant.* Si C est un code dont les coordonnées sont indexées par Ω, alors $\mathcal{P}_C(g)$, $g \in Sym(\Omega)$, est satisfaite si et seulement si $C^g = C$, i.e. $g \in Perm(C)$. La propriété $\mathcal{P}_C$ est de type sous-groupe.
- $\mathcal{P}_{C,C'}$ — *La propriété d'équivalence par permutation entre deux codes.* Si C et C' sont deux codes sur le même corps fini $\mathbf{F}_q$ et dont les coordonnées sont indexées par Ω, alors $\mathcal{P}_{C,C'}(g)$, $g \in Sym(\Omega)$, est satisfaite si et seulement si $C^g = C'$. La propriété $\mathcal{P}_{C,C'}$ est de type classe, $(\mathcal{P}_{C,C'})_{\mathsf{G}} = \mathcal{P}_C$, $(\mathcal{P}_{C,C'})_{\mathsf{D}} = \mathcal{P}_{C'}$.
- $\mathcal{A}_C$ — *La propriété d'automorphisme d'un code.* Si C est un code sur un corps fini $\mathbf{F}_q$ et dont les coordonnées sont indexées par Ω, alors $\mathcal{A}_C(g)$, $g \in Sym(\Omega_q)$, est satisfaite si et seulement si $g \in Sym_q(\Omega)$ et $C^g = C$, i.e. $g \in Aut(C)$. La propriété $\mathcal{A}_C$ est de type sous-groupe.

$\mathcal{A}_{C,C'}$ *La propriété d'isomorphisme des codes.* Si C et C' sont deux codes sur le même corps fini $\mathbf{F}_q$ et dont les coordonnées sont indexées par Ω, alors $\mathcal{A}_{C,C'}(g)$, $g \in Sym(\Omega_q)$, est satisfaite si et seulement si $g \in Sym_q(\Omega)$ et $C^g = C'$. La propriété $\mathcal{A}_{C,C'}$ est de type classe, $(\mathcal{A}_{C,C'})_{\mathsf{G}} = \mathcal{A}_C$, $(\mathcal{A}_{C,C'})_{\mathsf{D}} = \mathcal{A}_{C'}$.

Notez que nous réservons la notation $\mathcal{E}_G$ pour les groupes G qui sont “connus”, par exemple donnés par une base et un ensemble fort de générateurs. Cela veut dire que nous pouvons manipuler G, par exemple énumérer tous les éléments de G et trouver les G-orbites. Donc, par exemple si nous avons un code C et son groupe de permutations n'est pas connu, i.e. il n'est pas calculé, alors nous utiliserons la notation $\mathcal{P}_C$ pour indiquer la propriété telle que $\mathcal{P}_C(g)$ est satisfaite si et seulement si la permutation g appartient au groupe $Perm(C)$. Par contre, une fois que $Perm(C)$ est calculé, qu'un de ses ensembles de générateurs est connu, nous pourrons utiliser la notation $\mathcal{E}_G$ avec $G = Perm(C)$.

2.3 Recherche arrière et partitions

2.3.1 Procédure générale de recherche arrière

Dans cette section nous décrivons la procédure générale de recherche arrière. Cette section est inspirée de [Knu75] et [Leo82]. Considérons le problème suivant.

Supposons que nous ayons un ensemble fini Y et un entier positif s. Nous voulons trouver l'ensemble Z de tous les s-vecteurs $(\beta_1, \ldots, \beta_s)$, $\beta_h \in Y\ \forall h, 1 \leq h \leq s$, satisfaisant une propriété $\mathcal{P}(\beta_1, \ldots, \beta_s)$ (dans cette section comme en partie dans les suivantes, nous utiliserons le terme “propriété” dans son sens général, pas dans le sens strict de la définition 2.2.1). Souvent la recherche exhaustive est l'unique moyen de le faire : nous formons tous les s-vecteurs $(\beta_1, \ldots, \beta_s)$ d'éléments de Y et nous vérifions la propriété $\mathcal{P}(\beta_1, \ldots, \beta_s)$ pour chacun d'eux. Comment réduire le nombre de vecteurs à vérifier ?

L'approche générale de la recherche arrière consiste à trouver des propriétés intermédiaires $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$, $0 \leq h \leq s$, telles que

$$\mathcal{P}(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_s) \text{ implique } \mathcal{P}_h(\beta_1, \ldots, \beta_h) \text{ pour tout } h,\ 0 \leq h \leq s. \tag{2.6}$$

Autrement dit, si $(\beta_1, \ldots, \beta_h)$ ne satisfait pas la propriété $\mathcal{P}_h$, alors aucune extension $(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_s)$ de $(\beta_1, \ldots, \beta_h)$ ne peut satisfaire la condition de départ $\mathcal{P}$.

La recherche arrière essaie de trouver toutes les solutions $(\beta_1, \ldots, \beta_s)$ de notre problème en parcourant toutes les solutions partielles $(\beta_1, \ldots, \beta_h)$ satisfaisant $\mathcal{P}_h$, $0 \leq h \leq s$, en utilisant l'algorithme suivant.

Pour énumérer tous les éléments de Y, nous classons Y selon un ordre total quelconque — du premier (le plus petit) jusqu'au dernier (le plus grand) élément. Soit ∞ un élément n'appartenant pas à Y. Nous définissons la fonction *ÉlémentSuivant* comme suit :

$$\textit{ÉlémentSuivant}(\beta) = \begin{cases} \text{le premier élément de } Y & \text{si } \beta = \infty, \\ \text{l'élément suivant de } Y & \text{si } \beta \in Y \text{ et } \beta \text{ n'est pas le dernier} \\ & \quad \text{élément de } Y, \\ \infty & \text{si } \beta \text{ est le dernier élément de } Y. \end{cases}$$

L'algorithme de recherche arrière pour calculer Z fonctionne comme suit.

D'abord nous initialisons Z à $\emptyset$ et h à 1.

Supposons que nous soyons dans une étape intermédiaire avec la solution partielle courante $(\beta_1, \ldots, \beta_h)$ dont les extensions ont été déjà vérifiées. Nous prenons la solution partielle suivante, c'est-à-dire nous remplaçons β_h par l'élément suivant de Y. Nous obtenons deux cas, selon que β_h était le dernier élément de Y ou non.

- Si β_h n'était pas le dernier élément de Y, le nouveau β_h appartient à Y lui aussi. D'abord nous vérifions que $(\beta_1, \ldots, \beta_h)$ satisfait la propriété $\mathcal{P}_h$.

  - Si ce n'est pas le cas, nous savons d'après la condition (2.6) qu'il n'existe aucune solution $(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_s)$ commençant par $(\beta_1, \ldots, \beta_h)$, nous pouvons donc passer à la solution partielle suivante, c'est-à-dire nous recommençons la procédure avec la solution partielle courante $(\beta_1, \ldots, \beta_h)$.
  - Si c'est le cas, nous devons explorer les extensions de la solution partielle courante $(\beta_1, \ldots, \beta_h)$.
    - Si cette solution est déjà complète, c'est-à-dire $h = s$, et si elle satisfait la condition initiale $\mathcal{P}(\beta_1, \ldots, \beta_s)$, nous avons trouvé une solution à notre problème, nous la mettons dans Z et nous recommençons avec la solution partielle courante $(\beta_1, \ldots, \beta_h)$.
    - Si $h < s$, nous commençons à explorer toutes les extensions de $(\beta_1, \ldots, \beta_h)$, c'est-à-dire nous augmentons h d'une unité, nous initialisons la nouvelle coordonnée β_h à ∞, et nous recommençons avec ce vecteur $(\beta_1, \ldots, \beta_h)$ plus long d'une coordonnée (nous disons que nous *avançons*).

- Si β_h est le dernier élément de Y, cela veut dire que nous avons exploré déjà toutes les extensions de $(\beta_1, \ldots, \beta_{h-1})$ (c'est-à-dire nous avons exploré déjà toutes les extensions de $(\beta_1, \ldots, \beta_{h-1}, \beta_h)$ pour tout $\beta_h \in Y$, et nous venons de finir avec le dernier élément β_h de Y). Nous devons donc revenir un pas en arrière et choisir un autre β_{h-1}. Nous faisons cela en diminuant h d'une unité et en recommençant la procédure décrite avec la solution partielle courante $(\beta_1, \ldots, \beta_h)$ plus courte d'une unité (nous disons que nous *revenons en arrière*).

Nous nous arrêtons quand nous avons exploré toutes les solutions partielles $(\beta_1, \ldots, \beta_h)$, plus précisément, les extensions de toute solution partielle $(\beta_1, \ldots, \beta_h)$ satisfaisant $\mathcal{P}_h$. Comme nous avons commencé par initialiser h à 1, nous voyons facilement qu'après avoir exploré toutes les extensions du 0-vecteur (), h devient 0. Nous pouvons donc considérer la condition $h = 0$ comme une condition d'arrêt de l'algorithme de recherche arrière.

Voici la présentation formelle de la procédure de recherche arrière que nous venons de décrire.

Algorithme 2.3.1
Début

1. $Z := \emptyset$; [initialisation de l'ensemble de solutions]
2. si $\mathcal{P}_0(()) = faux$ alors [si le 0-vecteur () ne satisfait pas $\mathcal{P}_0$]
3.   retourner Z; [alors $Z = \emptyset$ d'après (2.6)]
4. fin si;
5. $h := 1$; [initialisation de la taille de la solution partielle courante]
6. $\beta_h := \infty$; [$(\beta_1, \ldots, \beta_h)$ sera la solution partielle courante]
7. tant que $h > 0$ faire
8.   si $\mathit{ÉlémentSuivant}(\beta_h) \neq \infty$ alors [nous changeons le dernier élément]
9. $\beta_h := \mathit{ÉlémentSuivant}(\beta_h)$; [de la solution partielle courante]
10.   si $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$ alors [si la propriété intermédiaire $\mathcal{P}_h$ est satisfaite]
11.   si $h = s$ alors [nous avons un s-vecteur]
12.   si $\mathcal{P}(\beta_1, \ldots, \beta_s)$ alors [nous avons trouvé une solution]
13. $Z := Z \cup \{(\beta_1, \ldots, \beta_s)\}$; [nous mettons $(\beta_1, \ldots, \beta_s)$ dans Z]
14.   fin si;
15.   sinon [avancer d'un pas]
16. $h := h + 1$; [la solution partielle courante devient plus longue]
17. $\beta_h := \infty$; [initialisation de la dernière coordonnée]
18.   fin si;
19.   fin si;
20.   sinon [on revient un pas en arrière]
21. $h := h - 1$; [la solution partielle courante devient plus courte]
22.   fin si;
23. fin tant que;
24. retourner Z;

Fin.

Remarque 2.3.2 Notez que si nous ne voulons trouver qu'une solution (comme, par exemple, dans le cas où on cherche un isomorphisme entre deux codes équivalents), après avoir trouvé la première solution nous pouvons la sortir et arrêter l'algorithme. Pour cela nous modifions l'algorithme comme suit : nous remplaçons la 13-ième ligne par

13'.   retourner $(\beta_1, \ldots, \beta_s)$;

2.3.1.1 L'arbre de recherche arrière

On peut représenter l'ensemble des solutions partielles de l'algorithme de recherche arrière par un arbre[2] $\mathbb{T}$ de hauteur au plus s dont[3] les noeuds sont étiquetés par les suites $(\beta_1, \ldots, \beta_h)$, $0 \leq h \leq s$, défini comme suit :

1. Si $\mathcal{P}_0(())$ n'est pas satisfait, alors l'arbre $\mathbb{T}$ est vide, sinon la racine de $\mathbb{T}$ est étiqueté par le 0-vecteur ().

2. Nous l'imaginons renversé, c'est-à-dire la racine est en haut et les feuilles sont en bas.
3. C'est-à-dire qui possède $s + 1$ niveaux, la racine étant au niveau 0.

2. Pour $1 \leq h \leq s$, les enfants du noeud de niveau $h-1$ étiqueté par $(\beta_1, \ldots, \beta_{h-1})$ sont exactement les noeuds étiquetés par $(\beta_1, \ldots, \beta_{h-1}, \beta_h)$, où $\beta_h \in Y$, tels que $\mathcal{P}_h(\beta_1, \ldots, \beta_{h-1}, \beta_h)$ est satisfait.

L'algorithme de recherche arrière parcourt cet arbre une branche après l'autre selon l'ordre lexicographique sur les suites $(\beta_1, \ldots, \beta_h)$ (notez que $(\beta'_1, \ldots, \beta'_{h'}) < (\beta''_1, \ldots, \beta''_{h''})$ si et seulement si soit $\beta'_1 = \beta''_1, \ldots, \beta'_i = \beta''_i$ et $\beta'_{i+1} < \beta''_{i+1}$ pour un certain $0 \leq i < \min(h', h'')$, soit $\beta'_1 = \beta''_1, \ldots, \beta'_i = \beta''_i$ pour $i = \min(h', h'')$ et $h' < h''$, où on compare les β'_j et β''_j selon l'ordre de Y). Plus précisément, nous faisons comme suit.

Nous initialisons Z à $\emptyset$, et nous tentons de commencer la traversée de $\mathbb{T}$ à la racine en calculant $\mathcal{P}_0(())$. Si $\mathcal{P}_0(())$ n'est pas satisfait, alors nous avons fini : $\mathbb{T}$ est vide et $Z = \emptyset$. Supposons maintenant que nous ayons atteint un noeud au niveau $h-1$, $1 \leq h \leq s$, étiqueté par $(\beta_1, \ldots, \beta_{h-1})$. Les enfants de ce noeud correspondent à un sous-ensemble de Y, donc pour chaque point β_h de Y (dans l'ordre de Y) nous vérifions si $\mathcal{P}_h(\beta_1, \ldots, \beta_{h-1}, \beta_h)$ est satisfait. Si c'est le cas, nous descendons du noeud courant à son enfant correspondant à β_h, et nous répétons le processus. Sinon pour ce β_h l'enfant n'existe pas, donc nous prenons l'élément suivant β_h dans Y, ou nous revenons au parent du noeud courant s'il n'en reste plus. Quand nous atteignons une feuille de l'arbre $\mathbb{T}$ au niveau s, i.e. un noeud étiqueté par $(\beta_1, \ldots, \beta_s)$ satisfaisant $\mathcal{P}_s$, nous vérifions si $\mathcal{P}$ est satisfait ; si c'est le cas, nous mettons $(\beta_1, \ldots, \beta_s)$ dans Z.

Quand la traversée de $\mathbb{T}$ est finie, Z est l'ensemble des s-vecteurs satisfaisant $\mathcal{P}$. De plus, nous obtenons les éléments de Z dans l'ordre lexicographique croissant décrit ci-dessus.

2.3.1.2 Remarques sur les propriétés intermédiaires

La condition (2.6) ne définit pas les propriétés intermédiaires $\mathcal{P}_h$ de façon unique, nous avons donc souvent une marge considérable de manoeuvre quand nous les choisissons. Par exemple, nous pouvons simplement choisir les propriétés intermédiaires $\mathcal{P}_h$ telles que $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$ est satisfaite pour tous les $(\beta_1, \ldots, \beta_h)$. Ce sont les propriétés les plus faibles satisfaisant (2.6), et dans ce cas l'algorithme que nous venons de décrire fera exactement la recherche exhaustive, car il ne pourra rejeter aucune solution partielle $(\beta_1, \ldots, \beta_h)$, $0 \leq h \leq s$.

De l'autre coté, la propriété la plus forte est obtenue quand $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$ est satisfaite si et seulement s'il existe $\beta_{h+1}, \ldots, \beta_s \in Y$ satisfaisant $\mathcal{P}(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_s)$. En général, les propriétés plus fortes limitent l'espace de recherche (nous rejetons plus de solutions partielles), mais demandent plus de calcul, il faut essayer donc de trouver un compromis convenable.

Le choix de propriétés intermédiaires $\mathcal{P}_h$ n'est pas toujours évident. Souvent la construction de propriétés convenables est la partie la plus difficile de la conception d'une procédure de recherche arrière pour un problème spécifique. Nous avons réussi à construire de bonnes propriétés intermédiaires pour les problèmes du groupe de permutations et de l'équivalence par permutation des codes linéaires, en utilisant la notion de *signature* introduite par Nicolas Sendrier (cf. [Sen97a, Sen99a, Sen99b] et chapitre 3). À notre connaissance, ce sont les premières propriétés intermédiaires destinées spécialement aux codes. L'algorithme de Leon [Leo82, Leo91] utilise pour les codes les mêmes propriétés intermédiaires que pour les matrices, et c'est sa principale limitation.

2.3.2 Application aux problèmes de type sous-groupe

Soit un ensemble fini Ω. Supposons que nous voulions trouver le sous-groupe $Sym(\Omega)_{\mathcal{Q}}$ de $Sym(\Omega)$, où $\mathcal{Q}$ est une propriété. Par exemple, nous voulons trouver $Perm(C)$ pour un code C, $\mathcal{Q}$ étant égale à $\mathcal{P}_C$.

Supposons que nous trouvions une base $B = (\alpha_1, \alpha_2, \ldots, \alpha_s)$ de $Sym(\Omega)_{\mathcal{Q}}$ (cf. définition 1.7.1 pour la définition d'une base et section 2.9 pour le calcul d'une base dans l'algorithme de Leon). Après tout, trouver une base d'un groupe n'est pas difficile et ne demande même pas de connaître le groupe de façon explicite, par exemple, Ω ordonné est une base (en général, très mauvaise) de tout groupe de permutations de Ω. Nous savons (cf. proposition 1.7.2) que tout élément g de H est déterminé de manière unique par l'image B^g de la base B par g. Pour parcourir les permutations de $Sym(\Omega)_{\mathcal{Q}}$, il suffit donc de parcourir les images de la base B, c'est-à-dire les s-vecteurs $(\beta_1, \ldots, \beta_s)$. Quelle sera la propriété initiale $\mathcal{P}(\beta_1, \ldots, \beta_s)$? Supposons que, étant donnée l'image de la base B par une permutation g de $Sym(\Omega)_{\mathcal{Q}}$, nous soyons capables de trouver g. Alors nous pouvons prendre pour propriété $\mathcal{P}(\beta_1, \ldots, \beta_s)$ la propriété qui est satisfaite si et seulement si nous sommes capables de trouver $g \in Sym(\Omega)_{\mathcal{Q}}$ telle que $B^g = (\beta_1, \ldots, \beta_s)$. Ainsi, si $(\beta_1, \ldots, \beta_s)$ est l'image de B par une permutation g de $Sym(\Omega)_{\mathcal{Q}}$, en vérifiant $\mathcal{P}(\beta_1, \ldots, \beta_s)$ nous pourrons en même temps trouver g, et sinon nous obtiendrons $\mathcal{P}(\beta_1, \ldots, \beta_s) = faux$. Maintenant pour trouver $Sym(\Omega)_{\mathcal{Q}}$, il suffit d'appliquer l'algorithme de recherche arrière, en substituant Ω à Y, avec la propriété $\mathcal{P}$ et des propriétés intermédiaires $\mathcal{P}_h$ dépendant de $\mathcal{Q}$. Si $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$, alors $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$ pour tout h, $0 \leq h \leq s$. Donc si nous montrons pour un certain h qu'il n'existe pas $g \in Sym(\Omega)_{\mathcal{Q}}$ tel que $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$, nous pouvons rejeter cette solution partielle $(\beta_1, \ldots, \beta_h)$. Donc une propriété intermédiaire $\mathcal{P}_h$ peut inclure toutes les conditions nécessaires de l'égalité $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$ pour un certain $g \in Sym(\Omega)_{\mathcal{Q}}$, par exemple, $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$ doit inclure une propriété qui est satisfaite si et seulement si $\beta_i \neq \beta_j$ pour $i \neq j$, car c'est une condition nécessaire pour qu'il existe une permutation g de Ω telle que $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$ (rappelons que par définition $\alpha_i \neq \alpha_j$ pour $i \neq j$). L'algorithme parcourt tous les s-vecteurs (les images possibles de la base B), non éliminés par les $\mathcal{P}_h$, et vérifie la propriété $\mathcal{P}$. Comme cela nous trouvons $Sym(\Omega)_{\mathcal{Q}}$.

En résumé, le calcul de $Sym(\Omega)_{\mathcal{Q}}$ se fait en deux étapes :

1. D'abord nous calculons
   - une base $B = (\alpha_1, \ldots, \alpha_s)$ de $Sym(\Omega)_{\mathcal{Q}}$ et
   - une fonction T qui, pour tout s-vecteur $(\beta_1, \ldots, \beta_s)$ d'éléments de Ω, vérifie s'il existe $g \in Sym(\Omega)_{\mathcal{Q}}$ tel que $B^g = (\beta_1, \ldots, \beta_s)$ et, si c'est le cas, qui trouve ce g.

2. Nous effectuons l'algorithme de recherche arrière 2.3.1 avec $Y = \Omega$, c'est-à-dire nous parcourons toutes les images partielles $(\beta_1, \ldots, \beta_h)$ de la base B en vérifiant $\mathcal{P}_h$:
   - si $\mathcal{P}_h(\beta_1, \ldots, \beta_h)$ est satisfaite,
     - si $h < s$, nous explorons les extensions $(\beta_1, \ldots, \beta_h, \beta_{h+1})$ de $(\beta_1, \ldots, \beta_h)$,
     - sinon nous avons déjà une image complète de la base B, nous appliquons T pour trouver $g \in Sym(\Omega)_{\mathcal{Q}}$ tel que $B^g = (\beta_1, \ldots, \beta_s)$. Si un tel g existe, nous le mettons en mémoire. Nous prenons le s-vecteur suivant et nous continuons.
   - sinon
     - nous prenons un autre β_h s'il y en a encore et nous continuons,
     - sinon nous raccourcissons la solution partielle, c'est-à-dire nous prenons la solution partielle $(\beta_1, \ldots, \beta_{h-1})$.

Remarque 2.3.3 Cette méthode (la recherche arrière classique n'utilisant pas les partitions) présente plusieurs désavantages par rapport à celle utilisant les partitions. Présentons maintenant quelques-uns de ces désavantages.

- L'efficacité de la méthode de recherche arrière dépend beaucoup de la taille de la base $B = (\alpha_1, \ldots, \alpha_s)$ — plus la taille est grande, plus l'espace de recherche est grand. Malheureusement, pour la plupart des propriétés $\mathcal{P}$, il n'est pas évident de trouver une base B de $Sym(\Omega)_{\mathcal{P}}$ plus courte que Ω (ou plus courte qu'une base A de G si $\mathcal{P}$ contient $\mathcal{E}_G$ pour un groupe G), cf., par exemple, [But82]. Nous sommes obligés de poser $B = \Omega$ (ou $B = A$), T étant alors trivial (ou T utilisant une BSGS de G). Il n'existe que quelques propriétés pour lesquelles nous sommes capables de construire une base non triviale (i.e. plus courte que Ω ou A), par exemple, pour le calcul du groupe d'automorphismes d'un code, cf. [Leo82]. Par contre, la recherche arrière utilisant les partitions permet de construire une base plus courte de manière uniforme pour toute propriété $\mathcal{P}$, cf. sections 2.4 et 2.9.

- Si nous trouvons une base $B = (\alpha_1, \ldots, \alpha_s)$ non triviale, l'algorithme de recherche arrière parcourt les s-vecteurs $(\beta_1, \ldots, \beta_s)$ qui sont susceptibles d'être les images de B par une permutation de $Sym(\Omega)_{\mathcal{P}}$. Mais alors il n'est pas évident de passer d'un s-vecteur $(\beta_1, \ldots, \beta_s)$ à une permutation g de $Sym(\Omega)_{\mathcal{P}}$ (si elle existe), il nous faut pour cela une fonction T qui, elle aussi, est différente pour chaque propriété $\mathcal{P}$. En utilisant les partitions nous évitons cet inconvénient, T est toujours triviale, car il est toujours facile à partir d'une partition ordonnée discrète de déduire la permutation correspondante (cf. section 2.3.3).

2.3.3 Méthode des partitions

Brendan D. McKay [McK78, McK81] a introduit les raffinements des partitions pour résoudre de manière plus efficace le problème d'isomorphisme entre deux graphes. Jeffrey

S. Leon [Leo91, Leo97] a appliqué cette idée pour une famille entière d'algorithmes de la théorie algorithmique des groupes. Dans cette section nous allons essayer de donner une idée sur le fonctionnement de cette amélioration dans le cas des problèmes de type sous-groupe, en la comparant avec la méthode de recherche arrière (un des avantages de la méthode des partitions est qu'elle permet de décrire de manière uniforme les problèmes de types classe et sous-groupe, tandis que la méthode de la section 2.3.2 (qui utilise la notion de *base*) ne le permet que pour les problèmes de type sous-groupe). Pour la description formelle et les définitions précises voir sections 2.4–2.9.

Suivant Leon, dans cette section nous appellerons la méthode de recherche arrière utilisant les partitions la *méthode des partitions (MP)*, et la méthode de recherche arrière sans partitions, décrite dans les sections 2.3.1 et 2.3.2, la *méthode de recherche arrière classique (MRAC)* ou tout simplement la *recherche arrière*.

Dans cette section, quand ce n'est pas indiqué autrement, les partitions seront ordonnées.

Soit Ω un ensemble fini et soit $n = |\Omega|$. Supposons que nous ayons un problème de type sous-groupe : nous voulons trouver $Sym(\Omega)_{\mathcal{P}}$ pour une propriété $\mathcal{P}$ de type sous-groupe. Dans la section 2.3.2 nous avons montré comment parcourir les images d'une base de $Sym(\Omega)_{\mathcal{P}}$ pour trouver tous les éléments de $Sym(\Omega)_{\mathcal{P}}$. Maintenant nous prenons une approche différente.

Toute permutation de Ω correspond à une partition discrète ordonnée de Ω : si une permutation g est

$$g = \begin{pmatrix} 1 & 2 & \cdots & n \\ a_1 & a_2 & \cdots & a_n \end{pmatrix},$$

g correspond à la partition discrète $(a_1|a_2|\cdots|a_n)$ (cf. section 1.5.1 pour cette notation des partitions ordonnées), et inversement, si nous avons une partition discrète $(a_1|a_2|\cdots|a_n)$, nous trouvons facilement le même g que ci-dessus. Donc il existe une bijection entre $Sym(\Omega)$ et l'ensemble de toutes les partitions ordonnées discrètes de Ω, et, de plus, il est facile à passer d'un ensemble à l'autre, c'est-à-dire il est facile à partir d'un élément de $Sym(\Omega)$ obtenir une partition discrète et vice versa (notez que, par contre, comme l'indique la remarque 2.3.3, dans la recherche arrière sans partitions il n'est pas évident de passer de l'ensemble de s-vecteurs à $Sym(\Omega)$). Au lieu de parcourir les éléments de $Sym(\Omega)$ en vérifiant la propriété $\mathcal{P}$ nous pouvons parcourir les partitions ordonnées discrètes. En gros, c'est ce que fait la méthode des partitions.

Nous allons l'expliquer par analogie avec la recherche arrière classique.

Dans MRAC nous avons un s-vecteur précalculé $B = (\alpha_1, \ldots, \alpha_s)$ (c'est une base de $Sym(\Omega)_{\mathcal{P}}$), et nous parcourons tous les s-vecteurs $(\beta_1, \ldots, \beta_s)$ susceptibles d'être l'image de B par une permutation g de $Sym(\Omega)_{\mathcal{P}}$. Dans MP cela se passe de façon analogue : nous avons une partition ordonnée discrète $\mathbf{\Pi}_s$ précalculée, et nous parcourons toutes les partitions $\mathbf{\Sigma}_s$ ordonnées discrètes susceptibles d'être l'image de $\mathbf{\Pi}_s$ par une permutation g de $Sym(\Omega)_{\mathcal{P}}$. Notez que, étant données deux partitions ordonnées discrètes $\mathbf{\Pi}_s$ et $\mathbf{\Sigma}_s$, il est évident de trouver une permutation g telle que ${\mathbf{\Pi}_s}^g = \mathbf{\Sigma}_s$.

Dans MRAC, pour parcourir tous les s-vecteurs, nous commençons par le 0-vecteur () et nous parcourons les solutions partielles (les h-vecteurs) $(\beta_1, \dots, \beta_h)$, $0 \leq h \leq s$, en vérifiant si $(\beta_1, \dots, \beta_h)$ satisfait la propriété intermédiaire $\mathcal{P}_h$, i.e. en vérifiant s'il est possible que dans $Sym(\Omega)_{\mathcal{P}}$ il existe un g tel que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$. Les solutions partielles de MP sont des partitions ordonnées. Dans MRAC, pour un s-vecteur $(\beta_1, \dots, \beta_s)$ nous avons une suite des solutions partielles $(), (\beta_1), (\beta_1, \beta_2), (\beta_1, \beta_2, \beta_3), \dots, (\beta_1, \dots, \beta_s)$ de plus en plus longues. De même dans MP : pour toute partition ordonnée discrète $\mathbf{\Sigma}_s$, nous avons une suite de partitions ordonnées $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3, \dots, \mathbf{\Sigma}_s$ de plus en plus fines, i.e. $\mathbf{\Sigma}_0 \geq \mathbf{\Sigma}_1 \geq \mathbf{\Sigma}_2 \geq \mathbf{\Sigma}_3 \geq \dots \geq \mathbf{\Sigma}_s$.

Dans MRAC nous précalculons une base $B = (\alpha_1, \dots, \alpha_s)$ de $Sym(\Omega)_{\mathcal{P}}$, nous pouvons donc considérer que nous avons précalculé une suite d'ensembles ordonnés $(), (\alpha_1), (\alpha_1, \alpha_2), (\alpha_1, \alpha_2, \alpha_3), \dots, (\alpha_1, \dots, \alpha_s)$ des vecteurs partiels de B. De même dans MP : nous précalculons une suite de partitions $\mathbf{\Pi}_0 \geq \mathbf{\Pi}_1 \geq \mathbf{\Pi}_2 \geq \mathbf{\Pi}_3 \geq \dots \geq \mathbf{\Pi}_s$.

Continuons de chercher des analogies entre MRAC et MP. Dans MRAC, nous parcourons les 1-vecteurs (β_1) en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1)^g = (\beta_1)$, pour ceux des (β_1) pour lesquels c'est le cas nous rallongeons (β_1) et nous parcourons tous les 2-vecteurs (β_1, β_2) en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \alpha_2)^g = (\beta_1, \beta_2)$, etc..., nous parcourons tous les h-vecteurs $(\beta_1, \dots, \beta_h)$, $1 \leq h \leq s$, en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$, etc... De même pour MP : nous parcourons les partitions ordonnées $\mathbf{\Sigma}_1$ en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que ${\mathbf{\Pi}_1}^g = \mathbf{\Sigma}_1$. Notez qu'il suffit de parcourir les partitions $\mathbf{\Sigma}_1$ telles que $\|\mathbf{\Pi}_1\| = \|\mathbf{\Sigma}_1\|$, car autrement $\mathbf{\Sigma}_1$ ne pourra être l'image de $\mathbf{\Pi}_1$ par aucune permutation $g \in Sym(\Omega)$. D'ailleurs, nous ne parcourons pas toutes les partitions $\mathbf{\Sigma}_1$ telles que $\|\mathbf{\Pi}_1\| = \|\mathbf{\Sigma}_1\|$, il suffit d'en parcourir quelques-unes (nous verrons plus loin comment le faire). Pour les partitions ordonnées $\mathbf{\Sigma}_1$ pour lesquelles il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que ${\mathbf{\Pi}_1}^g = \mathbf{\Sigma}_1$, nous construisons les partitions $\mathbf{\Sigma}_2$ et nous les parcourons (plus précisément, quelques-unes des partitions telles que $\|\mathbf{\Pi}_2\| = \|\mathbf{\Sigma}_2\|$) en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que ${\mathbf{\Pi}_2}^g = \mathbf{\Sigma}_2$, etc..., nous parcourons les partitions $\mathbf{\Sigma}_h$ en vérifiant s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que ${\mathbf{\Pi}_h}^g = \mathbf{\Sigma}_h$, etc...

Comment passer d'un niveau au niveau supérieur dans MRAC ? Par exemple, si nous avons une solution partielle $(\beta_1, \dots, \beta_h)$, comment obtenir une solution partielle plus longue $(\beta_1, \dots, \beta_h, \beta_{h+1})$? Nous pouvons voir cela comme suit. Supposons que $(\beta_1, \dots, \beta_h)$ satisfasse la propriété intermédiaire $\mathcal{P}_h$. Il peut donc exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$. Comme nous voulons trouver tous les g de cette sorte, nous divisons la recherche en n cas :

- d'abord nous essayons de trouver $g \in Sym(\Omega)_{\mathcal{P}}$ tels que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$ et ${\alpha_{h+1}}^g = \omega_1$,
- ensuite, tels que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$ et ${\alpha_{h+1}}^g = \omega_2$,
- ...
- enfin, tels que $(\alpha_1, \dots, \alpha_h)^g = (\beta_1, \dots, \beta_h)$ et ${\alpha_{h+1}}^g = \omega_n$,

où $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ et $\omega_1 < \omega_2 < \cdots < \omega_n$. Nous pouvons l'écrire plus brièvement comme suit : pour tout $\beta_{h+1} \in \Omega$, nous essayons de trouver $g \in Sym(\Omega)_{\mathcal{P}}$ tels que $(\alpha_1, \ldots, \alpha_h, \alpha_{h+1})^g = (\beta_1, \ldots, \beta_h, \beta_{h+1})$.

Voyons comment cela se passe dans MP. Nous avons une solution partielle $\mathbf{\Sigma}_h$, comment obtenir une solution partielle plus longue $\mathbf{\Sigma}_{h+1}$? C'est plus compliqué que dans MRAC. Cela se fait en deux étapes. La première est la même que dans MRAC : la division de la recherche. Supposons que $\mathbf{\Sigma}_h$ satisfasse $\mathcal{P}_h$, i.e. il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que ${\mathbf{\Pi}_h}^g = \mathbf{\Sigma}_h$. Nous choisissons un nombre $e_{h+1} \in I_n$ tel que la e_{h+1}-ième cellule $\Pi_{h,e_{h+1}}$ de $\mathbf{\Pi}_h$ contient au moins deux éléments, et nous choisissons un élément α_{h+1} de cette cellule. Comme $\mathbf{\Pi}_0, \mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_s$ sont précalculés, nous avons pu choisir $e_1, \ldots, e_s$ et $\alpha_1, \ldots, \alpha_s$ avant le parcours de l'arbre. De plus, nous avons choisi e_h, α_h, $1 \le h \le s$, au cours de calcul des $\mathbf{\Pi}_0, \mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_s$, et la partition $\mathbf{\Pi}_h$ dépend en fait de $e_1, \ldots, e_h$ et $\alpha_1, \ldots, \alpha_h$. Donc à part $\mathbf{\Pi}_h$, $0 \le h \le s$, nous précalculons aussi e_h, α_h, $1 \le h \le s$. S'il existe un ou plusieurs g de $Sym(\Omega)_{\mathcal{P}}$ tels que ${\mathbf{\Pi}_h}^g = \mathbf{\Sigma}_h$, alors ${\Pi_{h,e_{h+1}}}^g = \Sigma_{h,e_{h+1}}$. Les images possibles de α_{h+1} sont donc les éléments de $\Sigma_{h,e_{h+1}}$. Nous divisons la recherche comme suit : pour tout élément β_{h+1} de $\Sigma_{h,e_{h+1}}$ nous cherchons les $g \in Sym(\Omega)_{\mathcal{P}}$ tels que ${\mathbf{\Pi}_h}^g = \mathbf{\Sigma}_h$ et ${\alpha_{h+1}}^g = \beta_{h+1}$. D'après la propriété 1.5.10 cela revient à chercher les $g \in Sym(\Omega)_{\mathcal{P}}$ tels que $\mathcal{F}_{e_{h+1},\alpha_{h+1}}(\mathbf{\Pi}_h)^g = \mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h)$. Nous divisons donc la recherche en $\left|\Pi_{h,e_{h+1}}\right|$ cas, $\left|\Pi_{h,e_{h+1}}\right| \le n$.

La deuxième étape consiste à calculer $\mathbf{\Sigma}_{h+1}$ à partir de $\mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h)$. Cela implique les *$\mathcal{P}$-raffinements* (cf. définitions 2.4.1 et 2.4.7) avec lesquels nous allons faire maintenant brièvement connaissance.

Pour les problèmes de type sous-groupe, un *$\mathcal{P}$-raffinement* peut être défini comme une application E de $PartOrd(\Omega)$ dans $PartOrd(\Omega)$ satisfaisant deux conditions pour tous les $\mathbf{\Pi}, \mathbf{\Sigma} \in PartOrd(\Omega)$ et $g \in Sym(\Omega)$:

1. $E(\mathbf{\Pi}) \le \mathbf{\Pi}$ (c'est pourquoi nous l'appelons raffinement),

2. si $\mathcal{P}(g)$ et $\mathbf{\Pi}^g = \mathbf{\Sigma}$, alors $E(\mathbf{\Pi})^g = E(\mathbf{\Sigma})$

(cf. sections 2.4.1 et 2.4.3 pour plus de commentaires sur les $\mathcal{P}$-raffinements).

Nous avons maintenant $\mathbf{\Pi}' \overset{\text{déf}}{=} \mathcal{F}_{e_{h+1},\alpha_{h+1}}(\mathbf{\Pi}_h)$ et $\mathbf{\Sigma}' \overset{\text{déf}}{=} \mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h)$, et nous voulons trouver tous les $g \in Sym(\Omega)_{\mathcal{P}}$ tels que $\mathbf{\Pi}'^g = \mathbf{\Sigma}'$. Si nous choisissions maintenant de nouveau une cellule Π'_e de $\mathbf{\Pi}'$, comportant au moins deux éléments, et un élément $\alpha \in \Pi'_e$, alors il faudrait vérifier tous les éléments β de la cellule Σ'_e de $\mathbf{\Sigma}'$ s'ils sont les images de α par une permutation g de $Sym(\Omega)_{\mathcal{P}}$ satisfaisant $\mathbf{\Pi}'^g = \mathbf{\Sigma}'$. Supposons que pour certains $\beta \in \Sigma'_e$ aucun $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}'^g = \mathbf{\Sigma}'$ et $\alpha^g = \beta$ n'existe. Nous perdrons donc du temps en les considérant. Comment pouvons-nous l'éviter ? Comment supprimer au moins une partie de ces β inutiles ?

Nous pouvons le faire au moyen de $\mathcal{P}$-raffinements. Supposons que nous ayons un ensemble $\mathbf{R}$ de $\mathcal{P}$-raffinements. Nous essayons d'appliquer les $\mathcal{P}$-raffinements E de $\mathbf{R}$ à $\mathbf{\Pi}'$ et $\mathbf{\Sigma}'$. Supposons que nous obtenions deux partitions $\mathbf{\Pi}'' \overset{\text{déf}}{=} E(\mathbf{\Pi}')$ et $\mathbf{\Sigma}'' \overset{\text{déf}}{=} E(\mathbf{\Sigma}')$ strictement plus fines que $\mathbf{\Pi}'$ et $\mathbf{\Sigma}'$, respectivement. La condition 2 de la définition de $\mathcal{P}$-raffinement ci-dessus garantit que s'il existe $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}'^g = \mathbf{\Sigma}'$, alors $\mathbf{\Pi}''^g = \mathbf{\Sigma}''$ aussi. Avec un peu de chance $|\Sigma''_e| < |\Sigma'_e|$, nous avons donc réussi à séparer une partie des β inutiles. Sinon, $|\Sigma''_e| = |\Sigma'_e|$, mais nous avons réussi quand même à diviser une ou plusieurs autres cellules de $\mathbf{\Sigma}'$, ce qui s'avérera utile par la suite quand il faudra choisir les β suivants dans les autres cellules de $\mathbf{\Sigma}''$. Nous continuons donc d'appliquer les $\mathcal{P}$-raffinements de $\mathbf{R}$ à $\mathbf{\Pi}''$ et $\mathbf{\Sigma}''$ jusqu'à ce que nous estimons que nous ne réussirons plus à raffiner le couple courant de partitions. Alors nous posons $(\mathbf{\Pi}_{h+1}, \mathbf{\Sigma}_{h+1})$ le couple courant de partitions et nous recommençons comme c'est décrit ci-dessus (le choix de e_{h+2} et α_{h+2}, etc...).

Mais en pratique cela se passe un peu différemment : nous précalculons $\mathbf{\Pi}_0, \mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_s$. Nous commençons par la partition triviale $\mathbf{\Pi}' \overset{\text{déf}}{=} (\Omega)$ et nous appliquons des $\mathcal{P}$-raffinements $E_1, E_2, \ldots$ de $\mathbf{R}$ à $\mathbf{\Pi}'$. Si, par exemple, E_i réussit à raffiner $\mathbf{\Pi}'$, nous mémorisons E_i, nous posons $\mathbf{\Pi}'' = E_i(\mathbf{\Pi}')$ et nous commençons à appliquer les $\mathcal{P}$-raffinements de $\mathbf{R}$ à $\mathbf{\Pi}''$, etc... Quand nous estimons que nous ne pourrons plus raffiner $\mathbf{\Pi}''^{\cdots\prime}$ au moyen des $\mathcal{P}$-raffinements de $\mathbf{R}$, nous posons $\mathbf{\Pi}_0 = \mathbf{\Pi}''^{\cdots\prime}$, nous choisissons e_1 tel que $|\Pi_{0,e_1}| \geq 2$ et $\alpha_1 \in \Pi_{0,e_1}$, nous calculons $\mathbf{\Pi}' \overset{\text{déf}}{=} \mathcal{F}_{e_1,\alpha_1}(\mathbf{\Pi}_0)$ et nous essayons de le raffiner comme ci-dessus. De la même façon nous obtenons $\mathbf{\Pi}_1, e_2, \alpha_2, \mathbf{\Pi}_2, e_3, \alpha_3, \ldots$ et nous le faisons jusqu'à ce que nous obtenions une partition $\mathbf{\Pi}_s$ discrète. Cela signifie qu'avant le parcours de l'arbre nous avons déjà non seulement $\mathbf{\Pi}_h$, $0 \leq h \leq s$, e_h, α_h, $1 \leq h \leq s$, mais aussi une suite de $\mathcal{P}$-raffinements U_h, $0 \leq h \leq s$, où U_h est une composition des $\mathcal{P}$-raffinements qui ont réussi à raffiner les partitions $\mathbf{\Pi}''^{\cdots\prime}$ courantes, plus précisémant, U_h, $1 \leq h \leq s$, est tel que $\mathbf{\Pi}_h = U_h(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$, c'est-à-dire U_h est une composition des $\mathcal{P}$-raffinements qui ont raffiné $\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1})$ jusqu'à $\mathbf{\Pi}_h$, et de même, U_0 est tel que $\mathbf{\Pi}_0 = U_0((\Omega))$. Les $U_h, \mathbf{\Pi}_h$, $0 \leq h \leq s$, et les e_h, α_h, $1 \leq h \leq s$, composent une $\mathbf{R}$-*base* pour $\mathcal{P}$ (cf. définition 2.4.10). Voir section 2.9 pour la description plus précise du calcul d'une $\mathbf{R}$-base.

Au cours du parcours de l'arbre, quand nous avons une solution partielle courante $\mathbf{\Sigma}_h$ telle qu'il peut exister une permutation $g \in Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}_h{}^g = \mathbf{\Sigma}_h$, nous obtenons $\mathbf{\Sigma}_{h+1}$ comme suit : nous prenons $\beta_{h+1} \in \Sigma_{h,e_{h+1}}$ (l'image éventuel de α_{h+1} par une permutation $g \in Sym(\Omega)_{\mathcal{P}}$ satisfaisant $\mathbf{\Pi}_h{}^g = \mathbf{\Sigma}_h$), nous appliquons U_{h+1} à $\mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h)$ et nous posons $\mathbf{\Sigma}_{h+1} = U_{h+1}(\mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h))$.

Nous vérifions si $\mathbf{\Sigma}_{h+1}$ satisfait $\mathcal{P}_{h+1}$, c'est-à-dire s'il peut exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}_{h+1}{}^g = \mathbf{\Sigma}_{h+1}$ (cf. proposition 2.4.13 et section 2.5 pour les propriétés intermédiaires $\mathcal{P}_h$ dans MP, il est évident que $\mathcal{P}_h$ doit au moins inclure la vérification $\|\mathbf{\Pi}_h\| = \|\mathbf{\Sigma}_h\|$). Si c'est le cas, nous recommençons la procédure décrite avec $\mathbf{\Sigma}_{h+1}$ la solution partielle courante. Sinon, il faut revenir en arrière, car il ne peut pas exister $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}_{h+1}{}^g = \mathbf{\Sigma}_{h+1}$ et donc (cf. la condition 2 de la définition d'un $\mathcal{P}$-raffinement ci-dessus) tel que $\mathcal{F}_{e_{h+1},\alpha_{h+1}}(\mathbf{\Pi}_h)^g = \mathcal{F}_{e_{h+1},\beta_{h+1}}(\mathbf{\Sigma}_h)$. Il faut choisir un autre β_{h+1} dans $\Sigma_{h,e_{h+1}}$ et recommencer.

De cette manière, en commençant par la partition triviale (Ω), nous parcourons toutes les solutions partielles $\mathbf{\Sigma}_h$ satisfaisant les propriétés intermédiaires $\mathcal{P}_h$ en essayant d'obtenir les partitions discrètes à partir desquelles nous pouvons retrouver les permutations de $Sym(\Omega)_{\mathcal{P}}$.

Comme nous voyons, MP est bien plus compliqué que MRAC, mais elle permet au moyen de $\mathcal{P}$-raffinements de diminuer fortement l'espace de recherche des solutions.

Notez enfin que le s-vecteur $(\alpha_1, \ldots, \alpha_s)$ d'une **R**-base pour $\mathcal{P}$ est une base de $Sym(\Omega)_{\mathcal{P}}$ (cf. proposition 2.4.13). C'est un lien qui unit MP et MRAC. En utilisant ce lien nous pouvons décrire l'arbre de la méthode des partitions d'une manière semblable à l'arbre de recherche arrière décrit dans la section 2.3.1.1. Nous le décrivons dans la section 2.5.

2.4 Raffinements et R-base

Soit Ω un ensemble fini et soit n le cardinal de Ω. Dans cette section nous donnons les définitions d'un *raffinement* et d'une **R**-*base* (généralisation de la notion de *base* introduite dans la définition 1.7.1, p. 40) et décrivons leurs propriétés qui permettent d'effectuer la recherche arrière dans $Sym(\Omega)$.

Nous avons vu dans la section 2.3.3 que la notion de raffinement des partitions ordonnées est très importante pour la méthode des partitions. Nous définissons maintenant cette notion de façon formelle.

2.4.1 Raffinements élémentaires

Définition 2.4.1 *[Leo91, Déf.8] et [Leo97, Déf.9] Soit $\mathcal{P}$ une propriété sur Ω. Un* $\mathcal{P}$-raffinement élémentaire *est une paire $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ d'applications de $PartOrd(\Omega)$ dans $PartOrd(\Omega)$ telle que pour tous les $\mathbf{\Pi}, \mathbf{\Sigma} \in PartOrd(\Omega)$ et pour tout $g \in Sym(\Omega)$, on a :*

1. $E_{\mathsf{G}}(\mathbf{\Pi}) \leq \mathbf{\Pi}$ *et* $E_{\mathsf{D}}(\mathbf{\Pi}) \leq \mathbf{\Pi}$.
2. $|E_{\mathsf{G}}(\mathbf{\Pi})| \leq |\mathbf{\Pi}| + 1$ *et* $|E_{\mathsf{D}}(\mathbf{\Pi})| \leq |\mathbf{\Pi}| + 1$.
3. *Si* $\mathcal{P}(g)$ *et* $\mathbf{\Pi}^g = \mathbf{\Sigma}$, *alors* $E_{\mathsf{G}}(\mathbf{\Pi})^g = E_{\mathsf{D}}(\mathbf{\Sigma})$.
4. *Il existe un entier i, ne dépendant que de E, tel que*
   - $E_{\mathsf{G}}(\mathbf{\Pi})_\infty \subseteq \Pi_i$ *pour tout* $\mathbf{\Pi}$ *tel que la partition* $E_{\mathsf{G}}(\mathbf{\Pi})$ *est strictement plus fine que* $\mathbf{\Pi}$,
   - $E_{\mathsf{D}}(\mathbf{\Pi})_\infty \subseteq \Pi_i$ *pour tout* $\mathbf{\Pi}$ *tel que la partition* $E_{\mathsf{D}}(\mathbf{\Pi})$ *est strictement plus fine que* $\mathbf{\Pi}$.

Nous dirons simplement raffinement élémentaire *si la propriété $\mathcal{P}$ n'est pas importante ou est évidente dans le contexte.*

Les indices "G" et "D" sont des symboles formels choisis pour indiquer gauche et droite, correspondant à l'apparition de E_{G} et E_{D} du coté gauche et du coté droit, respectivement, de l'équation dans la condition 3.

Notez que les conditions 1 et 2 impliquent que soit $E_{\mathsf{G}}(\mathbf{\Pi}) = \mathbf{\Pi}$, soit $|E_{\mathsf{G}}(\mathbf{\Pi})| = |\mathbf{\Pi}| + 1$ (de même pour E_{D}), c'est-à-dire chaque composante d'un raffinement élémentaire soit laisse une partition $\mathbf{\Pi}$ inchangé, soit divise exactement *une* cellule de $\mathbf{\Pi}$ en deux parties non vides. C'est pourquoi nous utilisons le terme “élémentaire”.

Les conditions 1 et 3 de la définition 2.4.1 sont les plus importantes.

La condition 1 montre que chaque composante d'un raffinement élémentaire doit raffiner (ou laisser inchangées) toutes les partitions ordonnées de Ω, d'où le terme “raffinement”.

La condition 3 permet d'acquérir de l'information supplémentaire sur l'action de g. Supposons que nous ayons deux partitions $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ et que nous voulions trouver une permutation $g \in Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}^g = \mathbf{\Sigma}$ (ou bien montrer qu'une telle permutation n'existe pas). Si $\|\mathbf{\Pi}\| \neq \|\mathbf{\Sigma}\|$, g ne peut pas exister. Si $\|\mathbf{\Pi}\| = \|\mathbf{\Sigma}\|$, nous appliquons des raffinements élémentaires à $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ (c'est-à-dire nous appliquons la première composante d'un raffinement élémentaire à $\mathbf{\Pi}$ et la deuxième à $\mathbf{\Sigma}$) en essayant de raffiner $\mathbf{\Pi}$ et $\mathbf{\Sigma}$. Si nous y réussissons, la condition 3 garantit que nous avons acquis de l'information sur l'action de g. En effet, supposons que nous trouvions un raffinement élémentaire $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ tel que $E_{\mathsf{G}}(\mathbf{\Pi}) < \mathbf{\Pi}$ ou $E_{\mathsf{D}}(\mathbf{\Sigma}) < \mathbf{\Sigma}$. Si $\|E_{\mathsf{G}}(\mathbf{\Pi})\| \neq \|E_{\mathsf{D}}(\mathbf{\Sigma})\|$, alors g satisfaisant $E_{\mathsf{G}}(\mathbf{\Pi})^g = E_{\mathsf{D}}(\mathbf{\Sigma})$ ne peut pas exister et donc d'après la condition 3, $g \in Sym(\Omega)_{\mathcal{P}}$ satisfaisant $\mathbf{\Pi}^g = \mathbf{\Sigma}$ ne peut pas exister. Supposons que $\|E_{\mathsf{G}}(\mathbf{\Pi})\| = \|E_{\mathsf{D}}(\mathbf{\Sigma})\|$. La condition 4 montre qu'il existe i tel que E_{G} et E_{D} divisent la i-ième cellule de $\mathbf{\Pi}$ et $\mathbf{\Sigma}$, respectivement, en deux parties. S'il existe une permutation $g \in Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}^g = \mathbf{\Sigma}$, alors ${\Pi_i}^g = \Sigma_i$. Après avoir appliqué le raffinement E, nous avons appris plus : nous savons maintenant que ${E_{\mathsf{G}}(\mathbf{\Pi})_i}^g = E_{\mathsf{D}}(\mathbf{\Sigma})_i$ et ${E_{\mathsf{G}}(\mathbf{\Pi})_\infty}^g = E_{\mathsf{D}}(\mathbf{\Sigma})_\infty$, où $E_{\mathsf{G}}(\mathbf{\Pi})_i \cup E_{\mathsf{G}}(\mathbf{\Pi})_\infty = \Pi_i$, $\emptyset \subsetneq E_{\mathsf{G}}(\mathbf{\Pi})_i \subsetneq \Pi_i$, et $E_{\mathsf{D}}(\mathbf{\Sigma})_i \cup E_{\mathsf{D}}(\mathbf{\Sigma})_\infty = \Sigma_i$, $\emptyset \subsetneq E_{\mathsf{D}}(\mathbf{\Sigma})_i \subsetneq \Sigma_i$. Avant l'application de E nous savions que l'image d'un point γ de Π_i par g pouvait être n'importe quel point de Σ_i, il y avait donc $|\Sigma_i|$ images possibles. Après l'application de E nous apprenons que l'image de γ peut être un point de $E_{\mathsf{D}}(\mathbf{\Sigma})_i$ ou $E_{\mathsf{D}}(\mathbf{\Sigma})_\infty$ selon que γ appartient à $E_{\mathsf{G}}(\mathbf{\Pi})_i$ ou $E_{\mathsf{G}}(\mathbf{\Pi})_\infty$. Dans les deux cas l'ensemble des images possibles de γ diminue. Nous avons obtenu donc de l'information supplémentaire sur g. L'ensemble des g possibles a diminué. De cette manière nous pouvons diminuer l'espace de recherche de la méthode des partitions, comme c'est expliqué dans la section 2.3.3.

La condition 3 peut s'écrire autrement comme suit : $E_{\mathsf{G}}(\mathbf{\Pi})^g = E_{\mathsf{D}}(\mathbf{\Pi}^g)$ pour tout $\mathbf{\Pi} \in PartOrd(\Omega)$ et tout $g \in Sym(\Omega)$ satisfaisant $\mathcal{P}(g)$.

La condition 4 n'est pas une restriction importante. Elle signifie que *pour toute* partition $\mathbf{\Pi} \in PartOrd(\Omega)$ et *pour tout* $\mathsf{z} \in \{\mathsf{G}, \mathsf{D}\}$ l'application E_{z} divise la i-ième cellule de $\mathbf{\Pi}$ (ou laisse $\mathbf{\Pi}$ inchangée), ce i ne dépendant pas de $\mathbf{\Pi}$ et de z. Si une application E satisfait toutes les conditions de la définition 2.4.1 sauf la condition 4, nous pouvons le remplacer par n raffinements élémentaires $E_1, \ldots, E_n$ définis comme suit : pour tout i, $1 \leq i \leq n$, $E_i = (E_{i,\mathsf{G}}, E_{i,\mathsf{D}})$ et

$$E_{i,\mathsf{G}}(\mathbf{\Pi}) = \begin{cases} E_{\mathsf{G}}(\mathbf{\Pi}) & \text{si } E_{\mathsf{G}}(\mathbf{\Pi})_\infty \subseteq \Pi_i, \\ \mathbf{\Pi} & \text{sinon,} \end{cases}$$

$E_{i,\mathsf{D}}$ étant défini de façon analogue. C'est-à-dire $E_{i,\mathsf{G}}$ (respectivement $E_{i,\mathsf{D}}$) divise la i-ième cellule de $\mathbf{\Pi}$ en deux parties si E_{G} (respectivement E_{D}) le fait, sinon il laisse $\mathbf{\Pi}$ inchangée ; donc $E_{i,\mathsf{G}}$ et $E_{i,\mathsf{D}}$ ne divisent que la i-ième cellule des partitions, pas les autres, i.e. $E_i = (E_{i,\mathsf{G}}, E_{i,\mathsf{D}})$ satisfait la condition 4 en même temps que les autres conditions, nous remplaçons donc E par une suite de $\mathcal{P}$-raffinements élémentaires.

Tous nos raffinements (et ceux de Leon) satisfont cette condition, et nous la mettons dans la définition générale d'un raffinement élémentaire parce qu'elle permet d'augmenter l'efficacité de l'algorithme dans le cas général : quand nous avons deux partitions $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ et nous espérons qu'il existe une permutation g de $Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}^g = \mathbf{\Sigma}$, nous appliquons un raffinement E, et, en vertu de la condition 4, au lieu de vérifier que $\|E_{\mathsf{G}}(\mathbf{\Pi})\| = \|E_{\mathsf{D}}(\mathbf{\Sigma})\|$ (i.e. vérifier l'égalité des cardinaux de toutes les cellules correspondantes de $E_{\mathsf{G}}(\mathbf{\Pi})$ et $E_{\mathsf{D}}(\mathbf{\Sigma})$), il suffira de vérifier que $|E_{\mathsf{G}}(\mathbf{\Pi})_{\infty}| = |E_{\mathsf{D}}(\mathbf{\Sigma})_{\infty}|$ (i.e. ne vérifier que l'égalité des cardinaux des dernières cellules de $E_{\mathsf{G}}(\mathbf{\Pi})$ et $E_{\mathsf{D}}(\mathbf{\Sigma})$) ; pour les détails voir la proposition 2.10.3 et la proposition 2.10.6.

Nous donnons un résultat pour une conjonction de deux propriétés.

Proposition 2.4.2 *Si $\mathcal{P}$ et $\mathcal{Q}$ sont deux propriétés sur Ω et si E est un $\mathcal{P}$-raffinement élémentaire, alors E est un $\mathcal{P} \wedge \mathcal{Q}$-raffinement élémentaire.*

Preuve. La propriété $\mathcal{P}$ n'intervient que dans la condition 3 de la définition 2.4.1. Supposons que cette condition soit satisfaite pour $\mathcal{P}$. Il faut montrer qu'elle est satisfaite pour $\mathcal{P} \wedge \mathcal{Q}$, c'est-à-dire que pour tout $g \in Sym(\Omega)_{\mathcal{P}\wedge\mathcal{Q}}$ $(\mathbf{\Pi}^g = \mathbf{\Sigma} \Rightarrow E_{\mathsf{G}}(\mathbf{\Pi})^g = E_{\mathsf{D}}(\mathbf{\Sigma}))$. Mais comme c'est vrai pour tout $g \in Sym(\Omega)_{\mathcal{P}}$ et $Sym(\Omega)_{\mathcal{P}\wedge\mathcal{Q}} \subset Sym(\Omega)_{\mathcal{P}}$, c'est vrai aussi pour tout $g \in Sym(\Omega)_{\mathcal{P}\wedge\mathcal{Q}}$. □

Corollaire 2.4.3 *Si $\mathcal{P}$ et $\mathcal{Q}$ sont deux propriétés sur Ω, $\mathbf{R}_1$ et $\mathbf{R}_2$ sont des ensembles de $\mathcal{P}$- et $\mathcal{Q}$-raffinements élémentaires, respectivement, alors $\mathbf{R}_1 \cup \mathbf{R}_2$ est un ensemble de $\mathcal{P} \wedge \mathcal{Q}$-raffinements élémentaires.*

Preuve. D'après la proposition 2.4.2, chaque raffinement élémentaire de $\mathbf{R}_1 \cup \mathbf{R}_2$ est un $\mathcal{P} \wedge \mathcal{Q}$-raffinement élémentaire. □

2.4.2 Raffinements élémentaires symétriques

Définition 2.4.4 *[Leo91, Déf.10] et [Leo97, Déf.10] Soit $\mathcal{P}$ une propriété sur Ω. Un $\mathcal{P}$-raffinement élémentaire $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ est* symétrique *si $E_{\mathsf{G}} = E_{\mathsf{D}}$.*

Comme les problèmes de type sous-groupe peuvent être considérés comme des cas particuliers des problèmes de type classe, il est naturel de s'attendre à des simplifications dans ce cas. La proposition suivante montre la première de ces simplifications : si notre problème est de type sous-groupe, nos raffinements pour le résoudre sont symétriques.

Proposition 2.4.5 *[Leo91, Lemme 6] et [Leo97, Prop.2] Si $\mathcal{P}(id)$ est satisfait, alors tout $\mathcal{P}$-raffinement élémentaire est symétrique. En particulier, ceci est vérifié si $Sym(\Omega)_{\mathcal{P}}$ est un sous-groupe de $Sym(\Omega)$.*

Preuve. Soit $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ un $\mathcal{P}$-raffinement élémentaire. Comme $\mathbf{\Pi}^{id} = \mathbf{\Pi}$ pour tout $\mathbf{\Pi} \in \mathit{PartOrd}(\Omega)$, la condition 3 de la définition 2.4.1 implique que $E_{\mathsf{G}}(\mathbf{\Pi})^{id} = E_{\mathsf{D}}(\mathbf{\Pi})$, d'où $E_{\mathsf{G}} = E_{\mathsf{D}}$. □

Dans ce cas nous simplifions les notations.

Notation 2.4.6 [Leo91, Déf.11] et [Leo97, Déf.10] Si $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ est un $\mathcal{P}$-raffinement élémentaire symétrique, nous écrivons $E(\mathbf{\Pi})$ au lieu de $E_{\mathsf{G}}(\mathbf{\Pi})$ ou $E_{\mathsf{D}}(\mathbf{\Pi})$. C'est-à-dire nous considérons E comme une application de $\mathit{PartOrd}(\Omega)$ dans $\mathit{PartOrd}(\Omega)$ plutôt qu'une paire de telles applications.

2.4.3 Raffinements

Définition 2.4.7 *[Leo91, Déf.12] Si $\mathcal{P}$ est une propriété sur Ω, un* $\mathcal{P}$-raffinement *est une suite fini $(E_l, E_{l-1}, \ldots, E_1)$ de $\mathcal{P}$-raffinements élémentaires, où $l \geq 0$.*

Si $R = (E_l, E_{l-1}, \ldots, E_1)$ est un $\mathcal{P}$-raffinement, nous pouvons considérer une paire $(R_{\mathsf{G}}, R_{\mathsf{D}})$ d'applications de $\mathit{PartOrd}(\Omega)$ dans $\mathit{PartOrd}(\Omega)$ définies comme suit :

$$\begin{aligned} R_{\mathsf{G}}(\mathbf{\Pi}) &= E_{l,\mathsf{G}}(E_{l-1,\mathsf{G}}(\cdots E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi}))\cdots)) \text{ et} \\ R_{\mathsf{D}}(\mathbf{\Pi}) &= E_{l,\mathsf{D}}(E_{l-1,\mathsf{D}}(\cdots E_{2,\mathsf{D}}(E_{1,\mathsf{D}}(\mathbf{\Pi}))\cdots)) \end{aligned}$$

(c'est la raison pour laquelle nous numérotons les $\mathcal{P}$-raffinements élémentaires dans un $\mathcal{P}$-raffinement de droite à gauche — parce que la composition des applications se fait de droite à gauche). Si $l = 0$, R_{G} et R_{D} sont des applications identiques : $R_{\mathsf{G}}(\mathbf{\Pi}) = \mathbf{\Pi}$ et $R_{\mathsf{D}}(\mathbf{\Pi}) = \mathbf{\Pi}$ pour tout $\mathbf{\Pi} \in \mathit{PartOrd}(\Omega)$.

En fait, il est pratique de considérer R comme une paire $(R_{\mathsf{G}}, R_{\mathsf{D}})$ d'applications de $\mathit{PartOrd}(\Omega)$ dans $\mathit{PartOrd}(\Omega)$. En tant que telle, R satisfait les conditions 1 et 3 de la définition 2.4.1 (en remplaçant E_{G} et E_{D} par R_{G} et R_{D}, respectivement). En effet, d'après la condition 1 de la définition 2.4.1, nous avons

$$\begin{aligned} R_{\mathsf{G}}(\mathbf{\Pi}) &= E_{l,\mathsf{G}}(E_{l-1,\mathsf{G}}(\cdots E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi}))\cdots)) \\ &\leq E_{l-1,\mathsf{G}}(\cdots E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi}))\cdots) \\ &\leq \cdots \\ &\leq E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi})) \\ &\leq E_{1,\mathsf{G}}(\mathbf{\Pi}) \\ &\leq \mathbf{\Pi}, \end{aligned}$$

et de même pour $R_{\mathsf{D}}(\mathbf{\Pi})$, donc $(R_{\mathsf{G}}, R_{\mathsf{D}})$ satisfait la condition 1. De plus, si $\mathcal{P}(g)$ et $\mathbf{\Pi}^g = \mathbf{\Sigma}$, en tenant compte de la condition 3 de la définition 2.4.1 nous trouvons

$$\begin{aligned} E_{1,\mathsf{G}}(\mathbf{\Pi})^g &= E_{1,\mathsf{D}}(\mathbf{\Sigma}), \\ E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi}))^g &= E_{2,\mathsf{D}}(E_{1,\mathsf{D}}(\mathbf{\Sigma})), \\ &\cdots \\ E_{l,\mathsf{G}}(E_{l-1,\mathsf{G}}(\cdots E_{2,\mathsf{G}}(E_{1,\mathsf{G}}(\mathbf{\Pi}))\cdots))^g &= E_{l,\mathsf{D}}(E_{l-1,\mathsf{D}}(\cdots E_{2,\mathsf{D}}(E_{1,\mathsf{D}}(\mathbf{\Sigma}))\cdots)), \end{aligned}$$

donc $R_{\mathsf{G}}(\mathbf{\Pi})^g = R_{\mathsf{D}}(\mathbf{\Sigma})$ et la condition 3 est satisfaite pour $(R_{\mathsf{G}}, R_{\mathsf{D}})$.

Donc un raffinement élémentaire est la plus petite brique composant un raffinement. C'est le plus petit pas quand on applique un raffinement à une partition.

Plutôt que d'écrire $R = (E_l, E_{l-1}, \ldots, E_1)$, nous écrirons $R = E_l E_{l-1} \cdots E_1$ pour suggérer une composition (de droite à gauche) de $\mathcal{P}$-raffinements élémentaires, et nous considérerons un $\mathcal{P}$-raffinement élémentaire E comme un $\mathcal{P}$-raffinement tandis que du point de vue technique la suite (E) (et non E lui-même) est un $\mathcal{P}$-raffinement.

Notez que si $R = E_l E_{l-1} \cdots E_1$ et $Q = F_r F_{r-1} \cdots F_1$ sont des $\mathcal{P}$-raffinements, leur composition $RQ = E_l E_{l-1} \cdots E_1 F_r F_{r-1} \cdots F_1$ est aussi un $\mathcal{P}$-raffinement.

Notation 2.4.8 [Leo91, Déf.22] Si $\mathbf{R}$ est un ensemble de $\mathcal{P}$-raffinements élémentaires, alors $\mathbf{R}^*$ sera l'ensemble de $\mathcal{P}$-raffinements dont chaque composante élémentaire appartient à $\mathbf{R}$, c'est-à-dire

$$\mathbf{R}^* = \{(E_l, E_{l-1}, \ldots, E_1) \mid l \geq 0, E_i \in \mathbf{R}\ \forall i, 1 \leq i \leq l\}.$$

Les raffinements sont exactement ce dont nous avons besoin dans la méthode des partitions décrite dans la section 2.3.3.

Nous donnons encore une définition.

Définition 2.4.9 *[Leo91, Déf.25] et [Leo97, Déf.11] Soit* $\mathbf{R}$ *un ensemble de* $\mathcal{P}$*-raffinements élémentaires et soit* $\mathbf{\Pi}$ *une partition ordonnée de* Ω*. Nous dirons que* $\mathbf{\Pi}$ *est* $\mathbf{R}_{\mathsf{G}}$-irréductible *("irreducible" en anglais) si* $E_{\mathsf{G}}(\mathbf{\Pi}) = \mathbf{\Pi}$ *pour tout* $E \in \mathbf{R}$*, et que* $\mathbf{\Pi}$ *est* $\mathbf{R}_{\mathsf{G}}$-réductible *sinon. Nous définissons la* $\mathbf{R}_{\mathsf{D}}$*-irréductibilité et* $\mathbf{R}_{\mathsf{D}}$*-réductibilité de la même manière. Pour les* $\mathcal{P}$*-raffinements élémentaires symétriques, nous utilisons les termes* $\mathbf{R}$*-irréductible et* $\mathbf{R}$*-réductible.*

2.4.4 R-base

Rappelons que Ω est un ensemble fini et n est son cardinal.

Nous avons vu dans la section 2.3.3 qu'en calculant une base $(\alpha_1, \ldots, \alpha_s)$ de $Sym(\Omega)_{\mathcal{P}}$, nous précalculons des suites de $\mathcal{P}$-raffinements $(U_0, \ldots, U_s)$ et de partitions $(\mathbf{\Pi}_0, \ldots, \mathbf{\Pi}_s)$. Nous les appelons une $\mathbf{R}$-base. Plus précisément :

Définition 2.4.10 *[Leo91, Déf.27] et [Leo97, Déf.12] Soient* $\mathcal{P}$ *une propriété sur* Ω *et* $\mathbf{R}$ *un ensemble de* $\mathcal{P}$*-raffinements élémentaires. Une* $\mathbf{R}$-base *pour* $\mathcal{P}$ *est une paire* $\mathbf{B} \overset{\text{déf}}{=} (\mathbf{U}, \mathbf{T})$ *de suites :*

- *une suite* $\mathbf{U} = (U_0, \ldots, U_s)$ *de* $\mathcal{P}$*-raffinements de* $\mathbf{R}^*$*, et*
- *une suite* $\mathbf{T} = (T_1, \ldots, T_s)$ *d'applications de* $PartOrd(\Omega)$ *dans* $PartOrd(\Omega)$ *appartenant à* $\{\mathcal{F}_{j,\gamma} \mid 1 \leq j \leq n, \gamma \in \Omega\}$ *(soit* $(e_1, \ldots, e_s)$ *et* $(\alpha_1, \ldots, \alpha_s)$ *tels que* $\mathbf{T} = (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s})$*),*

telle que ces deux suites et la suite de partitions $(\mathbf{\Pi}_0, \ldots, \mathbf{\Pi}_s)$ *définie par*

$$\mathbf{\Pi}_0 = U_{0,\mathsf{G}}((\Omega)) \text{ et } \mathbf{\Pi}_h = U_{h,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1})) \text{ pour } h = 1, \ldots, s \tag{2.7}$$

satisfont les conditions suivantes :

1. $\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}) < \mathbf{\Pi}_{h-1}$ *pour tout* h, $1 \leq h \leq s$.
2. $\mathbf{\Pi}_s$ *est une partition discrète.*

Rappelons que $\mathcal{F}_{j,\gamma}$ est défini dans la notation 1.5.8, p. 29.
Notez que la condition 1 est équivalente à la condition suivante :

$$\alpha_h \in \Pi_{h-1,e_h} \text{ et } |\Pi_{h-1,e_h}| > 1 \text{ pour tout } h,\ 1 \leq h \leq s.$$

Nous noterons $\mathbf{\Pi}_{-1} = (\Omega)$.

Remarque 2.4.11 Par la suite nous supposons que, étant donnée une **R**-base **B**, les quantités suivantes de la définition 2.4.10 sont définies et peuvent être référencées commodément : s, $(U_0, \ldots, U_s)$, $B \overset{\text{déf}}{=} (\alpha_1, \ldots, \alpha_s)$, $(e_1, \ldots, e_s)$ et $(\mathbf{\Pi}_0, \ldots, \mathbf{\Pi}_s)$.

Nous introduisons un ordre sur les éléments de Ω, associé à une **R**-base **B**.

Notation 2.4.12 [Leo91, Déf.28] Soit $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ une **R**-base pour une propriété $\mathcal{P}$ sur Ω (notez qu'elle définit les $\alpha_1, \ldots, \alpha_s$). Alors $\prec$ est un ordre de Ω défini comme suit :

(a) $\alpha_1 \prec \cdots \prec \alpha_s$,

(b) les $\alpha_1, \ldots, \alpha_s$ sont plus petits (selon l'ordre $\prec$) que les éléments de $\Omega \setminus \{\alpha_1, \ldots, \alpha_s\}$,

(c) les éléments de $\Omega \setminus \{\alpha_1, \ldots, \alpha_s\}$ sont ordonnés suivant un ordre quelconque fixé.

2.4.5 Une proposition importante pour la méthode des partitions

Supposons que nous ayons une **R**-base $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ pour une propriété $\mathcal{P}$ sur Ω avec les quantités associées (cf. remarque 2.4.11). Supposons que nous ayons une suite $(\beta_1, \ldots, \beta_s)$ d'éléments de Ω. Comment trouver les permutations $g \in Sym(\Omega)_{\mathcal{P}}$ telles que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$? La proposition suivante prouve qu'il existe au plus une telle permutation, et montre explicitement comment vérifier si elle existe, et si c'est le cas, comment la trouver.

Proposition 2.4.13 *[Leo91, Prop. 7, p. 551] Soit* $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ *une* **R***-base pour une propriété* $\mathcal{P}$ *sur* Ω*, soit* $(\beta_1, \ldots, \beta_s)$ *une suite de points de* Ω.

(a) Si $\mathcal{P}$ *est de type sous-groupe, alors* $(\alpha_1, \ldots, \alpha_s)$ *est une base de* $Sym(\Omega)_{\mathcal{P}}$*. Si* $\mathcal{P}$ *est de type classe, alors* $(\alpha_1, \ldots, \alpha_s)$ *est une base de* $Sym(\Omega)_{\mathcal{P}_\mathsf{G}}$.

(b) Il existe au plus un élément g de $Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \dots, \alpha_s)^g = (\beta_1, \dots, \beta_s)$.

(c) Si g existe dans (b), et si la suite $(\mathbf{\Sigma}_0, \dots, \mathbf{\Sigma}_s)$ de partitions ordonnées de Ω est définie par

$$\mathbf{\Sigma}_0 = U_{0,\mathsf{D}}((\Omega)) \text{ et } \mathbf{\Sigma}_h = U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})) \text{ pour } h = 1, \dots, s, \tag{2.8}$$

alors

(i) $\mathbf{\Pi}_h{}^g = \mathbf{\Sigma}_h$ pour tout h, $0 \leq h \leq s$,

(ii) $\beta_h \in \Sigma_{h-1,e_h}$ pour tout h, $1 \leq h \leq s$,

(iii) $\|\mathbf{\Pi}_h\| = \|\mathbf{\Sigma}_h\|$ pour tout h, $0 \leq h \leq s$,

(iv) pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$ (cf. remarque 2.2.3, p. 55), on a fix $(\mathbf{\Sigma}_h) \in$ fix $(\mathbf{\Pi}_h)^G$ pour tout h, $0 \leq h \leq s$,

(v) $\mathbf{\Sigma}_s$ est discrète, et g est déterminé de manière unique par l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$.

Notez que la proposition 2.4.13 est constructive : étant donné $(\beta_1, \dots, \beta_s)$, elle suggère un algorithme pour déterminer s'il existe $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \dots, \alpha_s)^g = (\beta_1, \dots, \beta_s)$ et, si c'est le cas, pour calculer g explicitement même sans un ensemble fort de générateurs de $Sym(\Omega)_{\mathcal{P}}$. Tout simplement nous faisons les calculs de (2.8). Si $\mathbf{\Sigma}_s$ s'avère être non discrète, alors g n'existe pas, sinon l'équation $\mathbf{\Pi}_s{}^x = \mathbf{\Sigma}_s$ détermine une unique permutation x, et g existe (et est égale à x) si et seulement si $\mathcal{P}(x)$ est satisfaite.

Corollaire 2.4.14

$$|Sym(\Omega)_{\mathcal{P}}| \leq \prod_{h=1}^{s} |\Pi_{h-1,e_h}| \,.$$

Corollaire 2.4.15 *Si $\mathcal{P}$ est de type sous-groupe (respectivement, de type classe), alors $\Delta^{(h)} \subseteq \Pi_{h-1,e_h}$ pour tout h, $1 \leq h \leq s$, où $\Delta^{(h)}$ est la h-ième orbite de base de $Sym(\Omega)_{\mathcal{P}}$ (respectivement, de $Sym(\Omega)_{\mathcal{P}_\mathsf{G}}$) correspondant à la base $(\alpha_1, \dots, \alpha_s)$.*

2.5 Algorithme général

Soit Ω un ensemble fini, n son cardinal et $\mathcal{P}$ une propriété sur Ω.

Dans cette section nous présentons la méthode des partitions pour calculer $Sym(\Omega)_{\mathcal{P}}$, en supposant que nous connaissons déjà une $\mathbf{R}$-base $\mathbf{B} = ((U_0, \dots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \dots, \mathcal{F}_{e_s,\alpha_s}))$ pour $\mathcal{P}$. Nous présenterons les techniques pour construire une $\mathbf{R}$-base dans la section 2.9.

Pour l'instant nous ne posons aucune restriction sur la propriété $\mathcal{P}$, elle peut être de n'importe quel type décrit dans la section 2.2, l'algorithme de cette section marchera pour toute propriété. Dans les sections suivantes nous décrirons les modifications et les améliorations de la méthode des partitions pour les problèmes de type sous-groupe et de type classe.

La méthode des partitions a été présentée déjà dans les grandes lignes dans la section 2.3.3. Nous allons donner maintenant une description formelle.

La proposition 2.4.13 montre comment, étant données une $\mathbf{R}$-base $\mathbf{B} = (\mathbf{U}, \mathbf{T})$ et une suite $(\beta_1, \ldots, \beta_s)$ d'éléments de Ω, il est possible de déterminer s'il existe $g \in \mathit{Sym}(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$, et de le trouver si c'est le cas. Toutefois, pour concevoir un efficace algorithme de recherche arrière, nous devons pouvoir déterminer de manière efficace qu'un i-vecteur $(\beta_1, \ldots, \beta_i)$, $0 \leq i < s$, ne peut pas être étendu à un s-vecteur $(\beta_1, \ldots, \beta_i, \beta_{i+1}, \ldots, \beta_s)$ de $(\alpha_1, \ldots, \alpha_i, \alpha_{i+1}, \ldots, \alpha_s)^{\mathit{Sym}(\Omega)_{\mathcal{P}}}$, autrement dit que $(\beta_1, \ldots, \beta_i) \notin (\alpha_1, \ldots, \alpha_i)^{\mathit{Sym}(\Omega)_{\mathcal{P}}}$. En général, nous ne pouvons pas espérer exclure tous les i-vecteurs $(\beta_1, \ldots, \beta_i)$ n'ayant pas d'extensions, mais l'efficacité de la technique de recherche arrière dépend de notre capacité d'en éliminer un grand pourcentage même si i est bien plus petit que s.

Supposons que nous ayons un i-vecteur $(\beta_1, \ldots, \beta_i)$, $0 \leq i \leq s$. Comment déterminer si $(\beta_1, \ldots, \beta_i) \notin (\alpha_1, \ldots, \alpha_i)^{\mathit{Sym}(\Omega)_{\mathcal{P}}}$? Nous pouvons faire comme suit. Nous calculons les partitions ordonnées $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_i$ en utilisant (2.8). Après avoir calculé $\mathbf{\Sigma}_{h-1}$, $1 \leq h \leq i$, nous nous assurons que $\beta_h \in \Sigma_{h-1,e_h}$. Si ce n'est pas le cas, nous arrêtons le calcul immédiatement, car la proposition 2.4.13(cii) nous garantit que $(\beta_1, \ldots, \beta_h) \notin (\alpha_1, \ldots, \alpha_h)^{\mathit{Sym}(\Omega)_{\mathcal{P}}}$ et donc $(\beta_1, \ldots, \beta_i) \notin (\alpha_1, \ldots, \alpha_i)^{\mathit{Sym}(\Omega)_{\mathcal{P}}}$. Si $\beta_h \in \Sigma_{h-1,e_h}$, nous calculons $\mathbf{\Sigma}_h$ selon (2.8), et nous vérifions si $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$. Si ce n'est pas le cas, nous arrêtons le calcul immédiatement à cause de la proposition 2.4.13(ciii) comme ci-dessus. Et enfin, pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$, nous vérifions si $\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ (cf. section 3.6 pour la description d'une procédure pour le faire si $G = \mathit{Sym}_q(\Omega)$). Si $\mathit{fix}\,(\mathbf{\Sigma}_h) \notin \mathit{fix}\,(\mathbf{\Pi}_h)^G$, nous arrêtons le calcul comme précédemment à cause de la proposition 2.4.13(civ).

2.5.1 L'arbre de la méthode des partitions

Nous sommes prêts à formaliser la procédure de recherche arrière. Compte tenu de la proposition 2.4.13, à tout ensemble $\mathbf{R}$ de $\mathcal{P}$-raffinements élémentaires et à toute $\mathbf{R}$-base $\mathbf{B} = (\mathbf{U}, \mathbf{T})$, nous associons l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$, appelé l'*arbre de la méthode des partitions* ou tout simplement l'*arbre de recherche arrière*, de hauteur[4] s (ou peut-être moins si $\mathit{Sym}(\Omega)_{\mathcal{P}} = \emptyset$) dans lequel chaque noeud au niveau h, $0 \leq h \leq s$, est étiqueté par un couple $((\beta_1, \ldots, \beta_h), \mathbf{\Sigma}_h)$. Nous définissons cet arbre comme suit :

1. Si $\|U_{0,\mathsf{D}}((\Omega))\| \neq \|\mathbf{\Pi}_0\|$ ou $\mathit{fix}\,(U_{0,\mathsf{D}}((\Omega))) \notin \mathit{fix}\,(\mathbf{\Pi}_0)^G$ pour un groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$, alors $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ est vide. Sinon la racine de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ est étiqueté par $((), \mathbf{\Sigma}_0)$, où $()$ est une suite vide (un 0-vecteur), et $\mathbf{\Sigma}_0 = U_{0,\mathsf{D}}((\Omega))$.

2. Pour $1 \leq h \leq s$, les enfants d'un noeud de niveau $h-1$ étiqueté par un couple $((\beta_1, \ldots, \beta_{h-1}), \mathbf{\Sigma}_{h-1})$ sont exactement les noeuds étiquetés par des couples $((\beta_1, \ldots, \beta_{h-1}, \beta_h), \mathbf{\Sigma}_h)$, où $\beta_h \in \Sigma_{h-1,e_h}$ et $\mathbf{\Sigma}_h = U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$, tels que $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ et $\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$.

Soit $((\beta_1, \ldots, \beta_s), \mathbf{\Sigma}_s)$ un noeud au niveau s. D'après les propositions 2.4.13(b) et 2.4.13(cv), il existe au plus un élément g de $\mathit{Sym}(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$,

4. c'est-à-dire qui possède $s+1$ niveaux, la racine étant au niveau 0.

et s'il existe, il peut être déterminé de manière unique de l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$. Nous vérifions si la permutation g, déterminée par $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$, satisfait $\mathcal{P}$. Si c'est le cas, nous trouvons $g \in Sym(\Omega)_\mathcal{P}$ tel que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$. Et comme à tout $g \in Sym(\Omega)_\mathcal{P}$ correspond un noeud de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ au niveau s, nous trouvons tous les éléments de $Sym(\Omega)_\mathcal{P}$ en parcourant les noeuds de niveau s de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$.

Donc nous pouvons énumérer tous les éléments de $Sym(\Omega)_\mathcal{P}$ en effectuant la recherche arrière dans l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$.

Nous commençons par initialiser un ensemble L d'éléments de $Sym(\Omega)_\mathcal{P}$ à $\emptyset$. Nous tentons de commencer la traversée de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ à la racine en calculant son étiquette $((), \mathbf{\Sigma}_0)$. Si $\|U_{0,\mathsf{D}}((\Omega))\| \neq \|\mathbf{\Pi}_0\|$ ou *fix* $(U_{0,\mathsf{D}}((\Omega))) \notin$ *fix* $(\mathbf{\Pi}_0)^G$ pour un groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$, alors nous avons fini : $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ est vide et $Sym(\Omega)_\mathcal{P} = \emptyset$. Supposons maintenant que nous ayons atteint un noeud au niveau $h-1$, $1 \leq h \leq s$, étiqueté par $((\beta_1, \ldots, \beta_{h-1}), \mathbf{\Sigma}_{h-1})$. Les enfants de ce noeud correspondent à un sous-ensemble de Σ_{h-1,e_h}, donc pour chaque point β_h de Σ_{h-1,e_h}, nous essayons de calculer l'étiquette $((\beta_1, \ldots, \beta_{h-1}, \beta_h), \mathbf{\Sigma}_h)$ de l'enfant possible correspondant à β_h, en utilisant la formule $\mathbf{\Sigma}_h = U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$. Si $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ et si *fix* $(\mathbf{\Sigma}_h) \in$ *fix* $(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$, nous descendons du noeud courant à son enfant correspondant à β_h, et nous répétons le processus. Sinon pour ce β_h l'enfant n'existe pas, donc nous prenons l'élément suivant β_h de Σ_{h-1,e_h}, ou nous revenons au parent du noeud courant s'il n'en reste plus. Quand nous atteignons une feuille de l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ au niveau s, i.e. un noeud étiqueté par $((\beta_1, \ldots, \beta_s), \mathbf{\Sigma}_s)$, nous vérifions si $\mathcal{P}(g)$ est satisfait, où g est déterminé de manière unique par l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$. Si c'est le cas, nous mettons g dans L.

Quand la traversée de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ est finie, L est l'ensemble des éléments de $Sym(\Omega)_\mathcal{P}$. De plus, si nous considérons les enfants de chaque noeud dans l'ordre croissant des β_h selon l'ordre $\prec$ des éléments de Ω, correspondant à la $\mathbf{R}$-base (cf. notation 2.4.12), nous obtenons les éléments de $Sym(\Omega)_\mathcal{P}$ dans l'ordre croissant défini comme dans la section 1.4.7, p. 26, à partir de l'ordre $\prec$.

2.5.2 L'algorithme

Nous présentons maintenant l'algorithme formel 2.5.1 `Éléments` pour parcourir l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$. Il est basé sur l'algorithme 2.3.1 de recherche arrière, mais il incorpore les particularités de la méthode des partitions (cf. sections 2.3.3 et 2.5.1).

La numérotation des lignes de l'algorithme 2.5.1 concorde avec celle de l'algorithme 2.3.1 de recherche arrière.

Les principales différences entre ces deux algorithmes sont les suivantes.

Premièrement, comme nous traitons ici les groupes de permutations, nous devons calculer d'abord une base du groupe que nous voulons trouver, i.e. de $Sym(\Omega)_\mathcal{P}$. Pour calculer $Sym(\Omega)_\mathcal{P}$ nous utilisons la méthode des partitions, donc, en plus, nous devons calculer une $\mathbf{R}$-base pour $\mathcal{P}$. C'est ce que fait l'algorithme `Base` décrit dans la section 2.9. Nous appelons donc cet algorithme au début de notre algorithme (la ligne 0 de l'algorithme 2.5.1) pour obtenir une $\mathbf{R}$-base $(\mathbf{U}, \mathbf{T})$ avec toutes les quantités associées (cf. remarque 2.4.11).

La deuxième différence est l'utilisation des partitions ordonnées de Ω. Outre les suites $(\beta_1, \ldots, \beta_h)$, nous étiquetons les noeuds de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ par les partitions ordonnées $\mathbf{\Sigma}_h$ de Ω, il faut donc les calculer pour chaque enfant du noeud courant. Dans la boucle principale (lignes 7–23) de l'algorithme, nous le faisons juste après le choix de β_h (la 9-ième ligne), c'est pourquoi nous devons insérer la ligne 9' comportant le calcul de $\mathbf{\Sigma}_h$. Avant la boucle principale nous devons calculer l'étiquette de la racine de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$, nous insérons donc la ligne 1'.

Troisième différence : les propriétés intermédiaires $\mathcal{P}_h$ sont définies. Elles sont basées sur la proposition 2.4.13 et expliquées ci-dessus (cf. définition de l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$). Nous remplaçons donc partout (les lignes 2 et 10) la vérification des propriétés $\mathcal{P}_h$ par la vérification $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ et $\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$.

Quatrièmement, comme c'est expliqué dans cette section, au lieu de tester tous les β_h de Ω, il suffit de considérer les β_h de Σ_{h-1,e_h}. C'est pourquoi dans les lignes 6 et 17 nous initialisons l'ensemble Γ_h à Σ_{h-1,e_h} — l'ensemble des valeurs possibles de β_h, et dans la ligne 9 nous choisissons un point β_h de Γ_h. En réalité, nous choisissons *le plus petit* point de Γ_h selon l'ordre $\prec$ de Ω, défini dans la notation 2.4.12. Le fait de choisir les $\beta_h \in \Sigma_{h-1,e_h}$ dans l'ordre ascendant permet d'obtenir les éléments de $Sym(\Omega)_{\mathcal{P}}$ dans l'ordre ascendant.

Et la cinquième remarque : partout dans l'algorithme nous travaillons avec les partitions $\mathbf{\Sigma}_h$ et les suites $(\beta_1, \ldots, \beta_h)$, sauf quand $h = s$ (les lignes 12–14), car pour tester la propriété $\mathcal{P}$ nous devons avoir une permutation. Pour transformer la partition discrète courante $\mathbf{\Sigma}_s$ en une permutation g, nous introduisons la ligne 11'. Comme nous l'avons déjà remarqué, il est trivial de construire une permutation g telle que $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$, si $\mathbf{\Pi}_s$ et $\mathbf{\Sigma}_s$ sont discrètes.

Et voici l'algorithme lui-même.

Algorithme 2.5.1 *Éléments*$(\mathcal{P}, \mathbf{R})$

Entrée : $\mathcal{P}$ une propriété sur un ensemble fini Ω (rappelons que $\mathcal{P}$ contient au moins une propriété $\mathcal{E}_G$, cf. remarque 2.2.3),
$\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires.

Sortie : L l'ensemble (ordonné) des éléments de $Sym(\Omega)_{\mathcal{P}}$ (dans l'ordre croissant associé à l'ordre $\prec$ de Ω défini par la $\mathbf{R}$-base $(\mathbf{U}, \mathbf{T})$ pour $\mathcal{P}$).

Début

0. $(\mathbf{U}, \mathbf{T}) :=$ `Base` $(\mathbf{R})$;
1. $L := \emptyset$;
1'. $\mathbf{\Sigma}_0 := U_{0,\mathsf{D}}((\Omega))$;
2. `si` $\|\mathbf{\Sigma}_0\| \neq \|\mathbf{\Pi}_0\|$ ou
   ($\mathit{fix}\,(\mathbf{\Sigma}_0) \notin \mathit{fix}\,(\mathbf{\Pi}_0)^G$ pour un groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`
3. `retourner` L;
4. `fin si`;
5. $h := 1$;
6. $\Gamma_h := \Sigma_{h-1,e_h}$;
7. `tant que` $h > 0$ `faire`
8. `si` $|\Gamma_h| > 0$ `alors`
9. $\beta_h := \min(\Gamma_h)$; $\Gamma_h := \Gamma_h \setminus \{\beta_h\}$; [on prend min selon l'ordre $\prec$ défini par]

9'. $\mathbf{\Sigma}_h := U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$; [la **R**-base $(\mathbf{U}, \mathbf{T})$ (cf. notation 2.4.12, p. 71)]
10. `si` $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ et
($\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`
11. `si` $h = s$ `alors`
11'. $g :=$ l'unique solution de l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$; [$\mathbf{\Pi}_s$ et $\mathbf{\Sigma}_s$ sont discrètes]
12. `si` $\mathcal{P}(g)$ `alors`
13. $L := L \cup \{g\}$; [ajouter g à L]
14. `fin si`;
15. `sinon`
16. $h := h + 1$;
17. $\Gamma_h := \Sigma_{h-1,e_h}$;
18. `fin si`;
19. `fin si`;
20. `sinon`
21. $h := h - 1$;
22. `fin si`;
23. `fin tant que`;
24. `retourner` L;

Fin.

Bien sûr, l'algorithme ne donne pas tous les détails de l'exécution, car c'est une description générale, par exemple, l'application d'un $\mathcal{P}$-raffinement à une partition dépend forcément du problème que nous voulons résoudre, i.e. de $\mathcal{P}$. La vérification de la condition $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ est expliquée dans la section 2.10.4.1. La vérification de la condition $\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ dans le cas où $G = \mathit{Sym}_q(\Omega)$ est décrite dans la section 3.6. Notez que si $G = \mathit{Sym}(\Omega)$, cela ne vaut pas la peine de vérifier cette condition, car elle est triviale.

Notez encore que les lignes 9' et 10 (comme les lignes 1' et 2) ne s'exécute pas l'une après l'autre, plutôt elles se mélangent, cf. section 2.10.4.1 pour plus de détails.

2.6 Problèmes de type sous-groupe

L'algorithme 2.5.1 ***Éléments*** de la section 2.5 trouve tous les éléments de $\mathit{Sym}(\Omega)_{\mathcal{P}}$ et donc n'est applicable que si $|\mathit{Sym}(\Omega)_{\mathcal{P}}|$ est petit. Nous ne pouvons pas espérer faire mieux sans aucune supposition sur la structure de $\mathit{Sym}(\Omega)_{\mathcal{P}}$. Par contre, dans le cas où $\mathcal{P}$ est de type sous-groupe ou de type classe, il n'est pas nécessaire, et souvent il est même indésirable, de générer l'ensemble complet des éléments de $\mathit{Sym}(\Omega)_{\mathcal{P}}$.

Dans cette section nous allons décrire comment faire pour les problèmes de type sous-groupe, et dans la section suivante nous considérerons les problèmes de type classe.

Soit $\mathcal{P}$ une propriété de type sous-groupe sur un ensemble fini Ω. Pour trouver $\mathit{Sym}(\Omega)_{\mathcal{P}}$, il suffira de construire un ensemble fort de générateurs (un SGS) $X = \{s_1, \ldots, s_m\}$ associé à une base de $\mathit{Sym}(\Omega)_{\mathcal{P}}$. L'algorithme que nous présentons calcule un SGS associé à la base $B = (\alpha_1, \ldots, \alpha_s)$ définie par une **R**-base $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ pour $\mathcal{P}$.

Supposons que nous connaissions une $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$ avec toutes les quantités associées (cf. remarque 2.4.11).

L'algorithme repose sur la notation et la proposition suivantes.

Notation 2.6.1 [Leo91, Déf.7] Soit $<$ un ordre total de Ω. Il induit un ordre total $<$ dans $Sym(\Omega)$ (cf. section 1.4.7). Soit G un groupe de permutations de Ω. Nous notons $X(G,<)$ l'ensemble ordonné $(s_1,\ldots,s_m)$ de permutations de G, où $s_i = \min(G\backslash <s_1,\ldots,s_{i-1}>)$ et m est le plus petit entier tel que $<s_1,\ldots,s_m> = G$ (nous prenons le minimum min suivant l'ordre $<$).

Proposition 2.6.2 *[Leo91, Prop.3 et Prop.7] Soit $\mathcal{P}$ une propriété de type sous-groupe sur Ω. Soient $\prec$ un ordre de Ω et $B = (\alpha_1,\ldots,\alpha_s)$ la base de $Sym(\Omega)_{\mathcal{P}}$ associés à une $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$. Alors $X(Sym(\Omega)_{\mathcal{P}},\prec)$ est un ensemble fort de générateurs de $Sym(\Omega)_{\mathcal{P}}$ associé à la base B.*

L'algorithme 2.6.6 `Sous-groupe` que nous présentons dans cette section calcule l'ensemble fort de générateurs $X(Sym(\Omega)_{\mathcal{P}},\prec) = (s_1,\ldots,s_m)$ de $Sym(\Omega)_{\mathcal{P}}$.

Supposons que nous effectuions la traversée de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ dans l'ordre croissant. Nous obtenons donc les éléments de $Sym(\Omega)_{\mathcal{P}}$ dans l'ordre croissant.

Supposons que nous ayons trouvé déjà $(s_1,\ldots,s_j)$, $0 \leq j \leq m$. Nous cherchons alors $s_{j+1} = \min(Sym(\Omega)_{\mathcal{P}} \setminus K)$, où $K = <s_1,\ldots,s_j>$. En particulier, s_{j+1} est un élément minimal dans ses classes à droite et à gauche modulo K, et dans sa classe double $Ks_{j+1}K$, i.e. $s_{j+1} = \min(Ks_{j+1}K)$. Donc si nous déterminons pour un noeud $((\beta_1,\ldots,\beta_h),\mathbf{\Sigma}_h)$, $1 \leq h \leq s$, de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ qu'il n'existe pas de $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1,\ldots,\alpha_h)^g = (\beta_1,\ldots,\beta_h)$ et $g = \min(KgK)$, nous pouvons omettre ce noeud et tous ses descendants, nous ne les considérons pas, nous passons directement au voisin du noeud courant (c'est-à-dire nous prenons un autre β_h). Nous dirons dans ce cas que nous *taillons* (“prune” en anglais) l'arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ à ce noeud, ou que nous taillons ce noeud, ou que nous *éliminons* la branche de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ commençant à ce noeud.

Nous donnons des conditions nécessaires (facilement calculables) pour qu'un élément soit minimal dans sa classe double. Nous les utilisons dans notre implantation de l'algorithme de Leon. Ces conditions, données par Leon, sont des généralisations de résultats de Sims [Sim71a, Sim71b].

Rappelons que les propriétés $\mathcal{P}_{\mathsf{G}}$ et $\mathcal{P}_{\mathsf{D}}$ sont définies par les formules (2.2) et (2.3), p. 54, et que $\mathcal{L}(B,g) = \min\{j \mid \alpha_j \neq \beta_j\}$ si $g \neq id$, cf. notation 1.4.8, p. 24.

Proposition 2.6.3 *[Leo91, Prop.8] et [Leo97, Prop.6] Soit $\mathcal{P}$ une propriété de type sous-groupe ou de type classe sur un ensemble fini Ω. Soit $g \in Sym(\Omega)_{\mathcal{P}}$. Supposons que nous ayons une $\mathbf{R}$-base $(\mathbf{U},\mathbf{T})$ pour $\mathcal{P}$ avec les quantités associées (ces quantités incluent la base $B = (\alpha_1,\ldots,\alpha_s)$ de $Sym(\Omega)_{\mathcal{P}}$ et l'ordre $\prec$ de Ω). Soit $(\beta_1,\ldots,\beta_s) \stackrel{\text{déf}}{=} (\alpha_1,\ldots,\alpha_s)^g$, soit les partitions ordonnées $\mathbf{\Sigma}_0,\ldots,\mathbf{\Sigma}_s$ définies comme dans (2.8), soit K et M des sous-groupes de $Sym(\Omega)_{\mathcal{P}}$ si $\mathcal{P}$ est de type sous-groupe, ou des sous-groupes de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$,* *respectivement, si $\mathcal{P}$ est de type classe, et soit $1 \leq h \leq s$. Les conditions suivantes sont des conditions nécessaires pour $g = \min(KgM)$:*

1. *β_h ne se trouve pas parmi les derniers (selon l'ordre $\prec$) $\left|\Delta^{(h)}\right| - 1$ éléments de Σ_{h-1,e_h}, où $\Delta^{(h)}$ est la h-ième orbite de base de K associée à la base $B = (\alpha_1, \ldots, \alpha_s)$, c'est-à-dire $\Delta^{(h)} = {\alpha_h}^{K_{\alpha_1,\ldots,\alpha_{h-1}}}$ (cf. définition 1.7.4, p. 41).*
2. *Si $h = \mathcal{L}(B, g)$, alors $\beta_h = \min({\beta_h}^{M_{\alpha_1,\ldots,\alpha_{h-1}}})$.*
3. *Pour tout i, $\mathcal{L}(B, g) \leq i < h$, tel que $\alpha_h \in \Delta^{(i)}$, nous avons $\beta_i \prec \beta_h$.*

Remarque 2.6.4 Notez que [Leo91, Prop.8] donne d'autres conditions nécessaires pour qu'un élément soit minimal dans sa classe double, notamment

4. Pour tout i, $1 \leq i \leq h$, tel que $\alpha_h \in \Delta^{(i)}$, nous avons $\beta_i \preceq \min({\beta_h}^{M_{\beta_1,\ldots,\beta_{i-1}}})$ (généralisation des conditions 2 et 3).
5. Pour tout i, $1 \leq i \leq h$, et tout j, $1 \leq j \leq |\mathbf{\Pi}_h|$,
$$\left|\Delta^{(i)} \cap \Pi_{h,j}\right| \leq \left|\{\gamma \in \Sigma_{h,j} \mid \beta_i \preceq \min(\gamma^{M_{\beta_1,\ldots,\beta_{i-1}}})\}\right|.$$

Pour vérifier ces conditions, il faut trouver le stabilisateur de $\beta_1, \ldots, \beta_{i-1}$ dans M, et pour cela il faut effectuer un *changement de la base* (cf. section 1.7.6). Il faudrait le faire pour toute solution partielle $(\beta_1, \ldots, \beta_{i-1})$. Cela prend un temps considérable. Leon remarque [Leo97, p.143] que son expérience avec l'algorithme montre que pour certains problèmes "difficiles", la vérification de la condition 4 accélère considérablement l'algorithme (le gain de temps obtenu en taillant plus efficacement l'arbre est supérieur à la perte de temps pour effectuer les changements de la base), tandis que pour beaucoup de problèmes "faciles", elle ralentit légèrement l'algorithme. Quant à la condition 5, Leon précise qu'elle n'a pas été implantée et donc son utilité reste indécise. Nous n'avons pas implanté ces deux conditions, c'est pourquoi nous ne les discuterons pas. □

Donc pour les problèmes de type sous-groupe, nous modifions l'algorithme 2.5.1 ***Éléments*** de telle sorte que nous taillons immédiatement un noeud $((\beta_1, \ldots, \beta_h), \mathbf{\Sigma}_h)$ de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ si nous déterminons (au moyen de la proposition 2.6.3) qu'il n'existe pas de $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$ et $g = \min(KgK)$, où K désigne toujours $< s_1, \ldots, s_j >$ et $s_1, \ldots, s_j$ sont des permutations de $Sym(\Omega)_{\mathcal{P}}$ déjà trouvées.

Nous présentons ce nouvel algorithme ***Sous-groupe*** (cf. algorithme 2.6.6). Comme avant, la numérotation des lignes de cet algorithme correspond à celles des algorithmes 2.3.1 et 2.5.1. Nous présentons cet algorithme par comparaison avec l'algorithme 2.5.1 ***Éléments***, i.e. en accentuant les ajouts à l'algorithme 2.6.6 ***Sous-groupe*** par rapport à l'algorithme 2.5.1 ***Éléments***.

Après avoir choisi un nouveau β_h (ligne 9 de l'algorithme ***Sous-groupe***), nous vérifions si $(\beta_1, \ldots, \beta_h)$ satisfait les conditions de la proposition 2.6.3. Nous testons la condition 1 dans la ligne 8, la condition 2 dans la ligne 9b et la condition 3 dans la ligne 9c. Nous le faisons avant le calcul de $\mathbf{\Sigma}_h$ parce que, en général, la vérification des conditions de la proposition 2.6.3 est bien moins coûteuse que le calcul de $\mathbf{\Sigma}_h$ (ligne 9') et la vérification des conditions associées à $\mathbf{\Sigma}_h$ (ligne 10).

Rappelons qu'à tout moment du fonctionnement de l'algorithme $K =< X >$, où $X = \{s_1, \ldots, s_j\}$ est l'ensemble des permutations de $Sym(\Omega)_{\mathcal{P}}$ déjà trouvées. Nous initialisons X dans la ligne 1, et nous le mettons à jour chaque fois que nous trouvons un nouveau élément g de $Sym(\Omega)_{\mathcal{P}}$ (ligne 13). La mise à jour de X implique la mise à jour de K et la reconstruction de la r-ième orbite de base $\Delta^{(r)}$ de K, où $r = \mathcal{L}(B, g)$ (cf. notation 1.4.8, p. 24). Les orbites de base $\Delta^{(i)}$ de K sont utilisées dans les lignes 8 et 9c pour vérifier les conditions 1 et 3 de la proposition 2.6.3.

Dans les lignes 9b et 9c nous utilisons $\min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\}$. Notez que si $\beta_j = \alpha_j$ pour tout j, $1 \leq j \leq h$, nous définissons ce minimum comme étant égal à $s+1$. En fait, ainsi défini, ce minimum est égal à $\mathcal{L}(B, g)$ (cf. notation 1.4.8, p. 24), où g est la permutation la plus petite de $Sym(\Omega)_{\mathcal{P}}$ telle que $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$ (si elle existe ; comme nous parcourons l'arbre de la méthode des partitions selon l'ordre lexicographique sur les suites $(\beta_1, \ldots, \beta_h)$, à tout moment nous sommes en train de calculer ce g). En effet, si g existe, d'après la définition de $\mathcal{L}(B, g)$ nous obtenons

$$\begin{aligned} \mathcal{L}(B, g) &= \min\{j \mid 1 \leq j \leq s,\ {\alpha_j}^g \neq \alpha_j\} \\ &= \min\{j \mid 1 \leq j \leq s,\ \beta_j \neq \alpha_j\} \\ &= \min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\} \end{aligned}$$

si $g \neq id$ et $\mathcal{L}(B, g) = s + 1$ sinon. Nous pouvons donc utiliser ce minimum au lieu de $\mathcal{L}(B, g)$ pour la vérification des conditions de la proposition 2.6.3.

Notez qu'à tout moment du fonctionnement de l'algorithme toutes les permutations de K fixent $(\alpha_1, \ldots, \alpha_{r-1})$, où $r = \min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\}$. Donc si $h = r$, alors $M_{\alpha_1,\ldots,\alpha_{h-1}}$ de la condition 2 de la proposition 2.6.3 est égal à

$$M_{\alpha_1,\ldots,\alpha_{h-1}} = K_{\alpha_1,\ldots,\alpha_{h-1}} = K_{\alpha_1,\ldots,\alpha_{r-1}} = K =< X >,$$

c'est pourquoi nous écrivons K au lieu de $M_{\alpha_1,\ldots,\alpha_{h-1}}$ pour tester cette condition dans la ligne 9b de l'algorithme.

Il reste à discuter encore deux différences entre les algorithmes 2.5.1 ***Éléments*** et 2.6.6 ***Sous-groupe***.

Premièrement, comme $\mathcal{P}$ est de type sous-groupe, la proposition 2.4.5 montre que tous les $\mathcal{P}$-raffinements élémentaires sont symétriques, donc

$$\mathbf{\Sigma}_0 = U_{0,\mathsf{D}}((\Omega)) = U_{0,\mathsf{G}}((\Omega)) = \mathbf{\Pi}_0,$$

c'est pourquoi nous pouvons poser directement $\mathbf{\Sigma}_0 := \mathbf{\Pi}_0$ dans la ligne 1' de l'algorithme ***Sous-groupe*** et omettre la vérification des conditions de la ligne 2 de l'algorithme ***Éléments***. C'est pourquoi il manque les lignes 2–4 à l'algorithme ***Sous-groupe***.

Deuxièmement, supposons que nous trouvions un nouveau élément $g = s_{j+1}$ appartenant à $Sym(\Omega)_{\mathcal{P}}$. Nous avons donc $(\alpha_1, \ldots, \alpha_s)^{s_{j+1}} = (\beta_1, \ldots, \beta_s)$. Alors

$$s_{j+2} = \min(Sym(\Omega)_{\mathcal{P}} \setminus < s_1, \ldots, s_{j+1} >)$$

est la permutation suivante que nous cherchons. Soit

$$r = \min\{j \mid 1 \leq j \leq s,\ \beta_j \neq \alpha_j\}.$$

Alors s_{j+2} satisfait la condition suivante :

Propriété 2.6.5 $(\alpha_1, \ldots, \alpha_r)^{s_{j+2}} \neq (\alpha_1, \ldots, \alpha_r)^{s_{j+1}}$, *i.e. il existe* i, $1 \leq i \leq r$, *tel que* ${\alpha_i}^{s_{j+2}} \neq {\alpha_i}^{s_{j+1}}$.

Preuve. Si ce n'est pas le cas, alors $(\alpha_1, \ldots, \alpha_r)^{s_{j+2}} = (\alpha_1, \ldots, \alpha_r)^{s_{j+1}}$ et donc $s_{j+2}s_{j+1}^{-1}$ fixe $(\alpha_1, \ldots, \alpha_r)$.

D'après la définition de r, $\beta_r \neq \alpha_r$. Mais est-ce que $\beta_r \succ \alpha_r$? Oui. En effet, considérons les formules (2.7), p. 71, et (2.8), p. 72. Dans notre cas $\mathcal{P}$-raffinements élémentaires sont symétriques, donc $U_{h,\mathsf{G}} = U_{h,\mathsf{D}}$ dans ces formules. De plus, $(\alpha_1, \ldots, \alpha_{r-1}) = (\beta_1, \ldots, \beta_{r-1})$, donc $\mathcal{F}_{e_h,\alpha_h} = \mathcal{F}_{e_h,\beta_h}$ dans ces formules pour tout h, $1 \leq h \leq r-1$. Cela donne

$$\mathbf{\Pi}_0 = \mathbf{\Sigma}_0,\ \mathbf{\Pi}_1 = \mathbf{\Sigma}_1, \ldots,\ \mathbf{\Pi}_{r-1} = \mathbf{\Sigma}_{r-1}.$$

Donc $\Pi_{r-1,e_r} = \Sigma_{r-1,e_r}$. Mais $\beta_r \in \Sigma_{r-1,e_r}$, donc $\beta_r \in \Pi_{r-1,e_r}$. Et $|\Pi_{r-1,e_r}| > 1$ en vue de la condition 1 de la définition 2.4.10 de $\mathbf{R}$-base, p. 70. Par contre, dans la partition $\mathbf{\Pi}_{r-1}$, les $\alpha_1, \ldots, \alpha_{r-1}$ sont dans des cellules de cardinal 1 (à cause de la définition de $\mathcal{F}_{e_h,\alpha_h}$), donc $\beta_r \notin \{\alpha_1, \ldots, \alpha_{r-1}\}$. Comme $\beta_r \neq \alpha_r$, nous obtenons $\beta_r \succ \alpha_r$.

Donc s_{j+1} ne fixe pas α_r, et ${\alpha_r}^{s_{j+1}} \succ \alpha_r$. De l'autre coté, $s_{j+2}s_{j+1}^{-1}$ fixe $(\alpha_1, \ldots, \alpha_r)$. Selon la définition de l'ordre sur les permutations, nous obtenons $s_{j+2}s_{j+1}^{-1} \prec s_{j+1}$, ce qui signifie que $s_{j+2}s_{j+1}^{-1}$ se trouve déjà dans $K = < s_1, \ldots, s_{j+1} >$ (nous parcourons les permutations dans l'ordre croissant, donc nous avons rencontré $s_{j+2}s_{j+1}^{-1}$ avant s_{j+1}), donc $s_{j+2} \in Ks_{j+1} = K$, ce qui contredit la définition de s_{j+2}. □

Donc dans l'algorithme, après avoir trouvé une permutation $g \in Sym(\Omega)_{\mathcal{P}}$ telle que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$, nous pouvons éliminer toutes les extensions de $(\beta_1, \ldots, \beta_r)$, car pour aucune d'entre elles nous ne trouverons s_{j+2}. Nous devons changer directement β_r. Cela justifie l'instruction de la ligne 13a de l'algorithme.

Dernière petite remarque : par convention $\{id\} = <>$ (cf. remarque après la définition 1.4.6), c'est pourquoi nous ne mettons pas id dans X (ligne 12).

Et voilà l'algorithme lui-même.

Algorithme 2.6.6 `Sous-groupe`$(\mathcal{P}, \mathbf{R})$
Entrée : $\mathcal{P}$ une propriété sur un ensemble fini Ω (rappelons que $\mathcal{P}$ contient
    au moins une propriété $\mathcal{E}_G$, cf. remarque 2.2.3, p. 55),
  $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires.
Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$, toutes les quantités associées
    (cf. remarque 2.4.11, p. 71) incluses,
  X un ensemble fort de générateurs de $Sym(\Omega)_{\mathcal{P}}$
    associé à la base $B = (\alpha_1, \ldots, \alpha_s)$ correspondant à la $\mathbf{R}$-base $\mathbf{B}$.
Début

0. $(\mathbf{U}, \mathbf{T}) :=$ *Base* $(\mathbf{R})$;
1. $X := \emptyset$;
1'. $\boldsymbol{\Sigma}_0 := \boldsymbol{\Pi}_0$;
5. $h := 1$;
6. $\Gamma_h := \Sigma_{h-1,e_h}$;
7. `tant que` $h > 0$ `faire`
8. `si` $|\Gamma_h| \geq \left|\Delta^{(h)}\right|$ `alors` [la vérification de la cond. 1 de la prop. 2.6.3]
9. $\beta_h := \min(\Gamma_h)$; $\Gamma_h := \Gamma_h \setminus \{\beta_h\}$;
9b. `si` $(h \neq \min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\}$ ou $\beta_h = \min({\beta_h}^K))$ et [$K =< X >$]
9c. (pour tout i, $\min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\} \leq i \leq h-1$,
tel que $\alpha_h \in \Delta^{(i)}$, on a $\beta_i \prec \beta_h$) `alors`
9'. $\boldsymbol{\Sigma}_h := U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\boldsymbol{\Sigma}_{h-1}))$;
10. `si` $\|\boldsymbol{\Sigma}_h\| = \|\boldsymbol{\Pi}_h\|$ et
(*fix* $(\boldsymbol{\Sigma}_h) \in$ *fix* $(\boldsymbol{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`
11. `si` $h = s$ `alors`
11'. $g :=$ l'unique solution de l'équation $\boldsymbol{\Pi}_s{}^g = \boldsymbol{\Sigma}_s$;
12. `si` $\mathcal{P}(g)$ et $g \neq id$ `alors`
13. $X := X \cup \{g\}$; [ajouter g à X]
13a. $h := \min\{j \mid 1 \leq j \leq s,\ \beta_j \neq \alpha_j\}$;
14. `fin si`;
15. `sinon`
16. $h := h + 1$;
17. $\Gamma_h := \Sigma_{h-1,e_h}$;
18. `fin si`;
19. `fin si`;
19a. `fin si`;
20. `sinon`
21. $h := h - 1$;
22. `fin si`;
23. `fin tant que`;
24. `retourner` $(\mathbf{U}, \mathbf{T}), X$;

Fin.

Proposition 2.6.7 *[Leo91, p.559] L'algorithme 2.6.6* `Sous-groupe` *sort l'ensemble X égal à $X(Sym(\Omega)_{\mathcal{P}}, \prec)$, où $\prec$ est l'ordre associé à $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$.*

Donc d'après la proposition 2.6.2, X est un ensemble fort de générateurs de $Sym(\Omega)_{\mathcal{P}}$ associé à la base $(\alpha_1, \ldots, \alpha_s)$.

2.7 Problèmes de type classe

Dans cette section nous allons décrire l'algorithme général *RepClasse* pour un problème de type classe. Soit $\mathcal{P}$ une propriété de type classe sur un ensemble fini Ω. L'algorithme

RepClasse trouve un élément de $Sym(\Omega)_{\mathcal{P}}$. Supposons que nous ayons un ensemble $\mathbf{R}$ de $\mathcal{P}$-raffinements élémentaires et une $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$. L'algorithme 2.7.1 *RepClasse* ne diffère pas beaucoup de l'algorithme 2.6.6 *Sous-groupe*. Comme d'habitude, la numérotation des lignes des deux algorithmes va concorder.

Pour pouvoir tailler l'arbre de la méthode des partitions $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ (cf. section 2.5.1, p. 73) au moyen de la proposition 2.6.3, p. 77, il faut avoir des sous-groupes L_{G} et L_{D} de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$, respectivement. Nous pouvons, par exemple, les précalculer à l'aide de l'algorithme *Sous-groupe*. En général, l'algorithme *RepClasse* prend en entrée les sous-groupes L_{G} et L_{D} de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$, respectivement, mais si nous ne les connaissons pas, nous pouvons entrer dans l'algorithme les sous-groupes triviaux $\{id\}$.

Maintenant au lieu de chercher $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $g = \min(KgK)$, nous chercherons $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $g = \min(L_{\mathsf{G}} g L_{\mathsf{D}})$; nous modifions donc l'algorithme de façon adéquate.

L'ensemble $\Delta^{(i)}$ (lignes 8 et 9c de l'algorithme *RepClasse*) est la i-ième orbite de base de L_{G} pour la base $B = (\alpha_1, \ldots, \alpha_s)$ associée à la $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$. Notez que, étant donné un ensemble fort de générateurs de L_{G} associé à la base B, les $\Delta^{(i)}$, $1 \leq i \leq s$, sont facilement calculables (cf. section 1.7.5, p. 46). De même, étant donné un ensemble fort de générateurs de L_{D} associé à la base B, les orbites de $(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{h-1}}$ sont facilement calculables (nous les utilisons dans la ligne 9b de l'algorithme *RepClasse* pour vérifier la condition 2 de la proposition 2.6.3).

Nous supprimons la ligne 13a de l'algorithme *Sous-groupe*, car nous arrêtons l'exécution de l'algorithme dès que nous trouvons un élément quelconque de $Sym(\Omega)_{\mathcal{P}}$ (ligne 13 de l'algorithme *RepClasse*).

Le retour des lignes 2–4 reflète le fait que les $\mathcal{P}$-raffinements élémentaires ne sont plus symétriques.

Nous ne vérifions plus si $g \neq id$ (ligne 12), car le cas où $g = id$ est possible.

Algorithme 2.7.1 *RepClasse*$(\mathcal{P}, \mathbf{R}, L_{\mathsf{G}}, L_{\mathsf{D}})$

Entrée : $\mathcal{P}$ une propriété sur un ensemble fini Ω (rappelons que $\mathcal{P}$ contient
au moins une propriété $\mathcal{E}_G$, cf. remarque 2.2.3),
$\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
$L_{\mathsf{G}}, L_{\mathsf{D}}$ des sous-groupes de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$, respectivement,
donnés par des bases A_{G} et A_{D} et par
des ensembles forts de générateurs X_{G} et X_{D}, respectivement.

Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$, toutes les quantités associées
(cf. remarque 2.4.11) incluses,
g tel que $g \in Sym(\Omega)_{\mathcal{P}}$ si $Sym(\Omega)_{\mathcal{P}} \neq \emptyset$, $g = \emptyset$ sinon.

Début

0. $(\mathbf{U}, \mathbf{T}) :=$ *Base*$(\mathbf{R})$;
1. $g := \emptyset$;
1'. $\mathbf{\Sigma}_0 := U_{0,\mathsf{D}}((\Omega))$;
2. `si` $\|\mathbf{\Sigma}_0\| \neq \|\mathbf{\Pi}_0\|$ ou
   ($\mathit{fix}\,(\mathbf{\Sigma}_0) \notin \mathit{fix}\,(\mathbf{\Pi}_0)^G$ pour un groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`

3. `retourner` $(\mathbf{U}, \mathbf{T}), g$;
4. `fin si`;
5. $h := 1$;
6. $\Gamma_h := \Sigma_{h-1,e_h}$;
7. `tant que` $h > 0$ `faire`
8. `si` $|\Gamma_h| \geq \left|\Delta^{(h)}\right|$ `alors`
9. $\beta_h := \min(\Gamma_h)$; $\Gamma_h := \Gamma_h \setminus \{\beta_h\}$;
9b. `si` ($h \neq \min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\}$ ou $\beta_h = \min({\beta_h}^{(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{h-1}}})$) et
9c. (pour tout i, $\min\{j \mid 1 \leq j \leq h,\ \beta_j \neq \alpha_j\} \leq i \leq h-1$,
tel que $\alpha_h \in \Delta^{(i)}$, on a $\beta_i \prec \beta_h$) `alors`
9'. $\mathbf{\Sigma}_h := U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$;
10. `si` $\|\mathbf{\Sigma}_h\| = \|\mathbf{\Pi}_h\|$ et
($\mathit{fix}\ (\mathbf{\Sigma}_h) \in \mathit{fix}\ (\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`
11. `si` $h = s$ `alors`
11'. $g :=$ l'unique solution de l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$;
12. `si` $\mathcal{P}(g)$ `alors`
13. `retourner` $(\mathbf{U}, \mathbf{T}), g$;
14. `fin si`;
15. `sinon`
16. $h := h + 1$;
17. $\Gamma_h := \Sigma_{h-1,e_h}$;
18. `fin si`;
19. `fin si`;
19a. `fin si`;
20. `sinon`
21. $h := h - 1$;
22. `fin si`;
23. `fin tant que`;
24. `retourner` $(\mathbf{U}, \mathbf{T}), g$;

Fin.

2.8 Un ensemble de $\mathcal{E}_G$-raffinements élémentaires

Comme mentionné dans la remarque 2.2.3, p. 55, étant donnée une propriété $\mathcal{P}$ sur un ensemble fini Ω, nous supposons toujours que $\mathcal{P}$ contient une propriété $\mathcal{E}_G$ pour un groupe de permutations G. Parfois $G = \mathit{Sym}(\Omega)$, $\mathcal{E}_G$ est alors triviale (elle est toujours satisfaite), nous pouvons donc ne pas la considérer. Mais il arrive que $G \neq \mathit{Sym}(\Omega)$. Par exemple, des propriétés $\mathcal{A}_C$ et $\mathcal{A}_{C,C'}$ (pour des problèmes du groupe d'automorphismes d'un code C et d'isomorphisme des codes C et C') contiennent la propriété $\mathcal{E}_{\mathit{Sym}_q(\Omega)}$ (cf. section 1.6.3). Mais pour pouvoir utiliser l'algorithme des partitions pour une propriété $\mathcal{E}_G$, il faut avoir un ensemble de $\mathcal{E}_G$-raffinements élémentaires (de préférence efficaces). Dans cette section nous présentons un tel ensemble basé sur les orbites de stabilisateurs de G, introduit par

Leon [Leo91, Leo97].

Soit G un groupe de permutations d'un ensemble fini Ω, soient $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ des partitions ordonnées de Ω et soit

$$H = \{g \in G \mid \mathbf{\Pi}^g = \mathbf{\Sigma}\}.$$

Comment construire un $\mathcal{E}_G$-raffinement élémentaire, ou, plus précisément, en s'appuyant sur la définition 2.4.1, comment diviser une cellule de $\mathbf{\Pi}$ et la cellule correspondante de $\mathbf{\Sigma}$ en deux parties de façon que ces parties se correspondent par les permutations $g \in H$, c'est-à-dire diviser, par exemple, la i-ième cellule Π_i de $\mathbf{\Pi}$ en Π'_i et Π''_i et la i-ième cellule Σ_i de $\mathbf{\Sigma}$ en Σ'_i et Σ''_i de telle façon que $\Pi'^g_i = \Sigma'_i$ et $\Pi''^g_i = \Sigma''_i$ pour tout $g \in H$? Si nous pouvions le faire et si $\emptyset \subsetneq \Pi'_i \subsetneq \Pi_i$, nous obtiendrions de l'information supplémentaire sur les permutations g de H.

Soit $\mathbf{\Phi} \in \mathit{PartOrd}(\Omega)$ et $t, u \in G$ tels que $\mathit{fix}\left(\mathbf{\Phi}\right)^t = \mathit{fix}\left(\mathbf{\Pi}\right)$ et $\mathit{fix}\left(\mathbf{\Phi}\right)^u = \mathit{fix}\left(\mathbf{\Sigma}\right)$. Soit $g \in H$. Alors $tgu^{-1} \in G$ et

$$\mathit{fix}\left(\mathbf{\Phi}\right)^{tgu^{-1}} = \mathit{fix}\left(\mathbf{\Pi}\right)^{gu^{-1}} = \mathit{fix}\left(\mathbf{\Sigma}\right)^{u^{-1}} = \mathit{fix}\left(\mathbf{\Phi}\right),$$

donc $tgu^{-1} \in G_{\mathit{fix}(\mathbf{\Phi})}$. Par conséquent si Θ_j est la j-ième cellule de $\mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Phi})}\right)$ (cf. notation 1.4.15, p. 25), alors ${\Theta_j}^{tgu^{-1}} = \Theta_j$ (cf. lemme 1.4.14, p. 25), d'où ${\Theta_j}^{tg} = {\Theta_j}^u$. Et donc $\left(\Pi_i \cap {\Theta_j}^t\right)^g = {\Pi_i}^g \cap {\Theta_j}^{tg} = \Sigma_i \cap {\Theta_j}^u$. Nous avons trouvé un moyen de diviser Π_i et Σ_i en deux parties (nous divisons Π_i en $\Pi_i \cap {\Theta_j}^t$ et $\Pi_i \setminus {\Theta_j}^t$, Σ_i en $\Sigma_i \cap {\Theta_j}^u$ et $\Sigma_i \setminus {\Theta_j}^u$) de telle façon que g fasse correspondre les parties respectives de Π_i et Σ_i. C'est ce que nous cherchions.

Autrement dit, nous essayons de raffiner une partition ordonnée $\mathbf{\Pi}$ de Ω comme suit : étant donnés $\mathbf{\Phi} \in \mathit{PartOrd}(\Omega)$ et $t \in G$ tels que $\mathit{fix}\left(\mathbf{\Phi}\right)^t = \mathit{fix}\left(\mathbf{\Pi}\right)$, nous faisons l'intersection de la i-ième cellule Π_i de $\mathbf{\Pi}$ avec la j-ième cellule de $\mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Phi})}\right)$ pour tous les i et j possibles, et nous espérons diviser ainsi au moins une cellule de $\mathbf{\Pi}$. Voici la définition formelle de ce $\mathcal{E}_G$-raffinement élémentaire.

Notation 2.8.1 [Leo91, Déf.18] et [Leo97, Fig.2] Soit G un groupe de permutations d'un ensemble fini Ω, soit $n = |\Omega|$, soient $i, j \in I_n$ et soit $\mathbf{\Phi} \in \mathit{PartOrd}(\Omega)$. Pour toute partition ordonnée $\mathbf{\Pi}$ de Ω, le $\mathcal{E}_G$-raffinement élémentaire symétrique $D_{G,\mathbf{\Phi},i,j}$ est défini comme suit :

$$D_{G,\mathbf{\Phi},i,j}(\mathbf{\Pi}) = \begin{cases} \mathcal{F}_{i,{\Theta_j}^t}(\mathbf{\Pi}) & \text{si } \mathit{fix}\left(\mathbf{\Pi}\right) \in \mathit{fix}\left(\mathbf{\Phi}\right)^G, \\ \mathbf{\Pi} & \text{sinon,} \end{cases}$$

où Θ_j est la j-ième cellule de la partition $\mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Phi})}\right)$ et t est une permutation de G telle que $\mathit{fix}\left(\mathbf{\Phi}\right)^t = \mathit{fix}\left(\mathbf{\Pi}\right)$.

Nous définissons l'ensemble $\mathbf{D}_G$ de $\mathcal{E}_G$-raffinements élémentaires comme suit :

$$\mathbf{D}_G = \{D_{G,\mathbf{\Phi},i,j} \mid \mathbf{\Phi} \in \mathit{PartOrd}(\Omega), i, j \in I_n\}. \qquad \square$$

Notez que $D_{G,\mathbf{\Phi},i,j}$ ne dépend pas du choix de t. En effet, soit $t, t' \in G$ tels que

$$\mathit{fix}\left(\mathbf{\Phi}\right)^t = \mathit{fix}\left(\mathbf{\Pi}\right) \quad \text{et} \quad \mathit{fix}\left(\mathbf{\Phi}\right)^{t'} = \mathit{fix}\left(\mathbf{\Pi}\right).$$

Alors

$$\mathit{fix}\,(\mathbf{\Phi})^{t't^{-1}} = \mathit{fix}\,(\mathbf{\Pi})^{t^{-1}} = \mathit{fix}\,(\mathbf{\Phi})$$

et donc $t't^{-1} \in G_{\mathit{fix}(\mathbf{\Phi})}$, d'où

$$\Theta_j{}^{t't^{-1}} = \Theta_j \quad \text{et} \quad \Theta_j{}^{t'} = \Theta_j{}^{t}.$$

Proposition 2.8.2 *$D_{G,\mathbf{\Phi},i,j}$ est effectivement un $\mathcal{E}_G$-raffinement élémentaire symétrique.*

Preuve. Le fait que $D_{G,\mathbf{\Phi},i,j}$ est un $\mathcal{E}_G$-raffinement élémentaire découle du raisonnement ci-dessus. Il est symétrique parce que $\mathcal{E}_G$ est une propriété de type sous-groupe (cf. proposition 2.4.5, p. 68). □

Le calcul de $D_{G,\mathbf{\Phi},i,j}(\mathbf{\Pi})$ est pratique, l'unique tâche non triviale consiste de la détermination d'une permutation $t \in G$ telle que $\mathit{fix}\,(\mathbf{\Phi})^t = \mathit{fix}\,(\mathbf{\Pi})$. Nous n'expliquerons pas comment le faire dans le cas général où G est quelconque. Il ne nous faudra le faire que dans le cas où $G = \mathit{Sym}_q(\Omega)$, et dans ce cas c'est facile (voir section 3.6). Évidemment, si $G = \mathit{Sym}(\Omega)$, il est facile à voir que ce raffinement élémentaire est trivial, i.e. $D_{G,\mathbf{\Phi},i,j}(\mathbf{\Pi}) = \mathbf{\Pi}$ pour tout $D_{G,\mathbf{\Phi},i,j} \in \mathbf{D}_G$ et tout $\mathbf{\Pi} \in \mathit{PartOrd}(\Omega)$. Nous ne l'utiliserons donc que dans le cas $G \subsetneq \mathit{Sym}(\Omega)$.

Un lemme utile :

Lemme 2.8.3 *[Leo91, Lemmes 8 et 10] Soit G un groupe de permutations d'un ensemble fini Ω, soit $n = |\Omega|$ et soit $\mathbf{\Pi}$ une partition ordonnée de Ω.*

(a) Si $\mathbf{\Pi}$ est $\mathbf{D}_G$-réductible, alors il existe $i, j \in I_n$ tels que $D_{G,\mathbf{\Pi},i,j}(\mathbf{\Pi}) < \mathbf{\Pi}$.

(b) $\mathbf{\Pi}$ est $\mathbf{D}_G$-irréductible si et seulement si $\mathbf{\Pi} \lesssim \mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Pi})}\right)$ (cf. notation 1.5.6, p. 28, pour la définition de l'ordre $\lesssim$).

La première partie du lemme 2.8.3 montre que si on veut raffiner une partition ordonnée $\mathbf{\Pi}$ en utilisant les $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$, il suffit d'essayer les $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$ de la forme $D_{G,\mathbf{\Pi},i,j}$, où $i, j \in I_n$, c'est-à-dire nous pouvons n'essayer que les $\mathcal{E}_G$-raffinements élémentaires $D_{G,\mathbf{\Phi},i,j}$ avec $\mathbf{\Phi} = \mathbf{\Pi}$. Par conséquent au cours du calcul d'une $\mathbf{R}$-base $\mathbf{B}$ nous n'essayons que des $\mathcal{E}_G$-raffinements élémentaires de cette forme-là. Comme $\mathbf{\Phi} = \mathbf{\Pi}$, on peut prendre l'identité pour t dans la notation 2.8.1, donc Θ_j devient la j-ième cellule de $\mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Pi})}\right)$ et le calcul de $D_{G,\mathbf{\Phi},i,j}(\mathbf{\Pi})$ devient plus simple. Par contre, au cours du parcours de l'arbre de la méthode des partitions, il faudra utiliser les mêmes $\mathcal{E}_G$-raffinements élémentaires que nous avons utilisés en calculant la $\mathbf{R}$-base $\mathbf{B}$, il faudra donc appliquer $D_{G,\mathbf{\Pi},i,j}$ à une partition ordonnée $\mathbf{\Sigma}$ qui sera en général différente de $\mathbf{\Pi}$. Dans ce cas la détermination de t ne sera pas triviale et le calcul de $D_{G,\mathbf{\Pi},i,j}(\mathbf{\Sigma})$ sera plus compliqué (voir section 3.6 pour le cas $G = \mathit{Sym}_q(\Omega)$).

La deuxième partie montre quand il faut s'arrêter d'appliquer les $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$ à $\mathbf{\Pi}$, car si $\mathbf{\Pi}$ devient tel que $\mathbf{\Pi} \lesssim \mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Pi})}\right)$, nous ne pourrons plus raffiner $\mathbf{\Pi}$ au moyen des $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$.

Cela montre encore une fois que si $G = \mathit{Sym}(\Omega)$, $\mathbf{D}_G$ est inutile, car dans ce cas $\mathbf{\Pi} \lesssim \mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Pi})}\right)$, et donc $\mathbf{\Pi}$ est $\mathbf{D}_G$-irréductible, pour toute partition ordonnée $\mathbf{\Pi}$ de Ω.

2.9 Calcul d'une R-base

Soit $\mathcal{P}$ une propriété sur un ensemble fini Ω. Étant donné un ensemble $\mathbf{R}$ de $\mathcal{P}$-raffinements élémentaires, la construction d'une $\mathbf{R}$-base est relativement simple. Il s'agit de choisir les points d'une base $B = (\alpha_1, \ldots, \alpha_s)$ de $Sym(\Omega)_{\mathcal{P}}$ et, pour tout h, $h = 0, \ldots, s$, $(\alpha_1, \ldots, \alpha_h)$ étant fixé, de trouver un $\mathcal{P}$-raffinement U_h de $\mathbf{R}^*$ (cf. notation 2.4.8, p. 70) tel que $\mathbf{\Pi}_h \stackrel{\text{déf}}{=} U_{h,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$ soit le plus fine possible (si $h = 0$, $\mathbf{\Pi}_0 \stackrel{\text{déf}}{=} U_{0,\mathsf{G}}((\Omega))$).

Nous présentons l'algorithme 2.9.1 ***Base*** pour le faire. Les α_h, $1 \leq h \leq s$, sont choisis dans les lignes 6–7. Les U_h sont calculés dans les lignes 3 et 8 au moyen de la fonction ***Raffiner***$(\mathbf{R}, \mathbf{\Pi})$ (cf. l'algorithme 2.9.2) qui retourne aussi $U_{h,\mathsf{G}}(\mathbf{\Pi})$. La boucle principale (lignes 4–9) tourne jusqu'à ce que $\mathbf{\Pi}_h$ devient discrète. Notez que dans les lignes 6–7 les e_h et α_h choisis sont tels que $\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}) < \mathbf{\Pi}_{h-1}$, la condition 1 de la définition d'une $\mathbf{R}$-base (cf. définition 2.4.10, p. 70) est donc satisfaite et l'algorithme s'arrête après au plus $|\Omega| - 1$ tours.

Voici l'algorithme 2.9.1 ***Base***.

Algorithme 2.9.1 ***Base***$(\mathbf{R})$

Entrée : $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
où $\mathcal{P}$ est une propriété sur un ensemble fini Ω.

Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$ avec toutes les quantités associées
(cf. remarque 2.4.11, p. 71).

Début

1. $h := 0$;
2. $\mathbf{\Pi}_{h-1} := (\Omega)$;
3. $U_h, \mathbf{\Pi}_h :=$ ***Raffiner***$(\mathbf{R}, \mathbf{\Pi}_{h-1})$; $[\mathbf{\Pi}_0 = U_{0,\mathsf{G}}((\Omega))]$
4. `tant que` $\mathbf{\Pi}_h$ n'est pas discrète `faire`
5. $\quad h := h + 1$;
6. $\quad$ choisir $e_h \in I_n$ tel que $|\Pi_{h-1,e_h}| > 1$;
7. $\quad$ choisir $\alpha_h \in \Pi_{h-1,e_h}$;
8. $\quad U_h, \mathbf{\Pi}_h :=$ ***Raffiner***$(\mathbf{R}, \mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$; $[\mathbf{\Pi}_h = U_{h,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))]$
9. `fin tant que`;
10. $s := h$;
11. `retourner` $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$;

Fin.

L'algorithme ***Base*** n'est pas aussi simple qu'il peut en avoir l'air. Les lignes 6–8 n'expliquent pas entièrement comment nous choisissons (e_h, α_h) dans

$$\{(e, \alpha) \mid e \in I_n \text{ tel que } |\Pi_{h-1,e}| > 1, \alpha \in \Pi_{h-1,e}\}$$

et U_h dans $\mathbf{R}^*$. Tous les choix produisent une $\mathbf{R}$-base valable, mais l'efficacité de la procédure de recherche arrière (des algorithmes 2.5.1 ***Éléments***, 2.6.6 ***Sous-groupe*** et 2.7.1 ***RepClasse***) en dépend. Donc cela vaut probablement la peine de faire un effort supplémentaire pour choisir une **R**-base qui va accélérer la procédure ultérieure de recherche arrière.

Discutons un peu le choix des e_h et α_h. Si $\mathcal{P}$ est de type sous-groupe (respectivement, de type classe), alors $\Delta^{(h)} \subseteq \Pi_{h-1,e_h}$, où $\Delta^{(h)}$ est la h-ième orbite de base de $Sym(\Omega)_{\mathcal{P}}$ (respectivement, de $Sym(\Omega)_{\mathcal{P}_G}$) correspondant à la base $(\alpha_1, \ldots, \alpha_s)$ (cf. corollaire 2.4.15, p. 72). Par conséquent $\left(\Delta^{(h)}\right)^g \subseteq \Sigma_{h-1,e_h}$ pour toute permutation g de $Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}_{h-1}{}^g = \mathbf{\Sigma}_{h-1}$. Et $\left(\Delta^{(h)}\right)^g$ ne dépend pas de g, car s'il existe une autre permutation $g' \in Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}_{h-1}{}^{g'} = \mathbf{\Sigma}_{h-1}$, alors $\left(\Delta^{(h)}\right)^{g'} = \left(\Delta^{(h)}\right)^g$. En effet,

$$\mathit{fix}\,(\mathbf{\Pi}_{h-1})^{g'g^{-1}} = \mathit{fix}\,(\mathbf{\Sigma}_{h-1})^{g^{-1}} = \mathit{fix}\,(\mathbf{\Pi}_{h-1})\,,$$

donc $g'g^{-1}$ fixe tous les points de l'ensemble ordonné $\mathit{fix}\,(\mathbf{\Pi}_{h-1})$. Mais à partir de la construction des $\mathbf{\Pi}_i$, $0 \leq i \leq s$, il est facile à voir que tout α_i, $1 \leq i \leq h-1$, appartient à $\mathit{fix}\,(\mathbf{\Pi}_{h-1})$, donc $g'g^{-1}$ fixe les α_i, $1 \leq i \leq h-1$, c'est-à-dire $g'g^{-1} \in \left(Sym(\Omega)_{\mathcal{P}}\right)_{\alpha_1,\ldots,\alpha_{h-1}}$. Par conséquent $g'g^{-1}$ stabilise l'orbite $\Delta^{(h)} = {\alpha_h}^{(Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_{h-1}}}$ de α_h sous l'action de $\left(Sym(\Omega)_{\mathcal{P}}\right)_{\alpha_1,\ldots,\alpha_{h-1}}$ (cf. lemme 1.4.14, p. 25), c'est-à-dire

$$\left(\Delta^{(h)}\right)^{g'g^{-1}} = \Delta^{(h)} \quad \text{et} \quad \left(\Delta^{(h)}\right)^{g'} = \left(\Delta^{(h)}\right)^g.$$

Supposons que $\Delta^{(h)} \subsetneq \Pi_{h-1,e_h}$. Alors $\left(\Delta^{(h)}\right)^g \subsetneq \Sigma_{h-1,e_h}$ pour tout $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}_{h-1}{}^g = \mathbf{\Sigma}_{h-1}$. Au cours de la procédure de recherche arrière nous choisissons β_h dans Σ_{h-1,e_h} (cf. proposition 2.4.13(cii), p. 71). Donc si nous choisissons $\beta_h \in \Sigma_{h-1,e_h}$ tel que $\beta_h \notin \left(\Delta^{(h)}\right)^g$, il n'existera aucune permutation g de $Sym(\Omega)_{\mathcal{P}}$ telle que $\mathbf{\Pi}_{h-1}{}^g = \mathbf{\Sigma}_{h-1}$ et ${\alpha_h}^g = \beta_h$, c'est-à-dire nous perdrons du temps en considérant les $\beta_h \in \Sigma_{h-1,e_h} \setminus \left(\Delta^{(h)}\right)^g$. Mais nous ne connaissons pour l'instant aucun $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $\mathbf{\Pi}_{h-1}{}^g = \mathbf{\Sigma}_{h-1}$ (nous sommes à leur recherche), nous ne connaissons donc pas $\left(\Delta^{(h)}\right)^g$. Comment éviter cette perte du temps? En calculant une **R**-base nous pouvons nous efforcer de choisir les e_h tels que Π_{h-1,e_h} soit égal (ou presque) à $\Delta^{(h)}$. Alors Σ_{h-1,e_h} sera égal (ou presque) à $\left(\Delta^{(h)}\right)^g$.

Une heuristique à appliquer serait de choisir e_h tel que $|\Pi_{h-1,e_h}|$ soit minimal, i.e. tel que

$$|\Pi_{h-1,e_h}| = \min\{|\Pi_{h-1,e}| \mid e \in I_n, |\Pi_{h-1,e}| > 1\}.$$

Mais cela présente le désavantage suivant : en choisissant e_h ainsi, nous choisissons α_h tel que $\Delta^{(h)} = {\alpha_h}^{(Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_{h-1}}}$ est petit. Comme

$$|Sym(\Omega)_{\mathcal{P}}| = \left|\Delta^{(1)}\right| \left|\Delta^{(2)}\right| \cdots \left|\Delta^{(h)}\right| \left|\Delta^{(h+1)}\right| \cdots \left|\Delta^{(s)}\right|$$

(cf. la formule (1.10), p. 41) et $|Sym(\Omega)_{\mathcal{P}}|$ est fixé, choisir $\Delta^{(h)}$ plus petit fait augmenter

$$\left|\left(Sym(\Omega)_{\mathcal{P}}\right)_{\alpha_1,\ldots,\alpha_h}\right| = \frac{|Sym(\Omega)_{\mathcal{P}}|}{|\Delta^{(1)}|\,|\Delta^{(2)}| \cdots |\Delta^{(h)}|}$$

(cf. la formule (1.11), p. 42) et, probablement, s, ce qui veut dire que nous devrons considérer plus de permutations de $Sym(\Omega)_{\mathcal{P}}$ dans les niveaux plus bas de l'arbre de recherche arrière $\mathbb{T}_{\mathbf{R},\mathbf{B}}$, et que le chemin jusqu'aux partitions discrètes se rallonge, car la hauteur s de $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ augmente (ce qui fait augmenter la quantité de temps nécessaire pour calculer toute permutation de $Sym(\Omega)_{\mathcal{P}}$). Notre expérience avec l'algorithme indique que pour le problème du groupe de permutations des codes linéaires binaires, en utilisant notre ensemble de $\mathcal{P}_C$-raffinements élémentaires (défini dans la formule (3.6), p. 108), le $\mathbf{\Pi}_{h-1}$ obtenu est assez "proche" des $(Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_{h-1}}$-orbites, c'est-à-dire pour presque tout e_h possible nous avons $\Delta^{(h)} = \Pi_{h-1,e_h}$, il vaut mieux donc choisir e_h tel que $|\Pi_{h-1,e_h}|$ soit maximal, sauf, peut-être, pour certains codes "difficiles".

Nous présentons maintenant l'algorithme 2.9.2 ***Raffiner***. Dans les lignes 1–2 nous initialisons la partition courante $\mathbf{\Phi}$ et le $\mathcal{P}$-raffinement U. Dans la boucle principale (lignes 3–10) nous choisissons un $\mathcal{P}$-raffinement élémentaire E dans $\mathbf{R}$ (ligne 4), nous appliquons sa composante gauche E_{G} à la partition courante $\mathbf{\Phi}$ (ligne 5), et si nous arrivons à la raffiner, nous mettons E dans U (ligne 7) et nous mettons à jour la partition $\mathbf{\Phi}$ (ligne 8).

Algorithme 2.9.2 ***Raffiner***$(\mathbf{R}, \mathbf{\Pi})$
Entrée : $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
où $\mathcal{P}$ est une propriété sur un ensemble fini Ω,
$\mathbf{\Pi}$ une partition de Ω.
Sortie : U un $\mathcal{P}$-raffinement,
$\mathbf{\Phi}$ une partition de Ω telle que $\mathbf{\Phi} = U_{\mathsf{G}}(\mathbf{\Pi})$.
Début
1. $\mathbf{\Phi} := \mathbf{\Pi}$;
2. $U := (id, id)$; [le $\mathcal{P}$-raffinement trivial]
3. `tant que` la condition d'arrêt n'est pas satisfaite `faire`
4. choisir $E \in \mathbf{R}$;
5. $\mathbf{\Phi}' := E_{\mathsf{G}}(\mathbf{\Phi})$;
6. `si` $\mathbf{\Phi}' < \mathbf{\Phi}$ `alors`
7. $U := EU$;
8. $\mathbf{\Phi} := \mathbf{\Phi}'$;
9. `fin si`;
10. `fin tant que`;
11. `retourner` $U, \mathbf{\Phi}$;

Fin.

L'algorithme 2.9.2 ***Raffiner*** a l'air d'être simple, mais ce n'est pas le cas : les lignes 3 et 4 n'expliquent pas quand arrête-t-on la boucle principale et comment choisit-on un $\mathcal{P}$-raffinement élémentaire E dans $\mathbf{R}$. Pour le choix de E dans le cadre des problèmes liés aux codes voir section 3.8. Parlons un peu maintenant de la condition d'arrêt (ligne 3) de la boucle principale.

Si $|\mathbf{R}|$ est petit, il est possible d'essayer tous les E de $\mathbf{R}$, et d'arrêter la boucle quand on s'assure que la partition courante $\mathbf{\Phi}$ est $\mathbf{R}$-irréductible. Mais le plus souvent $|\mathbf{R}|$ est trop grand pour essayer tous les $E \in \mathbf{R}$, et, en plus, nous avons remarqué que pour le problème du groupe de permutations des codes linéaires binaires, en utilisant notre ensemble de $\mathcal{P}_C$-raffinements élémentaires (voir formule (3.6), p. 108), il suffisait de plusieurs essais (une dizaine ou une vingtaine) pour obtenir $\mathbf{\Phi}$ $\mathbf{R}$-irréductible. Nous pouvons donc prendre comme condition d'arrêt la condition suivante : “nous avons essayé b $\mathcal{P}$-raffinements élémentaires”, où b est un entier fixé d'avance. Actuellement dans notre implantation de cet algorithme pour le problème du groupe de permutations d'un code linéaire binaire nous utilisons la condition d'arrêt suivante : “nous avons fait b essais infructueux consécutifs”, où b est aussi fixé d'avance, c'est-à-dire nous arrêtons la boucle principale de ***Raffiner*** si les derniers b $\mathcal{P}$-raffinements élémentaires essayés n'ont pas réussi à raffiner la partition courante $\mathbf{\Phi}$. Voir section 3.8 pour plus de détails.

2.10 Implantation

Dans cette section nous donnons plus de détails sur le fonctionnement des algorithmes 2.5.1 ***Éléments***, 2.6.6 ***Sous-groupe***, 2.7.1 ***RepClasse*** et 2.9.1 ***Base***. Nous expliquons comment ces algorithmes fonctionnent en pratique et comment faire pour les accélérer.

Soit $\mathcal{P}$ une propriété sur un ensemble fini Ω.

2.10.1 Un vecteur auxiliaire

Le vecteur suivant est de grande utilité en réduisant l'espace de recherche de la méthode des partitions. Comme mentionné dans les commentaires sur la condition 4 de la définition 2.4.1 (voir p. 68), après avoir appliqué un $\mathcal{P}$-raffinement élémentaire $E = (E_{\mathsf{G}}, E_{\mathsf{D}})$ à deux partitions ordonnées $\mathbf{\Pi}$ et $\mathbf{\Sigma}$, il suffira de vérifier que $|E_{\mathsf{G}}(\mathbf{\Pi})_\infty| = |E_{\mathsf{D}}(\mathbf{\Sigma})_\infty|$ au lieu de vérifier $\|E_{\mathsf{G}}(\mathbf{\Pi})\| = \|E_{\mathsf{D}}(\mathbf{\Sigma})\|$, c'est-à-dire au lieu de vérifier l'égalité des cardinaux de toutes les cellules correspondantes de $E_{\mathsf{G}}(\mathbf{\Pi})$ et $E_{\mathsf{D}}(\mathbf{\Sigma})$, il suffira de ne vérifier que l'égalité des cardinaux des dernières cellules de $E_{\mathsf{G}}(\mathbf{\Pi})$ et $E_{\mathsf{D}}(\mathbf{\Sigma})$.

Notation 2.10.1 [Leo91, Déf.13] Si $R = E_l E_{l-1} \cdots E_1$ est un $\mathcal{P}$-raffinement, alors pour tout $\mathsf{z} \in \{\mathsf{G}, \mathsf{D}\}$ et tout $\mathbf{\Pi} \in \mathit{PartOrd}(\Omega)$, nous définissons $\mathit{TaillesdeDivisions}_{\mathsf{z}}(R, \mathbf{\Pi})$ comme suit :

$$\mathit{TaillesdeDivisions}_{\mathsf{z}}(R, \mathbf{\Pi}) = (|\Pi_{1,d_1}|, |\Pi_{2,d_2}|, \ldots, |\Pi_{l,d_l}|),$$

où $\mathbf{\Pi}_0, \ldots, \mathbf{\Pi}_l$ sont les partitions ordonnées définies par induction comme suit : $\mathbf{\Pi}_0 = \mathbf{\Pi}$ et $\mathbf{\Pi}_i = E_{i,\mathsf{z}}(\mathbf{\Pi}_{i-1})$, et où d_i indique $|\mathbf{\Pi}_{i-1}| + 1$.

Autrement dit, $\mathit{TaillesdeDivisions}_{\mathsf{G}}(R, \mathbf{\Pi})$ est un vecteur dont les composantes sont les cardinaux, au moment de la création, des nouvelles cellules créées par l'application des composantes gauches des raffinements élémentaires $E_1, \ldots, E_l$ de R à $\mathbf{\Pi}$. Notez que si $E_{i,\mathsf{G}}$ raffine la partition $\mathbf{\Pi}_{i-1} = E_{i-1,\mathsf{G}} \cdots E_{1,\mathsf{G}}(\mathbf{\Pi})$, alors $\Pi_{i,d_i} = \Pi_{i,\infty}$ — la dernière cellule de

$\mathbf{\Pi}_i$, sinon $\Pi_{i,d_i} = \emptyset$. C'est valable aussi pour $\mathit{TaillesdeDivisions}_{\mathsf{D}}(R, \mathbf{\Pi})$, en remplaçant G par D.

Proposition 2.10.2 *[Leo91, p.543] Si $R = E_l E_{l-1} \cdots E_1$ est un $\mathcal{P}$-raffinement, alors pour tous les $\mathbf{\Pi}, \mathbf{\Sigma} \in \mathit{PartOrd}(\Omega)$ et pour tout $g \in \mathit{Sym}(\Omega)$,*

$$\mathcal{P}(g) \textit{ et } \mathbf{\Pi}^g = \mathbf{\Sigma} \textit{ impliquent } \mathit{TaillesdeDivisions}_{\mathsf{G}}(R, \mathbf{\Pi}) = \mathit{TaillesdeDivisions}_{\mathsf{D}}(R, \mathbf{\Sigma}).$$

Preuve. Il suffit de remarquer que si $\mathbf{\Pi}', \mathbf{\Sigma}' \in \mathit{PartOrd}(\Omega)$ tels que $\mathbf{\Pi}'^g = \mathbf{\Sigma}'$ (en particulier, $|\mathbf{\Pi}'| = |\mathbf{\Sigma}'|$), alors pour tout $\mathcal{P}$-raffinement élémentaire E on a :

1. $E_{\mathsf{G}}(\mathbf{\Pi}')^g = E_{\mathsf{D}}(\mathbf{\Sigma}')$ (en vertu de la condition 3 de la définition 2.4.1, p. 66) et

2. $$\left| E_{\mathsf{G}}(\mathbf{\Pi}')_{|\mathbf{\Pi}'|+1} \right| = \left| E_{\mathsf{D}}(\mathbf{\Sigma}')_{|\mathbf{\Sigma}'|+1} \right|,$$
   car $|\mathbf{\Pi}'| = |\mathbf{\Sigma}'|$ et la partie 1 implique que
   $$|E_{\mathsf{G}}(\mathbf{\Pi}')_i| = |E_{\mathsf{D}}(\mathbf{\Sigma}')_i| \ \forall i \geq 1$$
   (si E raffine $\mathbf{\Pi}'$ et $\mathbf{\Sigma}'$, alors $E_{\mathsf{G}}(\mathbf{\Pi}')_{|\mathbf{\Pi}'|+1}$ est la dernière cellule de $E_{\mathsf{G}}(\mathbf{\Pi}')$, i.e. $E_{\mathsf{G}}(\mathbf{\Pi}')_{|\mathbf{\Pi}'|+1} = E_{\mathsf{G}}(\mathbf{\Pi}')_{\infty}$, sinon $E_{\mathsf{G}}(\mathbf{\Pi}')_{|\mathbf{\Pi}'|+1} = \emptyset$, et de même pour $E_{\mathsf{D}}(\mathbf{\Sigma}')_{|\mathbf{\Sigma}'|+1}$ en remplaçant G par D et $\mathbf{\Pi}'$ par $\mathbf{\Sigma}'$ ci-dessus).

Maintenant nous appliquons ce raisonnement pour

$$\mathbf{\Pi}' = E_{i,\mathsf{G}} E_{i-1,\mathsf{G}} \cdots E_{1,\mathsf{G}}(\mathbf{\Pi}) \quad \text{et} \quad \mathbf{\Sigma}' = E_{i,\mathsf{D}} E_{i-1,\mathsf{D}} \cdots E_{1,\mathsf{D}}(\mathbf{\Sigma})$$

pour $i = 0, 1, \ldots, l-1$ (dans cet ordre), et nous obtenons le résultat désiré. □

Proposition 2.10.3 *Soient $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ des partitions ordonnées de Ω telles que $\|\mathbf{\Pi}\| = \|\mathbf{\Sigma}\|$. Soit $R = E_l E_{l-1} \cdots E_1$ un $\mathcal{P}$-raffinement. Si*

$$\mathit{TaillesdeDivisions}_{\mathsf{G}}(R, \mathbf{\Pi}) = \mathit{TaillesdeDivisions}_{\mathsf{D}}(R, \mathbf{\Sigma}), \tag{2.9}$$

alors

$$\|E_{i,\mathsf{G}} E_{i-1,\mathsf{G}} \cdots E_{1,\mathsf{G}}(\mathbf{\Pi})\| = \|E_{i,\mathsf{D}} E_{i-1,\mathsf{D}} \cdots E_{1,\mathsf{D}}(\mathbf{\Sigma})\| \quad \textit{pour tout } i, 1 \leq i \leq l,$$

et, en particulier, $\|R_{\mathsf{G}}(\mathbf{\Pi})\| = \|R_{\mathsf{D}}(\mathbf{\Sigma})\|$.

Preuve. Supposons que la condition (2.9) soit satisfaite. Nous avons, en particulier,

$$\left| E_{1,\mathsf{G}}(\mathbf{\Pi})_{|\mathbf{\Pi}|+1} \right| = \left| E_{1,\mathsf{D}}(\mathbf{\Sigma})_{|\mathbf{\Sigma}|+1} \right|.$$

Montrons que $\|E_{1,\mathsf{G}}(\mathbf{\Pi})\| = \|E_{1,\mathsf{D}}(\mathbf{\Sigma})\|$.

Si $E_{1,\mathsf{G}}$ laisse $\mathbf{\Pi}$ inchangé, alors $E_{1,\mathsf{G}}(\mathbf{\Pi})_{|\mathbf{\Pi}|+1} = \emptyset$ et donc $E_{1,\mathsf{D}}(\mathbf{\Sigma})_{|\mathbf{\Sigma}|+1} = \emptyset$, par conséquent $E_{1,\mathsf{D}}$ laisse $\mathbf{\Sigma}$ inchangé aussi, d'où

$$\|E_{1,\mathsf{G}}(\mathbf{\Pi})\| = \|\mathbf{\Pi}\| = \|\mathbf{\Sigma}\| = \|E_{1,\mathsf{D}}(\mathbf{\Sigma})\|.$$

Si $E_{1,\mathsf{G}}$ raffine $\mathbf{\Pi}$, alors $E_{1,\mathsf{D}}$ raffine $\mathbf{\Sigma}$ aussi. D'après la condition 4 de la définition 2.4.1, p. 66, il existe un entier j tel que E_1 divise les j-ièmes cellules de $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ en deux parties, les autres cellules restant inchangées. Donc

$$\left|E_{1,\mathsf{G}}(\mathbf{\Pi})_i\right| = |\Pi_i| = |\Sigma_i| = \left|E_{1,\mathsf{D}}(\mathbf{\Sigma})_i\right| \ \forall\, i, i \neq j.$$

Et nous avons que

$$\left|E_{1,\mathsf{G}}(\mathbf{\Pi})_j\right| = |\Pi_j| - \left|E_{1,\mathsf{G}}(\mathbf{\Pi})_{|\mathbf{\Pi}|+1}\right| = |\Sigma_j| - \left|E_{1,\mathsf{D}}(\mathbf{\Sigma})_{|\mathbf{\Sigma}|+1}\right| = \left|E_{1,\mathsf{D}}(\mathbf{\Sigma})_j\right|.$$

Donc $\|E_{1,\mathsf{G}}(\mathbf{\Pi})\| = \|E_{1,\mathsf{D}}(\mathbf{\Sigma})\|$.

De la même façon on peut montrer que

$$\begin{aligned} \|E_{2,\mathsf{G}}E_{1,\mathsf{G}}(\mathbf{\Pi})\| &= \|E_{2,\mathsf{D}}E_{1,\mathsf{D}}(\mathbf{\Sigma})\|, \\ &\cdots \\ \|E_{l,\mathsf{G}}E_{l-1,\mathsf{G}}\cdots E_{1,\mathsf{G}}(\mathbf{\Pi})\| &= \|E_{l,\mathsf{D}}E_{l-1,\mathsf{D}}\cdots E_{1,\mathsf{D}}(\mathbf{\Sigma})\|, \end{aligned}$$

et donc en particulier $\|R_{\mathsf{G}}(\mathbf{\Pi})\| = \|R_{\mathsf{D}}(\mathbf{\Sigma})\|$. □

2.10.2 Quantités associées à une R-base

Notation 2.10.4 [Leo91, Déf.28] Étant donnée une $\mathbf{R}$-base $\mathbf{B}$ pour $\mathcal{P}$ avec toutes les quantités associées (cf. remarque 2.4.11, p. 71), nous définissons les quantités suivantes :

(i) $E^{(h)}_{l_h}E^{(h)}_{l_h-1}\cdots E^{(h)}_1$ est la décomposition de U_h, $0 \leq h \leq s$, en $\mathcal{P}$-raffinements élémentaires de $\mathbf{R}$, c'est-à-dire $\forall\, h, 0 \leq h \leq s$, $U_h = E^{(h)}_{l_h}E^{(h)}_{l_h-1}\cdots E^{(h)}_1$, où $E^{(h)}_i \in \mathbf{R}$ pour tout i, $1 \leq i \leq l_h$.

(ii) pour tout h, $0 \leq h \leq s$, la suite $(\mathbf{\Phi}^{(h)}_0, \ldots, \mathbf{\Phi}^{(h)}_{l_h})$ est définie par

$$\mathbf{\Phi}^{(h)}_0 = \begin{cases} \mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}) & \text{si } h \neq 0, \\ \mathbf{\Pi}_{h-1} & \text{sinon,} \end{cases} \qquad \text{et } \mathbf{\Phi}^{(h)}_i = E^{(h)}_{i,\mathsf{G}}(\mathbf{\Phi}^{(h)}_{i-1}) \text{ pour tout } i,\ 1 \leq i \leq l_h.$$

(iii)

$$\mathbf{a}_0 \stackrel{\text{déf}}{=} (a^{(0)}_1, \ldots, a^{(0)}_{l_0}) = \mathit{TaillesdeDivisions}_{\mathsf{G}}(U_0, (\Omega))$$

et

$$\mathbf{a}_h \stackrel{\text{déf}}{=} (a^{(h)}_1, \ldots, a^{(h)}_{l_h}) = \mathit{TaillesdeDivisions}_{\mathsf{G}}(U_h, \mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1})),\ 1 \leq h \leq s.$$

Propriété 2.10.5 *Les objets définis ci-dessus satisfont les relations suivantes :*

(i)

$$\mathbf{\Phi}_i^{(0)} = E_{i,\mathsf{G}}^{(0)} E_{i-1,\mathsf{G}}^{(0)} \cdots E_{1,\mathsf{G}}^{(0)}((\Omega)) \text{ pour tout } i,\ 0 \leq i \leq l_0,\ \text{ et}$$
$$\mathbf{\Phi}_i^{(h)} = E_{i,\mathsf{G}}^{(h)} E_{i-1,\mathsf{G}}^{(h)} \cdots E_{1,\mathsf{G}}^{(h)}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$$

pour tout i, $0 \leq i \leq l_h$, *et tout* h, $1 \leq h \leq s$ *;*

(ii) $\mathbf{\Phi}_{l_0}^{(0)} = U_{0,\mathsf{G}}((\Omega)) = \mathbf{\Pi}_0$ *et* $\mathbf{\Phi}_{l_h}^{(h)} = U_{h,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1})) = \mathbf{\Pi}_h$ *pour tout* h, $1 \leq h \leq s$ *;*

(iii) pour tout h, $0 \leq h \leq s$, *et tout* i, $1 \leq i \leq l_h$,

$$(a_i^{(h)}) = \mathit{TaillesdeDivisions}_{\mathsf{G}}(E_i^{(h)}, \mathbf{\Phi}_{i-1}^{(h)}) = \left(\left|\mathbf{\Phi}^{(h)}_{i,\left|\mathbf{\Phi}_{i-1}^{(h)}\right|+1}\right|\right).$$

Preuve. (i) découle de la notation 2.10.4(ii) ; (ii) de (i), de la notation 2.10.4(i) et de la formule (2.7), p. 71 ; (iii) de (i) et des notations 2.10.1 et 2.10.4. □

Étant donnée une **R**-base **B** pour $\mathcal{P}$, toutes les quantités de la notation 2.10.4 sont facilement calculables. Par la suite nous supposerons que, une fois **B** calculée, elles sont disponibles et peuvent être référencées commodément.

2.10.3 Une proposition importante

Supposons que nous ayons une **R**-base $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ pour $\mathcal{P}$ avec les quantités associées (cf. remarque 2.4.11, p. 71, et notation 2.10.4, p. 91). Supposons que nous ayons une suite $(\beta_1, \ldots, \beta_s)$ d'éléments de Ω. Comment trouver les permutations $g \in \mathit{Sym}(\Omega)_{\mathcal{P}}$ telles que $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$? La proposition 2.4.13, p. 71, répond à cette question. Nous donnons maintenant les versions plus détaillées des conditions (ciii) et (civ) de cette proposition.

Proposition 2.10.6 *Soit* $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$ *une* **R***-base pour une propriété* $\mathcal{P}$ *sur un ensemble fini* Ω, *soit* $(\beta_1, \ldots, \beta_s)$ *une suite de points de* Ω. *Si une permutation* g *de* $\mathit{Sym}(\Omega)_{\mathcal{P}}$ *telle que* $(\alpha_1, \ldots, \alpha_s)^g = (\beta_1, \ldots, \beta_s)$ *existe, et si la suite* $(\mathbf{\Sigma}_0, \ldots, \mathbf{\Sigma}_s)$ *de partitions ordonnées de* Ω *est définie par la formule (2.8), p. 72 (notons, de plus,* $\mathbf{\Sigma}_{-1} = (\Omega)$*), alors* g *est unique et*

(i) $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_0, (\Omega)) = \mathbf{a}_0$, *et* $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h, \mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})) = \mathbf{a}_h$ *pour tout* $h = 1, \ldots, s$,

(ii) pour tout groupe G *tel que la propriété* $\mathcal{P}$ *contient* $\mathcal{E}_G$ *(cf. remarque 2.2.3, p. 55), on a* $\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ *pour tout* h, $0 \leq h \leq s$, *et, de plus, ce résultat est valable pour toute paire de partitions intermédiaire entre* $(\mathbf{\Pi}_{h-1}, \mathbf{\Sigma}_{h-1})$ *et* $(\mathbf{\Pi}_h, \mathbf{\Sigma}_h)$, $0 \leq h \leq s$, *plus précisément,*

$$\mathit{fix}\left(E_{i,\mathsf{D}}^{(0)} E_{i-1,\mathsf{D}}^{(0)} \cdots E_{1,\mathsf{D}}^{(0)}((\Omega))\right) \in \mathit{fix}\left(E_{i,\mathsf{G}}^{(0)} E_{i-1,\mathsf{G}}^{(0)} \cdots E_{1,\mathsf{G}}^{(0)}((\Omega))\right)^G \ \forall\, i, 0 \leq i \leq l_0,$$

et

$$\mathit{fix}\left(E^{(h)}_{i,\mathsf{D}}E^{(h)}_{i-1,\mathsf{D}}\cdots E^{(h)}_{1,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))\right)\in \mathit{fix}\left(E^{(h)}_{i,\mathsf{G}}E^{(h)}_{i-1,\mathsf{G}}\cdots E^{(h)}_{1,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))\right)^G$$

$\forall\, i,\ 0\le i\le l_h,\ \forall\, h,\ 1\le h\le s.$

Notez que, compte tenu de la proposition 2.10.3, p. 90, la condition (i) implique que $\|\mathbf{\Pi}_h\|=\|\mathbf{\Sigma}_h\|$ pour tout h, $0\le h\le s$ (ce qui est annoncé dans la proposition 2.4.13(ciii), p. 71) et que, de plus, cette égalité est vraie pour toute paire de partitions intermédiaire entre $(\mathbf{\Pi}_{h-1},\mathbf{\Sigma}_{h-1})$ et $(\mathbf{\Pi}_h,\mathbf{\Sigma}_h)$, $0\le h\le s$, plus précisément,

$$\left\|E^{(0)}_{i,\mathsf{G}}E^{(0)}_{i-1,\mathsf{G}}\cdots E^{(0)}_{1,\mathsf{G}}((\Omega))\right\|=\left\|E^{(0)}_{i,\mathsf{D}}E^{(0)}_{i-1,\mathsf{D}}\cdots E^{(0)}_{1,\mathsf{D}}((\Omega))\right\|\ \ \forall\, i,0\le i\le l_0,$$

et

$$\left\|E^{(h)}_{i,\mathsf{G}}E^{(h)}_{i-1,\mathsf{G}}\cdots E^{(h)}_{1,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))\right\|=\left\|E^{(h)}_{i,\mathsf{D}}E^{(h)}_{i-1,\mathsf{D}}\cdots E^{(h)}_{1,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))\right\|$$

$\forall\, i,0\le i\le l_h,\ \forall\, h,1\le h\le s.$

2.10.4 Méthode des partitions

Dans cette section nous donnerons plus de détails sur le fonctionnement des algorithmes 2.5.1 ***Éléments***, 2.6.6 ***Sous-groupe*** et 2.7.1 ***RepClasse***.

2.10.4.1 Remarque pour tous les algorithmes

Dans cette section nous décrivons plus en détails l'exécution des lignes 1'–2 et 9'–10 des algorithmes 2.5.1 ***Éléments***, 2.6.6 ***Sous-groupe*** et 2.7.1 ***RepClasse***.

Les lignes 9' et 10 (comme les lignes 1' et 2) ne s'exécutent pas l'une après l'autre, plutôt elles se mélangent. Après avoir appliqué un $\mathcal{P}$-raffinement élémentaire composant U_h, nous vérifions l'égalité des cardinaux des cellules correspondantes des partitions courantes $\mathbf{\Pi}$ et $\mathbf{\Sigma}$, et nous vérifions que $\mathit{fix}\,(\mathbf{\Sigma})\in \mathit{fix}\,(\mathbf{\Pi})^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$. Comme c'est montré dans la proposition 2.10.3, pour vérifier l'égalité des cardinaux des cellules correspondantes de $\mathbf{\Pi}$ et $\mathbf{\Sigma}$, il suffit de vérifier l'égalité des cardinaux des dernières cellules de $\mathbf{\Pi}$ et $\mathbf{\Sigma}$. Concrètement, dans la ligne 10 nous vérifions la condition $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h,\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))=\mathbf{a}_h$. Voyons cela de plus près.

Nous savons que le $\mathcal{P}$-raffinement U_h, $0\le h\le s$, se décompose en $\mathcal{P}$-raffinements élémentaires comme suit : $U_h=E^{(h)}_{l_h}E^{(h)}_{l_h-1}\cdots E^{(h)}_1$ (cf. la notation 2.10.4(i)). Nous commençons l'exécution des lignes 9' et 10 par le calcul de $\mathbf{\Sigma}'\overset{\text{déf}}{=}\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})$. Ensuite nous calculons $\mathbf{\Sigma}''=E^{(h)}_{1,\mathsf{D}}(\mathbf{\Sigma}')$ et nous vérifions que le cardinal de la cellule $|\mathbf{\Sigma}'|+1$ de $\mathbf{\Sigma}''$ est égal à $a^{(h)}_1$ (la première composante du vecteur $\mathbf{a}_h$), i.e. nous vérifions l'égalité des premières composantes des vecteurs $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h,\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$ et $\mathbf{a}_h$. Si elles ne sont pas égales, nous arrêtons tout de suite le calcul de $\mathbf{\Sigma}_h$, car la proposition 2.10.6(i) nous garantit qu'il n'existe pas $g\in \mathit{Sym}(\Omega)_{\mathcal{P}}$ tel que $(\alpha_1,\dots,\alpha_s)^g=(\beta_1,\dots,\beta_s)$, cela ne vaut pas la peine de continuer le calcul. Nous arrêtons donc le calcul de $\mathbf{\Sigma}_h$ et nous agissons comme si la condition de la ligne 10 ne serait pas satisfaite, i.e. nous passons tout de suite à la 19-ième ligne. Si ces composantes sont égales, nous appliquons le $\mathcal{P}$-raffinement élémentaire suivant $E_2^{(h)}$ à la partition courante $\mathbf{\Sigma}''$, nous obtenons $\mathbf{\Sigma}''' \stackrel{\text{déf}}{=} E_{2,\mathsf{D}}^{(h)}(\mathbf{\Sigma}'')$ et nous vérifions si $\left|\Sigma'''_{|\Sigma''|+1}\right| = a_2^{(h)}$, etc... En résumé, le calcul de $\mathbf{\Sigma}_h = U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$ se fait par les pas élémentaires, et à chaque pas nous pouvons arrêter le calcul si la condition $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h, \mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})) = \mathbf{a}_h$ n'est pas satisfaite.

D'ailleurs, compte tenu de la proposition 2.10.6(ii), pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$, de la même manière à chaque pas nous vérifions que

$$\mathit{fix}\,(\mathbf{\Sigma}'') \in \mathit{fix}\left(\mathbf{\Phi}_1^{(h)}\right)^G,\ \mathit{fix}\,(\mathbf{\Sigma}''') \in \mathit{fix}\left(\mathbf{\Phi}_2^{(h)}\right)^G,\ \text{etc.}\ldots,$$

où

$$\mathit{fix}\left(\mathbf{\Phi}_i^{(h)}\right) = \mathit{fix}\left(E_{i,\mathsf{G}}^{(h)} E_{i-1,\mathsf{G}}^{(h)} \cdots E_{1,\mathsf{G}}^{(h)}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))\right),\ 0 \le i \le l_h,$$

sont précalculés pour tout $h,\ 0 \le h \le s$ (cf. notation 2.10.4 et propriété 2.10.5).

2.10.4.2 Problèmes de type sous-groupe

Comment réaliser en pratique la vérification des conditions de la proposition 2.6.3, p. 77?

Pour réaliser cela, nous maintenons plusieurs variables auxiliaires.

D'abord, une variable r, égale à tout moment à $\min\{j \mid 1 \le j \le h,\ \alpha_j \ne \beta_j\}$. Nous l'initialisons à $s+1$ (ligne 1a), et nous la mettons à jour dans la ligne 9a — après avoir pris un nouveau β_h, nous vérifions si $h < r$, et si c'est le cas et si $\beta_h \ne \alpha_h$, nous actualisons r. Notez que nous parcourons les solutions partielles $(\beta_1, \ldots, \beta_h)$ dans l'ordre lexicographique induit par l'ordre $\prec$ de la notation 2.4.12 (noté aussi $\prec$), c'est-à-dire nous parcourons d'abord les suites commençant par $(\alpha_1, \ldots, \alpha_{s-1})$, ensuite celles commençant par $(\alpha_1, \ldots, \alpha_{s-2})$, etc..., celles commençant par (α_1), et enfin celles qui commencent par (γ), $\gamma \in \Omega$, $\gamma \ne \alpha_1$. Donc au cours de l'exécution de l'algorithme r prendra les valeurs $s, s-1, s-2, \ldots, 1$. La variable r tiendra le rôle de $\mathcal{L}(B, g)$ de la proposition 2.6.3. En effet, si pour une solution partielle $(\beta_1, \ldots, \beta_h)$, il existe $g \in \mathit{Sym}(\Omega)_{\mathcal{P}}$, $g \ne \mathit{id}$, tel que $(\alpha_1, \ldots, \alpha_h)^g = (\beta_1, \ldots, \beta_h)$, alors

$$\mathcal{L}(B, g) = \min\{j \mid 1 \le j \le s,\ \alpha_j \ne \beta_j\} = \min\{j \mid 1 \le j \le h,\ \alpha_j \ne \beta_j\} = r.$$

Nous l'utiliserons donc au lieu de $\mathcal{L}(B, g)$ dans la vérification des conditions de la proposition 2.6.3. Notez qu'à tout moment les permutations obtenues jusqu'alors fixent la suite $(\alpha_1, \ldots, \alpha_{r-1})$.

Pour vérifier la condition 2 de la proposition 2.6.3, nous introduisons une variable *orbreps*. C'est une liste de taille $n = |\Omega|$, indexée par les éléments de Ω, dont chaque composante $\mathit{orbreps}\,[\gamma]$, $\gamma \in \Omega$, à tout moment garde le plus petit élément (suivant l'ordre $\prec$ de Ω défini dans la notation 2.4.12, p. 71) de l'orbite de γ sous l'action du sous-groupe

$K =< s_1, \dots, s_j >$ de $Sym(\Omega)_{\mathcal{P}}$ trouvé jusqu'alors (c'est-à-dire $s_1, \dots, s_j$ sont les permutations de $Sym(\Omega)_{\mathcal{P}}$ déjà trouvées à ce moment). Deux éléments γ_1, γ_2 de Ω se trouvent dans la même K-orbite si et seulement si $orbreps\,[\gamma_1] = orbreps\,[\gamma_2]$, et $\gamma \in \Omega$ est le plus petit dans son K-orbite si et seulement si $orbreps\,[\gamma] = \gamma$. Nous initialisons *orbreps* dans la ligne 1b de façon qu'elle corresponde à $K = \{id\}$. Quand nous trouvons un nouveau élément $g = s_{j+1}$ de $Sym(\Omega)_{\mathcal{P}}$ (lignes 11'–13), le nouveau $K =< s_1, \dots, s_j, s_{j+1} >$ devient plus grand que l'ancien K, il faut donc ajuster *orbreps* pour qu'elle corresponde à ce nouveau K. Nous le faisons dans la ligne 13b au moyen de la fonction ***Joindre***. Cette fonction (cf. algorithme 2.10.7) prend en entrée Ω, les orbites sous l'action de $K =< s_1, \dots, s_j >$ sous forme de *orbreps*, et la nouvelle permutation g. Elle sort les orbites sous l'action de $< K, g >$ sous forme de *orbreps* ajustée. Elle est basée sur la définition 1.5.7, p. 29. En effet, on peut montrer que la partition formée par les orbites sous l'action de $< K, g >$ est l'union de deux partitions : celle présentée par *orbreps* (i.e. formée par les orbites sous l'action de $K =< s_1, \dots, s_j >$), et celle induite par les cycles de g (cf. lemme 1.4.4, p. 22). La fonction ***Joindre*** parcourt tous les éléments γ de Ω et vérifie si γ et γ^g appartiennent aux K-orbites différentes (i.e. si $orbreps\,[\gamma] \neq orbreps\,[\gamma^g]$), si c'est le cas, elle joint les orbites de γ et γ^g (i.e. choisit le plus petit de $orbreps\,[\gamma]$ et $orbreps\,[\gamma^g]$ qui est par définition le plus petit élément de la réunion $O_1 \cup O_2$ des orbites O_1 et O_2 de γ et γ^g, respectivement, et affecte cette valeur minimale à toutes les composantes $orbreps\,[\phi]$, $\phi \in O_1 \cup O_2$).

Algorithme 2.10.7 ***Joindre***$(\Omega, orbreps, g)$

Entrée : Ω un ensemble fini,
orbreps une fonction de Ω dans Ω présentant une partition de Ω, plus précisément, $orbreps\,[\gamma]$, $\gamma \in \Omega$, est le plus petit élément de la cellule de γ,
g une permutation de Ω.

Sortie : *orbreps*1 une fonction de même type que *orbreps* présentant la partition de Ω égale à l'union (cf. définition 1.5.7, p. 29) de deux partitions : celle présentée par *orbreps*, et celle induite par les cycles de g (cf. lemme 1.4.4, p. 22).

Début

```
pour γ ∈ Ω faire
  si orbreps[γ] ≠ orbreps[γ^g] alors
    mini := min(orbreps[γ], orbreps[γ^g]);
    maxi := max(orbreps[γ], orbreps[γ^g]);
    orbreps[φ] := mini pour tout φ ∈ Ω tel que orbreps[φ] = maxi;
  fin si;
fin pour;
retourner orbreps;
```

Fin.

Pour tester la condition 2 de la proposition 2.6.3, nous prenons pour M le sous-groupe K de $Sym(\Omega)_{\mathcal{P}}$ déjà construit. Il nous faut avoir $M_{\alpha_1, \dots, \alpha_{h-1}}$, mais si

$$h = r = \min\{j \mid 1 \le j \le h,\ \alpha_j \neq \beta_j\},$$

toutes les permutations de K fixent $(\alpha_1, \ldots, \alpha_{r-1}) = (\alpha_1, \ldots, \alpha_{h-1})$; au lieu de $M_{\alpha_1,\ldots,\alpha_{h-1}}$ nous utilisons donc $K_{\alpha_1,\ldots,\alpha_{h-1}} = K$.

Pour tester la condition 2, il faut donc vérifier que si $h = r$, alors β_h est le plus petit dans sa K-orbite, i.e. comme nous l'avons vu, *orbreps* $[\beta_h] = \beta_h$. C'est ce que fait la ligne 9b de l'algorithme 2.10.8 ***Sous-groupe***.

Passons maintenant à la condition 3 de la proposition 2.6.3. Pour la vérifier, il faut savoir si $\alpha_h \in \Delta^{(i)}$, où $\Delta^{(i)}$ est la i-ième orbite de base de $K =< s_1, \ldots, s_j >$. Pour le faire d'une manière plus efficace, nous introduisons une liste Y à deux dimensions telle que $Y[i,h] = vrai$ si et seulement si $\alpha_h \in \Delta^{(i)}$, $1 \leq i < h \leq s$. Elle est initialisée (ligne 1c) de façon qu'elle corresponde à $K = \{id\}$. Quand on trouve un nouveau élément g de $Sym(\Omega)_{\mathcal{P}}$, on ajuste Y (ligne 13c). Au cours de l'exécution de l'algorithme, à tout moment nous avons déjà $(Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_r}$, et nous sommes en train de calculer $(Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_{r-1}}$ (en effet, nous avons déjà essayé toutes les solutions partielles courantes $(\alpha_1, \ldots, \alpha_{r-1}, \alpha_r, \beta_{r+1}, \ldots)$, et nous essayons maintenant $(\alpha_1, \ldots, \alpha_{r-1}, \beta_r, \ldots)$). Quand nous trouvons une nouvelle permutation g de $Sym(\Omega)_{\mathcal{P}}$, nous avons donc que $g \in (Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_{r-1}} \setminus (Sym(\Omega)_{\mathcal{P}})_{\alpha_1,\ldots,\alpha_r}$, et l'ajout de g ne change que la r-ième orbite de base $\Delta^{(r)}$. Donc pour ajuster Y il suffira de vérifier si $\alpha_j \in \Delta^{(r)}$, $r < j \leq s$. C'est exactement ce que fait la ligne 13c.

L'implantation de la condition 3 de la proposition 2.6.3 (ligne 9c de l'algorithme 2.10.8 ***Sous-groupe***) est alors évidente.

Il nous reste la condition 1 de la proposition 2.6.3. Nous introduisons une variable *orbtailles*. C'est une liste telle que *orbtailles* $[h] = \left|\Delta^{(h)}\right|$, $1 \leq h \leq s$, où $\Delta^{(h)}$ est la h-ième orbite de base du sous-groupe $K =< s_1, \ldots, s_j >$ de $Sym(\Omega)_{\mathcal{P}}$ trouvé jusqu'alors. Elle est initialisé (ligne 1d) de façon qu'elle corresponde à $K = \{id\}$. Nous l'ajustons (ligne 13d) chaque fois que nous trouvons un nouveau élément de $Sym(\Omega)_{\mathcal{P}}$. Comme c'est mentionné ci-dessus, ce n'est que la r-ième orbite de base $\Delta^{(r)}$ qui change, il suffit donc de recalculer *orbtailles* $[r]$.

La vérification de la condition 1 de la proposition 2.6.3 (ligne 8 de l'algorithme) est évidente.

Nous présentons maintenant cette version détaillée de l'algorithme ***Sous-groupe***.

Algorithme 2.10.8 ***Sous-groupe***$(\mathcal{P}, \mathbf{R})$

Entrée : $\mathcal{P}$ une propriété sur un ensemble fini Ω (rappelons que $\mathcal{P}$ contient
au moins une propriété $\mathcal{E}_G$, cf. remarque 2.2.3, p. 55),
$\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires.

Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$, toutes les quantités associées
(cf. remarque 2.4.11, p. 71, et notation 2.10.4, p. 91) incluses,
X un ensemble fort de générateurs de $Sym(\Omega)_{\mathcal{P}}$
associé à la base $B = (\alpha_1, \ldots, \alpha_s)$ correspondant à la $\mathbf{R}$-base $\mathbf{B}$.

Début

0. $(\mathbf{U}, \mathbf{T}) :=$ ***Base*** $(\mathbf{R})$;
1. $X := \emptyset$;

1a. $r := s + 1$;

1b. $orbreps\,[\gamma] := \gamma$ pour tout $\gamma \in \Omega$;
1c. $Y\,[i,j] := faux$ pour tous les i et$j$ tels que $1 \leq i < j \leq s$;
1d. $orbtailles\,[h] := 1$ pour tout $h,\ 1 \leq h \leq s$;
1'. $\mathbf{\Sigma}_0 := \mathbf{\Pi}_0$;
5. $h := 1$;
6. $\Gamma_h := \Sigma_{h-1,e_h}$;
7. `tant que` $h > 0$ `faire`
8. `si` $|\Gamma_h| \geq orbtailles\,[h]$ `alors` [la vérification de la cond. 1 de la prop. 2.6.3]
9. $\beta_h := \min(\Gamma_h)$; $\Gamma_h := \Gamma_h \setminus \{\beta_h\}$;
9a. `si` $h < r$ et $\alpha_h \neq \beta_h$ `alors` $r := h$; `fin si`; [la mise à jour de r]
9b. `si` ($r \neq h$ ou $orbreps\,[\beta_h] = \beta_h$) et
9c. (pour tout i, $r \leq i \leq h-1$, tel que $Y\,[i,h] = vrai$, on a $\beta_i \prec \beta_h$) `alors`
9'. $\mathbf{\Sigma}_h := U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$;
10. `si` $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h, \mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})) = \mathbf{a}_h$ et
($\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) `alors`
11. `si` $h = s$ `alors`
11'. $g :=$ l'unique solution de l'équation ${\mathbf{\Pi}_s}^g = \mathbf{\Sigma}_s$;
12. `si` $\mathcal{P}(g)$ et $g \neq id$ `alors`
13. $X := X \cup \{g\}$; [ajouter g à X]
13a. $h := r$;
13b. $orbreps :=$ ***Joindre*** $(\Omega, orbreps, g)$; [la mise à jour de $orbreps$,]
13c. $Y\,[r,j] := vrai$ pour les j, $r+1 \leq j \leq s$,
tels que $orbreps\,[\alpha_j] = \alpha_r$; [de Y,]
13d. $orbtailles\,[r] := |\{\gamma \in \Omega \mid orbreps\,[\gamma] = \alpha_r\}|$; [et de $orbtailles$]
14. `fin si`;
15. `sinon`
16. $h := h + 1$;
17. $\Gamma_h := \Sigma_{h-1,e_h}$;
18. `fin si`;
19. `fin si`;
19a. `fin si`;
20. `sinon`
21. $h := h - 1$;
22. `fin si`;
23. `fin tant que`;
24. `retourner` $(\mathbf{U}, \mathbf{T}), X$;
Fin.

2.10.4.3 Problèmes de type classe

Dans cette section nous allons décrire brièvement comment nous vérifions les conditions de la proposition 2.6.3, p. 77, dans l'algorithme ***RepClasse***.

Au lieu de chercher $g \in Sym(\Omega)_{\mathcal{P}}$ tel que $g = \min(KgK)$, nous chercherons g tel que

$g = \min(L_{\mathsf{G}} g L_{\mathsf{D}})$; nous modifions donc l'algorithme de façon adéquate.

Au début (ligne 0i) nous changeons la base de L_{G} pour obtenir un ensemble fort de générateurs de L_{G} associé à la base $B = (\alpha_1, \ldots, \alpha_s)$ correspondant à la $\mathbf{R}$-base $\mathbf{B}$ (cf. section 1.7.6 pour les références sur la procédure de changement de la base). Nous pouvons le faire, car la proposition 2.4.13, p. 71, indique que $(\alpha_1, \ldots, \alpha_s)$ est bien une base de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$. De même, nous changeons la base de L_{D} (ligne 0ii) pour obtenir un SGS de L_{D} associé à une base commençant par $(\alpha_1, \ldots, \alpha_s)$.

Ensuite (lignes 0iii et 0iv) au moyen de la procédure décrite dans la section 1.7.5 nous calculons les orbites de base $\Delta^{(h)}$ de L_{G} pour tout h, $1 \leq h \leq s$, et les orbites $O^{(h)}$ sous l'action du stabilisateur $(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{h-1}}$ pour tout h, $1 \leq h \leq s$. $\Delta^{(h)}$ va servir à l'initialisation de Y (ligne 1c) et *orbtailles* (ligne 1d), qui sont initialisés pour correspondre à L_{G} et permettront de tester les conditions 3 et 1, respectivement, de la proposition 2.6.3 (lignes 9c et 8, respectivement, de l'algorithme 2.10.9 ***RepClasse***). La vérification de la condition 2 de la proposition 2.6.3 (ligne 9b de l'algorithme) requiert avoir l'orbite de β_r sous l'action de $(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{r-1}}$, c'est pourquoi nous calculons les $O^{(h)}$, $1 \leq h \leq s$. Nous réinitialisons *orbreps* chaque fois que r change (ligne 9aii). Au début r est initialisé à $s+1$ (ligne 1a), or pour cette valeur nous n'aurons jamais à tester la condition 2 de la proposition 2.6.3 (car nous ne la testons que si $r = h$, mais h ne dépasse pas s), donc nous pouvons omettre la ligne 1b avec l'initialisation de *orbreps*.

Nous supprimons les lignes 13a–13d, car nous arrêtons l'exécution de l'algorithme dès que nous trouvons un élément quelconque de $Sym(\Omega)_{\mathcal{P}}$ (ligne 13).

Voici la version détaillée de l'algorithme ***RepClasse***.

Algorithme 2.10.9 *RepClasse*$(\mathcal{P}, \mathbf{R}, L_{\mathsf{G}}, L_{\mathsf{D}})$

Entrée : $\mathcal{P}$ une propriété sur un ensemble fini Ω (rappelons que $\mathcal{P}$ contient
  au moins une propriété $\mathcal{E}_G$, cf. remarque 2.2.3),
 $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
 $L_{\mathsf{G}}, L_{\mathsf{D}}$ des sous-groupes de $Sym(\Omega)_{\mathcal{P}_{\mathsf{G}}}$ et $Sym(\Omega)_{\mathcal{P}_{\mathsf{D}}}$, respectivement,
  donnés par des bases A_{G} et A_{D} et
  des ensembles forts de générateurs X_{G} et X_{D}, respectivement.

Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$, toutes les quantités associées
  (cf. remarque 2.4.11) incluses,
 g tel que $g \in Sym(\Omega)_{\mathcal{P}}$ si $Sym(\Omega)_{\mathcal{P}} \neq \emptyset$, $g = \emptyset$ sinon.

Début

0. $(\mathbf{U}, \mathbf{T}) :=$ *Base*$(\mathbf{R})$;
0i. Changement de la base de L_{G} : trouver un SGS de L_{G}
  associé à la base $B = (\alpha_1, \ldots, \alpha_s)$.
0ii. Changement de la base de L_{D} : trouver un SGS de L_{D}
  associé à une base commençant par $(\alpha_1, \ldots, \alpha_s)$.
0iii. Calculer les orbites de base $\Delta^{(h)}$ de L_{G} pour tout h, $1 \leq h \leq s$ (cf. section 1.7.5).
0iv. Calculer les orbites $O^{(h)}$ pour le stabilisateur $(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{h-1}}$ pour tout h, $1 \leq h \leq s$.
1. $g := \emptyset$;
1a. $r := s + 1$;

1c. $Y[i,h] := \mathit{vrai}$ pour tous les i et h, $1 \le i < h \le s$, tels que $\alpha_h \in \Delta^{(i)}$, et faux pour les autres i et h, $1 \le i < h \le s$;

1d. $\mathit{orbtailles}\,[h] := \left|\Delta^{(h)}\right|$ pour tout h, $1 \le h \le s$;

1'. $\mathbf{\Sigma}_0 := U_{0,\mathsf{D}}((\Omega))$;

2. si $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_0,(\Omega)) \neq \mathbf{a}_0$ ou ($\mathit{fix}\,(\mathbf{\Sigma}_0) \notin \mathit{fix}\,(\mathbf{\Pi}_0)^G$ pour un groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) alors

3.   retourner $(\mathbf{U},\mathbf{T}), g$;

4. fin si;

5. $h := 1$;

6. $\Gamma_h := \Sigma_{h-1,e_h}$;

7. tant que $h > 0$ faire

8.   si $|\Gamma_h| \ge \mathit{orbtailles}\,[h]$ alors

9. $\beta_h := \min(\Gamma_h)$; $\Gamma_h := \Gamma_h \setminus \{\beta_h\}$;

9a.   si $h < r$ et $\alpha_h \neq \beta_h$ alors

9ai. $r := h$;

9aii.   pour tout $\gamma \in \Omega$, $\mathit{orbreps}\,[\gamma] := \min(O)$, où $O \in O^{(r)}$ tel que $\gamma \in O$; [ajuster *orbreps* pour qu'il corresponde à $(L_{\mathsf{D}})_{\alpha_1,\ldots,\alpha_{r-1}}$]

9aiii.   fin si;

9b.   si ($r \neq h$ ou $\mathit{orbreps}\,[\beta_h] = \beta_h$) et

9c.   (pour tout i, $r \le i \le h-1$, tel que $Y[i,h] = \mathit{vrai}$, on a $\beta_i \prec \beta_h$) alors

9'. $\mathbf{\Sigma}_h := U_{h,\mathsf{D}}(\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1}))$;

10.   si $\mathit{TaillesdeDivisions}_{\mathsf{D}}(U_h,\mathcal{F}_{e_h,\beta_h}(\mathbf{\Sigma}_{h-1})) = \mathbf{a}_h$ et ($\mathit{fix}\,(\mathbf{\Sigma}_h) \in \mathit{fix}\,(\mathbf{\Pi}_h)^G$ pour tout groupe G tel que $\mathcal{P}$ contient $\mathcal{E}_G$) alors

11.   si $h = s$ alors

11'. $g :=$ l'unique solution de l'équation $\mathbf{\Pi}_s{}^g = \mathbf{\Sigma}_s$;

12.   si $\mathcal{P}(g)$ alors

13.   retourner $(\mathbf{U},\mathbf{T}), g$;

14.   fin si;

15.   sinon

16. $h := h + 1$;

17. $\Gamma_h := \Sigma_{h-1,e_h}$;

18.   fin si;

19.   fin si;

19a.   fin si;

20.   sinon

21. $h := h - 1$;

22.   fin si;

23. fin tant que;

24. retourner $(\mathbf{U},\mathbf{T}), g$;

Fin.

2.10.5 Calcul d'une R-base

En pratique les algorithmes 2.9.1 ***Base*** et 2.9.2 ***Raffiner*** calculent les quantités de la notation 2.10.4, p. 91. Nous donnons ici les versions longues de ces algorithmes incluant ces calculs.

L'algorithme ***Raffiner*** (cf. algorithme 2.10.10), au lieu de retourner seulement U_h et $\mathbf{\Pi}_h$, retourne les quantités de la notation 2.10.4, notamment,

- les $\mathcal{P}$-raffinements élémentaires $E^{(h)}_{l_h}, E^{(h)}_{l_h-1}, \ldots, E^{(h)}_1$ composant U_h (cf. notation 2.10.4(i)),
- les partitions intermédiaires $\mathbf{\Phi}^{(h)}_i = E^{(h)}_{i,\mathsf{G}} E^{(h)}_{i-1,\mathsf{G}} \cdots E^{(h)}_{1,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$, $0 \leq i \leq l_h$ (ou $\mathbf{\Phi}^{(0)}_i = E^{(0)}_{i,\mathsf{G}} E^{(0)}_{i-1,\mathsf{G}} \cdots E^{(0)}_{1,\mathsf{G}}((\Omega))$, $0 \leq i \leq l_0$, si $h = 0$; cf. notation 2.10.4(ii)) et
- les vecteurs $\mathbf{a}_h = \mathit{TaillesdeDivisions}_{\mathsf{G}}(U_h, \mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$ (ou $\mathbf{a}_0 = \mathit{TaillesdeDivisions}_{\mathsf{G}}(U_0, (\Omega))$ si $h = 0$, cf. notation 2.10.4(iii)).

Nous changeons donc cet algorithme comme suit : dans la ligne 2 nous initialisons à 1 le compteur i, nous appliquons la composante gauche d'un $\mathcal{P}$-raffinement élémentaire E à la partition courante $\mathbf{\Phi}_{i-1}$ (ligne 5), et si nous arrivons à la raffiner (ligne 6), nous mettons E (ligne 7) et la nouvelle partition $\mathbf{\Phi}'$ (ligne 8) en mémoire, nous calculons a_i — la dernière composante du vecteur $\mathit{TaillesdeDivisions}_{\mathsf{G}}(E_i E_{i-1} \cdots E_1, \mathbf{\Pi})$ (ligne 8'), et nous augmentons i (ligne 8"). Dans la ligne 10' nous initialisons l le nombre de $\mathcal{P}$-raffinements élémentaires trouvés.

Algorithme 2.10.10 ***Raffiner***$(\mathbf{R}, \mathbf{\Pi})$
Entrée : $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
où $\mathcal{P}$ est une propriété sur un ensemble fini Ω,
$\mathbf{\Pi}$ une partition de Ω.
Sortie : $(E_l, E_{l-1}, \ldots, E_1)$ un $\mathcal{P}$-raffinement (cf. définition 2.4.7, p. 69),
$(\mathbf{\Phi}_0, \ldots, \mathbf{\Phi}_l)$ une suite de partitions de Ω telle que
$\mathbf{\Phi}_0 = \mathbf{\Pi}$ et $\mathbf{\Phi}_i = E_{i,\mathsf{G}}(\mathbf{\Phi}_{i-1})$ pour tout i, $1 \leq i \leq l$,
$\mathbf{a} \stackrel{\text{déf}}{=} (a_1, \ldots, a_l)$ une suite d'entiers telle que
$\mathbf{a} = \mathit{TaillesdeDivisions}_{\mathsf{G}}(E_l E_{l-1} \cdots E_1, \mathbf{\Pi})$.

Début

1. $\mathbf{\Phi}_0 := \mathbf{\Pi}$;
2. $i := 1$;
3. `tant que` la condition d'arrêt n'est pas satisfaite `faire`
4.   choisir $E \in \mathbf{R}$;
5. $\mathbf{\Phi}' := E_{\mathsf{G}}(\mathbf{\Phi}_{i-1})$;
6.   `si` $\mathbf{\Phi}' < \mathbf{\Phi}_{i-1}$ `alors`
7. $E_i := E$;
8. $\mathbf{\Phi}_i := \mathbf{\Phi}'$;

8'. $a_i := \left| \Phi'_{|\mathbf{\Phi}_{i-1}|+1} \right|$;

8”. $i := i + 1$;
9. `fin si`;
10. `fin tant que`;
10’. $l := i - 1$;
11. `retourner` $(E_l, E_{l-1}, \ldots, E_1), (\mathbf{\Phi}_0, \ldots, \mathbf{\Phi}_l), (a_1, \ldots, a_l)$;
Fin.

Alors l’algorithme 2.9.1 ***Base*** se change de la façon suivante. Nous ajoutons les lignes 3’ et 8’ où nous initialisons la partition $\mathbf{\Pi}_h$, $0 \leq h \leq s$, après l’appel de l’algorithme 2.10.10 ***Raffiner***. Notez que pour tout h, $0 \leq h \leq s$, les $E_i^{(h)}$, $1 \leq i \leq l_h$, les $\mathbf{\Phi}_i^{(h)}$, $0 \leq i \leq l_h$, et les $\mathbf{a}_h$ sont présents implicitement dans la $\mathbf{R}$-base sortie par algorithme (ligne 11).

Algorithme 2.10.11 ***Base***$(\mathbf{R})$
Entrée : $\mathbf{R}$ un ensemble de $\mathcal{P}$-raffinements élémentaires,
où $\mathcal{P}$ est une propriété sur un ensemble fini Ω.
Sortie : $\mathbf{B}$ une $\mathbf{R}$-base pour $\mathcal{P}$ avec toutes les quantités associées
(cf. remarque 2.4.11, p. 71, et notation 2.10.4, p. 91).
Début
1. $h := 0$;
2. $\mathbf{\Pi}_{h-1} := (\Omega)$;
3. $U_h, (\mathbf{\Phi}_0^{(h)}, \ldots, \mathbf{\Phi}_{l_h}^{(h)}), \mathbf{a}_h :=$ ***Raffiner***$(\mathbf{R}, \mathbf{\Pi}_{h-1})$;
3’. $\mathbf{\Pi}_h := \mathbf{\Phi}_{l_h}^{(h)}$; $[\mathbf{\Pi}_0 = U_{0,\mathsf{G}}((\Omega))$, cf. propriété 2.10.5(ii), p. 92]
4. `tant que` $\mathbf{\Pi}_h$ n’est pas discrète `faire`
5. $h := h + 1$;
6. choisir $e_h \in I_n$ tel que $|\Pi_{h-1,e_h}| > 1$;
7. choisir $\alpha_h \in \Pi_{h-1,e_h}$;
8. $U_h, (\mathbf{\Phi}_0^{(h)}, \ldots, \mathbf{\Phi}_{l_h}^{(h)}), \mathbf{a}_h :=$ ***Raffiner***$(\mathbf{R}, \mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$;
8’. $\mathbf{\Pi}_h := \mathbf{\Phi}_{l_h}^{(h)}$; $[\mathbf{\Pi}_h = U_{h,\mathsf{G}}(\mathcal{F}_{e_h,\alpha_h}(\mathbf{\Pi}_{h-1}))$, cf. propriété 2.10.5(ii)]
9. `fin tant que`;
10. $s := h$;
11. `retourner` $((U_0, \ldots, U_s), (\mathcal{F}_{e_1,\alpha_1}, \ldots, \mathcal{F}_{e_s,\alpha_s}))$;
Fin.

2.11 Remarques sur la complexité

Nous n’étudions pas la complexité de la méthode de recherche arrière et de la méthode des partitions. Elle est très difficile à évaluer et dépend du problème à résoudre. Par exemple, pour un code donné C, l’arbre $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ pour le problème du groupe de permutations de C peut être “optimal” (i.e. ne fournissant que les permutations de $Sym(\Omega)_{\mathcal{P}_C}$) ou très “mauvais” (i.e. les permutations de $Sym(\Omega)_{\mathcal{P}_C}$ ne constituent qu’un petit pourcentage de toutes les permutations fournies par $\mathbb{T}_{\mathbf{R},\mathbf{B}}$), cela dépend de C, et nous ne le savons pas d'avance. Voici comment le caractère instable de la méthode de recherche arrière est décrit par Donald E. Knuth [Knu75, p.121] :

> Sometimes a backtrack program will run to completion in less than a second, while other applications seem to go on forever. The author once waited all night for the output from such a program, only to discover that the answers would not be forthcoming for about 10 centuries. A "slight increase" in one of the parameters of a backtrack routine might slow down the total running time by a factor of a thousand ; conversely, a "minor improvement" to the algorithm might cause a hundredfold improvement in speed ; and a sophisticated "major improvement" might actually make the program ten times slower. These great discrepancies in execution time are characteristic of backtrack programs, yet it is usually not obvious what will happen until the algorithm has been coded and run on a machine.

Chapitre 3

Nos raffinements pour les problèmes liés aux codes

3.1 Introduction

Dans le chapitre précédent nous avons présenté la méthode des partitions introduite par Jeffrey S. Leon [Leo91, Leo97]. On peut l'appliquer aux quatre problèmes liés aux codes, notamment pour les problèmes désignés par $\mathcal{P}_C$, $\mathcal{P}_{C,C'}$, $\mathcal{A}_C$ et $\mathcal{A}_{C,C'}$ (cf. notation 2.2.4, p. 56). Mais pour que la méthode des partitions fonctionne, il faut construire d'abord des ensembles de $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires (nous avons vu dans la section 1.6.3 que résoudre $\mathcal{A}_C$ et $\mathcal{A}_{C,C'}$ revient à résoudre $\mathcal{P}_C$ et $\mathcal{P}_{C,C'}$, respectivement, nous pouvons donc utiliser les mêmes ensembles de $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires pour les résoudre). C'est ce que nous ferons dans ce chapitre. Nous l'avons fait en collaboration avec Nicolas Sendrier (cf. [SeS99]). Nous reprenons la notion de signature, introduite par N. Sendrier pour son algorithme de séparation du support qui calcule une permutation entre deux codes équivalents par permutation dont les groupes de permutations sont triviaux. Nous ne décrivons pas cet algorithme ici, cf. [Sen96, Sen97b, Sen99a, Sen99b], mais nous introduisons les signatures (section 3.2), les généralisons (section 3.3), et montrons comment elles permettent de construire des $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires (section 3.4). Dans la section 3.5 nous introduirons une signature particulière, basée sur le hull. La section 3.6 décrit comment vérifier que $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^{Sym_q(\Omega)}$ pour $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ des partitions ordonnées de Ω_q. La section 3.7 récapitule les résultats et pour chaque problème $\mathcal{P}_C$, $\mathcal{P}_{C,C'}$, $\mathcal{A}_C$ ou $\mathcal{A}_{C,C'}$ explique comment le résoudre. L'implantation est discutée dans la section 3.8. La section 3.9 compare brièvement nos raffinements avec ceux de Leon. Et enfin la section 3.10 donne quelques exemples du temps d'exécution de notre implantation.

Dans tout ce chapitre n sera un entier positif non nul, q une puissance d'un nombre premier et Ω un ensemble fini de cardinal n utilisé pour indexer les coordonnées des vecteurs de $\mathbf{F}_q^n$.

3.2 Invariants et signatures

Cette section est basée sur les articles de Sendrier [Sen96, Sen97b, Sen99a, Sen99b].

Soit $\mathcal{L}_n$ l'ensemble des codes de longueur n sur $\mathbf{F}_q$, soit $\mathcal{L} = \bigcup_{n>0} \mathcal{L}_n$ l'ensemble des codes sur $\mathbf{F}_q$.

Définition 3.2.1 *Un* invariant *sur un ensemble E est une application $\mathcal{L} \to E$ telle que deux codes équivalents par permutation prennent la même valeur.*

Par exemple la longueur, la dimension, la distance minimale sont des invariants sur les entiers. L'énumérateur des poids est un invariant sur les polynômes à coefficients entiers (cf. propriété 1.6.5, p. 31).

Définition 3.2.2 *Une* signature *S sur un ensemble E est une application qui à tout code C de longueur n et à tout élément γ de Ω associe un élément $S(C, \gamma)$ de E et telle que pour toute permutation g de Ω et pour tout γ dans Ω, $S(C^g, \gamma^g) = S(C, \gamma)$.*

Les signatures existent. Par exemple, à chaque invariant $\mathcal{V}$ nous pouvons associer une signature $S : (C, \gamma) \mapsto \mathcal{V}(C_\gamma)$, où C_γ est le code C poinçonné en γ (cf. définition 1.3.11, p. 20). En effet :

Proposition 3.2.3 *Si $\mathcal{V}$ est un invariant, alors $S : (C, \gamma) \mapsto \mathcal{V}(C_\gamma)$ est une signature.*

Preuve. Nous avons

$$S(C^g, \gamma^g) = \mathcal{V}((C^g)_{\gamma^g}) \overset{Propr. 1.6.6}{=} \mathcal{V}((C_\gamma)^g) \overset{Déf. 3.2.1}{=} \mathcal{V}(C_\gamma) = S(C, \gamma)$$

pour tout $g \in Sym(\Omega)$. □

Proposition 3.2.4 *Si S et T sont des signatures, alors les applications $S^\perp$, définie par $S^\perp : (C, \gamma) \mapsto S(C^\perp, \gamma)$, et $S \times T$, définie par $S \times T : (C, \gamma) \mapsto (S(C, \gamma), T(C, \gamma))$, sont des signatures.*

Corollaire 3.2.5 *Si S est une signature, alors $S \times S^\perp$ est une signature.*

3.3 Signatures généralisées

Soient C un code de longueur n sur $\mathbf{F}_q$, S une signature sur un ensemble E, $\mathbf{\Pi}$ une partition ordonnée de Ω, L un sous-ensemble de I_n et $\gamma \in \Omega$.

Nous définissons l'application $\mathsf{T}_{S,L}$ comme suit :

$$\mathsf{T}_{S,L}(\mathbf{\Pi}, C, \gamma) = S(C_{\bigcup_{j \in L} \Pi_j}, \gamma). \tag{3.1}$$

Remarquons une propriété de $\mathsf{T}_{S,L}$:

Propriété 3.3.1 $\mathsf{T}_{S,L}(\mathbf{\Pi}^g, C^g, \gamma^g) = \mathsf{T}_{S,L}(\mathbf{\Pi}, C, \gamma)$ *pour tout $g \in Sym(\Omega)$.*

Preuve. Nous avons

$$\begin{aligned}
\mathsf{T}_{S,L}\,(\mathbf{\Pi}^g, C^g, \gamma^g) &\overset{(3.1)}{=} S((C^g)_{\bigcup_{j\in L}(\mathbf{\Pi}^g)_j}, \gamma^g) \\
&\overset{Propr.\,1.5.4}{=} S((C^g)_{\bigcup_{j\in L}(\Pi_j)^g}, \gamma^g) \\
&\overset{Propr.\,1.4.9}{=} S((C^g)_{\left(\bigcup_{j\in L}\Pi_j\right)^g}, \gamma^g) \\
&\overset{Propr.\,1.6.6}{=} S\left(\left(C_{\bigcup_{j\in L}\Pi_j}\right)^g, \gamma^g\right) \\
&\overset{Déf.\,3.2.2}{=} S(C_{\bigcup_{j\in L}\Pi_j}, \gamma) \\
&\overset{(3.1)}{=} \mathsf{T}_{S,L}\,(\mathbf{\Pi}, C, \gamma)
\end{aligned}$$

pour tout $g \in Sym(\Omega)$. □

Nous appelons *signatures généralisées* les applications satisfaisant cette propriété :

Définition 3.3.2 *Soient C un code de longueur n sur $\mathbf{F}_q$, $\mathbf{\Pi}$ une partition ordonnée de Ω, E un ensemble et $\gamma \in \Omega$. Une application R qui associe $\mathbf{\Pi}$, C et γ à un élément $\mathsf{R}\,(\mathbf{\Pi}, C, \gamma)$ de E est appelée* signature généralisée *sur E si*

$$\mathsf{R}\,(\mathbf{\Pi}^g, C^g, \gamma^g) = \mathsf{R}\,(\mathbf{\Pi}, C, \gamma) \text{ pour tout } g \in Sym(\Omega).$$

Nous pouvons construire des nouvelles signatures généralisées à partir des signatures généralisées données comme suit :

Proposition 3.3.3 *Soit P et Q des signatures généralisées. Alors $\mathsf{P}^\perp$, définie par*

$$\mathsf{P}^\perp(\mathbf{\Pi}, C, \gamma) = \mathsf{P}(\mathbf{\Pi}, C^\perp, \gamma),$$

et $\mathsf{P} \times \mathsf{Q}$, définie par

$$\mathsf{P} \times \mathsf{Q}(\mathbf{\Pi}, C, \gamma) = (\mathsf{P}(\mathbf{\Pi}, C, \gamma), \mathsf{Q}(\mathbf{\Pi}, C, \gamma)),$$

sont des signatures généralisées.

Preuve. Évidente. □

Corollaire 3.3.4 *Si R est une signature généralisée, alors $\overline{\mathsf{R}} \overset{\text{déf}}{=} \mathsf{R} \times \mathsf{R}^\perp$ est aussi une signature généralisée.*

3.4 Raffinements basés sur des signatures généralisées

Nous définissons nos $\mathcal{P}_C$-raffinements élémentaires au moyen des signatures généralisées :

Proposition 3.4.1 *Soient C un code de longueur n sur $\mathbf{F}_q$, $\mathbf{\Pi} \in PartOrd(\Omega)$, $i \in I_n$, R une signature généralisée sur un ensemble E et $e \in E$. L'application $Q_{C,\mathsf{R},i,e}$ de $PartOrd(\Omega)$ dans $PartOrd(\Omega)$, définie par*

$$Q_{C,\mathsf{R},i,e}(\mathbf{\Pi}) = \mathcal{F}_{i,\mathbf{\Phi}}(\mathbf{\Pi}), \text{ où } \mathbf{\Phi} = \{\gamma \in \Omega \mid \mathsf{R}(\mathbf{\Pi}, C, \gamma) = e\}, \tag{3.2}$$

est un $\mathcal{P}_C$-raffinement élémentaire symétrique.

Preuve. Il faut montrer que $Q_{C,\mathsf{R},i,e}$ satisfait les quatre conditions de la définition 2.4.1, p. 66.

Nous voyons facilement que la propriété 1.5.9(a), p. 29, implique la condition 1 de la définition 2.4.1, que la propriété 1.5.9(b) implique la condition 2, et que la propriété 1.5.9(c) implique la condition 4.

Soient $g \in Perm(C)$, $\mathbf{\Pi}, \mathbf{\Sigma} \in PartOrd(\Omega)$ tels que $\mathbf{\Pi}^g = \mathbf{\Sigma}$. Pour montrer la condition 3, il faut montrer que $Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})^g = Q_{C,\mathsf{R},i,e}(\mathbf{\Sigma})$. Rappelons que $Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})$ est défini dans (3.2), et que

$$Q_{C,\mathsf{R},i,e}(\mathbf{\Sigma}) = \mathcal{F}_{i,\mathbf{\Phi}'}(\mathbf{\Sigma}), \text{ où } \mathbf{\Phi}' = \{\gamma \in \Omega \mid \mathsf{R}(\mathbf{\Sigma}, C, \gamma) = e\}. \tag{3.3}$$

Mais g appartient à $Perm(C)$, donc, compte tenu de la définition 3.3.2,

$$\mathsf{R}(\mathbf{\Sigma}, C, \gamma) = \mathsf{R}(\mathbf{\Pi}^g, C, \gamma) = \mathsf{R}\left(\mathbf{\Pi}, C^{g^{-1}}, \gamma^{g^{-1}}\right) = \mathsf{R}\left(\mathbf{\Pi}, C, \gamma^{g^{-1}}\right).$$

Donc

$$\begin{aligned} \mathbf{\Phi}' &= \{\gamma \in \Omega \mid \mathsf{R}(\mathbf{\Sigma}, C, \gamma) = e\} \\ &= \{\gamma \in \Omega \mid \mathsf{R}\left(\mathbf{\Pi}, C, \gamma^{g^{-1}}\right) = e\} \\ &= \{\eta^g \mid \eta \in \Omega, \mathsf{R}(\mathbf{\Pi}, C, \eta) = e\} \\ &= \mathbf{\Phi}^g, \end{aligned} \tag{3.4}$$

car si γ parcourt Ω, alors $\eta \overset{\text{déf}}{=} \gamma^{g^{-1}}$ parcourt Ω aussi. Nous avons donc

$$Q_{C,\mathsf{R},i,e}(\mathbf{\Sigma}) \overset{(3.3)}{=} \mathcal{F}_{i,\mathbf{\Phi}'}(\mathbf{\Sigma}) \overset{(3.4)}{=} \mathcal{F}_{i,\mathbf{\Phi}^g}(\mathbf{\Pi}^g) \overset{Propr. 1.5.10}{=} \mathcal{F}_{i,\mathbf{\Phi}}(\mathbf{\Pi})^g \overset{(3.2)}{=} Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})^g,$$

ce qui complète la démonstration. □

Proposition 3.4.2 *Soient $C, \mathbf{\Pi}, i, \mathsf{R}, e$ définis comme dans la proposition 3.4.1. Soit C' un code de longueur n sur $\mathbf{F}_q$. Si $Q_{C,\mathsf{R},i,e}$ est défini par la formule (3.2), alors*

$$Q_{C,C',\mathsf{R},i,e} \overset{\text{déf}}{=} (Q_{C,\mathsf{R},i,e}, Q_{C',\mathsf{R},i,e})$$

est un $\mathcal{P}_{C,C'}$-raffinement élémentaire.

Preuve. Tout à fait comme dans la démonstration précédente nous montrons que les conditions 1, 2 et 4 de la définition 2.4.1 sont satisfaites. Pour montrer que la condition 3 est satisfaite, nous devons montrer que si $g \in Sym(\Omega)$, $\mathbf{\Pi}, \mathbf{\Sigma} \in PartOrd(\Omega)$ sont tels que $C^g = C'$ et $\mathbf{\Pi}^g = \mathbf{\Sigma}$, alors $Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})^g = Q_{C',\mathsf{R},i,e}(\mathbf{\Sigma})$. Rappelons que $Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})$ est défini par (3.2), et que

$$Q_{C',\mathsf{R},i,e}(\mathbf{\Sigma}) = \mathcal{F}_{i,\mathbf{\Phi}'}(\mathbf{\Sigma}), \text{ où } \mathbf{\Phi}' = \{\gamma \in \Omega \mid \mathsf{R}(\mathbf{\Sigma}, C', \gamma) = e\}.$$

Nous avons que

$$\mathsf{R}(\mathbf{\Sigma}, C', \gamma) = \mathsf{R}(\mathbf{\Pi}^g, C^g, \gamma) = \mathsf{R}\left(\mathbf{\Pi}, C, \gamma^{g^{-1}}\right)$$

et donc $\mathbf{\Phi}' = \mathbf{\Phi}^g$ comme dans la formule (3.4). D'où

$$Q_{C',\mathsf{R},i,e}(\mathbf{\Sigma}) = \mathcal{F}_{i,\mathbf{\Phi}'}(\mathbf{\Sigma}) = \mathcal{F}_{i,\mathbf{\Phi}^g}(\mathbf{\Pi}^g) = \mathcal{F}_{i,\mathbf{\Phi}}(\mathbf{\Pi})^g = Q_{C,\mathsf{R},i,e}(\mathbf{\Pi})^g.$$ □

3.5 Hull et signature associée

Comme nous l'avons vu, il est possible de construire des $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires au moyen d'une signature généralisée (voir section 3.4), une signature généralisée à l'aide d'une signature (voir la formule (3.1)), et une signature à partir d'un invariant (voir proposition 3.2.3). Nous présentons dans cette section un invariant et une signature à partir desquels nous construisons nos ensembles de $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires.

Rappelons que le hull $\mathcal{H}(C)$ d'un code C est défini comme son intersection avec son dual, c'est-à-dire $\mathcal{H}(C) = C \cap C^{\perp}$.

Proposition 3.5.1 *Soit C un code de longueur n sur $\mathbf{F}_q$, soit g une permutation de Ω. Alors $\mathcal{H}(C^g) = \mathcal{H}(C)^g$.*

Preuve. Il suffit de remarquer que $\left(C^{\perp}\right)^g = (C^g)^{\perp}$ et $(A \cap B)^g = A^g \cap B^g$. □

Corollaire 3.5.2 *Soit $\mathcal{V}$ un invariant et C un code. L'application $C \mapsto \mathcal{V}(\mathcal{H}(C))$ est un invariant.*

Preuve. C'est un corollaire direct de la proposition 3.5.1. □

Remarque 3.5.3 Cela n'est plus valable pour les codes équivalents (isomorphes), sauf dans les cas $q = 2$ et $q = 3$. En effet, en général, si deux codes sont isomorphes, leurs hulls ne le sont pas. Les cas $q = 2$ et $q = 3$ sont des exceptions. Dans le cas binaire l'équivalence et l'équivalence par permutation coïncide, et dans le cas ternaire tous les éléments non zéros du corps $\mathbf{F}_3$ sont leurs propres inverses.

Cela montre que, en général, si on a une propriété qui prend la même valeur pour les codes isomorphes (la notion qui est équivalente à celle d'un invariant pour les codes équivalents par permutation), elle peut prendre des valeurs différentes pour les hulls des codes isomorphes. On ne peut donc pas utiliser directement le hull pour tester l'isomorphisme ou calculer le groupe d'automorphismes. Nous contournons cette restriction à l'aide des propositions 1.6.21, p. 36, et 1.6.22, p. 38. □

Remarque 3.5.4 Cet invariant est valable pour les codes linéaires et non linéaires, mais il n'est pratique que pour les codes linéaires, parce que, premièrement, il existe des techniques pour accélérer le calcul du hull d'un code poinçonné si ce code est linéaire (cf. [Sen99a, Sen99b]), et, deuxièmement, pour des codes C et C' il est difficile d'évaluer les propriétés $\mathcal{P}_C(g)$ et $\mathcal{P}_{C,C'}(g)$ si C et C' ne sont pas linéaires, le temps d'évaluation n'est plus forcément polynomial en longueur de ces codes. Nous ne considérerons donc que des codes linéaires.

Nous avons donc à notre disposition tous les invariants appliqués au hull, par exemple, l'énumérateur des poids. L'intérêt pratique de ces invariants est dû au fait que la dimension moyenne du hull des codes linéaires est une petite constante strictement positive, par exemple, pour les codes linéaires binaires cette constante est égale à 0.7645..., pour les codes ternaires 0.4041..., pour les codes quaternaires 0.2794..., etc... (cf. [Sen97a, Sen99a, Sen99b]).

L'énumérateur des poids $C \mapsto \mathcal{W}(C)$ est un invariant, donc $C \mapsto \mathcal{W}(\mathcal{H}(C))$ est aussi un invariant (cf. corollaire 3.5.2). Nous construisons la signature S comme dans la proposition 3.2.3 : $S : (C, \gamma) \mapsto \mathcal{W}(\mathcal{H}(C_\gamma))$. Pour obtenir une signature "plus puissante", nous pouvons faire comme dans le corollaire 3.2.5, nous définissons donc la signature $\mathbf{S}$ comme suit : $\mathbf{S} = S \times S^\perp$. Plus précisément,

$$\mathbf{S} : (C, \gamma) \mapsto \left(\mathcal{W}(\mathcal{H}(C_\gamma)), \mathcal{W}\left(\mathcal{H}\left((C^\perp)_\gamma\right)\right)\right). \tag{3.5}$$

C'est Nicolas Sendrier qui a proposé cette signature. Il l'utilise dans son algorithme de séparation du support (nous ne le décrivons pas ici, cf. [Sen96, Sen97b, Sen99a, Sen99b]). Nous l'utilisons pour construire la signature généralisée $\overline{\mathsf{T}}_{\mathbf{S},L}$ (cf. formule (3.1) et corollaire 3.3.4), à partir de laquelle nous construisons les $\mathcal{P}_C$- et $\mathcal{P}_{C,C'}$-raffinements élémentaires $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},i,e}$ et $Q_{C,C',\overline{\mathsf{T}}_{\mathbf{S},L},i,e}$, respectivement (cf. propositions 3.4.1 et 3.4.2). Notons

$$\begin{aligned} \mathbf{Q}_C &= \{Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},i,e} \mid L \subset I_n, i \in I_n, e \in \mathbb{Z}[X]^4\} \text{ et} & (3.6)\\ \mathbf{Q}_{C,C'} &= \{Q_{C,C',\overline{\mathsf{T}}_{\mathbf{S},L},i,e} \mid L \subset I_n, i \in I_n, e \in \mathbb{Z}[X]^4\}, & (3.7)\end{aligned}$$

où $\mathbb{Z}[X]$ est l'ensemble des polynômes à coefficients entiers. Pourquoi $\mathbb{Z}[X]^4$? Parce que $\overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Pi}, C, \gamma)$ peut être vu comme une suite de quatre énumérateurs des poids, plus précisément,

$$\begin{aligned} \overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Pi}, C, \gamma) &\overset{Cor. 3.3.4}{=} \left(\mathsf{T}_{\mathbf{S},L}(\mathbf{\Pi}, C, \gamma), \mathsf{T}_{\mathbf{S},L}(\mathbf{\Pi}, C^\perp, \gamma)\right) \\ &\overset{(3.1)}{=} \left(\mathbf{S}\left(C_{\bigcup_{r\in L}\Pi_r}, \gamma\right), \mathbf{S}\left((C^\perp)_{\bigcup_{r\in L}\Pi_r}, \gamma\right)\right) \\ &\overset{(3.5)}{=} \Bigg(\mathcal{W}\left(\mathcal{H}\left(C_{(\bigcup_{r\in L}\Pi_r)\cup\{\gamma\}}\right)\right), \mathcal{W}\left(\mathcal{H}\left(\left(\left(C_{\bigcup_{r\in L}\Pi_r}\right)^\perp\right)_\gamma\right)\right), \\ &\qquad \mathcal{W}\left(\mathcal{H}\left((C^\perp)_{(\bigcup_{r\in L}\Pi_r)\cup\{\gamma\}}\right)\right), \mathcal{W}\left(\mathcal{H}\left(\left(\left((C^\perp)_{\bigcup_{r\in L}\Pi_r}\right)^\perp\right)_\gamma\right)\right)\Bigg). \end{aligned}$$

Donc l'ensemble des valeurs de $\overline{\mathsf{T}}_{\mathbf{S},L}$ est contenu dans $\mathbb{Z}[X]^4$.

Nous entrons les ensembles $\mathbf{Q}_C$ et $\mathbf{Q}_{C,C'}$ dans l'algorithme général de Leon pour les problèmes $\mathcal{P}_C$ et $\mathcal{P}_{C,C'}$, respectivement. Notre expérience avec l'algorithme montre que ces deux ensembles raffinent assez bien les partitions, cf. section 3.10. Bien sûr, cela ne marche que si le hull du code donné est assez petit pour pouvoir calculer son énumérateur des poids ; par exemple, dans le cas binaire la dimension du hull ne doit pas excéder 30.

3.6 Cas du groupe de permutations monomiales

Dans le cas où une propriété $\mathcal{P}$ contient une propriété $\mathcal{E}_G$ pour un groupe de permutations G, les algorithmes de recherche arrière (cf. algorithmes 2.5.1 ***Éléments***, 2.6.6 ***Sous-groupe*** et 2.7.1 ***RepClasse***) doivent vérifier si $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^G$, où $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ sont des partitions ordonnées (cf. lignes 2 et 10 de ces algorithmes). De plus, l'application d'un $\mathcal{E}_G$-raffinement élémentaire de $\mathbf{D}_G$ (cf. notation 2.8.1, p. 84) requiert de trouver une permutation $t \in G$ telle que $\mathit{fix}\,(\mathbf{\Sigma}) = \mathit{fix}\,(\mathbf{\Pi})^t$. Quand nous considérons les problèmes $\mathcal{P}_C$ et $\mathcal{P}_{C,C'}$, le groupe G est égal au groupe symétrique $\mathit{Sym}(\Omega)$, donc la vérification de cette condition et le calcul de t sont triviaux (en fait, nous ne les faisons même pas). Par contre, quand nous considérons les problèmes $\mathcal{A}_C$ et $\mathcal{A}_{C,C'}$, le groupe G devient $\mathit{Sym}_q(\Omega)$ (cf. définition 1.6.9, p. 32). Nous allons décrire dans cette section comment nous le faisons dans ce cas-là.

Il faut résoudre le problème suivant : étant données deux partitions ordonnées $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ de Ω_q, vérifier si $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^{\mathit{Sym}_q(\Omega)}$ et si c'est le cas trouver une permutation $t \in \mathit{Sym}_q(\Omega)$ telle que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$.

Nous vérifions d'abord que $|\mathit{fix}\,(\mathbf{\Pi})| = |\mathit{fix}\,(\mathbf{\Sigma})|$. Évidemment, si cette égalité n'est pas satisfaite, $\mathit{fix}\,(\mathbf{\Sigma}) \notin \mathit{fix}\,(\mathbf{\Pi})^{\mathit{Sym}_q(\Omega)}$.

Soit maintenant $|\mathit{fix}\,(\mathbf{\Pi})| = |\mathit{fix}\,(\mathbf{\Sigma})|$. Nous essayons de construire $t \in \mathit{Sym}_q(\Omega)$ tel que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$. Si nous n'y arrivons pas, cela signifiera que $\mathit{fix}\,(\mathbf{\Sigma}) \notin \mathit{fix}\,(\mathbf{\Pi})^{\mathit{Sym}_q(\Omega)}$, sinon nous aurons montré que $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^{\mathit{Sym}_q(\Omega)}$ et nous aurons trouvé t.

D'après le corollaire 1.6.13, p. 33, une permutation $t \in \mathit{Sym}_q(\Omega)$ est définie complètement par les images de $(1,\gamma)$, $1 \in \mathbf{F}_q^*$, $\gamma \in \Omega$. Nous chercherons la partie de ces images définie par l'équation $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$, et si nous y arrivons, nous fixerons le reste de ces images d'une manière arbitraire.

Supposons que $1 \leq i \leq |\mathit{fix}\,(\mathbf{\Pi})|$ et les couples $(a,\alpha) \in \Omega_q$ et $(b,\beta) \in \Omega_q$ soient les i-ièmes composantes de $\mathit{fix}\,(\mathbf{\Pi})$ et $\mathit{fix}\,(\mathbf{\Sigma})$, respectivement. Nous cherchons une permutation t de $\mathit{Sym}_q(\Omega)$ telle que $(a,\alpha)^t = (b,\beta)$. Comme t est une permutation monomiale,

$$(a,\alpha)^t = a(1,\alpha)^t, \quad \text{d'où} \quad (1,\alpha)^t = a^{-1}(b,\beta) = (a^{-1}b,\beta).$$

Donc si t existe, l'image de $(1,\alpha)$ par t doit être égale à $(a^{-1}b,\beta)$. Ainsi nous trouvons les images pour certains points $(1,\gamma)$, $\gamma \in \Omega$, plus précisément, nous pouvons trouver l'image d'un point $(1,\gamma)$ si et seulement s'il existe $u \in \mathbf{F}_q^*$ tel que (u,γ) se trouve dans $\mathit{fix}\,(\mathbf{\Pi})$.

Supposons que nous ayons exploré déjà $i-1$ composantes de $\mathit{fix}\,(\mathbf{\Pi})$ et $\mathit{fix}\,(\mathbf{\Sigma})$. Nous passons à la i-ième. Comme ci-dessus, nous trouvons l'image (v,β) d'un point $(1,\gamma)$, $\gamma \in \Omega$.

Nous vérifions si une image de ce point a déjà été trouvée.

- Si c'est le cas, cette image doit être la même.
  - Si elle est différente, $t \in Sym_q(\Omega)$ tel que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$ n'existe pas et nous arrêtons le calcul.
  - Sinon, nous continuons.
- Si c'est la première fois que nous trouvons l'image de $(1, \gamma)$, nous la mettons en mémoire et nous vérifions que β est différent de toutes les deuxièmes composantes des images déjà trouvées.
  - Si ce n'est pas le cas, $t \in Sym_q(\Omega)$ tel que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$ n'existe pas en vertu du corollaire 1.6.12, p. 33, nous nous arrêtons.
  - Sinon nous recommençons la procédure décrite avec les composantes $i+1$ de $\mathit{fix}\,(\mathbf{\Pi})$ et $\mathit{fix}\,(\mathbf{\Sigma})$.

En procédant ainsi, soit nous montrons que $t \in Sym_q(\Omega)$ tel que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$ n'existe pas, c'est-à-dire $\mathit{fix}\,(\mathbf{\Sigma}) \notin \mathit{fix}\,(\mathbf{\Pi})^{Sym_q(\Omega)}$, soit nous trouvons les images par t de certains points $(1, \gamma)$, $\gamma \in \Omega$, plus précisément, les images des points $(1, \gamma)$ tels que (d, γ) se trouve dans $\mathit{fix}\,(\mathbf{\Pi})$ pour certain $d \in \mathbf{F}_q^*$.

Supposons que nous trouvions en procédant ainsi les images $(v_1, \beta_1), \ldots, (v_j, \beta_j)$ de j points $(1, \gamma_1), \ldots, (1, \gamma_j)$. Par construction, les deuxièmes composantes de ces images sont toutes différentes, nous avons donc aussi j deuxièmes composantes différentes $\beta_1, \ldots, \beta_j$. Nous faisons correspondre par t les $n-j$ restants points $(1, \gamma)$, $\gamma \in \Omega \setminus \{\gamma_1, \ldots, \gamma_j\}$ aux $n-j$ restants points $(1, \beta)$, $\beta \in \Omega \setminus \{\beta_1, \ldots, \beta_j\}$, d'une manière arbitraire. Il est facile à voir que t définie de cette façon par les images de $(1, \gamma)$, $\gamma \in \Omega$, est une permutation de $Sym_q(\Omega)$.

3.7 Récapitulatif

Nous voulons donc résoudre quatre problèmes : étant donné(s) un (des) code(s) C (et C') de longueur n sur $\mathbf{F}_q$, trouver $Sym(\Omega)_{\mathcal{P}_C}$, $Sym(\Omega)_{\mathcal{P}_{C,C'}}$, $Sym_q(\Omega)_{\mathcal{A}_C}$ et $Sym_q(\Omega)_{\mathcal{A}_{C,C'}}$ (cf. section 2.2), où Ω est un ensemble fini de cardinal n utilisé pour indexer les coordonnées des mots de $\mathbf{F}_q^n$.

3.7.1 Problème du groupe de permutations d'un code

Considérons le problème suivant : étant donné un code linéaire C, trouver $Perm(C)$. Comme $Perm(C) = Sym(\Omega)_{\mathcal{P}_C}$, nous pouvons appliquer l'algorithme général de Leon 2.6.6 ***Sous-groupe***. Nous lui passons la propriété $\mathcal{P}_C$ et l'ensemble de $\mathcal{P}_C$-raffinements élémentaires symétriques $\mathbf{Q}_C$ de la formule (3.6). L'algorithme retourne une $\mathbf{Q}_C$-base $\mathbf{B}$ pour $\mathcal{P}_C$ (une base $B = (\alpha_1, \ldots, \alpha_s)$ de $Perm(C)$ incluse) et un ensemble fort de générateurs

X de $Perm(C)$ associé à la base B. Notre but est atteint. De plus, nous pouvons manipuler de $Perm(C)$ en utilisant les techniques décrites dans la section 1.7.

3.7.2 Problème de l'équivalence par permutation des codes

Considérons le problème suivant : étant donnés deux codes linéaires C et C' de la même longueur n sur le même corps fini $\mathbf{F}_q$, vérifier s'ils sont équivalents par permutation et si c'est le cas trouver une permutation $g \in Sym(\Omega)$ telle que $C^g = C'$. Ce problème est désigné par la propriété $\mathcal{P}_{C,C'}$. Pour le résoudre, nous appliquons l'algorithme général de Leon 2.7.1 ***RepClasse***, en lui passant la propriété $\mathcal{P}_{C,C'}$, l'ensemble de $\mathcal{P}_{C,C'}$-raffinements élémentaires $\mathbf{Q}_{C,C'}$ de la formule (3.7) et deux groupes triviaux $\{id\}$. L'algorithme retourne une $\mathbf{Q}_{C,C'}$-base $\mathbf{B}$ pour $\mathcal{P}_{C,C'}$ et un g tel que $g = \emptyset$ si C et C' ne sont pas équivalents par permutation, et $g \in Sym(\Omega)$ tel que $C^g = C'$ sinon. Notre problème est résolu.

3.7.3 Problème du groupe d'automorphismes d'un code

Considérons le problème suivant : étant donné un code linéaire C de longueur n sur $\mathbf{F}_q$, trouver $Aut(C)$. Nous savons que

$$\begin{aligned} Aut(C) &= \{g \in Sym(\Omega_q) \mid g \in Sym_q(\Omega), g \in Perm(C^*)\} \\ &= \{g \in Sym(\Omega_q) \mid \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}(g)\} \\ &= Sym(\Omega_q)_{\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}}, \end{aligned}$$

où C^* est le code étalé de C (cf. définition 1.6.20 et proposition 1.6.21, p. 36 ; notez que C^* est un code linéaire). Nous pouvons donc appliquer l'algorithme général de Leon 2.6.6 ***Sous-groupe*** avec la propriété $\mathcal{P} = \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}$ sur l'ensemble Ω_q et avec l'ensemble $\mathbf{D}_{Sym_q(\Omega)} \cup \mathbf{Q}_{C^*}$ de $\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}$-raffinements élémentaires symétriques (cf. la notation 2.8.1, p. 84, pour la définition de $\mathbf{D}_{Sym_q(\Omega)}$ et le corollaire 2.4.3, p. 68, pour la preuve que $\mathbf{D}_{Sym_q(\Omega)} \cup \mathbf{Q}_{C^*}$ est effectivement un ensemble de $\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}$-raffinements élémentaires). L'algorithme retourne une $\mathbf{D}_{Sym_q(\Omega)} \cup \mathbf{Q}_{C^*}$-base $\mathbf{B}$ pour la propriété $\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}$ (une base $B = (\alpha_1, \ldots, \alpha_s)$ de $Aut(C)$ incluse) et un ensemble fort de générateurs X de $Aut(C)$ associé à la base B. Nous avons trouvé ce que nous cherchions.

Il reste à discuter encore un détail. Comme dans ce cas la propriété donnée $\mathcal{P}$ contient $\mathcal{E}_G$ (avec $G = Sym_q(\Omega)$), l'algorithme devra vérifier si $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^G$ (cf. ligne 10 de l'algorithme 2.6.6 ***Sous-groupe***), où $\mathbf{\Pi}$ et $\mathbf{\Sigma}$ sont deux partitions ordonnées de Ω_q. D'ailleurs, pour appliquer les $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$, il faudra faire encore plus : il faudra trouver une permutation t de G telle que $\mathit{fix}\,(\mathbf{\Pi})^t = \mathit{fix}\,(\mathbf{\Sigma})$. Nous avons expliqué dans la section 3.6 comment le faire dans le cas où $G = Sym_q(\Omega)$.

3.7.4 Problème de l'équivalence des codes

Considérons le problème suivant : étant donnés deux codes linéaires C et C' de la même longueur n sur le même corps fini $\mathbf{F}_q$, vérifier s'ils sont équivalents et si c'est le cas trouver une permutation $g \in Sym_q(\Omega)$ telle que $C^g = C'$. Ce problème est désigné par la propriété $\mathcal{A}_{C,C'}$. La propriété $\mathcal{A}_{C,C'}(g)$ pour une permutation $g \in Sym(\Omega_q)$ est satisfaite si et seulement si $g \in Sym_q(\Omega)$ et $C^g = C'$. D'après la proposition 1.6.22, p. 38, cela est équivalent à $g \in Sym_q(\Omega)$ et $C^{*g} = C'^{*}$, où C^* est le code étalé de C. Donc $\mathcal{A}_{C,C'}(g)$ est satisfaite si et seulement si $\mathcal{E}_{Sym_q(\Omega)}(g)$ et $\mathcal{P}_{C^*,C'^*}(g)$ sont satisfaites, i.e. $\mathcal{A}_{C,C'} = \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*,C'^*}$. Nous pouvons donc appliquer l'algorithme général de Leon 2.7.1 ***RepClasse*** avec la propriété $\mathcal{P} = \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*,C'^*}$ sur l'ensemble Ω_q, l'ensemble $\mathbf{D}_{Sym_q(\Omega)} \cup \mathbf{Q}_{C^*,C'^*}$ de $\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*,C'^*}$-raffinements élémentaires et deux groupes $< f >$, où $f \in Sym_q(\Omega)$ est défini par $(1,\gamma)^f = (a,\gamma)$, $\gamma \in \Omega$, a étant un élément quelconque fixé de $\mathbf{F}_q^*$. En effet, $< f > \subset Perm(C^*)$ pour tout code linéaire C, car appliquer f à C^* revient à multiplier chaque mot de code de C par a, obtenant ainsi des mots de code de C à cause de la linéarité de C.

L'algorithme retourne une $\mathbf{D}_{Sym_q(\Omega)} \cup \mathbf{Q}_{C^*,C'^*}$-base $\mathbf{B}$ pour $\mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*,C'^*}$ et un g tel que $g = \emptyset$ si C^* et C'^* ne sont pas équivalents par une permutation de $Sym_q(\Omega)$ (i.e. si C et C' ne sont pas équivalents), et $g \in Sym_q(\Omega)$ tel que $C^g = C'$ sinon. Nous avons réussi à résoudre notre problème.

Voir la section 3.6 pour apprendre comment vérifier si $\mathit{fix}\,(\mathbf{\Sigma}) \in \mathit{fix}\,(\mathbf{\Pi})^{Sym_q(\Omega)}$ dont nous aurons besoin dans les lignes 2 et 10 de l'algorithme 2.7.1 ***RepClasse***.

3.8 Implantation

Nous avons implanté en langage C la méthode des partitions avec nos raffinements pour le problème du groupe de permutations d'un code linéaire binaire. Nous avons choisi C pour l'efficacité, mais cela a rendu la programmation ardue. La mise au point a été complexe (à peu près 3800 lignes du code source), a pris du temps et a abouti, entre autres, aux heuristiques que nous décrivons dans le reste de cette section.

Dans cette section nous précisons quelques points importants de l'implantation de la méthode des partitions utilisant nos raffinements pour les problèmes liés aux codes.

Soit $\mathcal{P}$ une propriété pour un problème lié aux codes, c'est-à-dire $\mathcal{P} = \mathcal{P}_C$, $\mathcal{P}_{C,C'}$, $\mathcal{A}_C$ ou $\mathcal{A}_{C,C'}$ pour des codes linéaires C et C'. Comme nous l'avons vu dans les sections 3.7.3 et 3.7.4, $\mathcal{A}_C = \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*}$ et $\mathcal{A}_{C,C'} = \mathcal{E}_{Sym_q(\Omega)} \wedge \mathcal{P}_{C^*,C'^*}$. Donc dans tous les quatre cas la propriété $\mathcal{P}$ est de la forme $\mathcal{E}_G \wedge \mathcal{P}_C$ (avec éventuellement $\mathcal{P}_{C,C'}$ remplaçant $\mathcal{P}_C$ et/ou $\mathcal{E}_G$ omis).

Rappelons que si $\mathcal{P} = \mathcal{P}_C$ ou $\mathcal{P}_{C,C'}$, alors $\mathcal{P}$ est une propriété sur Ω, et si $\mathcal{P} = \mathcal{A}_C$ ou $\mathcal{A}_{C,C'}$, alors $\mathcal{P}$ est une propriété sur Ω_q. Pour uniformiser les notations, notons Γ l'ensemble sur lequel $\mathcal{P}$ est défini, c'est-à-dire

$$\Gamma = \begin{cases} \Omega & \text{si } \mathcal{P} = \mathcal{P}_C \text{ ou } \mathcal{P}_{C,C'}, \\ \Omega_q & \text{si } \mathcal{P} = \mathcal{A}_C \text{ ou } \mathcal{A}_{C,C'}. \end{cases}$$

De même, notons $m = |\Gamma|$.

Nous n'expliquons pas comment accélérer le calcul du hull et le calcul du hull d'un code poinçonné, les articles de N. Sendrier [Sen99a, Sen99b] donnent les explications détaillées sur ce sujet.

Revenons à l'algorithme 2.9.2 ***Raffiner***. Comment choisir un $\mathcal{P}$-raffinement élémentaire dans l'ensemble $\mathbf{R}$ de $\mathcal{P}$-raffinements élémentaires (ligne 4 de cet algorithme) ?

Si $\mathcal{P}$ contient $\mathcal{E}_G$ pour un groupe de permutations G différent du groupe symétrique — dans notre cas G ne peut être que $Sym_q(\Omega)$,— alors $\mathbf{D}_G$ est un sous-ensemble de $\mathbf{R}$. Si c'est le cas, nous commençons par choisir des $\mathcal{P}$-raffinements élémentaires de $\mathbf{D}_G$, ils sont plus faciles à appliquer. À cause du lemme 2.8.3(a), p. 85, il suffit de les choisir de la forme $D_{G,\mathbf{\Phi},j,l}$, où $\mathbf{\Phi}$ est la partition courante de Γ que nous cherchons à raffiner (mise à jour dans la ligne 8) et $j, l \in I_m$. Le lemme 2.8.3(b) donne une condition, facile à vérifier, de $\mathbf{D}_G$-irréductibilité : $\mathbf{\Phi}$ est $\mathbf{D}_G$-irréductible si et seulement si $\mathbf{\Phi} \lesssim \mathbf{\Theta}\left(G_{\mathit{fix}(\mathbf{\Phi})}\right)$ (notez que $\mathbf{\Theta}\left(\left(Sym_q(\Omega)\right)_{\mathit{fix}(\mathbf{\Phi})}\right)$ est facile à calculer, cf. lemme 1.6.15, p. 34).

Si $\mathbf{R}$ ne contient pas $\mathbf{D}_G$ ou si nous atteignons une partition courante $\mathbf{\Phi}$ qui n'est plus $\mathbf{D}_G$-réductible, nous commençons à essayer des autres $\mathcal{P}$-raffinements élémentaires de $\mathbf{R}$. Comme ci-dessus pour la propriété $\mathcal{P}$, nous pouvons dire que dans notre cas $\mathbf{R}$ est de la forme $\mathbf{D}_G \cup \mathbf{Q}_C$ (avec éventuellement $\mathbf{Q}_{C,C'}$ au lieu de $\mathbf{Q}_C$ et/ou sans $\mathbf{D}_G$ — cela dépend de notre problème à résoudre). Nous choisissons donc maintenant des $\mathcal{P}$-raffinements élémentaires dans $\mathbf{Q}_C$ (ou $\mathbf{Q}_{C,C'}$, ce qui revient au même dans notre cas, car nous appliquons à la partition courante $\mathbf{\Phi}$ la composante gauche d'un $\mathcal{P}_{C,C'}$-raffinement élémentaire $Q_{C,C',\overline{\mathsf{T}}_{\mathbf{S},L},j,e} = (Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e}, Q_{C',\overline{\mathsf{T}}_{\mathbf{S},L},j,e})$ (ligne 5 de l'algorithme 2.9.2 ***Raffiner***), i.e. dans le cas $\mathcal{P}_{C,C'}$ comme dans le cas $\mathcal{P}_C$ nous appliquons à $\mathbf{\Phi}$ un $\mathcal{P}_C$-raffinement élémentaire $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e}$). Comment choisir un $\mathcal{P}_C$-raffinement élémentaire $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e}$ de $\mathbf{Q}_C$?

En pratique nous choisissons d'abord un sous-ensemble L de I_m (plus précisément, $L \subset I_{|\mathbf{\Phi}|}$, car $\Phi_l = \emptyset$ si $l > |\mathbf{\Phi}|$, et donc $\bigcup_{l\in L} \Phi_l = \bigcup_{l\in L\cap I_{|\mathbf{\Phi}|}} \Phi_l$, par conséquent les entiers $l \in I_m \setminus I_{|\mathbf{\Phi}|}$ n'ont pas d'influence sur l'application d'un $\mathcal{P}_C$-raffinement élémentaire $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e}$, cf. définitions de $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e}$ et de $\mathsf{T}_{\mathbf{S},L}$). Ensuite pour tout $\gamma \in \Gamma$ nous calculons $\overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Phi}, C, \gamma)$. Nous obtenons une partition $\mathbf{\Psi} = \{\Psi_1, \ldots, \Psi_r\}$ de Γ telle que r est le nombre des valeurs distincts de $\overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Phi}, C, \gamma)$ quand γ parcourt Γ, et $\delta, \eta \in \Gamma$ sont dans la même cellule de $\mathbf{\Psi}$ si et seulement si $\overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Phi}, C, \delta) = \overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Phi}, C, \eta)$. Pour tout Ψ_l, $1 \le l \le r$, il existe $e_l \in \mathbb{Z}[X]^4$ tel que $\Psi_l = \{\gamma \in \Gamma \mid \overline{\mathsf{T}}_{\mathbf{S},L}(\mathbf{\Phi}, C, \gamma) = e_l\}$. Alors nous appliquons tout $\mathcal{P}_C$-raffinement élémentaire de l'ensemble

$$Y_{C,L,\mathbf{\Phi}} = \{Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e_l} \mid j \in I_{|\mathbf{\Phi}|}, 1 \le l \le r\}$$

à la partition courante $\mathbf{\Phi}$. En pratique l'application de tous ces $\mathcal{P}_C$-raffinements élémentaires $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e_l}$ de $Y_{C,L,\mathbf{\Phi}}$ à $\mathbf{\Phi}$ revient à faire l'intersection des cellules de $\mathbf{\Phi}$ avec les cellules de $\mathbf{\Psi}$, car par définition $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e_l}(\mathbf{\Phi}) = \mathcal{F}_{j,\mathbf{\Psi}_l}(\mathbf{\Phi})$. Et nous gardons les $\mathcal{P}_C$-raffinements élémentaires $Q_{C,\overline{\mathsf{T}}_{\mathbf{S},L},j,e_l}$ qui réussissent à raffiner $\mathbf{\Phi}$, c'est-à-dire tels que $\emptyset \subsetneq \mathbf{\Phi}_j \cap \mathbf{\Psi}_l \subsetneq \mathbf{\Phi}_j$.

En résumé, d'abord nous choisissons un sous-ensemble L de $I_{|\mathbf{\Phi}|}$, et ensuite nous vérifions tous les $\mathcal{P}_C$-raffinements élémentaires de $Y_{C,L,\mathbf{\Phi}}$. Donc dans la ligne 4 de l'algorithme 2.9.2 ***Raffiner*** nous choisissons non un $\mathcal{P}_C$-raffinement élémentaire, mais un ensemble entier $Y_{C,L,\mathbf{\Phi}}$ de $\mathcal{P}_C$-raffinements élémentaires, et nous exécutons les lignes 5–9 pour tout $\mathcal{P}_C$-raffinement élémentaire E de cet ensemble.

Puis nous répétons : nous appliquons des $\mathcal{E}_G$-raffinements élémentaires de $\mathbf{D}_G$ et, quand la partition courante $\mathbf{\Phi}$ devient $\mathbf{D}_G$-irréductible, nous choisissons encore une fois un sous-ensemble L de $I_{|\mathbf{\Phi}|}$, etc... (évidemment, si $\mathbf{R}$ ne contient pas $\mathbf{D}_G$, nous choisissons tout de suite un sous-ensemble L de $I_{|\mathbf{\Phi}|}$).

Si nous voulons vérifier si la partition courante $\mathbf{\Phi}$ est $\mathbf{Q}_C$-irréductible, il faudra essayer tous les sous-ensembles L de $I_{|\mathbf{\Phi}|}$, il faudra donc faire $2^{|\mathbf{\Phi}|}$ essais. C'est trop si $|\mathbf{\Phi}|$ est grand. Toutefois, nous avons remarqué qu'en pratique pour le problème du groupe de permutations des codes linéaires binaires, en utilisant notre ensemble de $\mathcal{P}_C$-raffinements élémentaires (voir formule (3.6), p. 108), il suffisait d'essayer plusieurs L pour obtenir la partition courante $\mathbf{\Phi}$ $\mathbf{Q}_C$-irréductible. Comment choisir les $L \subset I_{|\mathbf{\Phi}|}$ et combien en essayer ?

Pour choisir $L \subset I_{|\mathbf{\Phi}|}$ nous combinons trois méthodes. D'abord, il parait que le cas le plus favorable ($Y_{C,L,\mathbf{\Phi}}$ contient le plus de $\mathcal{P}_C$-raffinements élémentaires raffinant $\mathbf{\Phi}$) est si L est tel que la dimension de $\mathcal{H}\left(C_{\bigcup_{l\in L}\Phi_l}\right)$ est égale à zéro (cf. [Sen99a, Sec.5.3.1] pour les commentaires sur ce sujet). Nous nous efforçons donc de choisir $L \subset I_{|\mathbf{\Phi}|}$ tel que $|\Phi_l| = 1$ $\forall l \in L$ (les positions contenues dans $\bigcup_{l\in L}\Phi_l$ sont déjà séparées des autres, i.e. elles sont dans *fix* $(\mathbf{\Phi})$) et $\mathcal{H}\left(C_{\bigcup_{l\in L}\Phi_l}\right) = \{\mathbf{0}\}$.

La deuxième méthode — la plus simple — consiste à essayer les $L \subset I_{|\mathbf{\Phi}|}$ tels que $|L| = 1$.

La troisième choisit un sous-ensemble L de $I_{|\mathbf{\Phi}|}$ au hasard — en pratique cette méthode est aussi assez “puissante”.

Et nous arrêtons l'exécution de la boucle principale de l'algorithme *Raffiner* après b choix infructueux consécutifs de L, où b est un nombre fixé d'avance, i.e. nous nous arrêtons si aucun $\mathcal{P}_C$-raffinement élémentaire des ensembles $Y_{C,L,\mathbf{\Phi}}$ associés aux b derniers L n'arrive pas à raffiner la partition courante $\mathbf{\Phi}$.

Discutons brièvement le choix de b. Si nous augmentons b, il y aura plus de chances d'obtenir $\mathbf{\Phi}$ $\mathbf{Q}_C$-irréductible, ce qui rend plus petit l'arbre de la méthode des partitions $\mathbb{T}_{\mathbf{R},\mathbf{B}}$ et donc permet de le parcourir plus rapidement — le coût de la recherche arrière diminue. Mais en même temps il faudra appliquer plus de $\mathcal{P}_C$-raffinements élémentaires à la partition courante $\mathbf{\Phi}$, ce qui augmente le coût de l'algorithme *Raffiner* et, en général, du calcul d'une $\mathbf{R}$-base (cf. algorithme 2.9.1 *Base*). Il s'agit donc de trouver un compromis convenable. Dans la figure 3.1 nous donnons deux exemples qui montrent comment le temps d'exécution de notre programme varie en fonction de b. Nous présentons le temps d'exécution pour deux codes : un code linéaire binaire $[35, 13]$ et un code cyclique binaire $[45, 13, 8]$. Nous voyons que le temps d'exécution est très instable quand $0 \leq b \leq 20$, il vaut mieux donc choisir b dans l'intervalle de 21 à 30. Nous avons remarqué que plus un code a de la “structure”, plus petite valeur de b on peut choisir. Par exemple, pour les codes cycliques binaires on peut choisir $10 \leq b \leq 20$, pour les codes résidus quadratiques binaires [8k+1,4k] il suffit de prendre b égal à 3 ou 4, etc...

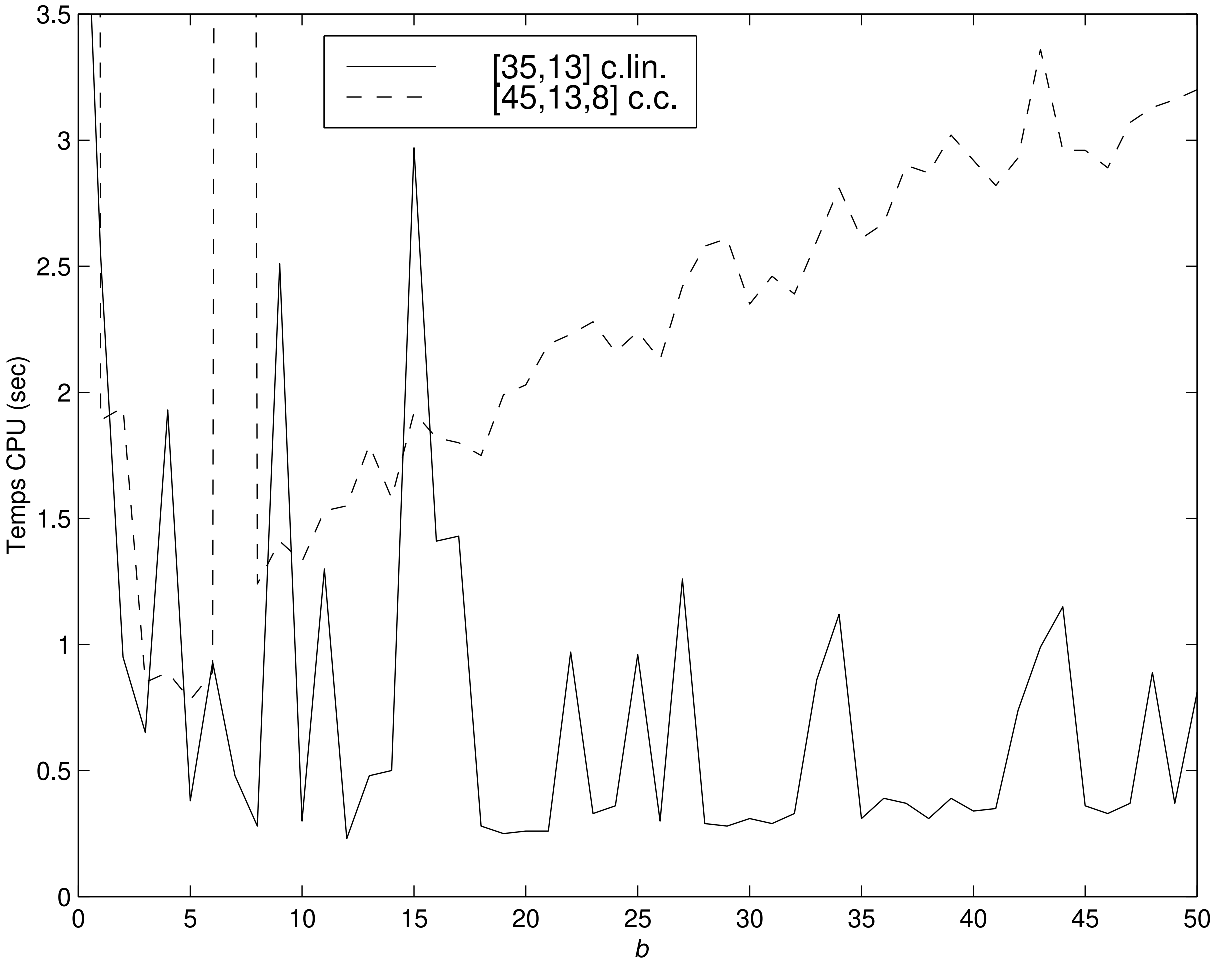

FIG. 3.1 – *Le temps d'exécution en fonction de b sur DEC Alpha EV56 500/400 (400 Mhz)*

3.9 Comparaison avec l'algorithme de Leon

L'algorithme de Leon n'a pas de raffinements propres aux codes, plus précisément, pour les problèmes liés aux codes, il utilise les mêmes raffinements que pour les matrices. Comment fonctionne-t-il ? En plus d'un code C, l'algorithme de Leon pour les problèmes liés aux codes prend en entrée un ensemble W de vecteurs invariant sous l'action de $Aut(C)$ (ou $Perm(C)$, cela dépend du problème considéré). Il considère cet ensemble W comme une matrice dont le groupe d'automorphismes (ou de permutations) il doit trouver, et il parcourt l'arbre de recherche arrière défini par cette matrice (plus précisément, par un ensemble de raffinements pour les matrices, nous n'en parlerons pas ici, cf. [Leo91, §9(f–g), §9(i), §10(f–g), §10(i)]). L'ensemble W peut être l'ensemble de tous les mots de code de C, $C^{\perp}$ ou $\mathcal{H}(C)$, l'ensemble de mots de code de poids minimal (ou de poids fixé) de C, $C^{\perp}$ ou $\mathcal{H}(C)$, une réunion de plusieurs de ces ensembles, etc... Mais W doit être "raisonnablement petit", sinon cela prend trop de temps de manipuler la matrice W, et doit "bien représenter" le code C, i.e. le groupe d'automorphismes (ou de permutations) de W ne doit pas être bien plus grand que $Aut(C)$ (ou $Perm(C)$), sinon dans l'arbre de recherche arrière il n'y aura qu'un petit pourcentage des feuilles fournissant les éléments de $Aut(C)$ (ou $Perm(C)$), même si presque toutes les feuilles fournissent les éléments de $Aut(W)$ (ou $Perm(W)$).

En pratique, quand un tel ensemble W est disponible, l'algorithme de Leon avec ses raffinements est un peu plus rapide qu'avec les nôtres, car l'application des raffinements de Leon demande bien moins de temps que celle des nôtres. Mais un tel ensemble n'est disponible que pour des petits codes (avec quelques exceptions). Par exemple, si C et $C^{\perp}$ sont trop grands pour calculer l'ensemble de mots de poids minimal et si $\mathcal{H}(C)$ est trop petit, on n'a pas de tel ensemble W, donc l'algorithme de Leon avec ses raffinements ne marche pas, mais il marche avec les nôtres, parce que la contrainte principale de nos raffinements est la taille du hull. Cela arrive très souvent, par exemple, les codes linéaires aléatoires possèdent un petit hull avec une probabilité pratiquement égale à 1 (cf. [Sen97a, Sen99a, Sen99b] et la remarque dans la section 3.5 sur la dimension moyenne du hull), tous les codes cycliques sur $\mathbf{F}_q$ de longueur $n = q^m + 1$, $m = 1, 2, \ldots$ possèdent un hull de dimension 0, etc...

3.10 Quelques exemples de temps d'exécution

Nous présentons deux exemples du temps d'exécution de notre implantation de l'algorithme de Leon avec nos raffinements pour le problème du groupe de permutations d'un code linéaire binaire.

D'abord, nous calculons le groupe de permutations pour tous les codes BCH au sens strict non dégénérés binaires de longueur $n = 2^m + 1$, $m = 4, 5, \ldots, 10, 11$ (cf. définitions 1.3.14, p. 21, 4.3.21 et 4.3.22, p. 134). Dans le tableau 3.1 nous donnons pour tout m le nombre de tels codes, le temps d'exécution minimal, moyen et maximal de notre implantation sur un DEC Alpha EV56 500/400 (400 Mhz). Notez que si la dimension d'un code

		nombre	Temps d'exécution (en sec de CPU)		
m	n	de codes	minimal	moyen	maximal
4	17	1	0.016	0.016	0.016
5	33	2	0.027	0.028	0.029
6	65	4	0.055	0.060	0.064
7	129	8	0.121	0.150	0.190
8	257	15	0.253	0.400	0.586
9	513	28	0.757	1.468	2.170
10	1025	50	2.619	5.795	10.152
11	2049	92	11.878	20.894	31.446

TAB. 3.1 – *Le temps d'exécution pour les codes BCH de longueur n*

C est supérieure à la moitié de sa longueur, nous calculons le groupe de permutations de son code dual $C^\perp$, car $Perm(C) = Perm(C^\perp)$ et la dimension de $C^\perp$ est inférieure à la dimension de C. Le coût de la construction de $C^\perp$ n'est pas inclus dans les nombres donnés. Notez de même que nous n'utilisons que les $\mathcal{P}_C$-raffinements élémentaires entraînant le calcul de l'énumérateur des poids d'un hull de dimension inférieure ou égale à 10, i.e. si nous rencontrons un $\mathcal{P}_C$-raffinement élémentaire demandant le calcul de l'énumérateur des poids d'un hull de dimension supérieure à 10, nous le jetons et nous choisissons un autre $\mathcal{P}_C$-raffinement élémentaire. Notez encore que nous posons $b = 10$ dans ces calculs (cf. section 3.8 pour le rôle de b).

Deuxième exemple : dans la figure 3.2 nous présentons le temps d'exécution sur la même machine de tous les codes résidus quadratiques binaires (cf. définition 4.3.23, p. 134) de la forme $[8k+1, 4k]$ de longueur jusqu'à 10000. Le hull de ces codes est $\{\mathbf{0}\}$. Nous posons $b = 3$, et nous ne calculons que les énumérateurs des poids des hulls de dimension inférieure ou égale à 2.

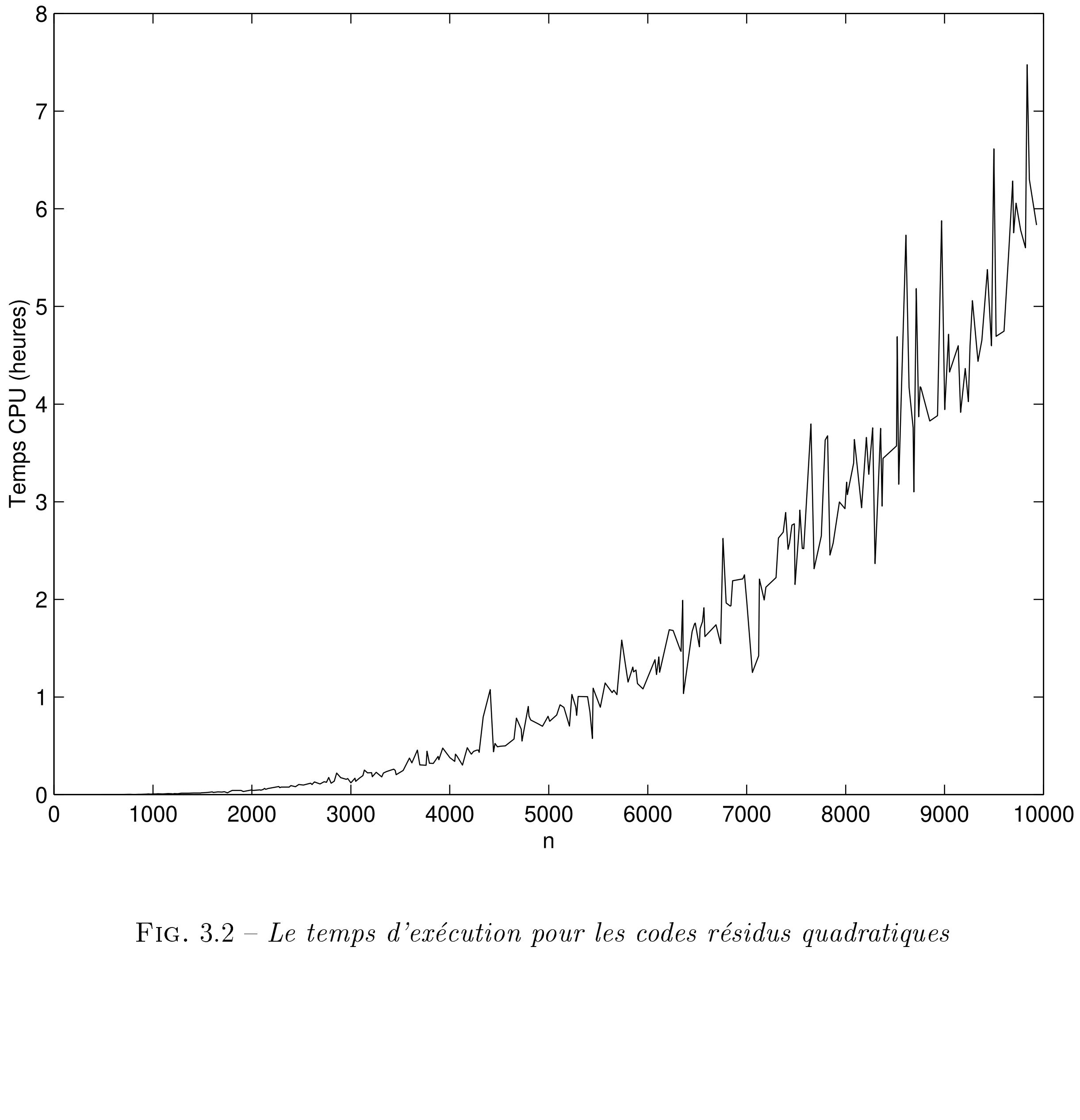

FIG. 3.2 – *Le temps d'exécution pour les codes résidus quadratiques*

Chapitre 4

Étude de la dimension du hull des codes cycliques

4.1 Introduction

La méthode des partitions avec nos raffinements ne marche que si le hull d'un code linéaire donné est petit (dans le cas binaire, par exemple, de dimension inférieure en général à vingt, cela dépend en fait de la longueur du code, plus la longueur est grande, plus la dimension du hull doit être petite). Les hulls de la plupart des codes linéaires sont petits, mais dans certaines classes importantes de codes il existe beaucoup de codes ayant un grand hull. Par exemple, P. Charpin et F. Levy-dit-Vehel [CL94] ont trouvé une famille infinie de codes affine-invariants auto-duaux. De même, nous avons remarqué que beaucoup de codes cycliques possèdent un grand hull. Nous nous sommes posés la question suivante : combien y-a-t-il des codes cycliques pour lesquels notre méthode n'est pas applicable? Nous avons fait des essais (cf. figure 4.1, p. 142) qui nous ont conduit à la conjecture suivante :

Le hull de la "plupart" de codes cycliques est "grand".

Nous présentons dans ce chapitre quelques résultats obtenus au cours de notre étude pour démontrer cette conjecture, que nous n'avons malheureusement pas pu établir.

D'abord nous rappelons, définissons et présentons les propriétés des outils utilisés dans notre étude, en particulier, les multi-ensembles (section 4.2.1), les polynômes minimaux (section 4.2.4.1), les racines de l'unité (section 4.2.4.3) et les classes cyclotomiques (section 4.2.4.4). Nous rappelons quelques résultats nécessaires de la théorie des nombres (section 4.2.2) et de la théorie des groupes (section 4.2.3).

Ensuite nous définissons l'ensemble de définition d'un code cyclique et étudions ses propriétés (section 4.3.1). Nous généralisons cette notion dans le cas où la longueur n du code et le cardinal q de son corps de base ne sont pas premiers entre eux (section 4.3.2). Dans la section 4.4 nous exprimons l'ensemble de définition du hull d'un code cyclique C en fonction de l'ensemble de définition de C. Cela va nous donner un moyen pour exprimer $Edimh_q(n)$ (cf. ci-dessous) par les cardinaux des classes cyclotomiques réversibles. Au passage nous étudions brièvement le nombre de codes cycliques de longueur donnée n

sur un corps donné $\mathbf{F}_q$ (section 4.5).

Nous définissons $Edimh_q(n)$ comme la dimension moyenne du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ (section 4.6.1). Soit $\mathcal{N}_q$ l'ensemble des diviseurs des entiers de la forme q^m+1, $m > 0$. Nous montrons que $Edimh_q(n) = 0$ si et seulement si $n \in \mathcal{N}_q$ (section 4.6.2). Pour continuer l'étude de $Edimh_q(n)$, nous l'exprimons en fonction de $B_{n,q}$ — la somme des cardinaux des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$ (sections 4.6.3–4.6.4). La définition de $B_{n,q}$ ne nous permet pas de le manipuler facilement, c'est pourquoi nous consacrons les sections 4.6.5, 4.7 et 4.8.1 à son étude. Dans la section 4.6.5 nous l'exprimons comme une somme de $\phi(l)$, où ϕ est la fonction d'Euler et l parcourt les diviseurs de n appartenant à $\mathcal{N}_q$. Pour expliciter cette formule nous nous intéressons à la structure de $\mathcal{N}_q$ (section 4.7). Cela nous permet d'obtenir une expression de $B_{n,q}$ plus facile à manipuler (section 4.8.1). Nous utilisons cette expression de $B_{n,q}$ pour étudier le comportement asymptotique de $Edimh_q(n)$ quand n tend vers l'infini (sections 4.8.2.2–4.8.2.3 et 4.8.3.1). Enfin nous montrons qu'il existe une constante $c > 0$ telle que $Edimh_q(n) \geq \frac{n}{c}$ pour tout $n \geq 1$, $n \notin \mathcal{N}_q$, et nous discutons la meilleure valeur possible de cette constante (sections 4.8.2.4, 4.8.3.3 et 4.8.4). Nous montrons donc que soit $Edimh_q(n)$ est nul (si et seulement si $n \in \mathcal{N}_q$), soit il est "grand" (de l'ordre de n).

Il reste la question de savoir si $\mathcal{N}_q$ est grand par rapport à $\mathbb{N}$, c'est-à-dire s'il y a beaucoup de n tels que $Edimh_q(n) = 0$. Nous conjecturons (section 4.9) que

$\mathcal{N}_q$ est négligeable par rapport à $\mathbb{N}$, autrement dit, presque tous les entiers positifs non nuls n'appartiennent pas à $\mathcal{N}_q$.

Si cette conjecture était vraie, il existerait "peu" d'entiers n tels que $Edimh_q(n)$ soit "petit", i.e. tels que notre méthode marcherait pour la plupart de codes cycliques de longueur n sur $\mathbf{F}_q$.

4.2 Définitions et rappels

4.2.1 Multi-ensembles

Définition 4.2.1 *Un* multi-ensemble *est un ensemble qui peut contenir un élément plusieurs fois.*

Par exemple, $\{1, 1, 2, 2, 2, 3, 6, 6\}$ est un multi-ensemble.

Définition 4.2.2 *Si un élément i est présent t fois dans un multi-ensemble E, nous dirons que la* multiplicité *de i dans E est t.*

Si i ne se trouve pas dans E, il sera de multiplicité 0 dans E.

Si I est un ensemble tel que tout $i \in I$ est de multiplicité t dans E, nous dirons que t est la multiplicité de I dans E.

Comme chaque élément peut apparaître dans un multi-ensemble beaucoup de fois, nous simplifions les notations.

Notation 4.2.3 Si i est de multiplicité t dans E, au lieu d'écrire i t fois nous écrirons $i\hat{}t$. Nous noterons $i\hat{}t \in E$. Si $t = 1$, on pourra écrire simplement i une fois. Si i ne se trouve pas dans E, nous écrirons $i\hat{}0 \in E$.

Si i est présent dans E au moins r fois, nous écrirons $i\hat{}\hat{}r \in E$. En général, nous utiliserons la notation $i\hat{}t \in E$ quand t est la multiplicité de i dans E, et $i\hat{}\hat{}r \in E$ quand $0 \leq r \leq t$.

De plus, dans $i\hat{}t$ et $i\hat{}\hat{}r$ on peut remplacer i par un ensemble I. Alors $I\hat{}t \subset E$ signifiera que $i\hat{}t \in E$ pour tout $i \in I$, et $I\hat{}\hat{}r \subset E$ signifiera que $i\hat{}\hat{}r \in E$ pour tout $i \in I$. De même, $I\hat{}t$ sera le multi-ensemble $\{i\hat{}t \mid i \in I\}$. □

Par exemple, nous écrirons $E = \{1\hat{}2, 2\hat{}3, 3\hat{}1, 6\hat{}2\}$ ou $E = \{1\hat{}2, 2\hat{}3, 3, 6\hat{}2\}$ au lieu d'écrire $E = \{1, 1, 2, 2, 2, 3, 6, 6\}$.

Nous écrivons $1\hat{}2 \in E$, $2\hat{}3 \in E$, $3\hat{}1 \in E$, $6\hat{}2 \in E$ et $1\hat{}\hat{}2 \in E$, $1\hat{}\hat{}1 \in E$, $2\hat{}\hat{}3 \in E$, $2\hat{}\hat{}2 \in E$, etc...

On peut définir pour les multi-ensembles les équivalents de certaines notions définies pour les ensembles.

Définition 4.2.4 *Soit E, F deux multi-ensembles.*

(i) *Si $E = \{i_1\hat{}t_1, \ldots, i_l\hat{}t_l\}$, le* cardinal *de E, noté $|E|$, est $\sum_{j=1}^{l} t_j$.*

(ii) *Nous disons que E est un* sous-ensemble *de F et nous écrivons $E \subset F$ si la condition suivante est satisfaite : si i est un élément de E de multiplicité $t > 0$, alors i doit être présent dans F et sa multiplicité dans F doit être supérieure ou égale à t. Plus précisément, $E \subset F$ si $i\hat{}\hat{}t \in F$ pour tout $i\hat{}t \in E$.*

(iii) *La* réunion *$E \cup F$ de E et F est définie par*

$$E \cup F = \{i\hat{}(t+s) \mid i\hat{}t \in E,\ i\hat{}s \in F,\ t \geq 0,\ s \geq 0,\ t+s > 0\}.$$

Autrement dit, on additionne les multiplicités d'éléments présents dans au moins un multi-ensemble E ou F.

(iv) *La* différence *$F \setminus E$ entre F et E est définie par*

$$F \setminus E = \{i\hat{}(s-t) \mid i\hat{}s \in F,\ s > 0,\ i\hat{}t \in E,\ t \geq 0, s-t > 0\}.$$

Autrement dit, on soustrait les multiplicités d'éléments présents dans F.

Notez que la notation $\subset$ est définie aussi dans la notation 4.2.3, mais d'une manière différente. Quand elle sera utilisée avec les notations $\hat{}$ ou $\hat{}\hat{}$ (par exemple, $I\hat{}t \subset E$), nous sous-entendrons le sens de la notation 4.2.3, sinon (par exemple, $E \subset F$) le sens de la définition 4.2.4(ii).

Il existe encore une façon pour généraliser la notion de réunion d'ensembles.

Notation 4.2.5 Soit E, F deux multi-ensembles. Nous notons $E + F$ le multi-ensemble

$$E + F = \{i\hat{}\max(s,t) \mid i\hat{}t \in E,\ i\hat{}s \in F,\ t \geq 0,\ s \geq 0,\ \max(s,t) > 0\}.$$

Autrement dit, on retient la multiplicité la plus grande pour tout élément présent dans au moins un multi-ensemble E ou F. □

On peut généraliser les notations $\cup$ et $+$ de façon évidente pour une suite de multi-ensembles $E_1, E_2, \ldots, E_l$.

Notation 4.2.6 Pour un entier positif n et pour un entier positif non nul l, nous noterons Z_n^l le multi-ensemble

$$Z_n^l = Z_n\hat{}l = \{0\hat{}l, 1\hat{}l, 2\hat{}l, \ldots, (n-1)\hat{}l\}.$$

Remarque 4.2.7 Notez que le cardinal de Z_n^l est $\left|Z_n^l\right| = nl$.

4.2.2 Rappels de théorie des nombres

On peut trouver les résultats de cette section dans, par exemple, [Apo76].

Soient n et a deux entiers positifs non nuls.

Définition 4.2.8 *La* fonction d'Euler $\phi(n)$ *est définie comme le nombre d'entiers positifs non nuls inférieurs ou égaux à n, premiers avec n, i.e.*

$$\phi(n) = \sum_{\substack{1 \leq i \leq n \\ (i,n)=1}} 1 = \left|\{i \mid 1 \leq i \leq n,\ (i,n) = 1\}\right|.$$

Notez que $\phi(1) = 1$.

Nous rappelons quelques propriétés de $\phi(n)$.

Propriété 4.2.9 *(i) La fonction ϕ est une fonction multiplicative, c'est-à-dire si $(n,a) = 1$, alors $\phi(na) = \phi(n)\phi(a)$.*

(ii) Si p est un nombre premier, alors $\phi(p) = p - 1$.

(iii) Si p est un nombre premier et i est un entier positif non nul, alors $\phi(p^i) = p^{i-1}(p-1)$.

(iv)

$$\phi(n) = n \prod_{\substack{p \mid n \\ p \text{ premier}}} \left(1 - \frac{1}{p}\right).$$

(v)

$$\sum_{d \mid n} \phi(d) = n.$$

Maintenant nous allons introduire l'ordre modulo n.

Théorème 4.2.10 (Euler—Fermat) *Si* $(n, a) = 1$, *alors* $a^{\phi(n)} \equiv 1 \bmod n$.

Définition 4.2.11 *Soit* $(n, a) = 1$. *Le plus petit entier positif non nul* r *tel que*

$$a^r \equiv 1 \bmod n$$

*est appelé l'*ordre de a modulo n *et est noté* $\mathrm{ord}_n(a)$.

Remarque 4.2.12 Notez que $a^r \equiv 1 \bmod n$ s'écrit aussi $n \mid a^r - 1$.

Notez encore que $\mathrm{ord}_1(a) = 1$ pour tout a dans $\mathbb{N}^*$.

Si $n > 1$, l'ordre $\mathrm{ord}_n(a)$ de a modulo n est l'ordre de a dans le groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^*$. □

Propriété 4.2.13 *Soit* $(n, a) = 1$. *Alors* $a^j \not\equiv a^i \bmod n$ *pour tous les* i *et* j *tels que* $1 \leq i < j \leq \mathrm{ord}_n(a)$.

Preuve. Supposons que $a^j \equiv a^i \bmod n$. Alors $a^{j-i} \equiv 1 \bmod n$ avec $1 \leq j - i < \mathrm{ord}_n(a)$, ce qui contredit la définition de $\mathrm{ord}_n(a)$. □

Définition 4.2.14 *Soit* $(n, a) = 1$. *Si* $\mathrm{ord}_n(a) = \phi(n)$, *nous appelons* a racine primitive modulo n.

Théorème 4.2.15 *Soit* p *un nombre premier,* $p > 2$. *Si* a *est une racine primitive modulo* p^2, *alors* a *est une racine primitive modulo* p^i *pour tout* $i \geq 1$.

4.2.3 Rappels sur les groupes

Nous rappelons quelques résultats sur les groupes, qui nous seront utiles dans ce chapitre.

Soit G un groupe multiplicatif fini. Notons $\mathrm{ord}(h)$ l'ordre de $h \in G$, i.e. le plus petit entier positif non nul r tel que $h^r = 1$, où 1 est l'élément neutre pour la multiplication de G.

Proposition 4.2.16 *Soit* h *un élément de* G, *soit* l *un entier positif non nul. Si* $h^l = 1$, *alors* $\mathrm{ord}(h) \mid l$.

Proposition 4.2.17 *Soit* h *un élément de* G, *soit* l *un entier positif non nul. Alors*

$$\mathrm{ord}(h^l) = \frac{\mathrm{ord}(h)}{(\mathrm{ord}(h), l)}.$$

Preuve. Soit $r = \text{ord}(h)$. Donc,

$$h^r = 1 \quad \text{et} \quad h^i \neq 1 \;\; \forall\, i,\; 1 \leq i \leq r-1.$$

Alors

$$\left(h^l\right)^{\frac{r}{(r,l)}} = \left(h^r\right)^{\frac{l}{(r,l)}} = 1 \quad \text{et donc} \quad j \overset{\text{déf}}{=} \text{ord}(h^l) \mid \tfrac{r}{(r,l)}.$$

Soit $m = \frac{r}{(r,l)j}$. Comme $\left(h^l\right)^j = h^{lj} = 1$, nous avons $r \mid lj = \frac{rl}{(r,l)m}$ (cf. proposition 4.2.16), donc $(r,l)\,m \mid l$. D'ailleurs, $(r,l)\,m = \frac{r}{j} \mid r$. Donc, $(r,l)\,m \mid (r,l)$, d'où $m = 1$ et $j = \frac{r}{(r,l)}$. □

Nous avons montré le résultat suivant.

Proposition 4.2.18 *Soit G un groupe multiplicatif cyclique fini d'ordre n engendré par g. Soit m un diviseur de n. Le groupe G possède exactement $\phi(m)$ éléments d'ordre m. En particulier, G possède $\phi(n)$ éléments d'ordre n.*

Preuve. Soit m un diviseur de n. La proposition 4.2.17 montre que $h \overset{\text{déf}}{=} g^i \in G$, $1 \leq i \leq n$, est d'ordre m si et seulement si $\frac{n}{(n,i)} = m$. Combien existe-t-il de i, $1 \leq i \leq n$, tels que $(n,i) = \frac{n}{m}$? i est forcément un multiple de $\frac{n}{m}$, i.e. $i \in \{\frac{jn}{m} \mid 1 \leq j \leq m\}$. Soit $i = \frac{jn}{m}$, $1 \leq j \leq m$. Comme $(i,n) = \left(\frac{jn}{m}, \frac{mn}{m}\right) = \frac{n}{m}\,(j,m)$, nous avons que $(i,n) = \frac{n}{m}$ si et seulement si $(j,m) = 1$. Il existe $\phi(m)$ éléments j, $1 \leq j \leq m$, tels que $(j,m) = 1$, il existe donc $\phi(m)$ éléments i, $1 \leq i \leq n$, tels que $(n,i) = \frac{n}{m}$ et donc $\phi(m)$ éléments de G d'ordre m. □

Corollaire 4.2.19 *Soit G un groupe multiplicatif cyclique fini d'ordre n engendré par g. Nous pouvons partitionner G en classes selon l'ordre des éléments ; la classe dont tous les éléments sont d'ordre m est non vide si et seulement si m est un diviseur de n ; il y a $\phi(m)$ éléments dans cette classe.*

Preuve. Conséquence directe des propositions 4.2.17 et 4.2.18. □

4.2.4 Rappels sur les corps finis

Nous rappelons certaines notions liées aux corps finis, notamment les polynômes minimaux, la conjugaison, les racines de l'unité et les classes cyclotomiques. Pour plus de détails voir [LN83] ou un des ouvrages sur les codes correcteurs d'erreurs, par exemple, [MS77], [PW72] ou [PW95].

Dans toute cette section q sera une puissance d'un nombre premier p, m et n seront deux entiers positifs non nuls, n sera premier avec q.

4.2.4.1 Polynôme minimal

Définition 4.2.20 *Soit* $\alpha \in \mathbf{F}_{q^m}$*. Le* polynôme minimal *de* α *sur* $\mathbf{F}_q$ *est le polynôme unitaire de plus bas degré* $f(X) \in \mathbf{F}_q[X]$ *vérifiant* $f(\alpha) = 0$*. Nous le notons* $M_\alpha(X)$*.*

Nous donnons quelques résultats sur le degré des polynômes minimaux, qui nous mènent au corollaire 4.2.23 utilisée dans la section 4.5 pour calculer le nombre de classes cyclotomiques.

Proposition 4.2.21 *[MS77, p.106] Soit* $\alpha \in \mathbf{F}_{q^m}$*. Soit* d *un entier positif non nul. Le degré* $\deg M_\alpha(X)$ *du polynôme minimal* $M_\alpha(X)$ *de* α *sur* $\mathbf{F}_q$ *est égal à* d *si et seulement si* d *est le plus petit entier positif non nul tel que* $\alpha^{q^d} = \alpha$*.*

Rappelons que l'*ordre* de α (dans le groupe multiplicatif $\mathbf{F}_{q^m}^*$) est le plus petit entier positif non nul l tel que $\alpha^l = 1$.

Lemme 4.2.22 *Soit* $\alpha \in \mathbf{F}_{q^m}$*. Soit* l *l'ordre de* α*. Soit* d *un entier positif non nul. Alors* d *est le plus petit entier positif non nul tel que* $\alpha^{q^d} = \alpha$ *si et seulement si* $d = \mathrm{ord}_l(q)$*.*

Preuve. Notons $r = \mathrm{ord}_l(q)$. D'après la définition de l'ordre de q modulo l (cf. définition 4.2.11), nous avons $l \mid q^r - 1$. Mais $\alpha^l = 1$, donc $\alpha^{q^r - 1} = 1$ et $\alpha^{q^r} = \alpha$. Et r est le plus petit entier positif non nul avec cette propriété, compte tenu de la même définition. Donc r est égal à d si et seulement si d est le plus petit entier positif non nul tel que $\alpha^{q^d} = \alpha$. □

Corollaire 4.2.23 *Soit* $\alpha \in \mathbf{F}_{q^m}$*. Soit* l *l'ordre de* α*. Alors*

$$\deg M_\alpha(X) = \mathrm{ord}_l(q).$$

Preuve. C'est une conséquence directe de la proposition 4.2.21 et du lemme 4.2.22. □

Proposition 4.2.24 *Soit* $\alpha \in \mathbf{F}_{q^m}$*. Soit* l *l'ordre de* α*. Alors*

$$M_\alpha(X) = \prod_{i=0}^{\mathrm{ord}_l(q)-1} (X - \alpha^{q^i}),$$

c'est-à-dire $\{\alpha, \alpha^q, \alpha^{q^2}, \alpha^{q^3}, \ldots, \alpha^{q^{\mathrm{ord}_l(q)-1}}\}$ *est l'ensemble des racines de* $M_\alpha(X)$*.*

Remarque 4.2.25 La proposition 4.2.21 et le corollaire 4.2.23 nous montrent que

$$\alpha^{q^{\mathrm{ord}_l(q)}} = \alpha.$$

Proposition 4.2.26 *Soit* $\alpha \in \mathbf{F}_{q^m}$*. Toutes les racines de* $M_\alpha(X)$ *sont de même ordre.*

4.2.4.2 Conjugaison

Définition 4.2.27 *La* conjugaison *dans* $\mathbf{F}_{q^m}$ *est la relation* R *définie par*

$$\alpha R \beta \quad si \quad M_\alpha(X) = M_\beta(X).$$

Proposition 4.2.28 *La conjugaison dans* $\mathbf{F}_{q^m}$ *est une relation d'équivalence.*

Définition 4.2.29 *Les* conjugués *d'un élément* α *de* $\mathbf{F}_{q^m}$ *sont les éléments de la classe d'équivalence de* α *pour la conjugaison dans* $\mathbf{F}_{q^m}$.

Proposition 4.2.30 *Soit* $\alpha \in \mathbf{F}_{q^m}$. *Soit* l *l'ordre de* α. *Les conjugués de* α *sont*

$$\alpha, \alpha^q, \alpha^{q^2}, \alpha^{q^3}, \ldots, \alpha^{q^{\mathrm{ord}_l(q)-1}}.$$

Ils sont distincts deux à deux.

Preuve. C'est une conséquence directe de la définition 4.2.27 et de la proposition 4.2.24. □

Remarque 4.2.31 En résumé, tous les éléments de $\mathbf{F}_{q^m}$ sont divisés en classes d'équivalence pour la conjugaison. Une classe d'équivalence est composée de toutes les racines d'un polynôme minimal sur $\mathbf{F}_q$. Donc

- il y a autant des classes d'équivalence que de polynômes minimaux différents des éléments de $\mathbf{F}_{q^m}$,
- le cardinal de toute classe est égal au degré du polynôme minimal correspondant.

4.2.4.3 Racines de l'unité

Rappelons que $(n, q) = 1$. Soit m un entier positif non nul tel que $n \mid q^m - 1$.

Définition 4.2.32 *On appelle* racine n-ième de l'unité sur $\mathbf{F}_q$, *un élément de* $\mathbf{F}_{q^m}$ *dont l'ordre divise* n, *on appelle* racine n-ième primitive de l'unité sur $\mathbf{F}_q$, *un élément de* $\mathbf{F}_{q^m}$ *d'ordre* n.

En particulier si $n = q^m - 1$, une racine primitive n-ième de l'unité sur $\mathbf{F}_q$ est un élément primitif de $\mathbf{F}_{q^m}$.

Les racines n-ièmes de l'unité sur $\mathbf{F}_q$ forment un sous-groupe du groupe multiplicatif $\mathbf{F}_{q^m}^*$. En effet, si β et γ sont deux racines n-ièmes de l'unité sur $\mathbf{F}_q$, $(\beta\gamma)^n = \beta^n \gamma^n = 1$ et donc $\beta\gamma$ est aussi une racine n-ième de l'unité sur $\mathbf{F}_q$. D'ailleurs, $(\beta^{-1})^n = (\beta^n)^{-1} = 1$. Donc les racines n-ièmes de l'unité sur $\mathbf{F}_q$ forment un sous-groupe de $\mathbf{F}_{q^m}^*$. Comme $\mathbf{F}_{q^m}^*$ est cyclique, ce sous-groupe est aussi cyclique.

Soit u l'entier tel que $un = q^m - 1$. Soit α un élément primitif de $\mathbf{F}_{q^m}$. Alors $\beta \overset{\text{déf}}{=} \alpha^u$ est une racine n-ième primitive de l'unité sur $\mathbf{F}_q$, car d'après la proposition 4.2.17, l'ordre de α^u est égal à $\frac{q^m-1}{(q^m-1,u)} = \frac{q^m-1}{u} = n$. Donc β est un générateur de ce sous-groupe qui est d'ordre n.

Ce sous-groupe est composé de toutes les racines de $X^n - 1$, i.e. la décomposition de $X^n - 1$ sur $\mathbf{F}_{q^m}$ est

$$X^n - 1 = \prod_{i=0}^{n-1}(X - \beta^i). \tag{4.1}$$

Soit γ une racine n-ième de l'unité sur $\mathbf{F}_q$. Ses conjugués dans $\mathbf{F}_{q^m}$ sont les puissances de γ (cf. proposition 4.2.30), donc ils sont aussi des racines n-ièmes de l'unité sur $\mathbf{F}_q$. La conjugaison dans $\mathbf{F}_{q^m}$ définit donc une relation d'équivalence dans l'ensemble des racines n-ièmes de l'unité sur $\mathbf{F}_q$. On peut alors dire les mêmes choses comme dans la remarque 4.2.31 : chaque classe d'équivalence est composée de toutes les racines d'un polynôme minimal, et

- il y a autant des classes d'équivalence que de polynômes minimaux différents des racines n-ièmes de l'unité sur $\mathbf{F}_q$,
- le cardinal de toute classe est égal au degré du polynôme minimal correspondant.

Nous obtenons aussi que

$$X^n - 1 = \prod_{\gamma} M_\gamma(X), \tag{4.2}$$

où γ parcourt un ensemble de représentants des classes d'équivalence, et, compte tenu de la proposition 4.2.24, que le polynôme minimal de $\gamma \overset{\text{déf}}{=} \beta^j$, $j \in Z_n$, est égal à

$$M_\gamma(X) = \prod_{i=0}^{\mathrm{ord}_l(q)-1} (X - \gamma^{q^i}) = \prod_{i=0}^{\mathrm{ord}_l(q)-1} (X - \beta^{jq^i}), \tag{4.3}$$

où l est l'ordre de γ, $l = \frac{n}{(n,j)}$ d'après la proposition 4.2.17.

Cas général Prenons maintenant le cas général où n et q ne sont pas forcément premiers entre eux. Soit $n = rp^s$, où r est premier avec p et $s \geq 0$ (p^s est la plus grande puissance de p qui divise n). Alors

$$X^n - 1 = X^{rp^s} - 1 = (X^r - 1)^{p^s}, \tag{4.4}$$

car nous travaillons sur le corps $\mathbf{F}_q$ de caractéristique p.

Puisque r est premier avec p, nous pouvons décomposer $X^r - 1$ comme ci-dessus, et en déduire la décomposition de $X^n - 1$. Plus précisément, si β est une racine r-ième primitive de l'unité sur $\mathbf{F}_q$, alors

$$X^r - 1 = \prod_{i=0}^{r-1}(X - \beta^i),$$

et donc

$$X^n - 1 = (X^r - 1)^{p^s} = \left(\prod_{i=0}^{r-1}(X - \beta^i)\right)^{p^s} = \prod_{i=0}^{r-1} (X - \beta^i)^{p^s}. \tag{4.5}$$

De même,

$$X^n - 1 = (X^r - 1)^{p^s} \overset{(4.2)}{=} \left(\prod_{\gamma} M_\gamma(X)\right)^{p^s} = \prod_{\gamma} M_\gamma(X)^{p^s},$$

où γ parcourt un ensemble de représentants des classes d'équivalence par conjugaison des racines r-ièmes de l'unité sur $\mathbf{F}_q$.

4.2.4.4 Classes cyclotomiques

Soit $(n, q) = 1$.

Soit β une racine n-ième primitive de l'unité sur $\mathbf{F}_q$. La relation d'équivalence sur les racines n-ièmes de l'unité sur $\mathbf{F}_q$, décrite dans la section 4.2.4.3, induit une relation d'équivalence dans l'ensemble Z_n comme suit : $i, j \in Z_n$ sont dans la même classe d'équivalence si et seulement si β^i et β^j sont dans la même classe. À la classe de $\gamma = \beta^j$, i.e. la classe $\{\gamma, \gamma^q, \gamma^{q^2}, \gamma^{q^3}, \ldots, \gamma^{q^{r-1}}\} = \{\beta^j, \beta^{jq}, \beta^{jq^2}, \beta^{jq^3}, \ldots, \beta^{jq^{r-1}}\}$, correspond la classe des exposants $\{j, jq, jq^2, jq^3, \ldots, jq^{r-1}\} \bmod n$, où r est le nombre de conjugués distincts de β^j. Nous savons que r est le plus petit entier positif non nul tel que $(\beta^j)^{q^r} = \beta^j$ (cf. proposition 4.2.30, corollaire 4.2.23 et proposition 4.2.21), autrement dit, tel que $jq^r \equiv j \bmod n$.

Définition 4.2.33 *Pour tout entier j, $j \in Z_n$, nous définissons la* classe cyclotomique *de j modulo n sur $\mathbf{F}_q$ comme l'ensemble*

$$Cl\,(j) = \{j, jq, jq^2, jq^3, \ldots, jq^{r-1}\} \bmod n,$$

où r est le plus petit entier positif non nul tel que $jq^r \equiv j \bmod n$.

Nous pouvons donc réécrire les résultats de la section 4.2.4.3. Nous avons que

$$r = \deg M_{\beta^j}(X) = \mathrm{ord}_l(q),$$

où l est l'ordre de β^j (cf. proposition 4.2.30). À partir de la formule (4.3) nous obtenons que le polynôme minimal de $\gamma = \beta^j$, $j \in Z_n$, est

$$M_\gamma(X) = \prod_{i \in Cl(j)} (X - \beta^i). \tag{4.6}$$

Le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$ est égal au nombre de polynômes minimaux différents des racines n-ièmes de l'unité sur $\mathbf{F}_q$. La formule (4.2) nous donne

$$X^n - 1 = \prod_{j} M_{\beta^j}(X), \tag{4.7}$$

où j parcourt un ensemble de représentants des classes cyclotomiques modulo n sur $\mathbf{F}_q$. Donc le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$ est égal au nombre de diviseurs irréductibles de $X^n - 1$ sur $\mathbf{F}_q$.

Classes cyclotomiques réversibles Nous notons $Cl\,(-j)$ la classe de $(-j) \bmod n$, $j \in Z_n$.

Propriété 4.2.34 *Si $Cl\,(i)$ est de cardinal r, alors $Cl\,(-i)$ est aussi de cardinal r.*

Preuve. $Cl\,(i)$ correspond à la classe d'équivalence de β^i. Soit l l'ordre de β^i. L'ordre l' de $\beta^{-i} = (\beta^i)^{-1} = (\beta^i)^{n-1}$ est (cf. proposition 4.2.17) $l' = \frac{l}{(l,n-1)}$, l' divise donc l. De la même façon nous montrons que l divise l', donc $l = l'$. Mais (cf. proposition 4.2.30) le cardinal r de $Cl\,(i)$ est égal à $\mathrm{ord}_l(q)$, donc le cardinal de $Cl\,(-i)$ est égal à $\mathrm{ord}_{l'}(q) = \mathrm{ord}_l(q) = r$. □

Définition 4.2.35 *Soit $i \in Z_n$ et $Cl\,(i)$ une classe cyclotomique modulo n sur $\mathbf{F}_q$. Si $Cl\,(i) = Cl\,(-i)$, nous dirons que $Cl\,(i)$ est* réversible.

Propriété 4.2.36 *Soit r un entier positif non nul. S'il existe une seule classe cyclotomique modulo n sur $\mathbf{F}_q$ de cardinal r, alors elle est réversible.*

Preuve. C'est une conséquence directe de la propriété 4.2.34. □

Définition 4.2.37 *Soit $f(X) \in \mathbf{F}_q[X]$. Le* polynôme réciproque *$f^*(X)$ de $f(X)$ est le polynôme $X^{\deg f(X)} f(X^{-1})$.*

Soit m un entier positif non nul tel que $n \mid q^m - 1$.

Propriété 4.2.38 *Soit $\gamma \in \mathbf{F}_{q^m}$. Son polynôme minimal $M_\gamma(X)$ sur $\mathbf{F}_q$ est son propre réciproque si et seulement si γ^{-1} est une racine de $M_\gamma(X)$.*

Propriété 4.2.39 *Le polynôme $M_\beta(X)$ est son propre réciproque si et seulement si*

$$\textit{il existe } i > 0 \textit{ tel que } q^i \equiv -1 \bmod n. \tag{4.8}$$

Preuve. $M_\beta(X)$ est son propre réciproque si et seulement si β^{-1} est une racine de $M_\beta(X)$ (cf. propriété 4.2.38), donc si et seulement s'il existe $1 \leq k \leq \mathrm{ord}_n(q)$ tel que $\beta^{-1} = \beta^{q^k}$ (cf. proposition 4.2.24 et remarque 4.2.25), par conséquent si et seulement si

$$\text{il existe } 1 \leq k \leq \mathrm{ord}_n(q) \text{ tel que } q^k \equiv -1 \bmod n. \tag{4.9}$$

D'ailleurs, si la condition (4.9) est satisfaite, la condition (4.8) est évidemment satisfaite, et réciproquement, si la condition (4.8) est satisfaite, alors $q^{i \bmod \mathrm{ord}_n(q)} \equiv -1 \bmod n$ (car $q^{\mathrm{ord}_n(q)} \equiv 1 \bmod n$, donc $q^{i + j\mathrm{ord}_n(q)} \equiv -1 \bmod n$ pour tout $j \in \mathbb{Z}$), par conséquent la condition (4.9) est satisfaite avec $k = i \bmod \mathrm{ord}_n(q)$ si $i \bmod \mathrm{ord}_n(q) > 0$, et $k = \mathrm{ord}_n(q)$ sinon. Donc les conditions (4.8) et (4.9) sont équivalentes. □

Propriété 4.2.40 *Si $M_\beta(X)$ est son propre réciproque, alors $M_{\beta^j}(X)$ est son propre réciproque pour tout j dans $\mathbb{N}$.*

Preuve. Comme dans la démonstration de la propriété 4.2.39 nous obtenons qu'il existe $i > 0$ tel que $\beta^{-1} = \beta^{q^i}$, donc $(\beta^{-1})^j = \left(\beta^{q^i}\right)^j$ et $(\beta^j)^{-1} = (\beta^j)^{q^i}$. Par conséquent $(\beta^j)^{-1}$ est une racine de $M_{\beta^j}(X)$ et, grâce à la propriété 4.2.38, $M_{\beta^j}(X)$ est son propre réciproque. □

La formule (4.6) montre qu'à chaque classe cyclotomique correspond un polynôme minimal, plus précisément, à $Cl(j)$ modulo n sur $\mathbf{F}_q$ correspond $M_{\beta^j}(X) = \prod_{i\in Cl(j)}(X - \beta^i)$.

Propriété 4.2.41 *Soit $j \in Z_n$. La classe cyclotomique $Cl(j)$ modulo n sur $\mathbf{F}_q$ est réversible si et seulement si $M_{\beta^j}(X)$ est son propre réciproque.*

Preuve. C'est une conséquence directe de la définition 4.2.35 et de la propriété 4.2.38. □

Cas général Si nous nous plaçons dans le cadre général, nous obtenons la décomposition suivante de $X^n - 1$. Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Alors

$$X^n - 1 \overset{(4.4)}{=} (X^r - 1)^{p^s} \overset{(4.7)}{=} \left(\prod_j M_{\beta^j}(X)\right)^{p^s} = \prod_j M_{\beta^j}(X)^{p^s}, \tag{4.10}$$

où β est une racine r-ième primitive de l'unité sur $\mathbf{F}_q$ et j parcourt un ensemble de représentants des classes cyclotomiques modulo r sur $\mathbf{F}_q$.

4.3 Codes cycliques

Soit $n \in \mathbb{N}^*$ et q une puissance d'un nombre premier p.

Proposition 4.3.1 *Soit C un code cyclique de longueur n sur $\mathbf{F}_q$ de polynôme générateur $g(X)$. Soit $h(X) = (X^n - 1)/g(X)$. Le polynôme $g^{\perp}(X) = X^{\deg h}h(X^{-1})$ est le polynôme générateur du dual $C^{\perp}$ du code C.*

Notons $g_C(X)$ le polynôme générateur d'un code cyclique C.

Proposition 4.3.2 *Soient C_1, C_2 des codes cycliques de longueur n sur $\mathbf{F}_q$. Nous avons :*

$$g_{C_1\cap C_2}(X) = \mathrm{ppcm}\left(g_{C_1}(X), g_{C_2}(X)\right).$$

Preuve. Comme

$$g_{C_1\cap C_2}(X) \in C_1 \cap C_2,$$

nous avons

$$g_{C_1\cap C_2}(X) \in C_1 \quad \text{et} \quad g_{C_1\cap C_2}(X) \in C_2.$$

Donc

$$g_{C_1}(X) \mid g_{C_1\cap C_2}(X) \quad \text{et} \quad g_{C_2}(X) \mid g_{C_1\cap C_2}(X) \quad \text{dans } \mathbf{F}_q[X]$$

(cf. théorème 1.3.17). Donc

$$\mathrm{ppcm}\,(g_{C_1}(X), g_{C_2}(X)) \mid g_{C_1\cap C_2}(X).$$

De l'autre coté,

$$g_{C_1}(X) \mid \mathrm{ppcm}\,(g_{C_1}(X), g_{C_2}(X)) \quad \text{et} \quad g_{C_2}(X) \mid \mathrm{ppcm}\,(g_{C_1}(X), g_{C_2}(X)),$$

donc $\mathrm{ppcm}\,(g_{C_1}(X), g_{C_2}(X))$ appartient à C_1 et C_2, par conséquent il appartient à $C_1 \cap C_2$ et donc

$$g_{C_1\cap C_2}(X) \mid \mathrm{ppcm}\,(g_{C_1}(X), g_{C_2}(X))\,.$$

D'où le résultat. □

4.3.1 Cas particulier : n est premier avec p

Les résultats de cette section peut être retrouvés dans [MS77].

Soit $(n, p) = 1$. Soit m l'ordre de q modulo n.

Comme c'est décrit dans la section 4.2.4.3, le polynôme $X^n - 1$ se factorise entièrement sur $\mathbf{F}_{q^m}$. Plus précisément, si β est une racine n-ième primitive de l'unité sur $\mathbf{F}_q$, alors

$$X^n - 1 \overset{(4.1)}{=} \prod_{i=0}^{n-1}(X - \beta^i).$$

Définition 4.3.3 *Soit $(n, p) = 1$. Soit m l'ordre de q modulo n et β une racine n-ième primitive de l'unité sur $\mathbf{F}_q$. L'*ensemble de définition *d'un code cyclique C de longueur n sur $\mathbf{F}_q$ engendré par un polynôme g est la partie T_C de Z_n telle que $i \in T_C \Leftrightarrow g(\beta^i) = 0$, i.e.*

$$T_C = \{i \in Z_n \mid g(\beta^i) = 0\}.$$

Propriété 4.3.4 $\deg g(X) = |T_C|$ *et* $\dim C = n - |T_C|$.

Propriété 4.3.5

$$C = \{f(X) \bmod (X^n - 1) \mid f(\beta^i) = 0\ \forall i \in T_C\}.$$

Propriété 4.3.6 *T_C est stable par multiplication par q modulo n, c'est-à-dire si $i \in T_C$, alors $iq \bmod n \in T_C$.*

Corollaire 4.3.7 *Si $i \in T_C$, alors $Cl\,(i) \subset T_C$, où $Cl\,(i)$ est la classe cyclotomique de i modulo n sur $\mathbf{F}_q$.*

Corollaire 4.3.8 *T_C est une réunion de classes cyclotomiques modulo n sur $\mathbf{F}_q$, c'est-à-dire*

$$T_C = \bigcup_i Cl\,(i)\,,$$

où i parcourt une partie d'un ensemble de représentants des classes cyclotomiques modulo n sur $\mathbf{F}_q$.

Proposition 4.3.9 *Si un code cyclique C a pour ensemble de définition T_C, son dual $C^\perp$ a pour ensemble de définition*

$$T_{C^\perp} = \{(n-s) \bmod n \mid s \in Z_n \setminus T_C\}.$$

Preuve. Notons $h(X) = (X^n - 1)/g(X)$. Comme β^s, $s \in Z_n$, sont les racines de $X^n - 1$, β^s, $s \in Z_n \setminus T_C$, sont les racines de $h(X)$. Donc β^{-s}, $s \in Z_n \setminus T_C$, sont les racines du polynôme générateur $g^\perp(X) = X^{\deg h} h(X^{-1})$ de $C^\perp$ (cf. proposition 4.3.1), d'où le résultat. □

Corollaire 4.3.10 *Soit $Cl\,(i_1), \ldots, Cl\,(i_j)$ toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$. Si*

$$T_C = \bigcup_{i \in I} Cl\,(i)\,,$$

où $I \subset \{i_1, \ldots, i_j\}$, alors

$$T_{C^\perp} = \bigcup_{i \in \{i_1, \ldots, i_j\} \setminus I} Cl\,(-i)\,.$$

Preuve. C'est une conséquence directe de la proposition 4.3.9 et du corollaire 4.3.8. □

Notons T_C l'ensemble de définition d'un code cyclique C.

Lemme 4.3.11 *Soient C_1, C_2 des codes cycliques de longueur n sur $\mathbf{F}_q$. Nous avons :*

$$T_{C_1 \cap C_2} = T_{C_1} \cup T_{C_2}.$$

Preuve. Nous avons

$$\begin{aligned} T_{C_1} \cup T_{C_2} &= \{i \in Z_n \mid g_{C_1}(\beta^i) = 0 \text{ ou } g_{C_2}(\beta^i) = 0\} \\ &= \{i \in Z_n \mid \mathrm{ppcm}\,(g_{C_1}, g_{C_2})\,(\beta^i) = 0\} \\ &\overset{Prop.4.3.2}{=} \{i \in Z_n \mid g_{C_1 \cap C_2}(\beta^i) = 0\} \\ &= T_{C_1 \cap C_2}. \end{aligned}$$

□

4.3.2 Cas général

Soit $n = rp^s$, où r est premier avec p et $s \geq 0$.

Soit C un code cyclique de longueur n sur $\mathbf{F}_q$. Son polynôme générateur est un diviseur de $X^n - 1 = \prod_{i=0}^{r-1} (X - \beta^i)^{p^s}$ (cf. théorème 1.3.17 et formule (4.5), p. 127), où β est une racine r-ième primitive de l'unité sur $\mathbf{F}_q$. Il est possible de le caractériser par ses racines et leurs multiplicités. Pour cela, nous introduisons la généralisation de l'ensemble de définition des codes cycliques.

Définition 4.3.12 *Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Soit β une racine r-ième primitive de l'unité sur $\mathbf{F}_q$. L'*ensemble de définition *d'un code cyclique C de longueur n sur $\mathbf{F}_q$ engendré par un polynôme g est le sous-ensemble (au sens de multi-ensembles, cf. définition 4.2.4(ii), p. 121) T_C de $Z_r^{p^s}$ (cf. notation 4.2.6, p. 122) suivant :*

$$T_C = \{i\hat{\ }t \mid i \in Z_r,\ \beta^i \text{ est une racine de } g(X) \text{ de multiplicité } t\}.$$

Nous obtenons des résultats généralisant ceux de la section 4.3.1.

Propriété 4.3.13 $\deg g(X) = |T_C|$ *et* $\dim C = n - |T_C|$.

Propriété 4.3.14

$$\begin{aligned} C &= \{f(X) \bmod (X^n - 1) \mid \forall i\hat{\ }t \in T_C, \\ & \quad \beta^i \textit{ est une racine de } f(X) \textit{ de multiplicité au moins } t\}. \end{aligned}$$

Propriété 4.3.15 *Si* $i\hat{\ }t \in T_C$, *alors* $(iq \bmod n)\hat{\ }t \in T_C$.

Corollaire 4.3.16 *Si* $i\hat{\ }t \in T_C$, *alors* $Cl\,(i)\hat{\ }t \subset T_C$, *où* $Cl\,(i)$ *est la classe cyclotomique de* i *modulo* r *sur* $\mathbf{F}_q$.

Rappelons que quand elle est utilisée conjointement avec la notation $\hat{\ }$, la notation $\subset$ sous-entend le sens de la notation 4.2.3, p. 120, et pas celui de la définition 4.2.4(ii), p. 121.

Corollaire 4.3.17

$$T_C = \bigcup_i Cl\,(i)\hat{\ }t_i,$$

où i *parcourt une partie d'un ensemble de représentants des classes cyclotomiques modulo* r *sur* $\mathbf{F}_q$, *et* $1 \leq t_i \leq p^s$ *(ou, ce qui signifie la même chose,* i *parcourt un ensemble de représentants des classes cyclotomiques modulo* r *sur* $\mathbf{F}_q$, *et* $0 \leq t_i \leq p^s$*).*

Proposition 4.3.18

$$T_{C^\perp} = \{((r-i) \bmod r)\hat{\ }t \mid i\hat{\ }t \in Z_r^{p^s} \setminus T_C\}.$$

Preuve. Analogue à la démonstration de la proposition 4.3.9. □

Corollaire 4.3.19 *Soit* $Cl\,(i_1)\,, \ldots, Cl\,(i_j)$ *toutes les classes cyclotomiques modulo* r *sur* $\mathbf{F}_q$. *Si*

$$T_C = \bigcup_{i \in I} Cl\,(i)\hat{\ }t_i,$$

où $I = \{i_1, \ldots, i_j\}$, $0 \leq t_i \leq p^s$, *alors*

$$T_{C^\perp} = \bigcup_{i \in I} Cl\,(-i)\hat{\ }(p^s - t_i).$$

Preuve. C'est une conséquence directe de la proposition 4.3.18 et du corollaire 4.3.17. □

Lemme 4.3.20 *Soient* C_1 *et* C_2 *des codes cycliques de longueur* n *sur* $\mathbf{F}_q$. *Nous avons :*

$$T_{C_1 \cap C_2} = T_{C_1} + T_{C_2}$$

(cf. notation 4.2.5, p. 121).

Preuve. Analogue à la démonstration du lemme 4.3.11. □

4.3.3 Les codes BCH

Dans le chapitre 3 nous avons mentionné les codes BCH et les codes résidus quadratiques. Pour que ce document soit complet, nous donnons ici les définitions de ces codes.

Définition 4.3.21 *Soit $(n, q) = 1$. Soit C un code cyclique le longueur n sur $\mathbf{F}_q$ et soit $g(X)$ son polynôme générateur. Le code C est un* code BCH de distance construite δ *si δ est le plus grand entier vérifiant*

$$\exists\, b \in \mathbb{Z} \text{ tel que } g(\alpha^b) = g(\alpha^{b+1}) = \cdots = g(\alpha^{b+\delta-2}) = 0,$$

où α est une racine primitive n-ième de l'unité sur $\mathbf{F}_q$.

Définition 4.3.22 *Un code BCH est dit* primitif *s'il existe un entier positif non nul m tel que $n = q^m - 1$, il est dit BCH* au sens strict *si $b = 1$.*

4.3.4 Les codes résidus quadratiques

Les codes résidus quadratiques sont des codes cycliques de longueur p première définis sur un corps fini $\mathbf{F}_l$, où l est un autre nombre premier qui est un résidu quadratique modulo p (i.e. l est un carré modulo p).

Lorsque $l = 2$, cette condition est équivalente à $p \equiv \pm 1 \bmod 8$.

Soient Q et N respectivement l'ensemble des carrés non nuls et des non carrés dans $\mathbf{F}_p$. Puisque l est un carré modulo p, l appartient à Q et les ensembles Q et N sont stables par multiplications par l modulo p.

Soit m l'ordre de l modulo p : $l^m \equiv 1 \bmod p$. Soit α une racine primitive p-ième de l'unité dans $\mathbf{F}_{l^m}$.

Les polynômes

$$q(X) = \textstyle\prod_{r\in Q}(X - \alpha^r) \qquad \text{et} \qquad n(X) = \prod_{r\in N}(X - \alpha^r)$$

sont des polynômes à coefficients dans $\mathbf{F}_l$. De plus

$$X^p - 1 = (X - 1)q(X)n(X).$$

Définition 4.3.23 *Les codes résidus quadratiques sont les codes cycliques $\mathcal{Q}$, $\overline{\mathcal{Q}}$, $\mathcal{N}$ et $\overline{\mathcal{N}}$ de polynômes générateurs respectifs $q(X)$, $(X - 1)q(X)$, $n(X)$ et $(X - 1)n(X)$. Leurs ensembles de définitions sont donc respectivement Q, $Q \cup \{0\}$, N et $N \cup \{0\}$.*

Remarque 4.3.24 Nous avons $\overline{\mathcal{Q}} \subset \mathcal{Q}$ et $\overline{\mathcal{N}} \subset \mathcal{N}$. De plus,

$$\dim \mathcal{Q} = \dim \mathcal{N} = (p + 1)/2 \qquad \text{et} \qquad \dim \overline{\mathcal{Q}} = \dim \overline{\mathcal{N}} = (p - 1)/2.$$

4.4 Hull des codes cycliques

Soit $n \in \mathbb{N}^*$ et q une puissance d'un nombre premier p.

Soit C un code cyclique de longueur n sur $\mathbf{F}_q$. Soit T_C son ensemble de définition.

Dans cette section nous allons exprimer l'ensemble de définition $T_{\mathcal{H}(C)}$ du hull $\mathcal{H}(C)$ de C en fonction de T_C.

Proposition 4.4.1 *Soit C un code cyclique. Nous avons :*

$$g_{\mathcal{H}(C)}(X) = \mathrm{ppcm}\left(g_C(X), g_{C^\perp}(X)\right).$$

Preuve. C'est une conséquence directe de la proposition 4.3.2 et de la définition du hull (cf. définition 1.3.10). □

4.4.1 Cas particulier : n est premier avec p

Proposition 4.4.2 *Soit C un code cyclique. Nous avons :*

$$T_{\mathcal{H}(C)} = T_C \cup T_{C^\perp}.$$

Preuve. C'est une conséquence directe du lemme 4.3.11 et de la définition du hull (cf. définition 1.3.10). □

Considérons les classes cyclotomiques modulo n sur $\mathbf{F}_q$. Nous obtenons les résultats suivants.

Lemme 4.4.3 *Soit $i \in Z_n$. Si $Cl(i)$ est réversible, alors $Cl(i) \subset T_{\mathcal{H}(C)}$ pour tout code cyclique de longueur n sur $\mathbf{F}_q$.*

Preuve.

- Si $Cl(i) \subset T_C$, alors $Cl(i) \subset T_{\mathcal{H}(C)} = T_C \cup T_{C^\perp}$;
- sinon $Cl(i) = Cl(-i) \subset T_{C^\perp}$ en vertu du corollaire 4.3.10, et donc $Cl(i) \subset T_{\mathcal{H}(C)}$. □

Corollaire 4.4.4 *Si toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$ sont réversibles, alors $T_{\mathcal{H}(C)} = Z_n$ et donc $\dim \mathcal{H}(C) = 0$ pour tout code cyclique C de longueur n sur $\mathbf{F}_q$.*

Preuve. C'est une conséquence directe du lemme 4.4.3 et de la propriété 4.3.4. □

Soit C un code cyclique de longueur n sur $\mathbf{F}_q$. Soit T_C son ensemble de définition.

Lemme 4.4.5 *Soit $i \in Z_n$. Si $Cl(i)$ n'est pas réversible, $Cl(i) \not\subset T_C$ et $Cl(-i) \not\subset T_C$, alors $Cl(i) \subset T_{\mathcal{H}(C)}$ et $Cl(-i) \subset T_{\mathcal{H}(C)}$.*

Preuve. Comme $Cl\,(i) \not\subset T_C$, nous avons $Cl\,(-i) \subset T_{C^\perp}$ en vertu du corollaire 4.3.10, et donc $Cl\,(-i) \subset T_{\mathcal{H}(C)}$.

De la même façon nous montrons que, comme $Cl\,(-i) \not\subset T_C$, $Cl\,(i) \subset T_{C^\perp} \subset T_{\mathcal{H}(C)}$. □

Lemme 4.4.6 *Soit $i \in Z_n$. Si $Cl\,(i)$ n'est pas réversible, $Cl\,(i) \subset T_C$ et $Cl\,(-i) \not\subset T_C$, alors $Cl\,(i) \subset T_{\mathcal{H}(C)}$ et $Cl\,(-i) \not\subset T_{\mathcal{H}(C)}$.*

Preuve. On a $Cl\,(i) \subset T_C$, donc $Cl\,(-i) \not\subset T_{C^\perp}$ en vertu du corollaire 4.3.10, et, comme $Cl\,(-i) \not\subset T_C$, $Cl\,(-i) \not\subset T_C \cup T_{C^\perp} = T_{\mathcal{H}(C)}$.

De plus, $Cl\,(i) \subset T_C \subset T_{\mathcal{H}(C)}$. □

Lemme 4.4.7 *Soit $i \in Z_n$. Si $Cl\,(i)$ n'est pas réversible, $Cl\,(i) \subset T_C$ et $Cl\,(-i) \subset T_C$, alors $Cl\,(i) \subset T_{\mathcal{H}(C)}$ et $Cl\,(-i) \subset T_{\mathcal{H}(C)}$.*

Preuve. Évidente, car $T_C \subset T_{\mathcal{H}(C)}$. □

Proposition 4.4.8 *Soit $i \in Z_n$. Alors $Cl\,(i) \not\subset T_{\mathcal{H}(C)}$ si et seulement si $Cl\,(i) \not\subset T_C$ et $Cl\,(-i) \subset T_C$. En particulier, si $Cl\,(i)$ est réversible, alors $Cl\,(i) \subset T_{\mathcal{H}(C)}$.*

Preuve. C'est une conséquence des lemmes 4.4.3 et 4.4.5–4.4.7. □

Corollaire 4.4.9 *Soit $i \in Z_n$. Au moins une de $Cl\,(i)$ et $Cl\,(-i)$ est contenue dans $T_{\mathcal{H}(C)}$.*

Preuve. Si $Cl\,(i) \not\subset T_{\mathcal{H}(C)}$, alors $Cl\,(-i) \subset T_C$ (cf. proposition 4.4.8) et donc $Cl\,(-i) \subset T_{\mathcal{H}(C)}$ (cf. proposition 4.4.2). □

4.4.2 Cas général

Soit $n = rp^s$, où r est premier avec p et $s \geq 0$.

Proposition 4.4.10 *Soit C un code cyclique. Nous avons :*

$$T_{\mathcal{H}(C)} = T_C + T_{C^\perp}.$$

Preuve. C'est une conséquence directe du lemme 4.3.20 et de la définition du hull. □

Soit C un code cyclique de longueur n sur $\mathbf{F}_q$. Considérons les classes cyclotomiques modulo r sur $\mathbf{F}_q$. Nous obtenons le résultat suivant, qui généralise ceux de la section 4.4.1.

Proposition 4.4.11 *Soit $i \in Z_r$, $0 \leq t, u \leq p^s$. Si $Cl\,(i)\hat{}t \subset T_C$, $Cl\,(-i)\hat{}u \subset T_C$, alors $Cl\,(i)\hat{}\max(t, p^s - u) \subset T_{\mathcal{H}(C)}$ et $Cl\,(-i)\hat{}\max(u, p^s - t) \subset T_{\mathcal{H}(C)}$.*

Preuve. $Cl\,(i)\hat{}t \subset T_C$ signifie, compte tenu du corollaire 4.3.19, que $Cl\,(-i)\hat{}(p^s - t) \subset T_{C^\perp}$, donc

$$Cl\,(-i)\hat{}\max(u, p^s - t) \subset T_C + T_{C^\perp} = T_{\mathcal{H}(C)}$$

(par construction de $T_C + T_{C^\perp}$ et en vertu de la proposition 4.4.10).

De la même façon nous obtenons que $Cl\,(i)\hat{}(p^s - u) \subset T_{C^\perp}$ et donc

$$Cl\,(i)\hat{}\max(t, p^s - u) \subset T_C + T_{C^\perp} = T_{\mathcal{H}(C)}. \qquad \square$$

Rappelons que quand elle est utilisée conjointement avec la notation $\hat{}$, la notation $\subset$ sous-entend le sens de la notation 4.2.3, p. 120, et pas celui de la définition 4.2.4(ii), p. 121.

Corollaire 4.4.12 *Soit $i \in Z_r$, $0 \le t \le p^s$. Si $Cl\,(i)$ est réversible et $Cl\,(i)\hat{}t \subset T_C$, alors $Cl\,(i)\hat{}\max(t, p^s - t) \subset T_{\mathcal{H}(C)}$.*

Preuve. C'est une conséquence directe de la proposition 4.4.11 et de la définition 4.2.35. $\square$

4.5 Le nombre de codes cycliques

Soit $n \in \mathbb{N}^*$ et q une puissance d'un nombre premier p. Soit $n = rp^s$, où r est premier avec p et $s \ge 0$.

Le théorème 1.3.17(iv) montre que le polynôme générateur d'un code cyclique de longueur n sur $\mathbf{F}_q$ divise $X^n - 1$ (dans $\mathbf{F}_q[X]$). Inversement, à chaque diviseur de $X^n - 1$ correspond un code cyclique de longueur n sur $\mathbf{F}_q$. Donc le nombre de codes cycliques distincts de longueur n sur $\mathbf{F}_q$ est égal au nombre des diviseurs de $X^n - 1$ dans $\mathbf{F}_q[X]$. La formule (4.10), p. 130, montre que $X^n - 1$ se factorise sur $\mathbf{F}_q$ en facteurs irréductibles comme suit :

$$X^n - 1 = \prod_{i=j_1}^{j_l} M_{\beta^i}(X)^{p^s},$$

où β est une racine r-ième primitive de l'unité sur $\mathbf{F}_q$, $M_{\beta^i}(X)$ est le polynôme minimal de β^i sur $\mathbf{F}_q$, l est le nombre de classes cyclotomiques modulo r sur $\mathbf{F}_q$, et $j_1, \ldots, j_l$ sont des représentants de ces classes cyclotomiques. D'ailleurs, chaque polynôme minimal correspond à une classe cyclotomique comme dans la formule (4.6), p. 128 :

$$M_{\beta^i}(X) = \prod_{s \in Cl(i)} (X - \beta^s).$$

Nous pouvons exprimer le même problème en termes d'ensembles de définition. L'ensemble de définition T_C d'un code cyclique C de longueur n sur $\mathbf{F}_q$ est une réunion (au sens des multi-ensembles, cf. définition 4.2.4(iii), p. 121) de classes cyclotomiques $Cl\,(i)$ modulo r sur $\mathbf{F}_q$ avec certaines multiplicités t_i :

$$T_C = \bigcup_i Cl\,(i)\hat{}t_i,$$

où i parcourt un ensemble de représentants des classes cyclotomiques modulo r sur $\mathbf{F}_q$ et $0 \leq t_i \leq p^s$ (cf. corollaire 4.3.17, p. 133). Inversement, à chaque tel multi-ensemble correspond un code cyclique de longueur n sur $\mathbf{F}_q$. Donc le nombre de codes cycliques de longueur n sur $\mathbf{F}_q$ est égal au nombre de sous-ensembles de $Z_r^{p^s}$ composés de classes cyclotomiques modulo r sur $\mathbf{F}_q$.

Notation 4.5.1 Soit $(n, p) = 1$. Notons $c_q(n)$ le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$.

Proposition 4.5.2 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Le nombre de codes cycliques de longueur n sur $\mathbf{F}_q$ est égal à $(p^s + 1)^{c_q(r)}$.*

Preuve. Le nombre de codes cycliques de longueur n sur $\mathbf{F}_q$ est égal au nombre de diviseurs de $X^n - 1$ et au nombre de sous-ensembles de $Z_r^{p^s}$ composés de classes cyclotomiques modulo r sur $\mathbf{F}_q$, comme c'est décrit ci-dessus. Il suffit donc de compter soit les uns, soit les autres. Comptons, par exemple, les sous-ensembles de $Z_r^{p^s}$ composés de classes cyclotomiques modulo r sur $\mathbf{F}_q$. Chaque classe cyclotomique peut être présente dans un sous-ensemble avec la multiplicité $0, 1, 2, \ldots, p^s$, donc il y a $p^s + 1$ possibilités pour chaque classe cyclotomique, $(p^s + 1)^2$ pour une paire de classes cyclotomiques, etc..., $(p^s + 1)^{c_q(r)}$ pour $c_q(r)$ classes cyclotomiques. □

Corollaire 4.5.3 *Si n est premier avec q, le nombre de codes cycliques de longueur n sur $\mathbf{F}_q$ est égal à $2^{c_q(n)}$.*

Preuve. Dans ce cas $p^s = 1$ et $r = n$. □

4.5.1 Le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit n premier avec q.

Étudions un peu le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$, autrement dit, le nombre de polynômes minimaux distincts des racines n-ièmes de l'unité sur $\mathbf{F}_q$ (cf. section 4.2.4.4).

Soit β une racine n-ième primitive de l'unité sur $\mathbf{F}_q$. Dans la section 4.2.4.3 nous avons montré que l'ensemble des racines n-ièmes de l'unité sur $\mathbf{F}_q$ forme un groupe multiplicatif cyclique $< \beta >$ d'ordre n engendré par β.

Nous savons (cf. formule (4.1), p. 127) que

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \beta^i).$$

D'après le corollaire 4.2.19, p. 124, nous pouvons partitionner $< \beta >$ en classes selon l'ordre des éléments β^i, $0 \leq i \leq n - 1$. Pour chaque diviseur l de n, il existe une classe de cardinal $\phi(l)$. Notons $\psi_l(X)$ le produit des $X-\gamma$, où γ parcourt la classe correspondant à l (i.e. γ parcourt tous les β^i, $0 \le i \le n-1$, d'ordre l). Alors

$$X^n - 1 = \prod_{l|n} \psi_l(X),$$

et $\deg \psi_l(X) = \phi(l)$.

Soit γ une racine n-ième de l'unité sur $\mathbf{F}_q$ d'ordre l, soit $M_\gamma(X)$ son polynôme minimal. La proposition 4.2.26, p. 125, montre que toutes les racines de $M_\gamma(X)$ sont de même ordre l, donc $M_\gamma(X) \overset{(4.3)}{=} \prod_{i=0}^{\mathrm{ord}_l(q)-1}(X-\gamma^{q^i})$ divise $\psi_l(X)$. D'ailleurs, le degré $\mathrm{ord}_l(q)$ de $M_\gamma(X)$ ne dépend pas de γ, il ne dépend que de l'ordre l de γ. Donc tout ψ_l, $l \mid n$, est un produit de polynômes minimaux de même degré $\mathrm{ord}_l(q)$ des racines n-ièmes de l'unité sur $\mathbf{F}_q$ d'ordre l. Cela signifie que le nombre de polynômes minimaux différents des racines n-ièmes de l'unité sur $\mathbf{F}_q$ est

$$\begin{aligned} c_q(n) &= \sum_{l|n} (\text{nombre de polynômes minimaux divisant } \psi_l(X)) \\ &= \sum_{l|n} \frac{\deg \psi_l(X)}{\deg M_\gamma(X)} \\ &= \sum_{l|n} \frac{\phi(l)}{\mathrm{ord}_l(q)}, \end{aligned}$$

où γ est une racine n-ième de l'unité d'ordre l.

Nous avons donc montré la proposition suivante :

Proposition 4.5.4 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n,q)=1$. Le nombre de classes cyclotomiques modulo n sur $\mathbf{F}_q$ (autrement dit, le nombre de diviseurs irréductibles de X^n-1 sur $\mathbf{F}_q$) est*

$$c_q(n) = \sum_{l|n} \frac{\phi(l)}{\mathrm{ord}_l(q)}$$

Il est difficile de manipuler une telle formule, et nous n'avons réussi à l'expliciter que dans un cas (très) particulier.

Proposition 4.5.5 *Soit t un nombre premier impair. Si q est une racine primitive modulo t^2, alors*

$$c_q(t^i) = 1 + i \text{ pour tout } i \ge 1.$$

Preuve. Compte tenu de la proposition 4.5.4,

$$c_q(t^i) = \sum_{l|t^i} \frac{\phi(l)}{\mathrm{ord}_l(q)} = \sum_{j=0}^{i} \frac{\phi(t^j)}{\mathrm{ord}_{t^j}(q)}.$$

Le théorème 4.2.15, p. 123, montre que si q est une racine primitive modulo t^2, alors c'est une racine primitive modulo t^j pour tout $j \geq 1$, donc $\mathrm{ord}_{t^j}(q) = \phi(t^j)$ pour tout $j \geq 1$. Par conséquent

$$c_q(t^i) = \sum_{j=0}^{i} \frac{\phi(t^j)}{\mathrm{ord}_{t^j}(q)} = \sum_{j=0}^{i} 1 = i + 1. \qquad \square$$

4.6 La dimension moyenne du hull

Dans la suite du chapitre nous allons étudier la dimension moyenne du hull des codes cycliques.

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p.

4.6.1 Définitions

Notons $C(n, q)$ l'ensemble des codes cycliques de longueur n sur $\mathbf{F}_q$.

Notation 4.6.1 Nous notons $Edimh_q(n)$ la dimension moyenne du hull des codes cycliques de longueur n sur $\mathbf{F}_q$, plus précisément,

$$Edimh_q(n) = \frac{\sum_{C \in C(n,q)} \dim \mathcal{H}(C)}{|C(n,q)|}.$$

Remarque 4.6.2 Comme il existe une correspondance entre l'ensemble des codes cycliques de longueur n sur $\mathbf{F}_q$ et l'ensemble des diviseurs de $X^n - 1$ sur $\mathbf{F}_q$, nous avons

$$Edimh_q(n) = \frac{\sum_{g | X^n - 1} \dim \mathcal{H}(< g >)}{\text{nombre de diviseurs de } X^n - 1 \text{ sur } \mathbf{F}_q},$$

où $< g >$ est le code cyclique de longueur n sur $\mathbf{F}_q$ de polynôme générateur g.

Remarque 4.6.3 Il est aussi possible de définir $Edimh_q(n)$ en termes de la théorie des probabilités. Soit Y la variable aléatoire qui prend pour valeur $\dim \mathcal{H}(C)$ pour un code cyclique C de longueur n sur $\mathbf{F}_q$ choisi de façon aléatoire. Supposons que nous choisissions un code cyclique d'une manière uniforme, c'est-à-dire la probabilité d'être choisi est $1/|C(n,q)|$ pour chaque code. Nous définissons $Edimh_q(n)$ comme la valeur moyenne $E(Y)$ de Y. Cette définition est équivalente à celle de la notation 4.6.1. En effet, soit $\{y_1, y_2, \ldots, y_l\}$ l'ensemble des valeurs possibles de Y, et la probabilité que Y devienne y_i est p_i, $1 \leq i \leq l$. Nous avons

$$p_i = \frac{|\{C \in C(n,q) \mid \dim \mathcal{H}(C) = y_i\}|}{|C(n,q)|}.$$

Alors la valeur moyenne

$$
\begin{aligned}
E(Y) &= \sum_{i=1}^{l} p_i y_i \\
&= \frac{\sum_{i=1}^{l} \left|\{C \in C(n,q) \mid \dim \mathcal{H}(C) = y_i\}\right| y_i}{|C(n,q)|} \\
&= \frac{\sum_{i=1}^{l} \sum_{C \in \{C \in C(n,q) \mid \dim \mathcal{H}(C) = y_i\}} \dim \mathcal{H}(C)}{|C(n,q)|} \\
&= \frac{\sum_{C \in C(n,q)} \dim \mathcal{H}(C)}{|C(n,q)|}.
\end{aligned}
$$

□

Au début de notre travail nous avons calculé $Edimh_2(n)$ pour des petites valeurs impaires de n (cf. figure 4.1). Son comportement nous a paru assez étonnant : soit elle est nulle, soit elle est entre $\frac{2n}{15}$ et $\frac{n}{4}$. Dans ce travail nous jetons un peu de lumière sur ce sujet.

D'abord nous étudions les zéros de $Edimh_q(n)$ (section 4.6.2) et exprimons $Edimh_q(n)$ en fonction des cardinaux de classes cyclotomiques modulo r sur $\mathbf{F}_q$ (sections 4.6.3 et 4.6.4), où $n = rp^s$, r est premier avec p et $s \geq 0$.

4.6.2 Zéros de la dimension moyenne

Dans cette section nous étudions les entiers positifs non nuls n tels que $Edimh_q(n) = 0$. Montrons d'abord le résultat suivant.

Lemme 4.6.4 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Si n et q ne sont pas premiers entre eux, alors $Edimh_q(n)$ est non nul.*

Preuve. Soit $n = rp^s$, $(r,p) = 1$ et $s > 0$. Alors $p^s > 1$. Il est donc possible de trouver t et u (éventuellement égaux) tels que $0 < t, u < p^s$. Comme alors $0 < p^s - u < p^s$, nous obtenons que $0 < \max(t, p^s - u) < p^s$.

Soit $i \in Z_r$ et soient $Cl(i)$ et $Cl(-i)$ les classes cyclotomiques de i et de $-i$, respectivement, modulo r sur $\mathbf{F}_q$. Soit C un code cyclique de longueur n sur $\mathbf{F}_q$ tel que $Cl(i)\hat{}t \subset T_C$ et $Cl(-i)\hat{}u \subset T_C$. D'après la proposition 4.4.11, p. 136, $Cl(i)\hat{}\max(t, p^s - u) \subset T_{\mathcal{H}(C)}$. Mais $\max(t, p^s - u) < p^s$, donc $T_{\mathcal{H}(C)} \subsetneq Z_r^{p^s}$ et $\left|T_{\mathcal{H}(C)}\right| < \left|Z_r^{p^s}\right| = rp^s = n$, d'où $\dim \mathcal{H}(C) > 0$ (cf. propriété 4.3.13, p. 133). Donc la dimension moyenne du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ est aussi strictement positive. □

Théorème 4.6.5 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Alors $Edimh_q(n) = 0$ si et seulement s'il existe $i > 0$ tel que $q^i \equiv -1 \bmod n$.*

Preuve.

– Soit $Edimh_q(n) = 0$. D'après le lemme 4.6.4, n et q sont premiers entre eux.

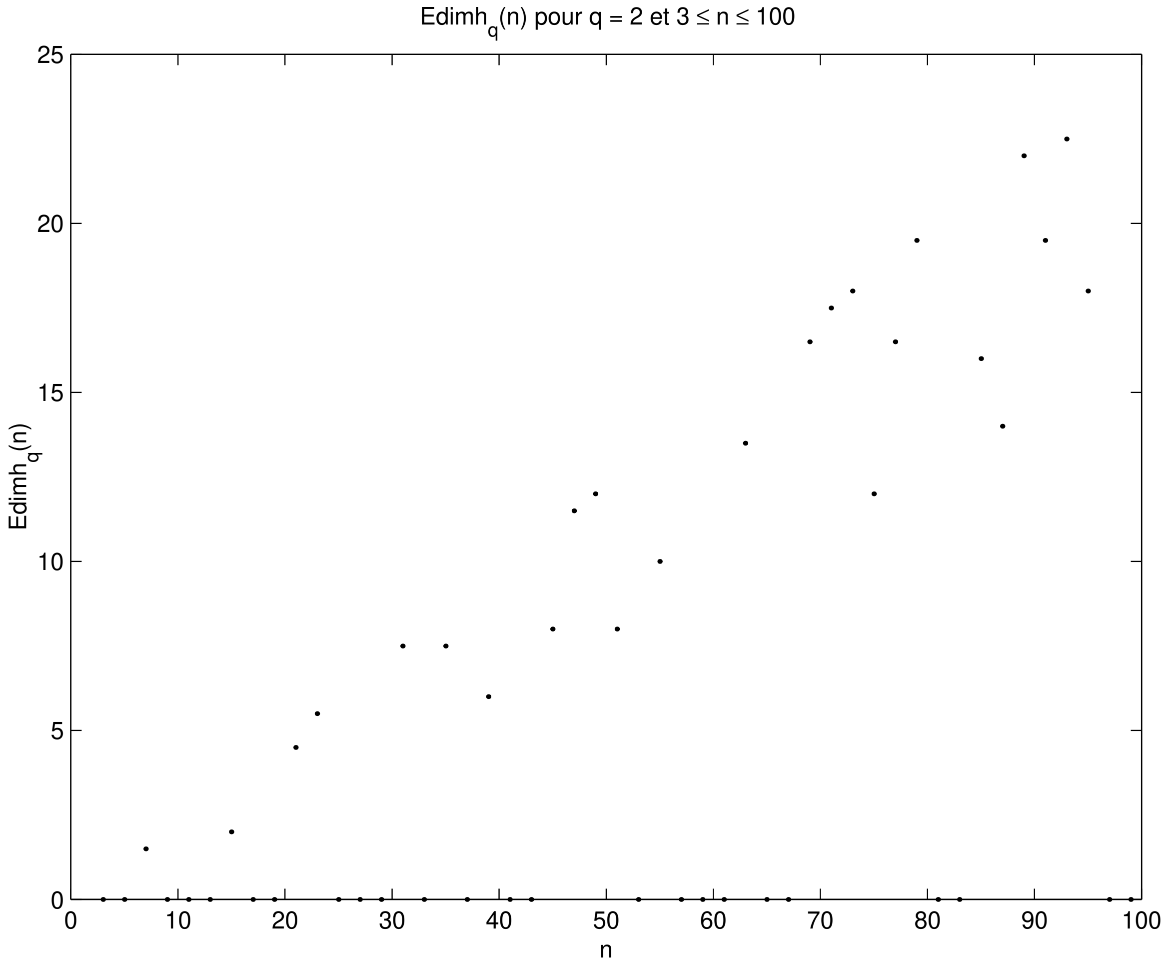

FIG. 4.1 – *Le comportement de* $Edimh_2(n)$ *pour des petites valeurs impaires de* n

Montrons que toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$ sont réversibles.

Soit $i \in Z_n$ tel que la classe cyclotomique $Cl\,(i)$ modulo n sur $\mathbf{F}_q$ n'est pas réversible. Alors il existe un code cyclique C de longueur n sur $\mathbf{F}_q$ tel que $Cl\,(i) \not\subset T_C$ et $Cl\,(-i) \subset T_C$, donc $Cl\,(i) \not\subset T_{\mathcal{H}(C)}$ (cf. proposition 4.4.8, p. 136) et $T_{\mathcal{H}(C)} \subsetneq Z_n$, d'où $\dim \mathcal{H}\,(C) > 0$ (cf. propriété 4.3.4, p. 131) et par conséquent $Edimh_q(n) > 0$. Cela contredit notre hypothèse, donc toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$ sont réversibles, $Cl\,(1)$ incluse. Les propriétés 4.2.41 et 4.2.39, p. 129, montrent qu'il existe $i \geq 0$ tel que $q^i \equiv -1 \bmod n$.

– Supposons qu'il existe $i > 0$ tel que $q^i \equiv -1 \bmod n$. Alors n est premier avec q. En effet, soit $(n, q) = d$. La congruence $q^i \equiv -1 \bmod n$ est équivalente à la condition $n \mid q^i + 1$. Nous avons $d \mid n$, donc $d \mid q^i + 1$. Mais $d \mid q$, donc $d \mid q^i$, d'où

$$d \mid (q^i + 1, q^i) = 1 \qquad \text{et} \qquad d = 1.$$

Soit β une racine n-ième primitive de l'unité sur $\mathbf{F}_q$. La propriété 4.2.39 montre que le polynôme minimal $M_\beta(X)$ de β est son propre réciproque, ce qui, compte tenu des propriétés 4.2.40 et 4.2.41, signifie que toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$ sont réversibles. Le corollaire 4.4.4, p. 135, complète alors la démonstration. □

4.6.3 Cas particulier : n est premier avec p

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Nous considérons d'abord le cas plus simple où n et q sont premiers entre eux. Nous obtenons le résultat suivant.

Proposition 4.6.6 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n, q) = 1$. Alors*

$$Edimh_q(n) = \frac{1}{4}\left(n - B_{n,q}\right),$$

où

$$B_{n,q} = \sum_i |Cl\,(i)| \tag{4.11}$$

et i parcourt un ensemble de représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$.

Preuve. Nous allons utiliser la définition de $Edimh_q(n)$ en termes de la théorie des probabilités (cf. remarque 4.6.3, p. 140).

Supposons que nous choisissions un code cyclique C de longueur n sur $\mathbf{F}_q$ avec la probabilité uniforme $1/\left|C(n,q)\right|$. Choisir un code cyclique C revient à choisir son ensemble de définition T_C (cf. définition 4.3.3, p. 131). L'ensemble T_C est une réunion de classes cyclotomiques modulo n sur $\mathbf{F}_q$ (cf. corollaire 4.3.8). Soit J un ensemble de représentants des classes cyclotomiques modulo n sur $\mathbf{F}_q$. Choisir un ensemble de définition T_C revient à choisir un sous-ensemble I de J, car chaque sous-ensemble I de J définit un unique code cyclique C de longueur n sur $\mathbf{F}_q$ d'ensemble de définition $T_C = \bigcup_{i\in I} Cl\,(i)$, et vice versa. Comment choisir un sous-ensemble I de J de façon uniforme? En mettant dans I chaque $i \in J$ avec la probabilité $1/2$, et indépendemment des autres $j \in J$, $j \neq i$. En effet, la probabilité d'obtenir un code cyclique C donné est alors $\frac{1}{2}\cdot\frac{1}{2}\cdots\frac{1}{2}$ ($|J|$ fois), car avec la probabilité $1/2$ nous incluons une classe cyclotomique dans l'ensemble de définition si c'est nécessaire pour obtenir T_C donné, et avec la même probabilité $1/2$ nous ne mettons pas de classe cyclotomique dans l'ensemble de définition si elle ne se trouve pas dans T_C. Donc cette probabilité est égale à $\frac{1}{2^{|J|}} = \frac{1}{2^{c_q(n)}}$, ce qui est égal, compte tenu du corollaire 4.5.3, p. 138, à $\frac{1}{|C(n,q)|}$.

Nous définissons les variables aléatoires supplémentaires suivantes : α_i, $i \in J$, prend pour valeur 0 si $Cl\,(i) \subset T_C$, et 1 sinon, avec les probabilités $1/2$; β_i, $i \in J$, prend pour valeur 0 si $Cl\,(i) \subset T_{\mathcal{H}(C)}$, et 1 sinon.

Compte tenu de la propriété 4.3.4 et de la définition 4.3.3, p. 131, nous obtenons que la variable aléatoire Y définie dans la remarque 4.6.3, p. 140, est égale à

$$Y = \dim \mathcal{H}\,(C) = n - \left|T_{\mathcal{H}(C)}\right| = \left|Z_n \setminus T_{\mathcal{H}(C)}\right|.$$

Comme $T_{\mathcal{H}(C)}$ est une réunion de classes cyclotomiques, $Z_n \setminus T_{\mathcal{H}(C)}$ l'est aussi, et, par définition de β_i, $Cl\,(i) \subset Z_n \setminus T_{\mathcal{H}(C)}$ si et seulement si $\beta_i = 1$. Donc

$$Z_n \setminus T_{\mathcal{H}(C)} = \bigcup_{\substack{i\in J\\ \beta_i=1}} Cl\,(i)\,,$$

d'où

$$Y = \left|Z_n \setminus T_{\mathcal{H}(C)}\right| = \sum_{i\in J,\,\beta_i=1} |Cl\,(i)| = \sum_{i\in J} \beta_i\,|Cl\,(i)|$$

et

$$Edimh_q(n) = E\,(Y) = \sum_{i\in J} |Cl\,(i)|\, E\,(\beta_i)\,.$$

Pour trouver $E\,(\beta_i)$, nous allons chercher des relations entre β_i et α_i et utiliser le fait que les probabilités des α_i sont connues.

Soit $i \in J$. D'après la proposition 4.4.8, p. 136,

$$Cl\,(i) \subset Z_n \setminus T_{\mathcal{H}(C)} \text{ si et seulement si } Cl\,(i) \subset Z_n \setminus T_C \text{ et } Cl\,(-i) \subset T_C.$$

Donc

$$\beta_i = 1 \text{ si et seulement si } \alpha_i = 1 \text{ et } \alpha_{-i} = 0, \tag{4.12}$$

où pour tout i, $i \in J$, α_{-i} est la variable aléatoire qui prend pour valeur 0 si $Cl\,(-i) \subset T_C$, et 1 sinon, avec les probabilités $1/2$. Notez que si t est le représentant de la classe $Cl\,(-i)$

dans J, alors $\alpha_t = \alpha_{-i}$. Compte tenu de (4.12), nous voyons facilement que les β_i et α_i satisfont la relation suivante :

$$\beta_i = \alpha_i(1 - \alpha_{-i}).$$

Si la classe cyclotomique $Cl\,(i)$ modulo n sur $\mathbf{F}_q$ est réversible, les variables aléatoires α_i et α_{-i} ne sont pas indépendantes, car alors $\alpha_i = \alpha_{-i}$. Donc dans ce cas

$$\beta_i = \alpha_i(1 - \alpha_i) \quad \text{et} \quad \beta_i = 0$$

quelle que soit la valeur de α_i. D'où

$$E\,(\beta_i) = 0 \tag{4.13}$$

pour tout $i \in J$ tel que $Cl\,(i)$ soit réversible.

Si $Cl\,(i)$ n'est pas réversible, α_i et α_{-i} sont indépendantes, car nous choisissons l'appartenance ou non de $Cl\,(i)$ et de $Cl\,(-i)$ à l'ensemble de définition T_C indépendamment l'un de l'autre. Donc

$$E\,(\beta_i) = E\,(\alpha_i - \alpha_i\alpha_{-i}) = E\,(\alpha_i) - E\,(\alpha_i)\,E\,(\alpha_{-i}) = \frac{1}{2} - \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}, \tag{4.14}$$

car

$$E\,(\alpha_i) = 0\cdot\frac{1}{2} + 1\cdot\frac{1}{2} = \frac{1}{2}.$$

Notons J_1 le sous-ensemble de J qui regroupe les représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$, et J_2 le sous-ensemble de J contenant les représentants des classes cyclotomiques non réversibles. Évidemment, $J = J_1 \cup J_2$. Nous obtenons alors

$$E\,(Y) = \sum_{i\in J} |Cl\,(i)|\,E\,(\beta_i) = \sum_{i\in J_1} |Cl\,(i)|\,E\,(\beta_i) + \sum_{i\in J_2} |Cl\,(i)|\,E\,(\beta_i) = \frac{1}{4}\sum_{i\in J_2} |Cl\,(i)|$$

à cause de formules (4.13) et (4.14). Il reste à remarquer que, comme $\sum_{i\in J} |Cl\,(i)| = n$, nous avons

$$\sum_{i\in J_2} |Cl\,(i)| = n - \sum_{i\in J_1} |Cl\,(i)|\,, \tag{4.15}$$

d'où

$$Edimh_q(n) = E\,(Y) = \frac{1}{4}\sum_{i\in J_2} |Cl\,(i)| = \frac{1}{4}\left(n - \sum_{i\in J_1} |Cl\,(i)|\right). \qquad \square$$

Cela nous fournit immédiatement une majoration de $Edimh_q(n)$:

Corollaire 4.6.7 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n, q) = 1$. Alors*

$$Edimh_q(n) \leq \frac{1}{4}\,(n-1) < \frac{n}{4}.$$

Preuve. La classe cyclotomique $Cl\,(0) = \{0\}$ modulo n sur $\mathbf{F}_q$ est toujours réversible, donc

$$B_{n,q} = \sum_i |Cl\,(i)| \geq |Cl\,(0)| = 1, \tag{4.16}$$

où i parcourt un ensemble de représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$. Cela et la proposition 4.6.6 nous donnent immédiatement le résultat. □

Remarquons au passage que :

Proposition 4.6.8 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n, q) = 1$. La dimension maximale du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ est $\frac{1}{2}(n - B_{n,q})$, i.e. $2Edimh_q(n)$, où $B_{n,q}$ est défini par (4.11). Plus précisément,*

$$maxdimh_q(n) \stackrel{\text{déf}}{=} \max\{\dim \mathcal{H}\,(C) \mid C \in C(n, q)\} = \frac{1}{2}(n - B_{n,q}).$$

Preuve. Comme $\dim \mathcal{H}\,(C) = n - \left|T_{\mathcal{H}(C)}\right|$ (cf. propriété 4.3.4, p. 131), la dimension de $\mathcal{H}\,(C)$ est maximale quand le cardinal de $T_{\mathcal{H}(C)}$ est minimal, où C est un code cyclique de longueur n sur $\mathbf{F}_q$.

Soit $i \in Z_n$. Si la classe cyclotomique $Cl\,(i)$ modulo n sur $\mathbf{F}_q$ est réversible, elle est contenue dans $T_{\mathcal{H}(C)}$ (cf. proposition 4.4.8, p. 136).

Sinon, d'après la même proposition, $Cl\,(i) \not\subset T_{\mathcal{H}(C)}$ si et seulement si $Cl\,(i) \not\subset T_C$ et $Cl\,(-i) \subset T_C$. D'ailleurs, au moins une des classes $Cl\,(i)$ et $Cl\,(-i)$ est contenue dans $T_{\mathcal{H}(C)}$ (cf. corollaire 4.4.9).

Donc $T_{\mathcal{H}(C)}$ est minimal si et seulement s'il contient exactement une classe cyclotomique de chaque paire $Cl\,(i)$ et $Cl\,(-i)$, par conséquent si et seulement si $Z_n \setminus T_{\mathcal{H}(C)}$ contient exactement une classe cyclotomique de chaque paire de classes non réversibles $Cl\,(i)$ et $Cl\,(-i)$. Comme $|Cl\,(i)| = |Cl\,(-i)|$ pour tout $i \in Z_n$ (cf. propriété 4.2.34, p. 129), $T_{\mathcal{H}(C)}$ est minimal si et seulement si

$$\left|Z_n \setminus T_{\mathcal{H}(C)}\right| = \frac{1}{2} \sum_{i \in J_2} |Cl\,(i)|\,,$$

où J_2 est un ensemble de représentants des classes cyclotomiques non réversibles modulo n sur $\mathbf{F}_q$. Compte tenu de (4.15), nous obtenons que si C est un code cyclique de longueur n sur $\mathbf{F}_q$ tel que $\dim \mathcal{H}\,(C) = maxdimh_q(n)$, alors

$$\dim \mathcal{H}\,(C) = n - \left|T_{\mathcal{H}(C)}\right| = \left|Z_n \setminus T_{\mathcal{H}(C)}\right| = \frac{1}{2} \sum_{i \in J_2} |Cl\,(i)| = \frac{1}{2}\,(n - B_{n,q})\,.$$ □

4.6.4 Cas général

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Considérons maintenant le cas général où n et q ne sont pas liés par la condition $(n, q) = 1$.

Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Dans ce cas l'expression de $Edimh_q(n)$ est plus compliquée.

Nous donnons d'abord un lemme technique.

Lemme 4.6.9 *Si v est un entier positif non nul, alors*

$$\sum_{j=1}^{v}(2j+1)j = \frac{v(4v+5)(v+1)}{6}.$$

Preuve. Nous allons raisonner par induction.

Quand $v = 1$, l'égalité est évidemment vraie.

Supposons qu'elle soit vraie pour $v - 1$, et démontrons la pour v.

$$\begin{aligned}
\sum_{j=1}^{v}(2j+1)j &= \sum_{j=1}^{v-1}(2j+1)j + (2v+1)v \\
&= \frac{(v-1)(4(v-1)+5)v}{6} + (2v+1)v \\
&= \frac{v}{6}(4v^2 + v - 4v - 1 + 12v + 6) \\
&= \frac{v}{6}(4v^2 + 9v + 5) \\
&= \frac{v(v+1)(4v+5)}{6}.
\end{aligned}$$

□

Nous utiliserons la notation suivante :

Notation 4.6.10 Si v est un entier positif non nul,

$$\delta_v = \begin{cases} 1 & \text{si } v \text{ est pair,} \\ 0 & \text{sinon.} \end{cases}$$

Autrement dit, $\delta_v = v + 1 \bmod 2$ pour tout $v \in \mathbb{N}^*$. □

Et voici l'expression de $Edimh_q(n)$ en fonction de $B_{r,q}$:

Proposition 4.6.11 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Alors*

$$Edimh_q(n) = n\left(\frac{1}{3} - \frac{1}{6(p^s+1)}\right) - B_{r,q}\left(\frac{p^s+1}{12} + \frac{2-3\delta_{p^s}}{12(p^s+1)}\right),$$

où $B_{r,q}$ est défini dans (4.11) et δ_{p^s} est défini dans la notation 4.6.10.

Preuve. La démonstration suit les grandes lignes de la démonstration de la proposition 4.6.6, p. 143.

Soit J un ensemble de représentants des classes cyclotomiques modulo r sur $\mathbf{F}_q$. Le corollaire 4.3.17, p. 133, montre que l'ensemble de définition T_C (cf. définition 4.3.12) d'un code cyclique C de longueur n sur $\mathbf{F}_q$ est égal à $\bigcup_{i\in J} Cl\,(i)\hat{\ } t_i$, où $0 \leq t_i \leq p^s$ et $Cl\,(i)$ est la classe cyclotomique de i modulo r sur $\mathbf{F}_q$.

Choisir un code cyclique au hasard revient à choisir son ensemble de définition de ce type, i.e. choisir les t_i pour tout $i \in J$. Chaque t_i a $p^s + 1$ valeurs possibles (de 0 à p^s). Pour chaque $i \in J$ choisissons t_i indépendemment des autres t_j, $j \in J$, $j \neq i$, avec la même probabilité uniforme $1/(p^s+1)$ pour chaque valeur possible de t_i.

Montrons que de cette façon nous obtenons chaque code cyclique $C \in C(n,q)$ avec la même probabilité uniforme $1/\left|C(n,q)\right|$. En effet, soit $C \in C(n,q)$. Quelle est la probabilité de l'obtenir en choisissant ainsi les t_i ? Soit son ensemble de définition $T_C = \bigcup_{i\in J} Cl\,(i)\hat{\ } v_i$. La probabilité d'obtenir $t_i = v_i$ est $1/(p^s+1)$ pour tout $i \in J$, et comme elles sont indépendantes, la probabilité d'obtenir $(t_i)_{i\in J} = (v_i)_{i\in J}$ est

$$\left(\frac{1}{p^s+1}\right)^{|J|} = \frac{1}{(p^s+1)^{|J|}} = \frac{1}{(p^s+1)^{c_q(r)}} = \frac{1}{|C(n,q)|}$$

à cause de la notation 4.5.1 et de la proposition 4.5.2, p. 138.

Compte tenu de la définition 4.3.12 et de la propriété 4.3.13, p. 133, nous obtenons que la variable aléatoire Y définie dans la remarque 4.6.3, p. 140, est égale à

$$Y = \dim \mathcal{H}\,(C) = n - \left|T_{\mathcal{H}(C)}\right|.$$

Nous définissons les variables aléatoires supplémentaires suivantes : α_i et α_{-i}, $i \in J$, prennent pour valeur la multiplicité (cf. définition 4.2.2, p. 120) de la classe cyclotomique $Cl\,(i)$ modulo r sur $\mathbf{F}_q$ (respectivement, $Cl\,(-i)$) dans le multi-ensemble T_C avec la probabilité uniforme $1/(p^s+1)$ (l'ensemble des valeurs de α_i est donc $\{0, 1, \ldots, p^s\}$) ; β_i, $i \in J$, prend pour valeur la multiplicité de la classe cyclotomique $Cl\,(i)$ dans $T_{\mathcal{H}(C)}$.

Nous avons donc

$$Y = n - \left|\bigcup_{i\in J} Cl\,(i)\hat{\ }\beta_i\right| = n - \sum_{i\in J} \beta_i \left|Cl\,(i)\right|$$

et

$$Edimh_q(n) = E\,(Y) = E\left(n - \sum_{i\in J} \beta_i \left|Cl\,(i)\right|\right) = n - \sum_{i\in J} \left|Cl\,(i)\right| E\,(\beta_i)\,. \qquad (4.17)$$

Pour trouver la moyenne $E\,(\beta_i)$, nous allons chercher des relations entre β_i et α_i et utiliser le fait que les probabilités des α_i sont connues.

La proposition 4.4.11, p. 136, nous donne la relation entre β_i et α_i :

$$\beta_i = \max(\alpha_i, p^s - \alpha_{-i}). \qquad (4.18)$$

Nous allons calculer $E(\beta_i)$. Par définition,

$$E(\beta_i) = \sum_{j=0}^{p^s} \Pr\{\beta_i = j\}\, j, \tag{4.19}$$

où nous notons $\Pr\{\beta_i = j\}$ la probabilité que la variable aléatoire β_i soit égale à j.

Soit $i \in J$ tel que $Cl(i)$ est réversible. Alors $\alpha_i = \alpha_{-i}$ et $\beta_i = \max(\alpha_i, p^s - \alpha_i)$. Comme $\max(\alpha_i, p^s - \alpha_i) \geq p^s/2$ quelle que soit la valeur de α_i,

$$\Pr\{\beta_i = j\} = \Pr\{\max(\alpha_i, p^s - \alpha_i) = j\} = 0 \quad \text{pour tout } j,\ j < p^s/2.$$

Si $j > p^s/2$, alors $j \neq p^s - j$, donc

$$\begin{aligned}
\Pr\{\beta_i = j\} &= \Pr\{\max(\alpha_i, p^s - \alpha_i) = j\} \\
&= \Pr\{\alpha_i = j \vee \alpha_i = p^s - j\} \\
&= \Pr\{\alpha_i = j\} + \Pr\{\alpha_i = p^s - j\} \\
&= \frac{1}{p^s+1} + \frac{1}{p^s+1} \\
&= \frac{2}{p^s+1},
\end{aligned}$$

où $\vee$ et $\wedge$ signifient respectivement “ou” et “et”.

Si p^s est pair, $p^s/2$ est un entier. Si $j = p^s/2$,

$$\Pr\{\beta_i = p^s/2\} = \Pr\{\max(\alpha_i, p^s - \alpha_i) = p^s/2\} = \Pr\{\alpha_i = p^s/2\} = \frac{1}{p^s+1}.$$

Rappelons que

$$\sum_{a=a_1}^{a_2} a = \frac{a_1 + a_2}{2}(a_2 - a_1 + 1)\ \forall a_1, a_2 \in \mathbb{Z}. \tag{4.20}$$

Soit p^s impair. Alors

$$\begin{aligned}
E(\beta_i) &\overset{(4.19)}{=} \sum_{j=0}^{p^s} \Pr\{\beta_i = j\}\, j \\
&= \sum_{j=0}^{(p^s-1)/2} \Pr\{\beta_i = j\}\, j + \sum_{j=(p^s+1)/2}^{p^s} \Pr\{\beta_i = j\}\, j \\
&= \sum_{j=0}^{(p^s-1)/2} 0 \cdot j + \sum_{j=(p^s+1)/2}^{p^s} \frac{2}{p^s+1} j \\
&= \frac{2}{p^s+1} \sum_{j=(p^s+1)/2}^{p^s} j
\end{aligned}$$

$$
\begin{aligned}
&\overset{(4.20)}{=} \quad \frac{2}{p^s+1} \cdot \frac{\frac{p^s+1}{2}+p^s}{2}\left(p^s-\frac{p^s+1}{2}+1\right) \\
&= \quad \frac{2}{p^s+1} \cdot \frac{3p^s+1}{4} \cdot \frac{p^s+1}{2} \\
&= \quad \frac{3p^s+1}{4}.
\end{aligned}
$$

Si maintenant p^s est pair,

$$
\begin{aligned}
E(\beta_i) &\overset{(4.19)}{=} \sum_{j=0}^{p^s} \Pr\{\beta_i = j\}\, j \\
&= \sum_{j=0}^{p^s/2-1} \Pr\{\beta_i = j\}\, j + \Pr\{\beta_i = p^s/2\}\, p^s/2 + \sum_{j=p^s/2+1}^{p^s} \Pr\{\beta_i = j\}\, j \\
&= 0 + \frac{1}{p^s+1} \cdot \frac{p^s}{2} + \sum_{j=p^s/2+1}^{p^s} \frac{2}{p^s+1} j \\
&= \frac{1}{p^s+1} \cdot \frac{p^s}{2} + \frac{2}{p^s+1} \sum_{j=p^s/2+1}^{p^s} j \\
&\overset{(4.20)}{=} \frac{1}{p^s+1} \cdot \frac{p^s}{2} + \frac{2}{p^s+1} \cdot \frac{\frac{p^s}{2}+1+p^s}{2}\left(p^s - \frac{p^s}{2} - 1 + 1\right) \\
&= \frac{p^s}{2(p^s+1)} + \frac{1}{p^s+1} \cdot \frac{3p^s+2}{2} \cdot \frac{p^s}{2} \\
&= \frac{p^s}{2(p^s+1)}\left(\frac{3p^s+2}{2}+1\right) \\
&= \frac{p^s}{2(p^s+1)} \cdot \frac{3p^s+4}{2} \\
&= \frac{(3p^s+4)p^s}{4(p^s+1)} \\
&= \frac{(3p^s+4)(p^s+1-1)}{4(p^s+1)} \\
&= \frac{3p^s+4}{4} - \frac{3p^s+4}{4(p^s+1)} \\
&= \frac{3p^s+1}{4} + \frac{3}{4} - \frac{3p^s+3+1}{4(p^s+1)} \\
&= \frac{3p^s+1}{4} - \frac{1}{4(p^s+1)}.
\end{aligned}
$$

Nous pouvons écrire ces deux cas (où p^s est quelconque) de manière plus compacte comme suit :

$$E(\beta_i) = \frac{3p^s+1}{4} - \frac{\delta_{p^s}}{4(p^s+1)},$$

où δ_{p^s} est défini dans la notation 4.6.10.

Maintenant soit $i \in J$ tel que $Cl(i)$ n'est pas réversible. Dans ce cas les variables aléatoires α_i et α_{-i} sont indépendantes. Il faut trouver $\Pr\{\beta_i = j\}$ pour tout j, $0 \leq j \leq p^s$, pour pouvoir calculer $E(\beta_i)$ à l'aide de la formule (4.19).

Nous allons utiliser la formule suivante :

$$\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\}.$$

Nous obtenons

$$\begin{aligned}
\Pr\{\beta_i = j\} &= \Pr\{\max(\alpha_i, p^s - \alpha_{-i}) = j\} \\
&= \Pr\{(\alpha_i = j \wedge p^s - \alpha_{-i} \leq j) \vee (p^s - \alpha_{-i} = j \wedge \alpha_i \leq j)\} \\
&= \Pr\{\alpha_i = j \wedge \alpha_{-i} \geq p^s - j\} + \Pr\{\alpha_{-i} = p^s - j \wedge \alpha_i \leq j\} \\
&\quad - \Pr\{\alpha_i = j \wedge \alpha_{-i} \geq p^s - j \wedge \alpha_{-i} = p^s - j \wedge \alpha_i \leq j\} \\
&= \Pr\{\alpha_i = j\} \Pr\{\alpha_{-i} \geq p^s - j\} + \Pr\{\alpha_{-i} = p^s - j\} \Pr\{\alpha_i \leq j\} \\
&\quad - \Pr\{\alpha_i = j \wedge \alpha_{-i} = p^s - j\} \\
&= \Pr\{\alpha_i = j\} \sum_{l=p^s-j}^{p^s} \Pr\{\alpha_{-i} = l\} + \Pr\{\alpha_{-i} = p^s - j\} \sum_{l=0}^{j} \Pr\{\alpha_i = l\} \\
&\quad - \Pr\{\alpha_i = j\} \Pr\{\alpha_{-i} = p^s - j\} \\
&= \frac{1}{p^s+1} \sum_{l=p^s-j}^{p^s} \frac{1}{p^s+1} + \frac{1}{p^s+1} \sum_{l=0}^{j} \frac{1}{p^s+1} - \frac{1}{p^s+1} \cdot \frac{1}{p^s+1} \\
&= \frac{j+1}{(p^s+1)^2} + \frac{j+1}{(p^s+1)^2} - \frac{1}{(p^s+1)^2} \\
&= \frac{2j+1}{(p^s+1)^2},
\end{aligned}$$

et donc

$$\begin{aligned}
E(\beta_i) &= \sum_{j=0}^{p^s} \Pr\{\beta_i = j\} j \\
&= \sum_{j=0}^{p^s} \frac{2j+1}{(p^s+1)^2} j \\
&= \frac{1}{(p^s+1)^2} \sum_{j=0}^{p^s} (2j+1)j \\
&\overset{Lem.\,4.6.9}{=} \frac{p^s(4p^s+5)(p^s+1)}{6(p^s+1)^2}
\end{aligned}$$

$$= \frac{p^s(4p^s+5)}{6(p^s+1)}.$$

Notons J_1 le sous-ensemble de J qui regroupe les représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$, et J_2 le sous-ensemble de J contenant les représentants des classes cyclotomiques non réversibles. Évidemment, $J = J_1 \cup J_2$. Alors

$$\begin{aligned}
Edimh_q(n) &= E(Y) \\
&\overset{(4.17)}{=} n - \sum_{i\in J} |Cl(i)|\, E(\beta_i) \\
&= n - \sum_{i\in J_1} |Cl(i)|\, E(\beta_i) - \sum_{i\in J_2} |Cl(i)|\, E(\beta_i) \\
&= n - \sum_{i\in J_1} |Cl(i)| \left(\frac{3p^s+1}{4} - \frac{\delta_{p^s}}{4(p^s+1)}\right) - \sum_{i\in J_2} |Cl(i)| \frac{p^s(4p^s+5)}{6(p^s+1)} \\
&= rp^s - B_{r,q}\left(\frac{3p^s+1}{4} - \frac{\delta_{p^s}}{4(p^s+1)}\right) - (r - B_{r,q})\frac{p^s(4p^s+5)}{6(p^s+1)} \\
&= rp^s\left(1 - \frac{4p^s+5}{6(p^s+1)}\right) - B_{r,q}\left(\frac{3p^s+1}{4} - \frac{\delta_{p^s}}{4(p^s+1)} - \frac{p^s(4p^s+5)}{6(p^s+1)}\right) \\
&= n\left(\frac{1}{3} - \frac{1}{6(p^s+1)}\right) - B_{r,q}\left(\frac{3p^s+1}{4} - \frac{4p^s}{6} - \frac{p^s}{6(p^s+1)} - \frac{\delta_{p^s}}{4(p^s+1)}\right) \\
&= n\left(\frac{1}{3} - \frac{1}{6(p^s+1)}\right) - B_{r,q}\left(\frac{p^s+3}{12} - \frac{1}{6} + \frac{1}{6(p^s+1)} - \frac{\delta_{p^s}}{4(p^s+1)}\right) \\
&= n\left(\frac{1}{3} - \frac{1}{6(p^s+1)}\right) - B_{r,q}\left(\frac{p^s+1}{12} + \frac{2-3\delta_{p^s}}{12(p^s+1)}\right).
\end{aligned}$$

□

Nous obtenons une majoration de $Edimh_q(n)$:

Corollaire 4.6.12 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Alors*

$$Edimh_q(n) < \frac{n}{3} \quad \text{pour tout } n \geq 1.$$

Preuve. D'après la proposition 4.6.11,

$$\frac{Edimh_q(n)}{n} = \frac{1}{3} - \frac{1}{6(p^s+1)} - \frac{B_{r,q}}{r}\left(\frac{p^s+1}{12p^s} + \frac{2-3\delta_{p^s}}{12p^s(p^s+1)}\right).$$

Comme $p^s \geq 1$, $B_{r,q} \geq 1$ (cf. formule (4.16), p. 146) et $\delta_{p^s} \leq 1$, nous obtenons que $\frac{1}{6(p^s+1)} > 0$ et que

$$\frac{B_{r,q}}{r}\left(\frac{p^s+1}{12p^s} + \frac{2-3\delta_{p^s}}{12p^s(p^s+1)}\right) = \frac{B_{r,q}}{r}\cdot\frac{(p^s+1)^2+2-3\delta_{p^s}}{12p^s(p^s+1)} \geq \frac{B_{r,q}}{r}\cdot\frac{4+2-3}{12p^s(p^s+1)} > 0,$$

donc

$$\frac{Edimh_q(n)}{n} < \frac{1}{3}. \qquad \square$$

Nous avons vu que si $(n, q) = 1$, $maxdimh_q(n) = 2Edimh_q(n)$ (cf. proposition 4.6.8). Cela n'est plus valable dans le cas général.

Proposition 4.6.13 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $n = rp^s$, où r est premier avec p et $s \geq 0$. Alors la dimension maximale du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ est $\frac{1}{2}(n - (1 - \delta_{p^s})B_{r,q})$, autrement dit,*

$$maxdimh_q(n) = \begin{cases} \frac{1}{2}(n - B_{r,q}) & si\ p^s\ est\ impair, \\ \frac{n}{2} & sinon. \end{cases}$$

Preuve. Soit J un ensemble de représentants des classes cyclotomiques modulo r sur $\mathbf{F}_q$. Notons J_1 le sous-ensemble de J qui regroupe les représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$, et J_2 le sous-ensemble de J contenant les représentants des classes cyclotomiques non réversibles. Évidemment, $J = J_1 \cup J_2$. Soit $C \in C(n, q)$. Pour tout $i \in J$, soient β_i et β_{-i} les multiplicités de $Cl\,(i)$ et $Cl\,(-i)$, respectivement, dans $T_{\mathcal{H}(C)}$, et soient α_i et α_{-i} les multiplicités de $Cl\,(i)$ et $Cl\,(-i)$, respectivement, dans T_C. Comme dans la démonstration de la proposition 4.6.11, nous obtenons

$$\begin{aligned} \dim \mathcal{H}\,(C) &= n - \left|T_{\mathcal{H}(C)}\right| \\ &= n - \sum_{i \in J} \beta_i \left|Cl\,(i)\right| \\ &= n - \sum_{i \in J_1} \left|Cl\,(i)\right| \beta_i - \sum_{i \in J_2} \left|Cl\,(i)\right| \beta_i \\ &= n - \sum_{i \in J_1} \left|Cl\,(i)\right| \beta_i - \sum_{i \in J_2,\, 0<i<r/2} \left(\left|Cl\,(i)\right| \beta_i + \left|Cl\,(-i)\right| \beta_{-i}\right) \\ &= n - \sum_{i \in J_1} \left|Cl\,(i)\right| \beta_i - \sum_{i \in J_2,\, 0<i<r/2} \left|Cl\,(i)\right| (\beta_i + \beta_{-i}). \end{aligned}$$

Donc maximiser $\dim \mathcal{H}\,(C)$ revient à minimiser β_i pour $i \in J_1$ et minimiser $\beta_i + \beta_{-i}$ pour $i \in J_2$, $0 < i < r/2$.

Comme nous avons vu dans la démonstration de la proposition 4.6.11, si $i \in J_1$, alors la valeur minimale de β_i est $p^s/2$ si p^s est pair, et $(p^s + 1)/2$ sinon. En commun, c'est $(p^s + 1 - \delta_{p^s})/2$.

Maintenant soit $i \in J_2$, $0 < i < r/2$. Quelle est la valeur minimale de

$$R \overset{\text{déf}}{=} \beta_i + \beta_{-i} \overset{(4.18)}{=} \max(\alpha_i, p^s - \alpha_{-i}) + \max(\alpha_{-i}, p^s - \alpha_i)\ ?$$

Comme $\max(a, b) \geq a$ pour tous les a et b, nous avons

$$R \geq \alpha_i + p^s - \alpha_i = p^s,$$

et cette valeur minimale est atteinte quand $\alpha_i = p^s - \alpha_{-i}$.

Donc

$$\begin{aligned}
maxdimh_q(n) &= n - \min\left(\sum_{i\in J_1} |Cl\,(i)|\,\beta_i\right) - \min\left(\sum_{i\in J_2,\,0<i<r/2} |Cl\,(i)|\,(\beta_i+\beta_{-i})\right) \\
&= n - \sum_{i\in J_1} |Cl\,(i)|\,\frac{p^s+1-\delta_{p^s}}{2} - \sum_{i\in J_2,\,0<i<r/2} |Cl\,(i)|\,p^s \\
&= n - B_{r,q}\frac{p^s+1-\delta_{p^s}}{2} - \frac{r-B_{r,q}}{2}p^s \\
&= \frac{n}{2} - B_{r,q}\left(\frac{p^s+1-\delta_{p^s}}{2} - \frac{p^s}{2}\right) \\
&= \frac{n}{2} - \frac{B_{r,q}}{2}(1-\delta_{p^s}).
\end{aligned}$$

□

4.6.5 L'étude de $B_{n,q}$

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Dans les sections précédentes nous avons exprimé la dimension moyenne du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ au moyen de

$$B_{r,q} = \sum_{i\in J_1} |Cl\,(i)|\,,$$

où $n = rp^s$, r est premier avec q et J_1 est un ensemble de représentants des classes cyclotomiques réversibles modulo r sur $\mathbf{F}_q$ (cf. propositions 4.6.6, p. 143, et 4.6.11, p. 147). Maintenant nous allons étudier $B_{r,q}$ pour pouvoir étudier le comportement de la dimension moyenne du hull des codes cycliques.

Nous montrons d'abord le résultat suivant :

Proposition 4.6.14 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n,q)=1$. Alors*

$$B_{n,q} = \sum_{\substack{l\mid n \\ \exists i>0\,:\,q^i\equiv -1 \bmod l}} \phi(l),$$

où i dépend de q et de l.

Preuve. Soit β une racine n-ième primitive de l'unité sur $\mathbf{F}_q$.

Comme dans la démonstration de la proposition 4.5.4, nous montrons qu'on peut partitionner l'ensemble $\{\beta^j \mid 0 \le j < n\}$ des racines n-ièmes de l'unité sur $\mathbf{F}_q$ en classes suivantes : pour chaque diviseur l de n il existe une classe R_l de cardinal $\phi(l)$ des éléments d'ordre l. La classe R_l est l'ensemble des racines l-ièmes primitives de l'unité sur $\mathbf{F}_q$.

Soit γ une racine l-ième primitive de l'unité sur $\mathbf{F}_q$. La propriété 4.2.40, p. 129, montre que $M_\gamma(X)$ est son propre réciproque si et seulement si le polynôme minimal de chaque racine l-ième primitive de l'unité sur $\mathbf{F}_q$ est son propre réciproque. Compte tenu de la propriété 4.2.41, soit toutes les classes cyclotomiques modulo n sur $\mathbf{F}_q$ correspondant aux polynômes minimaux des éléments de R_l sont réversibles, soit aucune d'entre elles. Et, d'après la propriété 4.2.39, elles sont réversibles si et seulement si

$$\text{il existe } i > 0 \text{ tel que } q^i \equiv -1 \bmod l. \tag{4.21}$$

Soit J_1 un ensemble de représentants des classes cyclotomiques réversibles modulo n sur $\mathbf{F}_q$. Nous avons donc

$$B_{n,q} = \sum_{i \in J_1} |Cl\,(i)| = \sum_{l|n} \sum_{i \in J_1,\, \beta^i \in R_l} |Cl\,(i)|\,.$$

Mais si un diviseur l de n ne vérifie pas la condition (4.21), toutes les classes cyclotomiques $Cl\,(i)$ modulo n sur $\mathbf{F}_q$ telles que $\beta^i \in R_l$ sont non réversibles, donc $\sum_{i\in J_1,\,\beta^i\in R_l} |Cl\,(i)| = 0$. Sinon toutes les classes cyclotomiques $Cl\,(i)$ telles que $\beta^i \in R_l$ sont réversibles, et la somme de leurs cardinaux est égale au cardinal de R_l, c'est-à-dire $\sum_{i\in J_1,\,\beta^i\in R_l} |Cl\,(i)| = |R_l| = \phi(l)$. Ce qui nous donne le résultat désiré. □

Une question se pose donc : pour quels $l > 0$ existe-t-il $i > 0$ tel que $q^i \equiv -1 \bmod l$? Ou, alternativement, pour quels $l > 0$ existe-t-il $i > 0$ tel que $l \mid q^i + 1$?

Nous étudions cette question dans la section suivante.

4.7 Étude des entiers positifs non nuls

Dans ce chapitre q désigne le cardinal d'un corps fini, c'est donc une puissance d'un nombre premier. Mais les résultats de cette section seront valables pour $q > 0$ quelconque. Soit donc q un entier positif non nul.

4.7.1 Définitions

Notation 4.7.1 Nous notons

$$\mathcal{N}_q = \{n \in \mathbb{N}^* \mid \exists l > 0 \text{ tel que } q^l \equiv -1 \bmod n\},$$

où l dépend de q et de n.

Remarque 4.7.2 Notez que $q^l \equiv -1 \bmod n$ est équivalent à $n \mid q^l + 1$. Nous pouvons donc définir $\mathcal{N}_q$ comme l'ensemble des diviseurs des entiers de la forme $q^l + 1$, $l > 0$, plus précisément,

$$\mathcal{N}_q = \bigcup_{l>0} \{n : n \mid q^l + 1\}.$$

Remarque 4.7.3 Si $n = 1$, alors $n \in \mathcal{N}_q$ pour tout $q > 0$ et $n \mid q^l + 1$ pour tout $l > 0$.

Si $n = 2$, alors $n \in \mathcal{N}_q$ si et seulement si q est impair, et si c'est le cas, $n \mid q^l + 1$ pour tout $l > 0$.

Si $q^l \equiv -1 \bmod n$ et $\mathrm{ord}_n(q) \mid l$, alors, comme $q^{\mathrm{ord}_n(q)} \equiv 1 \bmod n$, nous obtenons que $-1 \equiv 1 \bmod n$, ce qui n'est possible que si $n = 1$ ou $n = 2$. Nous traiterons ces deux cas à part et supposerons généralement que $n > 2$.

Notez aussi que si $q = 1$, alors 1 et 2 sont des éléments uniques de $\mathcal{N}_1$, car d'après la définition $\mathcal{N}_1 = \{n \in \mathbb{N}^* \mid \exists l > 0 \text{ tel que } n \mid 1^l + 1\} = \{1, 2\}$. Ce cas n'est pas intéressant, et nous supposerons désormais que $q \geq 2$. □

Remarque 4.7.4 Notez que si $n \in \mathcal{N}_q$, alors il est premier avec q. En effet, si $(n, q) = d$, alors $d \mid n$ et donc $d \mid q^l + 1$, $l > 0$, et $d \mid q$ et donc $d \mid q^l$, d'où $d \mid \big(q^l, q^l + 1\big) = 1$ et $d = 1$.

D'abord nous allons donner plusieurs définitions équivalentes de $\mathcal{N}_q$.

Rappelons que $\mathrm{ord}_n(q)$ est l'ordre de q modulo n (cf. définition 4.2.11, p. 123).

Proposition 4.7.5 *Soit $n > 2$, soit $l > 0$. Si $q^l \equiv -1 \bmod n$, alors $\mathrm{ord}_n(q)$ est pair et*

$$l \equiv \frac{\mathrm{ord}_n(q)}{2} \bmod \mathrm{ord}_n(q) \ ;$$

de plus, nous avons alors

$$q^v \equiv -1 \bmod n \textit{ pour tout } v > 0 \textit{ tel que } v \equiv \tfrac{\mathrm{ord}_n(q)}{2} \bmod \mathrm{ord}_n(q).$$

Preuve. Soit $q^l \equiv -1 \bmod n$, $l > 0$. Puisque $q^{\mathrm{ord}_n(q)} \equiv 1 \bmod n$ et $q^j \bmod n$ sont distincts pour $1 \leq j \leq \mathrm{ord}_n(q)$ (cf. propriété 4.2.13, p. 123), nous obtenons que si l' est le plus petit entier positif non nul satisfaisant $q^{l'} \equiv -1 \bmod n$, alors il est compris entre 1 et $\mathrm{ord}_n(q)$. D'ailleurs, $l' \neq \mathrm{ord}_n(q)$, car $n > 2$ (cf. remarque 4.7.3). Donc $1 \leq l' < \mathrm{ord}_n(q)$. De plus, pour les mêmes raisons, les autres v tels que $q^v \equiv -1 \bmod n$ satisfont $v \equiv l' \bmod \mathrm{ord}_n(q)$.

Il reste à montrer que $l' = \mathrm{ord}_n(q)/2$.

Comme $q^{l'} \equiv -1 \bmod n$, nous avons $q^{2l'} \equiv 1 \bmod n$ et donc $\mathrm{ord}_n(q) \mid 2l'$. Par conséquent $2l'$ est de la forme $i\mathrm{ord}_n(q)$, $i \geq 1$. Mais $l' < \mathrm{ord}_n(q)$, d'où $2l' < 2\mathrm{ord}_n(q)$, il ne reste donc qu'une possibilité $2l' = \mathrm{ord}_n(q)$. D'où $l' = \mathrm{ord}_n(q)/2$. □

Remarque 4.7.6 Autrement dit, si $n \in \mathcal{N}_q$, $n > 2$, alors $q^l \equiv -1 \bmod n$, $l > 0$, si et seulement si $l \equiv \frac{\mathrm{ord}_n(q)}{2} \bmod \mathrm{ord}_n(q)$. En d'autres termes, si $n \in \mathcal{N}_q$, $n > 2$, alors $n \mid q^l + 1$, $l > 0$, si et seulement si l est de la forme $\frac{\mathrm{ord}_n(q)}{2} + i\mathrm{ord}_n(q)$ pour un $i \geq 0$.

Nous obtenons donc la deuxième définition de $\mathcal{N}_q$.

Proposition 4.7.7

$$\mathcal{N}_q = \{i \in \{1, 2\} \mid (i, q) = 1\} \cup \{n > 2 \mid q^{\mathrm{ord}_n(q)/2} \equiv -1 \bmod n\}.$$

Preuve. C'est une conséquence directe de la définition de $\mathcal{N}_q$ et de la proposition 4.7.5. □

Notez que l'ensemble

$$\{i \in \{1,2\} \mid (i,q)=1\} = \begin{cases} \{1\} & \text{si } q \text{ est pair,} \\ \{1,2\} & \text{sinon.} \end{cases}$$

Pour établir la troisième définition de $\mathcal{N}_q$, montrons la proposition suivante :

Proposition 4.7.8 *Soit $n > 2$. Soit β et t tels que $\phi(n) = 2^\beta t$, t impair. Alors $n \in \mathcal{N}_q$ si et seulement s'il existe i, $0 \leq i < \beta$, tel que $q^{2^i t} \equiv -1 \bmod n$. En d'autres termes, $n \in \mathcal{N}_q$ si et seulement s'il existe j, $1 \leq j \leq \beta$, tel que $q^{\phi(n)/2^j} \equiv -1 \bmod n$.*

Preuve. Soit $n \in \mathcal{N}_q$, $n > 2$. Soit α et s tels que $\mathrm{ord}_n(q) = 2^\alpha s$. D'après la proposition 4.7.7,

$$q^{\mathrm{ord}_n(q)/2} \equiv q^{2^{\alpha-1}s} \equiv -1 \bmod n,$$

c'est-à-dire $n \mid q^{2^{\alpha-1}s} + 1$. D'ailleurs, grâce au théorème 4.2.10 et à la proposition 4.2.16, p. 123, $\mathrm{ord}_n(q) \mid \phi(n)$, d'où $\alpha \leq \beta$ et $s \mid t$.

En général, pour tous les entiers positifs non nuls a et b et pour tout entier positif impair c nous avons

$$a^{bc} + 1 = \left(a^b + 1\right)\left(a^{b(c-1)} - a^{b(c-2)} + a^{b(c-3)} - \cdots - a^b + 1\right),$$

donc $a^b + 1 \mid a^{bc} + 1$.

Dans notre cas $q^{2^{\alpha-1}s} + 1 \mid q^{2^{\alpha-1}t} + 1$, car t/s est impair. Donc

$$n \mid q^{2^{\alpha-1}t} + 1 = q^{\phi(n)/2^{\beta-\alpha+1}} + 1.$$

Nous posons $i = \alpha - 1$ et $j = \beta - \alpha + 1$ et nous obtenons le résultat désiré.

La réciproque est évidente. □

Remarque 4.7.9 La troisième définition de $\mathcal{N}_q$ est donc la suivante :

$$\begin{aligned} \mathcal{N}_q \;=\; & \{i \in \{1,2\} \mid (i,q)=1\} \cup \\ & \{n > 2 \mid \exists i,\ 0 \leq i < \beta,\ \text{tel que } q^{2^i t} \equiv -1 \bmod n,\ \text{où } \phi(n) = 2^\beta t,\ t \text{ impair}\}, \end{aligned}$$

où, évidemment, i dépend de q et de n.

Une définition équivalente serait

$$\begin{aligned} \mathcal{N}_q \;=\; & \{i \in \{1,2\} \mid (i,q)=1\} \cup \\ & \{n > 2 \mid \exists j,\ 1 \leq j \leq \beta,\ \text{tel que } q^{\phi(n)/2^j} \equiv -1 \bmod n,\ \text{où } \phi(n) = 2^\beta t,\ t \text{ impair}\}. \end{aligned}$$ □

4.7.2 Quelques propriétés

Corollaire 4.7.10 *Soit $n > 2$. Si $\mathrm{ord}_n(q)$ est impair, alors $n \notin \mathcal{N}_q$.*

Preuve. Elle découle de la proposition 4.7.5. □

Si n est un nombre premier impair, la situation est plus simple.

Proposition 4.7.11 *Si n est un nombre premier impair, alors $n \in \mathcal{N}_q$ si et seulement si $\mathrm{ord}_n(q)$ est pair.*

Preuve. Le corollaire 4.7.10 implique que si $n \in \mathcal{N}_q$, $\mathrm{ord}_n(q)$ est pair.

Pour montrer la réciproque, remarquons que

$$n \mid q^{\mathrm{ord}_n(q)} - 1 = \left(q^{\mathrm{ord}_n(q)/2} - 1\right)\left(q^{\mathrm{ord}_n(q)/2} + 1\right).$$

Comme n est un nombre premier, il doit diviser soit l'un soit l'autre. Par définition de $\mathrm{ord}_n(q)$ nous avons $n \nmid q^{\mathrm{ord}_n(q)/2} - 1$, donc $n \mid q^{\mathrm{ord}_n(q)/2} + 1$ et par conséquent $n \in \mathcal{N}_q$. □

Même si elle est facile à montrer, cette proposition est forte, car elle permet de dire si un nombre premier n appartient à $\mathcal{N}_q$ en calculant seulement $\mathrm{ord}_n(q)$ et en considérant sa parité, il n'est donc pas nécessaire de calculer $q^{\mathrm{ord}_n(q)/2} + 1$ et de vérifier si n le divise.

Remarque 4.7.12 Cela n'est plus valable pour les n composés, c'est-à-dire si $\mathrm{ord}_n(q)$ est pair, n peut ne pas appartenir à $\mathcal{N}_q$. Par exemple, si $n = 15$ et $q = 2$, $\mathrm{ord}_{15}(2) = 4$, mais $2^{\mathrm{ord}_{15}(2)/2} \equiv 5 \not\equiv -1 \bmod 15$ et $15 \notin \mathcal{N}_2$.

Propriété 4.7.13 *Soit $n \geq 1$. Alors $\mathrm{ord}_d(q) \mid \mathrm{ord}_n(q)$ pour tout d, $d \mid n$.*

Preuve. Cela découle du fait que si $q^l \equiv 1 \bmod n$, alors $q^l \equiv 1 \bmod d$. □

Corollaire 4.7.14 *Soit $n > 2$. Si $\mathrm{ord}_n(q)$ est impair, alors $d \notin \mathcal{N}_q$ pour tout d, $d \mid n$, $d > 2$.*

Preuve. C'est une conséquence directe de la propriété 4.7.13 et du corollaire 4.7.10. □

Propriété 4.7.15 *Si $n \in \mathcal{N}_q$, alors $d \in \mathcal{N}_q$ pour tout d, $d \mid n$.*

Preuve. Si $n \in \mathcal{N}_q$, alors $q^l \equiv -1 \bmod n$ pour un $l > 0$, donc $q^l \equiv -1 \bmod d$ et $d \in \mathcal{N}_q$. □

Corollaire 4.7.16 *Si $n \geq 1$ et $n \notin \mathcal{N}_q$, alors $nk \notin \mathcal{N}_q$ pour tout $k \in \mathbb{N}^*$.*

Preuve. C'est une conséquence directe de la propriété 4.7.15. □

Nous voyons que l'ensemble des n n'appartenant pas à $\mathcal{N}_q$ doit être assez grand : il contient tous les n dont $\mathrm{ord}_n(q)$ est impair, et tous les multiples de chacun de ses éléments.

4.7.3 Trois propositions importantes

Nous avons remarqué que si $a, b \in \mathcal{N}_q$, parfois leur produit ab appartient à $\mathcal{N}_q$, parfois non. De quoi cela depend-il ? Nous répondrons à cette question dans la section 4.7.5 où nous divisons $\mathcal{N}_q$ en classes d'équivalence satisfaisant certaines propriétés. Cette division repose sur les trois propositions suivantes.

Rappelons que $2^\alpha \parallel a$ signifie que 2^α est la plus grande puissance de 2 divisant a, i.e. $2^\alpha \mid a$ et $2^{\alpha+1} \nmid a$.

Soit $q \geq 2$, soit n un entier positif non nul, $(n, q) = 1$.

Proposition 4.7.17 *Soit* $\alpha \in \mathbb{N}$ *tel que* $2^\alpha \parallel \mathrm{ord}_n(q)$. *Si* $n \in \mathcal{N}_q$, *alors* $d \in \mathcal{N}_q$ *et* $2^\alpha \parallel \mathrm{ord}_d(q)$ *pour tout* d, $d \mid n$, $d > 2$.

Preuve. Si $n \in \mathcal{N}_q$, alors $d \in \mathcal{N}_q$ pour tout d, $d \mid n$ (cf. propriété 4.7.15). Soit $n \in \mathcal{N}_q$, $d \mid n$, $d > 2$. Soit $n \mid q^l + 1$. Alors $d \mid q^l + 1$. D'après la remarque 4.7.6, il existe $i \geq 0$ et $j \geq 0$ tels que

$$l = \mathrm{ord}_n(q)/2 + i\mathrm{ord}_n(q) \quad \text{ et } \quad l = \mathrm{ord}_d(q)/2 + j\mathrm{ord}_d(q),$$

donc

$$\mathrm{ord}_n(q)(1 + 2i) = \mathrm{ord}_d(q)(1 + 2j).$$

Comme $1 + 2i$ et $1 + 2j$ sont impairs, $2^\alpha \parallel \mathrm{ord}_n(q)$ si et seulement si $2^\alpha \parallel \mathrm{ord}_d(q)$. □

Lemme 4.7.18 *Soit* $l > 0$. *Si* $n \mid q^l + 1$, $nd \mid q^{ld} + 1$ *pour tout diviseur impair* d *de* n.

Preuve. Nous avons

$$q^{ld} + 1 = \left(q^l + 1\right)\left(q^{l(d-1)} - q^{l(d-2)} + q^{l(d-3)} - \cdots - q^l + 1\right) = \left(q^l + 1\right)\sum_{i=0}^{d-1} q^{li}(-1)^i.$$

Comme $n \mid q^l + 1$, il reste à montrer que $d \mid \sum_{i=0}^{d-1} q^{li}(-1)^i$.

Mais $n \mid q^l + 1$ signifie que $d \mid q^l + 1$, donc $q^l \equiv -1 \bmod d$. D'où

$$\sum_{i=0}^{d-1} q^{li}(-1)^i \equiv \sum_{i=0}^{d-1}(-1)^i(-1)^i \equiv \sum_{i=0}^{d-1}(-1)^{2i} \equiv \sum_{i=0}^{d-1} 1 \equiv d \equiv 0 \bmod d,$$

donc $d \mid \sum_{i=0}^{d-1} q^{li}(-1)^i$. □

Proposition 4.7.19 *Soit* $\alpha \in \mathbb{N}$ *tel que* $2^\alpha \parallel \mathrm{ord}_n(q)$. *Si* $n \in \mathcal{N}_q$, *alors* $dn \in \mathcal{N}_q$ *et* $2^\alpha \parallel \mathrm{ord}_{dn}(q)$ *pour tout* d, $d \mid n$, d *impair.*

Preuve. Si $n \in \mathcal{N}_q$, alors $dn \in \mathcal{N}_q$ pour tout d, $d \mid n$, d impair (cf. lemme 4.7.18).

Soit $n \in \mathcal{N}_q$, $d \mid n$, d impair. Soit $n \mid q^l + 1$. Alors $nd \mid q^{ld} + 1$. D'après la remarque 4.7.6, il existe $i \geq 0$ et $j \geq 0$ tels que

$$l = \mathrm{ord}_n(q)/2 + i\mathrm{ord}_n(q) \quad \text{et} \quad ld = \mathrm{ord}_{nd}(q)/2 + j\mathrm{ord}_{nd}(q),$$

donc

$$\mathrm{ord}_n(q)d(1+2i) = \mathrm{ord}_{nd}(q)(1+2j).$$

Comme $d(1+2i)$ et $1+2j$ sont impairs, $2^\alpha \parallel \mathrm{ord}_n(q)$ si et seulement si $2^\alpha \parallel \mathrm{ord}_{nd}(q)$. □

Lemme 4.7.20 *Soient n_1 et n_2 dans $\mathbb{N}^*$ tels que $(n_1, q) = (n_2, q) = 1$. Alors*

$$\mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q) = \mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right).$$

Preuve. Comme

$$q^{\mathrm{ord}_{n_1}(q)} \equiv 1 \bmod n_1 \quad \text{et} \quad \mathrm{ord}_{n_1}(q) \mid \mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right),$$

nous avons

$$q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} \equiv 1 \bmod n_1.$$

De même pour n_2 :

$$q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} \equiv 1 \bmod n_2.$$

Donc

$$n_1 \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} - 1 \quad \text{et} \quad n_2 \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} - 1,$$

d'où

$$\mathrm{ppcm}\left(n_1, n_2\right) \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} - 1.$$

Donc

$$q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)} \equiv 1 \bmod \mathrm{ppcm}\left(n_1, n_2\right)$$

et par conséquent

$$\mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q) \mid \mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right).$$

De l'autre coté, soit

$$l = \frac{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)}{\mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q)}.$$

Compte tenu de la propriété 4.7.13 et du fait que $ab = \mathrm{ppcm}\left(a,b\right)\left(a,b\right) \ \forall a, b \in \mathbb{N}^*$,

$$\mathrm{ord}_{n_1}(q) \mid \mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q) = \frac{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)}{l} = \frac{\mathrm{ord}_{n_1}(q)\mathrm{ord}_{n_2}(q)}{l\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)},$$

donc

$$l\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right) \mid \mathrm{ord}_{n_2}(q).$$

De la même façon nous obtenons que

$$l\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right) \mid \mathrm{ord}_{n_1}(q).$$

Donc

$$l\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right) \mid \left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right),$$

d'où $l \mid 1$ et $l = 1$. □

Proposition 4.7.21 *Soient n_1 et n_2 dans $\mathcal{N}_q$ tels que $n_1 > 2$ et $n_2 > 2$. Soit α_1 et α_2 dans $\mathbb{N}$ tels que $2^{\alpha_1} \parallel \mathrm{ord}_{n_1}(q)$ et $2^{\alpha_2} \parallel \mathrm{ord}_{n_2}(q)$. Alors*

$$\mathrm{ppcm}\left(n_1, n_2\right) \in \mathcal{N}_q \textit{ si et seulement si } \alpha_1 = \alpha_2.$$

De plus, si c'est le cas, $2^{\alpha_1} \parallel \mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q)$.

Preuve. Soient $\mathrm{ord}_{n_1}(q) = 2^{\alpha_1}m_1$, m_1 impair, et $\mathrm{ord}_{n_2}(q) = 2^{\alpha_2}m_2$, m_2 impair. Alors

$$\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right) = \mathrm{ppcm}\left(2^{\alpha_1-1}m_1, 2^{\alpha_2-1}m_2\right) = \frac{2^{\alpha_1-1}m_1 2^{\alpha_2-1}m_2}{\left(2^{\alpha_1-1}m_1, 2^{\alpha_2-1}m_2\right)}.$$

Soit $\alpha_1 = \alpha_2$. Alors

$$\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right) = \frac{2^{\alpha_1-1}m_1 m_2}{(m_1, m_2)} = \frac{\mathrm{ord}_{n_1}(q)}{2} \cdot \frac{m_2}{(m_1, m_2)}.$$

Comme

$$q^{\mathrm{ord}_{n_1}(q)/2} \equiv -1 \bmod n_1$$

et $\frac{m_2}{(m_1,m_2)}$ est impair, nous obtenons

$$q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1 = q^{\frac{\mathrm{ord}_{n_1}(q)}{2} \cdot \frac{m_2}{(m_1,m_2)}} + 1 \equiv (-1)^{\frac{m_2}{(m_1,m_2)}} + 1 \equiv -1 + 1 \equiv 0 \bmod n_1,$$

d'où

$$n_1 \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1.$$

De même pour n_2 :

$$n_2 \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1.$$

Donc

$$\mathrm{ppcm}\left(n_1, n_2\right) \mid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1$$

ce qui implique $\mathrm{ppcm}\left(n_1, n_2\right) \in \mathcal{N}_q$.

D'après le lemme 4.7.20,

$$\begin{aligned}
\mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q) &= \mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right) \\
&= \mathrm{ppcm}\left(2^{\alpha_1}m_1, 2^{\alpha_2}m_2\right) \\
&= 2^{\alpha_1}\mathrm{ppcm}\left(m_1, m_2\right),
\end{aligned}$$

donc $2^{\alpha_1} \parallel \mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q)$.

De l'autre coté, soit $\alpha_1 \neq \alpha_2$. Prenons, par exemple, le cas $\alpha_1 > \alpha_2$ (l'autre cas se montre de façon analogue). Alors

$$\begin{aligned}
\mathrm{ppcm}\,(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2) &= \frac{2^{\alpha_1-1}m_1m_2}{(2^{\alpha_1-\alpha_2-1}m_1, m_2)} \\
&= \frac{2^{\alpha_1-\alpha_2}m_1 2^{\alpha_2-1}m_2}{(m_1, m_2)} \\
&= \frac{\mathrm{ord}_{n_2}(q)}{2} \cdot \frac{2^{\alpha_1-\alpha_2}m_1}{(m_1, m_2)}.
\end{aligned}$$

Comme

$$q^{\mathrm{ord}_{n_2}(q)/2} \equiv -1 \bmod n_2$$

et $\frac{2^{\alpha_1-\alpha_2}m_1}{(m_1,m_2)}$ est pair, nous obtenons

$$q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1 = q^{\frac{\mathrm{ord}_{n_2}(q)}{2}\cdot\frac{2^{\alpha_1-\alpha_2}m_1}{(m_1,m_2)}} + 1 \equiv (-1)^{\frac{2^{\alpha_1-\alpha_2}m_1}{(m_1,m_2)}} + 1 \equiv 1 + 1 \not\equiv 0 \bmod n_2,$$

car $n_2 > 2$. Donc

$$n_2 \nmid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1,$$

d'où

$$\mathrm{ppcm}\,(n_1, n_2) \nmid q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1.$$

Si $\mathrm{ppcm}\,(n_1, n_2) \in \mathcal{N}_q$, d'après la proposition 4.7.7 et le lemme 4.7.20, nous avons

$$\begin{aligned}
\mathrm{ppcm}\,(n_1, n_2) \mid q^{\mathrm{ord}_{\mathrm{ppcm}(n_1,n_2)}(q)/2} + 1 &= q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q), \mathrm{ord}_{n_2}(q)\right)/2} + 1 \\
&= q^{\mathrm{ppcm}\left(\mathrm{ord}_{n_1}(q)/2, \mathrm{ord}_{n_2}(q)/2\right)} + 1,
\end{aligned}$$

mais nous venons de montrer le contraire, donc $\mathrm{ppcm}\,(n_1, n_2) \notin \mathcal{N}_q$. □

Remarque 4.7.22 En résumé, la situation est la suivante :

- Soit n un nombre premier impair et $i > 0$. Si $n^i \in \mathcal{N}_q$, alors $n^j \in \mathcal{N}_q$ pour tout $j \geq 0$. En effet, $n^j \in \mathcal{N}_q$ pour tout j, $0 \leq j \leq i$, grâce à la proposition 4.7.17, et $n^j \in \mathcal{N}_q$ pour tout j, $j > i$, grâce à l'application répétée de la proposition 4.7.19. De plus, si $\alpha \geq 0$ est tel que $2^\alpha \parallel \mathrm{ord}_{n^i}(q)$, alors $2^\alpha \parallel \mathrm{ord}_{n^j}(q)$ pour tout $j \geq 0$ (grâce aux mêmes propositions).

- Soient n_1 et n_2 des nombres premiers impairs distincts, $i_1 > 0$, $i_2 > 0$, $2^{\alpha_1} \parallel \mathrm{ord}_{n_1^{i_1}}(q)$, $2^{\alpha_2} \parallel \mathrm{ord}_{n_2^{i_2}}(q)$, $n_1^{i_1} \in \mathcal{N}_q$ et $n_2^{i_2} \in \mathcal{N}_q$. Alors $n_1^{i_1}n_2^{i_2} \in \mathcal{N}_q$ si et seulement si $\alpha_1 = \alpha_2$ (cf. proposition 4.7.21).

  Si c'est le cas, $2^{\alpha_1} \parallel \mathrm{ord}_{n_1^{i_1}n_2^{i_2}}(q)$ et, de plus, $n_1^{j_1}n_2^{j_2} \in \mathcal{N}_q$ et $2^{\alpha_1} \parallel \mathrm{ord}_{n_1^{j_1}n_2^{j_2}}(q)$ pour tous les $j_1 \geq 0$ et $j_2 \geq 0$ (cela se montre comme ci-dessus).

– Cela se généralise facilement : soient $n_1, \ldots, n_l$ des nombres premiers impairs distincts, $i_k > 0$, $2^{\alpha_k} \parallel \mathrm{ord}_{n_k^{i_k}}(q)$ et $n_k^{i_k} \in \mathcal{N}_q$ pour tout k, $1 \leq k \leq l$. Alors $\prod_{k=1}^{l} n_k^{i_k} \in \mathcal{N}_q$ si et seulement si $\alpha_1 = \alpha_2 = \alpha_3 = \cdots = \alpha_l$. En effet, il suffit de répéter l'application de la proposition 4.7.21.

  Si c'est le cas, $2^{\alpha_1} \parallel \mathrm{ord}_{\prod_{k=1}^{l} n_k^{i_k}}(q)$ et, de plus, $\prod_{k=1}^{l} n_k^{j_k} \in \mathcal{N}_q$ et $2^{\alpha_1} \parallel \mathrm{ord}_{\prod_{k=1}^{l} n_k^{j_k}}(q)$ pour tous les $j_1 \geq 0, \ldots, j_l \geq 0$. □

Mais que se passe-t-il si $n = 2$? C'est le sujet de la section suivante.

4.7.4 Le cas pair

Dans cette section nous étudions l'appartenance des nombres pairs à $\mathcal{N}_q$. Nous commençons par l'étude des puissances de 2. Évidemment, si q est pair, $2^i \notin \mathcal{N}_q$ pour tout $i \geq 1$ (cf. remarque 4.7.4). Supposons que q soit impair.

Nous allons utiliser le lemme suivant.

Lemme 4.7.23 *[IR82, p. 43] Soit $i > 2$. Alors $(\mathbb{Z}/2^i\mathbb{Z})^*$ n'est pas cyclique ; il est produit de deux groupes cycliques, l'un d'ordre 2 engendré par -1, l'autre d'ordre 2^{i-2} engendré par 5.*

Preuve. Soit $i > 2$. D'abord nous montrerons que pour tout $t \geq 0$,

$$5^{2^t} = 1 + 2^{t+2}h_t \text{ avec } h_t \text{ impair.} \tag{4.22}$$

En effet, si $t = 0$, l'égalité devient $5 = 1 + 4h_t$ avec $h_t = 1$ impair. Supposons qu'elle soit vraie pour $t - 1$. Alors

$$5^{2^t} = \left(5^{2^{t-1}}\right)^2 = \left(1 + 2^{t+1}h_{t-1}\right)^2 = 1 + 2^{t+2}h_{t-1} + 2^{2t+2}h_{t-1}^2 = 1 + 2^{t+2}h_{t-1}\left(1 + 2^t h_{t-1}\right)$$

avec $h_t \overset{\text{déf}}{=} h_{t-1}\left(1 + 2^t h_{t-1}\right)$ impair, car h_{t-1} est impair et $t \geq 1$.

L'ordre de 5 dans $(\mathbb{Z}/2^i\mathbb{Z})^*$ divise $\phi(2^i) = 2^{i-1}$ (d'après le théorème 4.2.10 et la proposition 4.2.16, p. 123), c'est donc une puissance de 2 ; d'après (4.22)

$$5^{2^{i-3}} = 1 + 2^{i-1}h_{i-3} \not\equiv 1 \bmod 2^i$$

et

$$5^{2^{i-2}} = 1 + 2^i h_{i-2} \equiv 1 \bmod 2^i,$$

d'où il s'ensuit que 2^{i-2} est l'ordre de 5.

Par conséquent le cardinal du sous-groupe de $(\mathbb{Z}/2^i\mathbb{Z})^*$ engendré par 5 est 2^{i-2}. D'ailleurs, $5 \equiv 1 \bmod 4$, et donc toutes les puissances de 5 sont congrues à 1 modulo 4. Il y a $\phi(2^i) = 2^{i-1}$ éléments dans $(\mathbb{Z}/2^i\mathbb{Z})^*$, et une moitié, i.e. 2^{i-2}, d'entre eux sont congrus à 1 modulo 4. Donc le sous-groupe engendré par 5 est constitué de tous les éléments de $(\mathbb{Z}/2^i\mathbb{Z})^*$ congrus à 1 modulo 4.

Par ailleurs, -1 (qui est d'ordre 2) n'appartient pas à ce sous-groupe. Nous pouvons donc écrire :

$$\begin{array}{rcl} \text{si } q & \equiv & 1 \bmod 4, \quad q \equiv 5^j \equiv (-1)^{(q-1)/2} 5^j \bmod 2^i, \quad \text{et} \\ \text{si } q & \equiv & 3 \bmod 4, \quad -q \equiv 5^k \quad \text{d'où} \quad q \equiv (-1)^{(q-1)/2} 5^k \bmod 2^i. \end{array}$$

□

Proposition 4.7.24 *Soit q un entier positif impair. Soit $i \geq 2$. Soit $\beta \geq 0$ tel que $2^\beta \parallel q+1$. Alors $2^i \in \mathcal{N}_q$ si et seulement si $i \leq \beta$. De plus, si $2^i \in \mathcal{N}_q$, alors $2 \parallel \mathrm{ord}_{2^i}(q)$.*

Preuve. D'après la définition de $\mathcal{N}_q$, $2^i \in \mathcal{N}_q$ si et seulement s'il existe $l > 0$ tel que $q^l \equiv -1 \bmod 2^i$ (cf. notation 4.7.1). Quelles conditions doivent satisfaire q et i pour qu'une puissance de q soit égale à -1 dans $(\mathbb{Z}/2^i\mathbb{Z})^*$?

Soit $i > 2$. D'après le lemme 4.7.23, il existe $j \geq 0$ et $k \geq 0$ tels que $q \equiv (-1)^j 5^k \bmod 2^i$, donc $q^l \equiv (-1)^{jl} 5^{kl} \bmod 2^i$.

De l'autre coté, $-1 \in (\mathbb{Z}/2^i\mathbb{Z})^*$ s'exprime aussi comme une puissance de -1 et 5 de façon suivante : $-1 \equiv (-1)^t 5^s \bmod 2^i$ si et seulement si t est impair et l'ordre de 5 modulo 2^i divise s, i.e. $2^{i-2} \mid s$, car nous avons vu dans la démonstration du lemme 4.7.23 que l'ordre de 5 modulo 2^i est 2^{i-2}.

Donc $q^l \equiv -1 \bmod 2^i$ si et seulement si jl est impair et $2^{i-2} \mid kl$. Supposons que ce soit le cas. Comme jl est impair, l est impair aussi, donc $2^{i-2} \mid k$. Soit $k = 2^{i-2}u$, $u \geq 0$. Alors $5^k = \left(5^{2^{i-2}}\right)^u \equiv 1 \bmod 2^i$, car 2^{i-2} est l'ordre de 5 dans $(\mathbb{Z}/2^i\mathbb{Z})^*$. Donc $q \equiv (-1)^j 5^k \equiv -1 \bmod 2^i$.

Nous avons donc montré que si $i > 2$ et $2^i \in \mathcal{N}_q$, alors $q \equiv -1 \bmod 2^i$, i.e. $2^i \mid q+1$, d'où $i \leq \beta$.

Dans le cas où $i = 2$, nous procédons de la même façon : q est impair, par conséquent $q \in (\mathbb{Z}/4\mathbb{Z})^* = < -1 >$. Soit $k \geq 0$ tel que $q \equiv (-1)^k \bmod 4$. Alors $q^l \equiv (-1)^{kl} \bmod 4$. Considérons l'équation $q^l \equiv -1 \bmod 4$. Elle devient $(-1)^{kl} \equiv -1 \bmod 4$, donc elle est satisfaite si et seulement si kl est impair. Si c'est le cas, k est impair aussi, par conséquent $q \equiv (-1)^k \equiv -1 \bmod 4$, d'où $4 \mid q+1$ et donc $i = 2 \leq \beta$.

La réciproque est évidente, c'est-à-dire si $2^i \mid q+1$, alors $q \equiv -1 \bmod 2^i$ et donc $2^i \in \mathcal{N}_q$ (d'après la définition de $\mathcal{N}_q$). De plus, si $n > 2$, le plus petit entier positif non nul r tel que $q^r \equiv -1 \bmod n$ est égal à $\mathrm{ord}_n(q)/2$ (cf. proposition 4.7.5). Dans notre cas $n = 2^i$ et $r = 1$, donc $\mathrm{ord}_{2^i}(q) = 2$ et $2 \parallel \mathrm{ord}_{2^i}(q)$. □

Remarque 4.7.25 Nous faisons le point.

2 appartient à $\mathcal{N}_q$ si et seulement si q est impair. Si c'est le cas, $2^0 \parallel \mathrm{ord}_2(q) = 1$.

Si $i \geq 2$, $2^i \in \mathcal{N}_q$ si et seulement si $2^i \mid q+1$, et si c'est le cas, $2 \parallel \mathrm{ord}_{2^i}(q) = 2$.

Il nous reste à étudier le produit d'une puissance 2^i de 2 et d'un nombre impair $t > 1$.

Soit d'abord $i \geq 2$. Est-ce que $2^i t \in \mathcal{N}_q$?

Évidemment, si 2^i ou t n'appartient pas à $\mathcal{N}_q$, leur produit $2^i t$ n'appartient pas non plus à $\mathcal{N}_q$ (à cause du corollaire 4.7.16).

Supposons que $2^i \in \mathcal{N}_q$ et $t \in \mathcal{N}_q$. D'après la proposition 4.7.21, ppcm $(2^i, t) = 2^i t \in \mathcal{N}_q$ si et seulement si $\alpha_1 = \alpha_2$, où $2^{\alpha_1} \parallel \mathrm{ord}_{2^i}(q)$, $2^{\alpha_2} \parallel \mathrm{ord}_t(q)$. Mais nous venons de voir (cf. proposition 4.7.24) que $2 \parallel \mathrm{ord}_{2^i}(q)$, donc $\alpha_1 = 1$. Nous obtenons donc que $2^i t \in \mathcal{N}_q$ si et seulement si $2 \parallel \mathrm{ord}_t(q)$. Et si c'est le cas, $2 \parallel \mathrm{ord}_{2^i t}(q)$.

Il nous reste le cas $i = 1$. Soit $t \in \mathcal{N}_q$, $t > 1$. Est-ce que $2t \in \mathcal{N}_q$?

D'après la remarque 4.7.4, si q est pair, $2t \notin \mathcal{N}_q$. Soit maintenant q impair. Soit $t \mid q^l + 1$, $l > 0$. Comme t est impair et q^l+1 est pair, nous obtenons que $2t \mid q^l+1$, donc $2t \in \mathcal{N}_q$. De plus, comme $t \mid q^{\mathrm{ord}_t(q)/2} + 1$ (cf. proposition 4.7.7), $2t \mid q^{\mathrm{ord}_t(q)/2} + 1$ et donc (cf. remarque 4.7.6)

$$\mathrm{ord}_t(q)/2 = \mathrm{ord}_{2t}(q)/2 + s\,\mathrm{ord}_{2t}(q) \text{ pour un } s \geq 0.$$

D'où $\mathrm{ord}_t(q) = \mathrm{ord}_{2t}(q)(1+2s)$, donc $\mathrm{ord}_{2t}(q) \mid \mathrm{ord}_t(q)$. Mais $\mathrm{ord}_t(q) \mid \mathrm{ord}_{2t}(q)$ (cf. propriété 4.7.13), donc $\mathrm{ord}_{2t}(q) = \mathrm{ord}_t(q)$.

En résumé, $t \in \mathcal{N}_q$ si et seulement si $2t \in \mathcal{N}_q$, et si c'est le cas, $2^\alpha \parallel \mathrm{ord}_{2t}(q)$ où $\alpha \geq 0$ est tel que $2^\alpha \parallel \mathrm{ord}_t(q)$. □

4.7.5 Classes d'équivalence

Les remarques 4.7.22 et 4.7.25 résument les résultats des sections 4.7.3 et 4.7.4. Nous voyons qu'il est possible d'introduire dans $\mathcal{N}_q$ la relation d'équivalence suivante : $n_1 \sim n_2$ si et seulement si $\alpha_1 = \alpha_2$, où $2^{\alpha_1} \parallel \mathrm{ord}_{n_1}(q)$ et $2^{\alpha_2} \parallel \mathrm{ord}_{n_2}(q)$.

Notation 4.7.26 Notons $\mathcal{P}_{q,\alpha}$, $\alpha \geq 0$, les classes de cette relation d'équivalence :

$$\mathcal{P}_{q,\alpha} = \{n \in \mathcal{N}_q : 2^\alpha \parallel \mathrm{ord}_n(q)\}.$$

Résumons les propriétés de $\mathcal{P}_{q,\alpha}$.

Proposition 4.7.27 *Soit $\beta \geq 0$ tel que $2^\beta \parallel q + 1$.*

(a) $\mathcal{P}_{q,0} = \{i \in \{1, 2\} \mid (i, q) = 1\}$.

(b) $n \in \mathcal{P}_{q,1}$ si et seulement si n est de la forme $2^\gamma s_1^{i_1} \cdots s_r^{i_r}$, où s_k est un nombre premier impair, $s_k \in \mathcal{P}_{q,1}$, $i_k > 0$ pour tout k, $1 \leq k \leq r$, et soit $r \geq 1$ et $0 \leq \gamma \leq \beta$, soit $r = 0$ et $2 \leq \gamma \leq \beta$.

(c) Soit $\alpha \geq 2$. Alors $n \in \mathcal{P}_{q,\alpha}$ si et seulement si n est de la forme $2^\gamma s_1^{i_1} \cdots s_r^{i_r}$, où $r \geq 1$, s_k est un nombre premier impair, $s_k \in \mathcal{P}_{q,\alpha}$, $i_k > 0$ pour tout k, $1 \leq k \leq r$, et $0 \leq \gamma \leq 1$.

Preuve. C'est une conséquence directe des remarques 4.2.12, p. 123, 4.7.22 et 4.7.25. □

Exemple 4.7.28 Soit $q = 2$. Alors tous les n impairs, $n \leq 100$, se divisent comme suit :

$$\begin{aligned}
\mathcal{P}_{2,0} &= \{1\},\\
\mathcal{P}_{2,1} &\supset \{3, 9, 11, 19, 27, 33, 43, 57, 59, 67, 81, 83, 99\},\\
\mathcal{P}_{2,2} &\supset \{5, 13, 25, 29, 37, 41, 53, 61, 65\},\\
\mathcal{P}_{2,3} &\supset \{17\},\\
\mathcal{P}_{2,4} &\supset \{97\},\\
\mathbb{N} \setminus \mathcal{N}_q &\supset \{7, 15, 21, 23, 31, 35, 39, 45, 47, 49, 51, 55, 63,\\
&\quad\ 69, 71, 73, 75, 77, 79, 85, 87, 89, 91, 93, 95\}.
\end{aligned}$$

En particulier, les nombres premiers “engendrant” $\mathcal{P}_{2,\alpha}$ (i.e. tout $n \in \mathcal{P}_{2,\alpha}$ est un produit de puissances de ces nombres premiers) sont

$$\begin{aligned}
\mathcal{P}_{2,1} &\supset \{3, 11, 19, 43, 59, 67, 83\},\\
\mathcal{P}_{2,2} &\supset \{5, 13, 29, 37, 41, 53, 61\},\\
\mathcal{P}_{2,3} &\supset \{17\},\\
\mathcal{P}_{2,4} &\supset \{97\},\\
\mathbb{N} \setminus \mathcal{N}_q &\supset \{7, 23, 31, 47, 71, 73, 79, 89\}.
\end{aligned}$$

Nous voyons que le nombre de nombres premiers n'est pas si différent dans $\mathcal{P}_{2,1}$, $\mathcal{P}_{2,2}$ et $\mathbb{N} \setminus \mathcal{N}_q$, tandis que le nombre d'éléments de $\mathbb{N} \setminus \mathcal{N}_q$ augmente bien plus vite que le nombre d'éléments de $\mathcal{P}_{2,1}$ ou $\mathcal{P}_{2,2}$. C'est dû au fait que pour qu'un nombre appartienne à $\mathcal{P}_{q,\alpha}$, tous ses diviseurs premiers impairs doivent appartenir à $\mathcal{P}_{q,\alpha}$. Cela nous a conduit à la conjecture 4.9.3. □

Nous présentons une minoration des éléments de ces classes d'équivalence.

Propriété 4.7.29 *Si $n \in \mathcal{P}_{q,\alpha}$, $\alpha \geq 1$, alors $n \geq 2^\alpha + 1$.*

Preuve. Soit t le plus petit élément de $\mathcal{P}_{q,\alpha}$. D'après la proposition 4.7.27, c'est un nombre premier impair.

Compte tenu de la définition 4.2.11, du théorème 4.2.10 et de la proposition 4.2.16, p. 123, $\mathrm{ord}_t(q) \mid \phi(t) = t - 1$. D'après la définition de $\mathcal{P}_{q,\alpha}$, $2^\alpha \parallel \mathrm{ord}_t(q) \mid t - 1$, donc $2^\alpha \leq t - 1$, ce qui signifie que $2^\alpha + 1 \leq t \leq n$. □

4.7.6 La décomposition d'un entier

Soit $q \geq 2$. La partition de $\mathcal{N}_q$ en classes $\mathcal{P}_{q,\alpha}$, $\alpha \geq 0$, induit la décomposition d'un entier positif non nul n, présentée dans la proposition 4.7.32.

Définition 4.7.30 *Soit $n \in \mathbb{N}^*$, $(n, q) = 1$. Un* diviseur $\mathcal{P}_{q,\alpha}$-maximal de n, *$\alpha \geq 0$, est un diviseur m_α de n tel que soit $m_\alpha = 1$, soit $m_\alpha \in \mathcal{P}_{q,\alpha}$ et $m_\alpha d \notin \mathcal{P}_{q,\alpha}$ pour tout d, $d \mid \frac{n}{m_\alpha}$.*

Notez que la condition "$m_\alpha d \notin \mathcal{P}_{q,\alpha}$ pour tout d, $d \mid \frac{n}{m_\alpha}$" est équivalente à "$d \notin \mathcal{P}_{q,\alpha}$ pour tout d, $d \mid n$, tel que $m_\alpha \mid d$".

Proposition 4.7.31 *Soit $n \in \mathbb{N}^*$, $(n,q)=1$. Soit $\gamma_q \geq 0$ tel que $2^{\gamma_q} \parallel q+1$. Soit $\alpha \geq 0$. Un diviseur $\mathcal{P}_{q,\alpha}$-maximal m_α de n existe et est unique. Si $n = 2^\beta \prod_{k=1}^r s_k^{i_k}$ est la décomposition de n en facteurs premiers, alors :*

- *si $\alpha = 0$, $m_0 = 2^{\min(\beta,1)}$,*
- *si $\alpha = 1$, $m_1 = 2^{\min(\beta,\gamma_q)} \prod_{k\in K_1} s_k^{i_k}$,*
- *si $\alpha \geq 2$, $m_\alpha = 2^{\min(\beta,1)} \prod_{k\in K_\alpha} s_k^{i_k}$,*

où $K_\alpha \subset \{1,\ldots,r\}$, $\alpha \geq 0$, est l'ensemble d'indices k tels que $s_k \in \mathcal{P}_{q,\alpha}$ (par convention, si K_α est vide, le produit $\prod_{k\in K_\alpha} s_k^{i_k}$ est égal à 1).

En particulier, si n est impair, $m_\alpha = \prod_{k\in K_\alpha} s_k^{i_k}$ pour tout $\alpha \geq 0$.

Preuve. Soit $\alpha = 0$. Nous savons que $\mathcal{P}_{q,0} = \{i \in \{1,2\} \mid (i,q) = 1\}$ (cf. proposition 4.7.27(a)). Si n est pair, alors q est impair et $m_0 = 2$, car, d'après la définition de m_0, soit $m_0 = 1$, soit $m_0 \in \mathcal{P}_{q,0} = \{1,2\}$. Si $m_0 = 1$, alors $2m_0 \in \mathcal{P}_{q,\alpha}$ pour $2 \mid n$, ce qui contredit la définition de m_0, donc $m_0 = 2$. Si n est impair, $m_0 = 1$, car l'autre possible candidat $2 \in \mathcal{P}_{q,0}$ n'est pas un diviseur de n. Donc en général $m_0 = 2^{\min(\beta,1)}$.

Soit $\alpha = 1$. Alors $\prod_{k\in K_1} s_k^{i_k} \mid m_1$, car si, par exemple, $s_u^{i_u} \nmid m_1$ pour un $u \in K_1$, alors $m_1 s_u \in \mathcal{P}_{q,1}$, car $m_1 \in \mathcal{P}_{q,1}$ et $s_u \in \mathcal{P}_{q,1}$, et $s_u \mid \frac{n}{m_1}$, ce qui contredit la définition de m_1. D'ailleurs, si $\beta \leq \gamma_q$, $2^\beta \in \mathcal{P}_{q,1}$ (cf. proposition 4.7.27(b)), et de la même façon on montre que $2^\beta \mid m_1$. Si $\beta \geq \gamma_q$, 2^{γ_q} est la puissance maximale de 2 appartenant à $\mathcal{P}_{q,1}$, donc de la même façon on montre que $2^{\gamma_q} \mid m_1$. Ces deux cas s'écrivent $2^{\min(\beta,\gamma_q)} \mid m_1$.

De l'autre coté, m_1 ne peut pas avoir de diviseur n'appartenant pas à $\mathcal{P}_{q,1}$ à cause de la proposition 4.7.17. Donc $m_1 = 2^{\min(\beta,\gamma q)} \prod_{k\in K_1} s_k^{i_k}$.

Si $\alpha \geq 2$, la démonstration est analogue. Nous tenons compte du fait que la puissance maximale 2^j de 2 divisant un élément de $\mathcal{P}_{q,\alpha}$ ne peut pas excéder $j = 1$, donc nous remplaçons γ_q par 1. □

Proposition 4.7.32 *Soit $n \in \mathbb{N}^*$, $(n,q)=1$. Il est possible d'exprimer n de façon unique comme suit :*

$$n = 2^\beta d' d_1 d_2 d_3 \cdots,$$

où $n/2^\beta$ est impair, d_α, $\alpha \geq 1$, est le diviseur $\mathcal{P}_{q,\alpha}$-maximal de $n/2^\beta$, et $d' = \frac{n}{2^\beta d_1 d_2 d_3 \cdots}$.

Preuve. Soit $\beta \geq 0$ tel que $2^\beta \parallel n$. Alors $n/2^\beta$ est impair.

Si $n = 2^\beta \prod_{k=1}^r s_k^{i_k}$ est la décomposition de n en facteurs premiers, il est possible de diviser l'ensemble $\{1,\ldots,r\}$ en classes $u', u_1, u_2, u_3, \ldots$ telles que u_α, $\alpha \geq 1$, contient les indices des s_k appartenant à $\mathcal{P}_{q,\alpha}$, et u' contient les indices des s_k n'appartenant pas à

$\mathcal{N}_q$. Si nous notons $d' = \prod_{k\in u'} s_k^{i_k}$, nous obtenons que $n = 2^\beta d' d_1 d_2 d_3 \cdots$, car, d'après la proposition 4.7.31, $d_\alpha = \prod_{k\in u_\alpha} s_k^{i_k}$, et donc

$$2^\beta d' d_1 d_2 d_3 \cdots = 2^\beta \prod_{k\in u'\cup u_1\cup u_2\cup\cdots} s_k^{i_k} = 2^\beta \prod_{k=1}^{r} s_k^{i_k} = n. \qquad \square$$

Remarque 4.7.33 D'après la démonstration de la proposition 4.7.32, si $n = 2^\beta \prod_{k=1}^{r} s_k^{i_k}$ est la décomposition de n en facteurs premiers, alors $d' = \prod_{k\in u'} s_k^{i_k}$, où $u' \subset \{1,\ldots,r\}$ est l'ensemble des indices k tels que $s_k \notin \mathcal{N}_q$.

Notation 4.7.34 Notons $\mathcal{P}_{q,\infty}$ l'ensemble des entiers impairs d' tels que chaque diviseur premier t de d' n'appartient pas à $\mathcal{N}_q$.

Remarque 4.7.35 Évidemment, comme n est fini, à partir d'un certain indice les nombres d_α deviennent égaux à 1, i.e. $\exists v \geq 0$ tel que $d_\alpha = 1\ \forall\alpha,\ \alpha \geq v$.

Remarque 4.7.36 Si $n = 2^\beta d' d_1 d_2 d_3 \cdots$ est la décomposition de n comme dans la proposition 4.7.32, nous obtenons, compte tenu de la proposition 4.7.31, que le diviseur $\mathcal{P}_{q,\alpha}$-maximal m_α de n est :

- si $\alpha = 0$, $m_0 = 2^{\min(\beta,1)}$,
- si $\alpha = 1$, $m_1 = 2^{\min(\beta,\gamma_q)} d_1$,
- si $\alpha \geq 2$, $m_\alpha = 2^{\min(\beta,1)} d_\alpha$.

En particulier, si $\beta = 0$, $m_\alpha = d_\alpha$ pour tout $\alpha \geq 0$ (nous notons $d_0 = 1$).
De même, si $\beta \geq 1$, $m_0 = 2$, $m_1 = 2^{\min(\beta,\gamma_q)} d_1$ et $m_\alpha = 2d_\alpha$ pour tout $\alpha \geq 2$. $\square$

Notation 4.7.37 Comme la décomposition est unique, pour chaque entier n les β, d' et d_i, $i \geq 1$, sont définis et uniques. Nous pouvons donc les considérer comme des fonctions de n, d'où la notation $\beta(n)$, $d'(n)$, $d_i(n)$, $i \geq 1$.

Remarque 4.7.38 Soit $\gamma_q \geq 0$ tel que $2^{\gamma_q} \parallel q+1$. Soit n un entier premier avec q.

D'après la proposition 4.7.27, $n = 2^\beta d' d_1 d_2 d_3 \cdots$ appartient à $\mathcal{P}_{q,\alpha}$, $\alpha \geq 2$, si et seulement s'il est de la forme $n = 2^\beta d_\alpha$, où $0 \leq \beta \leq 1$ et $d_\alpha > 1$. De même, il appartient à $\mathcal{P}_{q,1}$ si et seulement s'il est de la forme $n = 2^\beta d_1$, où soit $d_1 = 1$ et $2 \leq \beta \leq \gamma_q$, soit $d_1 > 1$ et $0 \leq \beta \leq \gamma_q$. Et il appartient à $\mathcal{P}_{q,0}$ si et seulement s'il est égal à 1 ou 2.

Donc, $n = 2^\beta d' d_1 d_2 d_3 \cdots$ n'appartient pas à $\mathcal{N}_q$ si et seulement si soit $\beta > \gamma_q$, soit $d' > 1$, soit $\beta \geq 2$ et $d_\alpha > 1$ pour un certain $\alpha \leq 2$, soit $d_{\alpha_1} > 1$ et $d_{\alpha_2} > 1$ pour deux distincts $\alpha_1 \geq 1$ et $\alpha_2 \geq 1$. $\square$

Remarque 4.7.39 Il est clair que pour tout $\alpha \geq 1$, si $d_\alpha > 1$, alors $d_\alpha \geq 3$ (parce que c'est un nombre impair). De même pour d' : si $d' > 1$, alors $d' \geq 3$.

4.8 La dimension moyenne du hull – suite

Soit n un entier positif non nul et soit q une puissance d'un nombre premier p.

Dans cette section nous utilisons les résultats de la section 4.7 pour étudier le comportement (en particulier, le comportement asymptotique quand n tend vers l'infini) de $Edimh_q(n)$.

4.8.1 L'étude de $B_{n,q}$ – suite

Soit n et q premiers entre eux.

Dans la proposition 4.6.14, p. 154, nous avons montré que

$$B_{n,q} = \sum_{\substack{l|n \\ \exists i>0\,:\,q^i\equiv -1 \bmod l}} \phi(l).$$

En utilisant la notation 4.7.1, p. 155, nous pouvons écrire

$$B_{n,q} = \sum_{\substack{l|n \\ l\in\mathcal{N}_q}} \phi(l).$$

Nous explicitons cette expression de $B_{n,q}$ en utilisant la proposition 4.7.32 :

Proposition 4.8.1 *Soit n un entier positif non nul et soit q une puissance d'un nombre premier p. Soit $(n,q)=1$. Soit $n = 2^\beta d' d_1 d_2 d_3 \cdots$ la décomposition de n comme dans la proposition 4.7.32. Soit $\gamma_q \geq 0$ tel que $2^{\gamma_q} \parallel q+1$. Alors*

$$B_{n,q} = 2^{\min(\gamma_q,\beta)} d_1 + 2^{\min(\beta,1)} \sum_{\alpha\geq 2}(d_\alpha - 1). \tag{4.23}$$

En particulier, si n est impair, alors

$$B_{n,q} = d_1 + \sum_{\alpha\geq 2}(d_\alpha - 1) = 1 + \sum_{\alpha\geq 1}(d_\alpha - 1).$$

Preuve. Nous avons $B_{n,q} = \sum_{\substack{l|n \\ l\in\mathcal{N}_q}} \phi(l)$. Quels sont les diviseurs l de n appartenant à $\mathcal{N}_q$? Exactement ceux qui divisent un des diviseurs maximaux m_α de n appartenant à $\mathcal{P}_{q,\alpha}$, car si $l \in \mathcal{N}_q$, l appartient à un des $\mathcal{P}_{q,\alpha}$, donc c'est un produit de puissances de diviseurs premiers de n appartenant à $\mathcal{P}_{q,\alpha}$ et d'une puissance correspondante de 2.

Supposons d'abord que $\beta = 0$. Compte tenu de la remarque 4.7.36, $m_\alpha = d_\alpha$, nous avons donc

$$B_{n,q} = \sum_{\substack{l|n \\ l\in\mathcal{N}_q}} \phi(l) = \phi(1) + \sum_{\alpha\geq 1} \sum_{\substack{d|d_\alpha \\ d\neq 1}} \phi(d), \tag{4.24}$$

où, comme 1 est un diviseur de tout d_α, pour ne pas l'inclure dans la somme plusieurs fois, nous le mettons à coté (notez que la somme $\sum_{\alpha\geq 1}\cdots$ peut être considérée comme une somme finie à cause de la remarque 4.7.35). Les d_α sont premiers entre eux, donc nous ne risquons pas d'inclure dans la somme un diviseur $d \neq 1$ plusieurs fois.

Mais $\sum_{d|d_\alpha} \phi(d) = d_\alpha$ (cf. propriété 4.2.9(v)), donc

$$\sum_{\substack{d|d_\alpha \\ d\neq 1}} \phi(d) = \sum_{d|d_\alpha} \phi(d) - \phi(1) = d_\alpha - 1.$$

En substituant cela dans (4.24), nous obtenons

$$B_{n,q} = 1 + \sum_{\alpha\geq 1}(d_\alpha - 1) = d_1 + \sum_{\alpha\geq 2}(d_\alpha - 1).$$

Soit maintenant $\beta \geq 1$. De la même façon, compte tenu de la remarque 4.7.36, nous obtenons que

$$\begin{aligned}
B_{n,q} &= \sum_{\substack{l|n \\ l\in\mathcal{N}_q}} \phi(l) \\
&= \phi(1) + \phi(2) + \sum_{\substack{d|2^{\min(\gamma_q,\beta)}d_1 \\ d\neq 1,\ d\neq 2}} \phi(d) + \sum_{\alpha\geq 2} \sum_{\substack{d|2d_\alpha \\ d\neq 1,\ d\neq 2}} \phi(d) \\
&= 1 + 1 + 2^{\min(\gamma_q,\beta)}d_1 - 2 + \sum_{\alpha\geq 2}(2d_\alpha - 2) \\
&= 2^{\min(\gamma_q,\beta)}d_1 + 2\sum_{\alpha\geq 2}(d_\alpha - 1).
\end{aligned}$$

Les deux cas peuvent s'écrire comme suit :

$$B_{n,q} = 2^{\min(\gamma_q,\beta)}d_1 + 2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1).$$ □

Rappelons que $\mathcal{P}_{q,\infty}$ est défini dans la notation 4.7.34.

Corollaire 4.8.2 *Nous avons : $B_{n,q} = 1$ si et seulement si $n \in \mathcal{P}_{q,\infty}$.*

Preuve. Il est facile à voir que $n \in \mathcal{P}_{q,\infty}$ si et seulement si la décomposition de n comme dans la proposition 4.7.32 est $n = d'$, i.e. $\beta = 0$ et $d_\alpha = 1$ $\forall\alpha \geq 1$. Il suffit alors d'utiliser la formule (4.23) pour obtenir le résultat désiré. □

Remarquons au passage quelques propriétés de $B_{n,q}$:

Propriété 4.8.3 *Soit q une puissance d'un nombre premier impair p. Soit t un nombre impair tel que $(t, q) = 1$. Soit $\gamma_q \geq 0$ tel que $2^{\gamma_q} \parallel q + 1$. Soit d_1 le diviseur $\mathcal{P}_{q,1}$-maximal de t. On a :*

(i) $B_{2t,q} = 2B_{t,q}$.

(ii) $B_{2^it,q} = B_{2^{\gamma_q}t,q}$ *pour tout* i, $i \geq \gamma_q$. *En particulier, si* $\gamma_q = 1$, $B_{2^it,q} = B_{2t,q} = 2B_{t,q}$ *pour tout* $i \geq 1$.

(iii) $B_{2^it,q} = (2^i - 2)d_1 + B_{2t,q}$ *pour tout* i, $1 \leq i \leq \gamma_q$.

(iv) $B_{2^it,q} = (2^{\min(\gamma_q,i)} - 2)d_1 + 2B_{t,q}$ *pour tout* $i \geq 1$.

Preuve. C'est une conséquence directe de la proposition 4.8.1. □

4.8.2 Premier cas : n est premier avec p

Dans cette section nous considérons n et q premiers entre eux.
D'après la proposition 4.6.6, p. 143,

$$Edimh_q(n) = \frac{1}{4}\left(n - B_{n,q}\right).$$

La proposition 4.8.1 nous montre que

$$B_{n,q} = 2^{\min(\gamma_q,\beta)}d_1 + 2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1),$$

où $n = 2^\beta d' d_1 d_2 d_3 \cdots$ est la décomposition de n comme dans la proposition 4.7.32, et $\gamma_q \geq 0$ est tel que $2^{\gamma_q} \parallel q+1$. Nous obtenons donc que

$$Edimh_q(n) = \frac{1}{4}\left(n - 2^{\min(\gamma_q,\beta)}d_1 - 2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1)\right). \tag{4.25}$$

4.8.2.1 Une propriété technique

Présentons d'abord une corollaire de la propriété 4.8.3.

Corollaire 4.8.4 *Soit* q *une puissance d'un nombre premier impair* p. *Soit* t *un nombre impair tel que* $(t,q) = 1$. *Soit* $\gamma_q \geq 0$ *tel que* $2^{\gamma_q} \parallel q+1$. *Soit* d_1 *le diviseur* $\mathcal{P}_{q,1}$*-maximal de* t. *Nous avons les égalités suivantes :*

(i) $Edimh_q(2t) = 2Edimh_q(t)$,

(ii) $Edimh_q(2^it) = Edimh_q(2^{\gamma_q}t) + t\left(2^i - 2^{\gamma_q}\right)/4$ *pour tout* i, $i \geq \gamma_q$,

(iii) $Edimh_q(2^it) = 2Edimh_q(t) + (t - d_1)(2^{i-1} - 1)/2$ *pour tout* i, $1 \leq i \leq \gamma_q$,

(iv) $Edimh_q(2^it) = 2Edimh_q(t) + \left(2^{i-1}t - t + d_1 - 2^{\gamma_q-1}d_1\right)/2$ *pour tout* i, $i \geq \gamma_q$,

(v) $Edimh_q(2^it) = 2Edimh_q(t) + \left(2^{i-1}t - t + d_1 - 2^{\min(\gamma_q,i)-1}d_1\right)/2$ *pour tout* $i \geq 1$.

Preuve.

(i) C'est une conséquence directe de la proposition 4.6.6 et de la propriété 4.8.3(i).

(ii) Compte tenu de la proposition 4.6.6, p. 143, et de la propriété 4.8.3(ii), nous obtenons que

$$\begin{aligned} Edimh_q(2^it) &= \frac{1}{4}(2^it - B_{2^it,q}) \\ &= \frac{1}{4}(2^it - B_{2^{\gamma_q}t,q}) \\ &= \frac{1}{4}(2^it - 2^{\gamma_q}t + 4Edimh_q(2^{\gamma_q}t)) \\ &= Edimh_q(2^{\gamma_q}t) + t(2^i - 2^{\gamma_q})/4. \end{aligned}$$

(iii) Compte tenu de la proposition 4.6.6 et des propriétés 4.8.3(iii) et 4.8.3(i), nous obtenons que

$$\begin{aligned} Edimh_q(2^it) &= \frac{1}{4}(2^it - B_{2^it,q}) \\ &= \frac{1}{4}(2^it - (2^i-2)d_1 - 2B_{t,q}) \\ &= \frac{1}{4}(2^it - (2^i-2)d_1 - 2(t - 4Edimh_q(t))) \\ &= 2Edimh_q(t) + \frac{1}{4}(2^i(t-d_1) + 2(d_1 - t)) \\ &= 2Edimh_q(t) + \frac{1}{4}(t-d_1)(2^i-2). \end{aligned}$$

(iv) C'est une conséquence de (ii) et (iii).

(v) C'est une conséquence de (iii) et (iv). □

4.8.2.2 Limites

Pour étudier le comportement asymptotique de $Edimh_q(n)$ quand n tend vers l'infini, nous allons étudier la suite

$$\left(\frac{Edimh_q(n)}{n}\right)_{n\geq 1,\,(n,q)=1}. \tag{4.26}$$

Compte tenu de la formule (4.25) et de la décomposition $n = 2^\beta d' d_1 d_2 d_3 \cdots$, nous obtenons que

$$\frac{Edimh_q(n)}{n} = \frac{1}{4}\left(1 - \frac{2^{\min(\gamma_q,\beta)}d_1}{n} - \frac{2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1)}{n}\right)$$

$$
\begin{aligned}
&= \frac{1}{4}\Bigg(1 - \frac{1}{2^{\max(\beta-\gamma_q,0)}d'd_2d_3\cdots} \\
&\quad - \sum_{\alpha\geq 2}\left(\frac{1}{2^{\max(\beta-1,0)}d'd_1d_2\cdots d_{\alpha-1}d_{\alpha+1}\cdots} - \frac{2^{\min(\beta,1)}}{n}\right)\Bigg).
\end{aligned}
$$

Fixons un entier positif non nul m, $(m,q)=1$. Soit $2^\beta d'd_1d_2d_3\cdots$ sa décomposition de la proposition 4.7.32. Fixons un entier $i \geq 2$. Notez que si t appartient à $\mathcal{P}_{q,i}$, t impair, alors la décomposition de mt est $2^\beta d'd_1d_2d_3\cdots d_{i-1}\cdot(d_it)\cdot d_{i+1}\cdots$, i.e. $\beta(mt)=\beta(m)$, $d'(mt)=d'(m)$, $d_i(mt)=d_i(m)t$ et $d_\alpha(mt)=d_\alpha(t)$ pour tout $\alpha\geq 1$, $\alpha\neq i$. Donc si t parcourt les nombres impairs de $\mathcal{P}_{q,i}$, les $d_\alpha(mt)$, $\alpha\geq 1$, $\alpha\neq i$, ne changent pas, $d'(mt)$ et $\beta(mt)$ non plus, ce n'est que $d_i(mt)=d_i(m)t$ qui change. Par conséquent si nous prenons la sous-suite (la suite extraite) de la suite (4.26) avec n de la forme $n=mt$, m et t comme ci-dessus, t tend vers l'infini, alors

$$
\begin{aligned}
\lim_{\substack{t\to\infty,\, t\in\mathcal{P}_{q,i}\\ t \text{ impair}}} \frac{Edimh_q(mt)}{mt} &= \lim_{\substack{t\to\infty,\, t\in\mathcal{P}_{q,i}\\ t \text{ impair}}} \frac{1}{4}\Bigg(1 - \frac{2^{\min(\gamma_q,\beta)}d_1}{mt} - \frac{2^{\min(\beta,1)}d_it}{mt} \\
&\quad + \frac{2^{\min(\beta,1)}}{mt} - \frac{2^{\min(\beta,1)}\sum_{\alpha\geq 2,\alpha\neq i}(d_\alpha-1)}{mt}\Bigg) \qquad (4.27)\\
&= \frac{1}{4}\left(1 - \frac{2^{\min(\beta,1)}d_i}{m}\right)\\
&= \frac{1}{4}\left(1 - \frac{1}{2^{\max(\beta-1,0)}d'd_1d_2\cdots d_{i-1}d_{i+1}\cdots}\right).
\end{aligned}
$$

De la même façon

$$
\begin{aligned}
\lim_{\substack{t\to\infty,\, t\in\mathcal{P}_{q,1}\\ t \text{ impair}}} \frac{Edimh_q(mt)}{mt} &= \frac{1}{4}\left(1 - \frac{2^{\min(\beta,\gamma_q)}d_1}{m}\right)\\
&= \frac{1}{4}\left(1 - \frac{1}{2^{\max(\beta-\gamma_q,0)}d'd_2d_3\cdots}\right),\\
\lim_{t\to\infty,\, t\in\mathcal{P}_{q,\infty}} \frac{Edimh_q(mt)}{mt} &= \frac{1}{4},\\
\lim_{j\to\infty} \frac{Edimh_q(2^jm)}{2^jm} &= \frac{1}{4},\\
\lim_{\substack{t_1\to\infty,\, t_1\in\mathcal{P}_{q,\alpha_1},\, t_1 \text{ impair}\\ t_2\to\infty,\, t_2\in\mathcal{P}_{q,\alpha_2},\, t_2 \text{ impair}}} \frac{Edimh_q(mt_1t_2)}{mt_1t_2} &= \frac{1}{4} \text{ pour tous les } \alpha_1\geq 1, \alpha_2\geq 1, \alpha_1\neq\alpha_2,
\end{aligned}
$$

etc...

4.8.2.3 Les points d'accumulation

Quelles conclusions pouvons-nous tirer de ces limites ? D'abord, il est clair que la suite (4.26) ne possède pas de limite, car il existe des sous-suites qui tendent vers des limites distinctes. Ensuite, nous avons le résultat suivant :

Proposition 4.8.5 *Soit q une puissance d'un nombre premier p. Soit k un entier positif non nul tel que $(k, q) = 1$. Il existe une sous-suite de la suite (4.26) dont la limite, quand n tend vers l'infini, est $\frac{1}{4}\left(1 - \frac{1}{k}\right)$.*

Preuve. Soit $2^\eta s' s_1 s_2 s_3 \cdots$ la décomposition de k comme dans la proposition 4.7.32. Soit r un entier positif non nul tel que $s_r = 1$ (il en existe forcément un, cf. remarque 4.7.35). Posons $\beta = \eta + 1$, $d' = s'$, $d_i = s_i$ pour tout $i \geq 1$, et $m = 2^\beta d' d_1 d_2 d_3 \cdots$ (en fait, cela revient à poser $m = 2k$. Notez que si $\eta = 0$, nous pouvons poser $m = k$ et continuer la démonstration comme ci-dessous, car $\max(\beta - 1, 0) = 0$ si $\beta = 0$ ou $\beta = 1$). Soit t parcourant $\mathcal{P}_{q,r}$, t impair. Alors d'après la formule (4.27)

$$\begin{aligned}
\lim_{\substack{t\to\infty,\, t\in\mathcal{P}_{q,r} \\ t \text{ impair}}} \frac{Edimh_q(mt)}{mt} &= \frac{1}{4}\left(1 - \frac{1}{2^{\max(\beta-1,0)} d' d_1 d_2 \cdots d_{r-1} d_{r+1} \cdots}\right) \\
&= \frac{1}{4}\left(1 - \frac{1}{2^\eta s' s_1 s_2 \cdots s_{r-1} s_r s_{r+1} \cdots}\right) \\
&= \frac{1}{4}\left(1 - \frac{1}{k}\right).
\end{aligned}$$

□

La proposition suivante montre qu'il n'existe pas d'autres limites des sous-suites de la suite (4.26).

Proposition 4.8.6 *Soit q une puissance d'un nombre premier p. Si une sous-suite de la suite (4.26) converge, sa limite appartient à*

$$\left\{\frac{1}{4}\right\} \cup \left\{\frac{1}{4}\left(1 - \frac{1}{k}\right) \mid k \in \mathbb{N}^*, (k, q) = 1\right\}.$$

Preuve. Prenons une sous-suite convergente de la suite (4.26). Soit J l'ensemble de ses indices. Cette sous-suite

$$\left(\frac{Edimh_q(n)}{n}\right)_{n\in J} \tag{4.28}$$

définit une suite $(d_\alpha(n))_{n\in J}$ pour tout $\alpha \geq 1$ (cf. notation 4.7.37, p. 168).

– Supposons d'abord que toutes ces suites $(d_\alpha(n))_{n\in J}$, $\alpha \geq 1$, soient bornées. Alors évidemment

$$\begin{aligned}
&\lim_{n\to\infty,\, n\in J} \frac{Edimh_q(n)}{n} \\
&= \lim_{n\to\infty,\, n\in J} \frac{1}{4}\left(1 - \frac{2^{\min(\beta(n),\gamma_q)} d_1(n)}{n} - \frac{2^{\min(\beta(n),1)} \sum_{\alpha\geq 2} (d_\alpha(n) - 1)}{n}\right) = \frac{1}{4}.
\end{aligned}$$

– Soit une et une seule de ces suites n'est pas bornée, par exemple, $(d_i(n))_{n\in J}$ pour un $i \geq 2$. Alors

$$\begin{aligned}
&\lim_{n\to\infty,\, n\in J} \frac{Edimh_q(n)}{n} \\
&= \lim_{n\to\infty,\, n\in J} \frac{1}{4}\left(1 - \frac{2^{\min(\beta(n),1)} d_i(n)}{n}\right) \\
&= \lim_{n\to\infty,\, n\in J} \frac{1}{4}\left(1 - \frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n)\cdots d_{i-1}(n) d_{i+1}(n)\cdots}\right).
\end{aligned} \tag{4.29}$$

Comme la suite (4.28) converge, la suite

$$\left(\frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n)\cdots d_{i-1}(n) d_{i+1}(n)\cdots}\right)_{n\in J}$$

converge aussi. Si cette dernière converge vers 0, alors la suite (4.28) converge vers $\frac{1}{4}$, et c'est ce qu'il fallait démontrer. Sinon, la suite

$$\left(2^{\max(\beta(n)-1,0)} d'(n) d_1(n)\cdots d_{i-1}(n) d_{i+1}(n)\cdots\right)_{n\in J}$$

converge aussi. Mais c'est une suite d'entiers, donc à un certain moment elle se stabilise et devient égale à un $k \geq 1$. Ce k est premier avec q, car n parcourt les entiers premiers avec q. Donc

$$\lim_{n\to\infty,\, n\in J} \frac{Edimh_q(n)}{n} = \frac{1}{4}\left(1 - \frac{1}{k}\right),\ k \geq 1,\ (k,q) = 1.$$

Nous pouvons raisonner de la même façon si $i = 1$.

– Supposons enfin qu'au moins deux des suites $(d_\alpha(n))_{n\in J}$, $\alpha \geq 1$, ne soient pas bornées. Par exemple, soient $(d_{\alpha_1}(n))_{n\in J}, \ldots, \left(d_{\alpha_j}(n)\right)_{n\in J}$ non bornées, $j \geq 2$. Pour simplifier, supposons que $\alpha_i \geq 2$ pour tout i, $1 \leq i \leq j$ (si $\alpha_i = 1$ pour un certain i, $1 \leq i \leq j$, alors dans la démonstration il suffira de remplacer $2^{\max(\beta(n)-1,0)}$ par $2^{\max(\beta-\gamma_q,0)}$ dans le terme qui correspond à $d_1(n)$, tout le reste restant le même). De la même façon que dans la formule (4.29) nous obtenons

$$\begin{aligned}
&\lim_{n\to\infty,\, n\in J} \frac{Edimh_q(n)}{n} \\
&= \lim_{n\to\infty,\, n\in J} \frac{1}{4}\left(1 - \sum_{i=1}^{j} \frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n)\cdots d_{\alpha_i-1}(n) d_{\alpha_i+1}(n)\cdots}\right).
\end{aligned}$$

La suite (4.28) converge, donc la suite

$$\left(\sum_{i=1}^{j} \frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n)\cdots d_{\alpha_i-1}(n) d_{\alpha_i+1}(n)\cdots}\right)_{n\in J} \tag{4.30}$$

converge aussi. Supposons que la suite (4.30) converge vers un $\omega \neq 0$ (si $\omega = 0$, la proposition est démontrée). La suite $(d_{\alpha_1}(n))_{n\in J}$ n'est pas bornée, il existe donc une sous-suite $(d_{\alpha_1}(n))_{n\in K}$, $K \subset J$, tendant vers l'infini. Donc la suite

$$\left(\sum_{i=2}^{j} \frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n) \cdots d_{\alpha_i-1}(n) d_{\alpha_i+1}(n) \cdots}\right)_{n\in K}$$

tend vers 0. Cela signifie que la suite

$$\left(\frac{1}{2^{\max(\beta(n)-1,0)} d'(n) d_1(n) \cdots d_{\alpha_1-1}(n) d_{\alpha_1+1}(n) \cdots}\right)_{n\in K}$$

tend vers le même $\omega \neq 0$. Donc la suite

$$\left(2^{\max(\beta(n)-1,0)} d'(n) d_1(n) \cdots d_{\alpha_1-1}(n) d_{\alpha_1+1}(n) \cdots\right)_{n\in K}$$

converge aussi et, comme c'est une suite d'entiers, elle se stabilise, elle devient égale à un k, $(k,q) = 1$, $k = \frac{1}{\omega}$.

Donc une sous-suite de (4.30) converge vers $\frac{1}{k}$. Comme la suite (4.30) converge, elle converge aussi vers $\frac{1}{k}$, et la suite (4.28) converge vers $\frac{1}{4}\left(1 - \frac{1}{k}\right)$, $k \geq 1$ et $(k,q) = 1$. □

Nous avons donc montré que l'ensemble des points d'accumulation et des valeurs prises une infinité de fois de la suite (4.26) est $\{\frac{1}{4}\} \cup \{\frac{1}{4}\left(1 - \frac{1}{k}\right) \mid k \in \mathbb{N}^*, (k,q) = 1\}$. Dans le tableau 4.1 nous donnons les valeurs de $\frac{1}{4}\left(1 - \frac{1}{k}\right)$ pour $1 \leq k \leq 9$.

k	1	2	3	4	5	6	7	8	9
$\frac{1}{4}\left(1 - \frac{1}{k}\right)$	0	$\frac{1}{8}$	$\frac{1}{6}$	$\frac{3}{16}$	$\frac{1}{5}$	$\frac{5}{24}$	$\frac{3}{14}$	$\frac{7}{32}$	$\frac{2}{9}$
	0	0.125	0.166...	0.1875	0.2	0.208...	0.214...	0.21875	0.222...

TAB. 4.1 – *Les valeurs de* $\frac{1}{4}\left(1 - \frac{1}{k}\right)$ *pour* $1 \leq k \leq 9$

4.8.2.4 Comparaison avec n

Le théorème 4.6.5, p. 141, montre que $Edimh_q(n) = 0$ si et seulement si $n \in \mathcal{N}_q$. Dans cette section nous allons montrer que pour $n \notin \mathcal{N}_q$, $(n,q) = 1$, $Edimh_q(n)$ est "grand", plus précisément, il existe une constante $c > 0$ telle que $Edimh_q(n) \geq \frac{n}{c}$. Avec la majoration du corollaire 4.6.7, p. 145, nous obtiendrons donc que

$$\frac{n}{c} \leq Edimh_q(n) < \frac{n}{4},$$

ce qui signifiera que, pour les $n \notin \mathcal{N}_q$, $Edimh_q(n)$ est du même ordre que n.

Notez que $Edimh_q(n) \geq \frac{n}{c}$ est équivalent à dire que le zéro n'est pas un point d'accumulation de la suite (4.26).

Théorème 4.8.7 *Soit q une puissance d'un nombre premier p. Il existe une constante $\epsilon > 0$ telle que $Edimh_q(n) \geq \epsilon n$ pour tout $n \in \mathbb{N}^*$, $n \notin \mathcal{N}_q$ et $(n,q) = 1$.*

Preuve. Comme $Edimh_q(n) = \frac{1}{4}(n - B_{n,q})$ (cf. proposition 4.6.6, p. 143), $Edimh_q(n) \geq \epsilon n$ est équivalent à $B_{n,q} \leq (1-4\epsilon)n$. Autrement dit, il faut montrer qu'il existe une constante $\delta < 1$ telle que $B_{n,q} \leq \delta n$, i.e. $\frac{B_{n,q}}{n} \leq \delta$ pour tout n, $n \notin \mathcal{N}_q$ et $(n,q) = 1$.

Soit n, $n \notin \mathcal{N}_q$, $(n,q) = 1$. Nous allons considérer plusieurs cas selon la forme de n. Soit $n = 2^\beta d' d_1 d_2 d_3 \cdots$ la décomposition de n comme dans la proposition 4.7.32. Rappelons que, compte tenu de la proposition 4.8.1, p. 169,

$$\frac{B_{n,q}}{n} = \frac{2^{\min(\gamma_q,\beta)} d_1 + 2^{\min(\beta,1)} \sum_{\alpha\geq 2}(d_\alpha - 1)}{2^\beta d' d_1 d_2 d_3 \cdots},$$

où $\gamma_q \geq 0$ est tel que $2^{\gamma_q} \parallel q+1$.

Soit $\{\alpha_1, \alpha_2, \ldots, \alpha_j\}$, $j \geq 0$, l'ensemble ordonné (i.e. $1 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_j$) des indices tels que $d_{\alpha_i} > 1\ \forall\, i$, $1 \leq i \leq j$, et $d_\alpha = 1\ \forall\, \alpha \geq 1$, $\alpha \notin \{\alpha_1, \alpha_2, \ldots, \alpha_j\}$. Nous considérons les six cas suivants :

1. $j = 0$ et $d' = 1$,
2. $j = 0$ et $d' > 1$,
3. $j = 1$ et $d' = 1$,
4. $j = 1$ et $d' > 1$,
5. $j = 2$,
6. $j \geq 3$.

Cas 1. D'après la remarque 4.7.38, p. 168, $n \notin \mathcal{N}_q$ signifie que $\beta \geq \gamma_q + 1$. Dans ce cas

$$\frac{B_{n,q}}{n} = \frac{2^{\min(\gamma_q,\beta)}}{2^\beta} = \frac{2^{\gamma_q}}{2^\beta} = \frac{1}{2^{\beta-\gamma_q}} \leq \frac{1}{2}.$$

Omettons pour l'instant le cas 2 et passons directement au :

Cas 3. Soit $\alpha_1 = 1$. D'après la remarque 4.7.38, $n \notin \mathcal{N}_q$ signifie que $\beta \geq \gamma_q + 1$. Dans ce cas

$$\frac{B_{n,q}}{n} = \frac{2^{\min(\gamma_q,\beta)} d_1}{2^\beta d_1} = \frac{2^{\min(\gamma_q,\beta)}}{2^\beta} \leq \frac{1}{2}$$

comme dans le cas 1.

Soit maintenant $\alpha_1 \geq 2$. D'après la même remarque, pour que n n'appartienne pas à $\mathcal{N}_q$, β doit être supérieur ou égal à 2. Dans ce cas

$$\frac{B_{n,q}}{n} = \frac{2^{\min(\gamma_q,\beta)} + 2^{\min(\beta,1)}(d_{\alpha_1} - 1)}{2^\beta d_{\alpha_1}} = \frac{2^{\min(\gamma_q,\beta)}}{2^\beta d_{\alpha_1}} + \frac{2d_{\alpha_1}}{2^\beta d_{\alpha_1}} - \frac{2}{2^\beta d_{\alpha_1}} \leq \frac{1}{d_{\alpha_1}} + \frac{1}{2} - 0 \leq \frac{1}{3} + \frac{1}{2} = \frac{5}{6},$$

compte tenu de la remarque 4.7.39.

Avant de passer au quatre cas restants, simplifions un peu $\frac{B_{n,q}}{n}$:

$$\begin{aligned}
\frac{B_{n,q}}{n} &= \frac{2^{\min(\gamma_q,\beta)}d_1 + 2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1)}{2^\beta d' d_1 d_2 d_3 \cdots} \\
&= \frac{\frac{2^{\min(\gamma_q,\beta)}}{2^\beta}d_1 + \frac{2^{\min(\beta,1)}}{2^\beta}\sum_{\alpha\geq 2}(d_\alpha - 1)}{d' d_1 d_2 d_3 \cdots} \\
&\leq \frac{d_1 + \sum_{\alpha\geq 2}(d_\alpha - 1)}{d' d_1 d_2 d_3 \cdots} \\
&= \frac{1 + \sum_{\alpha\geq 1}(d_\alpha - 1)}{d' d_1 d_2 d_3 \cdots} \\
&= \frac{1 + \sum_{i=1}^{j}(d_{\alpha_i} - 1)}{d' d_{\alpha_1} d_{\alpha_2} \cdots d_{\alpha_j}} \\
&= \frac{1 - j + \sum_{i=1}^{j} d_{\alpha_i}}{d' d_{\alpha_1} d_{\alpha_2} \cdots d_{\alpha_j}}.
\end{aligned}$$

Cas 2.

$$\frac{B_{n,q}}{n} \leq \frac{1 - j + \sum_{i=1}^{j} d_{\alpha_i}}{d' d_{\alpha_1} d_{\alpha_2} \cdots d_{\alpha_j}} = \frac{1}{d'} \leq \frac{1}{3}.$$

Cas 4.

$$\frac{B_{n,q}}{n} \leq \frac{1 - j + \sum_{i=1}^{j} d_{\alpha_i}}{d' d_{\alpha_1} d_{\alpha_2} \cdots d_{\alpha_j}} = \frac{d_{\alpha_1}}{d' d_{\alpha_1}} = \frac{1}{d'} \leq \frac{1}{3}.$$

Pour les deux cas restants nous allons simplifier encore la majoration de $\frac{B_{n,q}}{n}$. Soit i_0, $1 \leq i_0 \leq j$, tel que $d_{\alpha_{i_0}} = \max_{1\leq i\leq j} d_{\alpha_i}$. Alors, comme $j \geq 2$ et $d' \geq 1$,

$$\frac{B_{n,q}}{n} \leq \frac{1 - j + \sum_{i=1}^{j} d_{\alpha_i}}{d' d_{\alpha_1} d_{\alpha_2} \cdots d_{\alpha_j}} \leq \frac{\sum_{i=1}^{j} d_{\alpha_i}}{\prod_{i=1}^{j} d_{\alpha_i}} \leq \frac{\sum_{i=1}^{j} d_{\alpha_{i_0}}}{\prod_{i=1}^{j} d_{\alpha_i}} = \frac{j d_{\alpha_{i_0}}}{\prod_{1\leq i\leq j} d_{\alpha_i}} = \frac{j}{\prod\limits_{\substack{1\leq i\leq j\\ i\neq i_0}} d_{\alpha_i}}.$$

Cas 5. L'ensemble d'indices est $\{\alpha_1, \alpha_2\}$, donc i_0 est égal à 1 ou 2. Notons i_1 l'autre indice, i.e. si $i_0 = 1$, $i_1 = 2$, et si $i_0 = 2$, $i_1 = 1$. Alors

$$\frac{B_{n,q}}{n} \leq \frac{j}{\prod\limits_{\substack{1\leq i\leq j\\ i\neq i_0}} d_{\alpha_i}} = \frac{2}{d_{\alpha_{i_1}}} \leq \frac{2}{3}.$$

Cas 6. Soit j_0, le plus grand des i, $1 \leq i \leq j$, $i \neq i_0$, i.e. si $i_0 = j$, alors $j_0 = j - 1$, sinon $j_0 = j$. Donc $j_0 \geq j - 1$.

Notez que $\alpha_i \geq i$ pour tout i, $1 \leq i \leq j$, car les indices $\alpha_1, \ldots, \alpha_j$ sont ordonnés : $1 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_j$.

Nous avons :

$$j < 2^{j-1} \leq 2^{j_0} \leq 2^{\alpha_{j_0}} < 2^{\alpha_{j_0}} + 1 \leq d_{\alpha_{j_0}},$$

la dernière inégalité s'inspirant de la propriété 4.7.29, p. 166. Donc $j < d_{\alpha_{j_0}}$. Alors

$$\frac{B_{n,q}}{n} \leq \frac{j}{\prod\limits_{\substack{1\leq i\leq j\\ i\neq i_0}} d_{\alpha_i}} < \frac{1}{\prod\limits_{\substack{1\leq i\leq j\\ i\neq i_0,\, i\neq j_0}} d_{\alpha_i}} \leq \frac{1}{3}.$$

Nous obtenons que dans tous les six cas $\frac{B_{n,q}}{n}$ est inférieur ou égal à $\frac{5}{6} < 1$, ce qui complète la démonstration. □

Nous avons donc montré qu'il existe une constante $\epsilon > 0$ telle que $Edimh_q(n) \geq \epsilon n$ pour tout $n \geq 1$ tel que $n \notin \mathcal{N}_q$ et $(n,q) = 1$. Nous allons donner quelques remarques sur la valeur de ϵ.

Comme nous l'avons vu dans la section 4.8.2.3, l'ensemble des points d'accumulation et des valeurs prises une infinité de fois de la suite (4.26) est

$$\left\{\frac{1}{4}\right\} \cup \left\{\frac{1}{4}\left(1-\frac{1}{k}\right) \mid k \in \mathbb{N}^*, (k,q) = 1\right\}.$$

Nous savons que $\frac{Edimh_q(n)}{n} = 0$ si et seulement si $n \in \mathcal{N}_q$. Prenons la sous-suite

$$\left(\frac{Edimh_q(n)}{n}\right)_{n\in\mathbb{N}^*,\,(n,q)=1,\,n\notin\mathcal{N}_q}. \tag{4.31}$$

Cette sous-suite ne prend jamais pour valeur zéro, et, comme zéro n'était pas un point d'accumulation de la suite (4.26), il ne l'est pas pour la suite (4.31). Donc l'ensemble des points d'accumulation et des valeurs prises une infinité de fois de la suite (4.31) est $\{\frac{1}{4}\} \cup \{\frac{1}{4}\left(1-\frac{1}{k}\right) \mid k \geq 2, (k,q) = 1\}$. Le plus petit élément de cet ensemble est égal à $\frac{1}{4}\left(1-\frac{1}{2}\right) = \frac{1}{8}$ si $2 \nmid q$, et $\frac{1}{4}\left(1-\frac{1}{3}\right) = \frac{1}{6}$ sinon. Donc

$$\liminf_{n\to\infty,\, n\notin\mathcal{N}_q,\,(n,q)=1} \frac{Edimh_q(n)}{n} = \begin{cases} \frac{1}{6} & \text{si } 2 \mid q,\\ \frac{1}{8} & \text{sinon.}\end{cases} \tag{4.32}$$

Cela veut dire que pour n “grands”, nous pouvons prendre pour ϵ une valeur “proche” de $\frac{1}{6}$ (ou de $\frac{1}{8}$, selon le cas). Plus précisément, pour tout $\sigma > 0$ il existe un $N > 0$ tel que pour tout n, $n \geq N$, $n \notin \mathcal{N}_q$, $(n,q) = 1$, on a $\frac{Edimh_q(n)}{n} \geq \frac{1}{6} - \sigma$ (évidemment, $\frac{1}{8}$ remplace ici $\frac{1}{6}$ si $2 \nmid q$). La figure 4.2 montre que même pour les petites valeurs de n, $n \notin \mathcal{N}_q$, $(n,q) = 1$, nous pouvons prendre pour ϵ une constante proche de $\frac{1}{6}$ (ou de $\frac{1}{8}$).

4.8.3 Deuxième cas : n n'est pas premier avec p

Ce cas est plus compliqué que le cas précédent. Il est bien plus difficile d'expliciter les points d'accumulation et les valeurs prises une infinité de fois de la suite

$$\left(\frac{Edimh_q(n)}{n}\right)_{n\in\mathbb{N}^*,\,(n,q)>1} \tag{4.33}$$

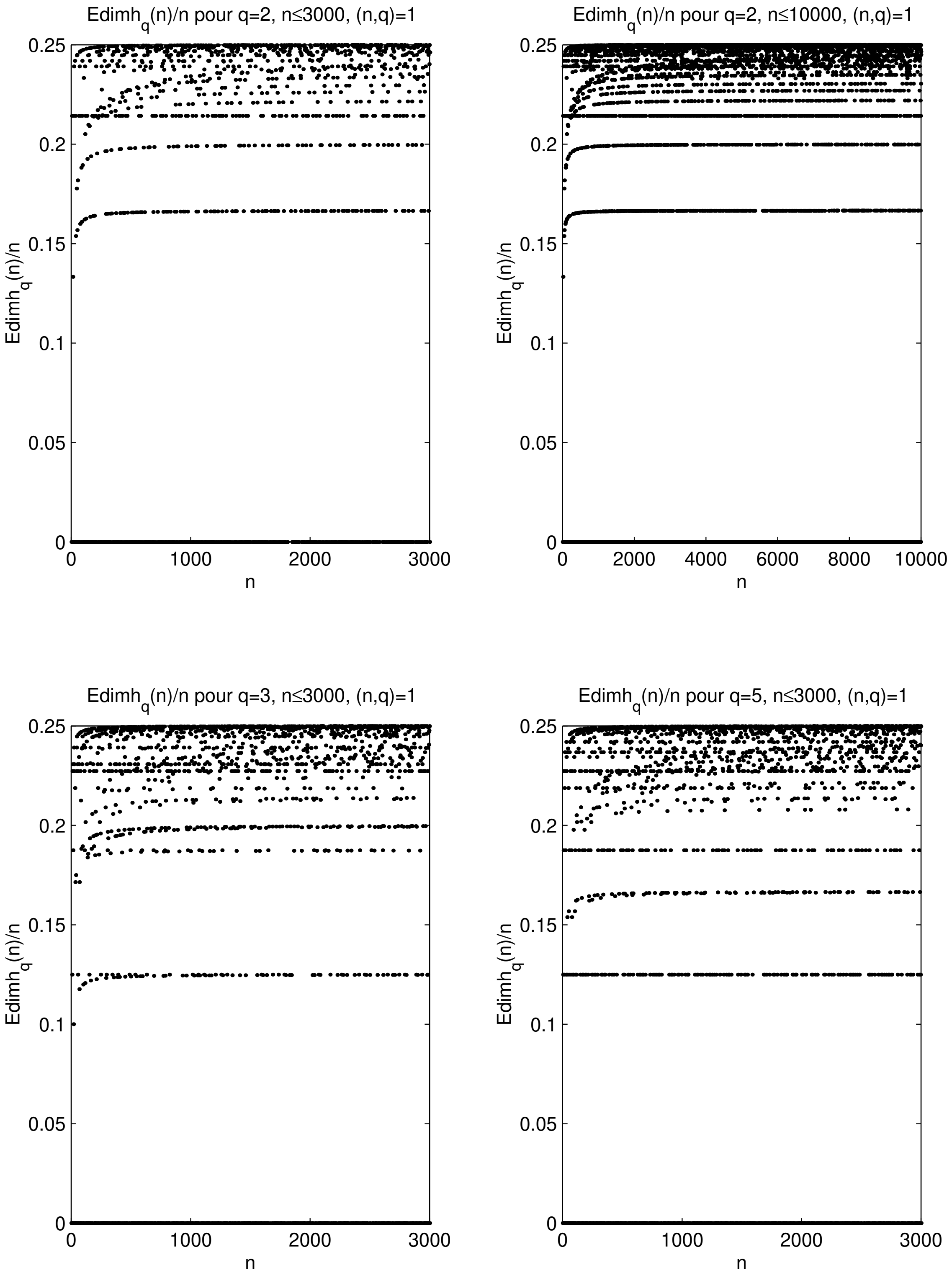

FIG. 4.2 – *Les valeurs de* $\frac{Edimh_q(n)}{n}$ *pour* $q = 2, 3, 5$

(cf. section 4.8.3.1). Mais nous arriverons quand même à montrer qu'il existe une constante $c > 0$ telle que $Edimh_q(n) \geq \frac{n}{c}$ pour tout n, $n \in \mathbb{N}^*$, $(n, q) > 1$ (cf. section 4.8.3.3).

Rappelons que $Edimh_q(n) \neq 0$ pour tout n, $n \in \mathbb{N}^*$ et $(n, q) > 1$ (cf. lemme 4.6.4, p. 141).

Soit, comme toujours, q une puissance d'un nombre premier p, et soit n un entier, $n = rp^s$, $(r, p) = 1$ et $s > 0$.

4.8.3.1 Limites

Dans cette section nous allons chercher des limites des sous-suites de la suite (4.33).

Rappelons que, compte tenu des propositions 4.6.11, p. 147, et 4.8.1, p. 169,

$$\frac{Edimh_q(n)}{n} = \frac{1}{3} - \frac{1}{6(p^s+1)} - \frac{B_{r,q}}{r}\left(\frac{1}{12} + \frac{1}{12p^s} + \frac{2 - 3\delta_{p^s}}{12p^s(p^s+1)}\right)$$

et

$$\frac{B_{r,q}}{r} = \frac{2^{\min(\gamma_q,\beta)}d_1 + 2^{\min(\beta,1)}\sum_{\alpha\geq 2}(d_\alpha - 1)}{2^\beta d' d_1 d_2 d_3 \cdots},$$

où $r = 2^\beta d' d_1 d_2 d_3 \cdots$ est la décomposition de r comme dans la proposition 4.7.32, et $\gamma_q \geq 0$ est tel que $2^{\gamma_q} \parallel q + 1$.

Fixons une partie des $s, \beta, d', d_1, d_2, d_3, \ldots$ et faisons tendre les autres vers l'infini. Par exemple, fixons tout sauf β. Alors

$$\lim_{\beta\to\infty} \frac{Edimh_q(n)}{n} = \frac{1}{3} - \frac{1}{6(p^s+1)}.$$

De la même façon nous obtenons que

$$\begin{aligned}
\lim_{d'\to\infty} \frac{Edimh_q(n)}{n} &= \lim_{\substack{d_{\alpha_1}\to\infty,\, \alpha_1\geq 1,\, \alpha_1 \text{ fixé} \\ d_{\alpha_2}\to\infty,\, \alpha_2\geq 1,\, \alpha_2 \text{ fixé},\, \alpha_1\neq\alpha_2}} \frac{Edimh_q(n)}{n} \\
&= \frac{1}{3} - \frac{1}{6(p^s+1)}, \\
\lim_{d_1\to\infty} \frac{Edimh_q(n)}{n} &= \frac{1}{3} - \frac{1}{6(p^s+1)} \\
&\quad - \frac{1}{2^{\max(\beta-\gamma_q,0)}d' d_2 d_3 \cdots}\left(\frac{1}{12} + \frac{1}{12p^s} + \frac{2 - 3\delta_{p^s}}{12p^s(p^s+1)}\right), \\
\lim_{\substack{d_\alpha\to\infty \\ \alpha\geq 2,\, \alpha \text{ fixé}}} \frac{Edimh_q(n)}{n} &= \frac{1}{3} - \frac{1}{6(p^s+1)} - \frac{1}{2^{\max(\beta-1,0)}d' d_1 d_2 \cdots d_{\alpha-1}d_{\alpha+1}\cdots} \\
&\quad \times \left(\frac{1}{12} + \frac{1}{12p^s} + \frac{2 - 3\delta_{p^s}}{12p^s(p^s+1)}\right),
\end{aligned}$$

$$\begin{aligned}
\lim_{\substack{\beta\to\infty\\ s\to\infty}} \frac{Edimh_q(n)}{n} &= \lim_{\substack{d'\to\infty\\ s\to\infty}} \frac{Edimh_q(n)}{n}\\
&= \lim_{\substack{d_{\alpha_1}\to\infty,\, d_{\alpha_2}\to\infty\\ s\to\infty,\,\alpha_1\geq 1,\,\alpha_2\geq 1,\,\alpha_1\neq\alpha_2}} \frac{Edimh_q(n)}{n}\\
&= \frac{1}{3},\\
\lim_{s\to\infty} \frac{Edimh_q(n)}{n} &= \frac{1}{3}-\frac{1}{12}\cdot\frac{B_{r,q}}{r},\\
\lim_{\substack{d_1\to\infty\\ s\to\infty}} \frac{Edimh_q(n)}{n} &= \frac{1}{3}-\frac{1}{12}\cdot\frac{1}{2^{\max(\beta-\gamma_q,0)}d'd_2d_3\cdots},\\
\lim_{\substack{d_\alpha\to\infty,\,\alpha\geq 2\\ s\to\infty}} \frac{Edimh_q(n)}{n} &= \frac{1}{3}-\frac{1}{12}\cdot\frac{1}{2^{\max(\beta-1,0)}d'd_1d_2\cdots d_{\alpha-1}d_{\alpha+1}\cdots}.
\end{aligned}$$

Nous voyons que ce cas est bien plus complexe que le cas précédent (cf. section 4.8.2.2). Par exemple, pour chaque r tel que $(r,p)=1$, il existe une sous-suite de la suite (4.33) dont la limite est égale à $\frac{1}{3}-\frac{1}{12}\cdot\frac{B_{r,q}}{r}$.

La figure 4.3 représente $\frac{Edimh_q(n)}{n}$ pour $q=2,3,5$, $n\leq 3000$ pour trois cas : $(n,q)=1$, $(n,q)>1$ et le cas général qui joint les deux premiers cas.

4.8.3.2 Une formule utile

En utilisant les propositions 4.6.6, p. 143, et 4.6.11, p. 147, nous exprimons $Edimh_q(n)$ par $Edimh_q(r)$:

$$\begin{aligned}
Edimh_q(n) &= n\left(\frac{1}{3}-\frac{1}{6(p^s+1)}\right)-B_{r,q}\left(\frac{p^s+1}{12}+\frac{2-3\delta_{p^s}}{12(p^s+1)}\right)\\
&= n\left(\frac{1}{3}-\frac{1}{6(p^s+1)}\right)-(r-4Edimh_q(r))\left(\frac{p^s+1}{12}+\frac{2-3\delta_{p^s}}{12(p^s+1)}\right)\\
&= n\left(\frac{1}{3}-\frac{1}{6(p^s+1)}\right)+4Edimh_q(r)\left(\frac{p^s+1}{12}+\frac{2-3\delta_{p^s}}{12(p^s+1)}\right)\\
&\quad -r\left(\frac{p^s+1}{12}+\frac{2-3\delta_{p^s}}{12(p^s+1)}\right)\\
&= \frac{n}{3}-\frac{n}{6(p^s+1)}-\frac{n}{12}-\frac{r}{12}-\frac{r(2-3\delta_{p^s})}{12(p^s+1)}\\
&\quad +Edimh_q(r)\left(\frac{p^s+1}{3}+\frac{2-3\delta_{p^s}}{3(p^s+1)}\right)\\
&= \frac{n}{4}-\frac{2rp^s+rp^s+r+2r-3r\delta_{p^s}}{12(p^s+1)}+Edimh_q(r)\left(\frac{p^s+1}{3}+\frac{2-3\delta_{p^s}}{3(p^s+1)}\right)
\end{aligned}$$

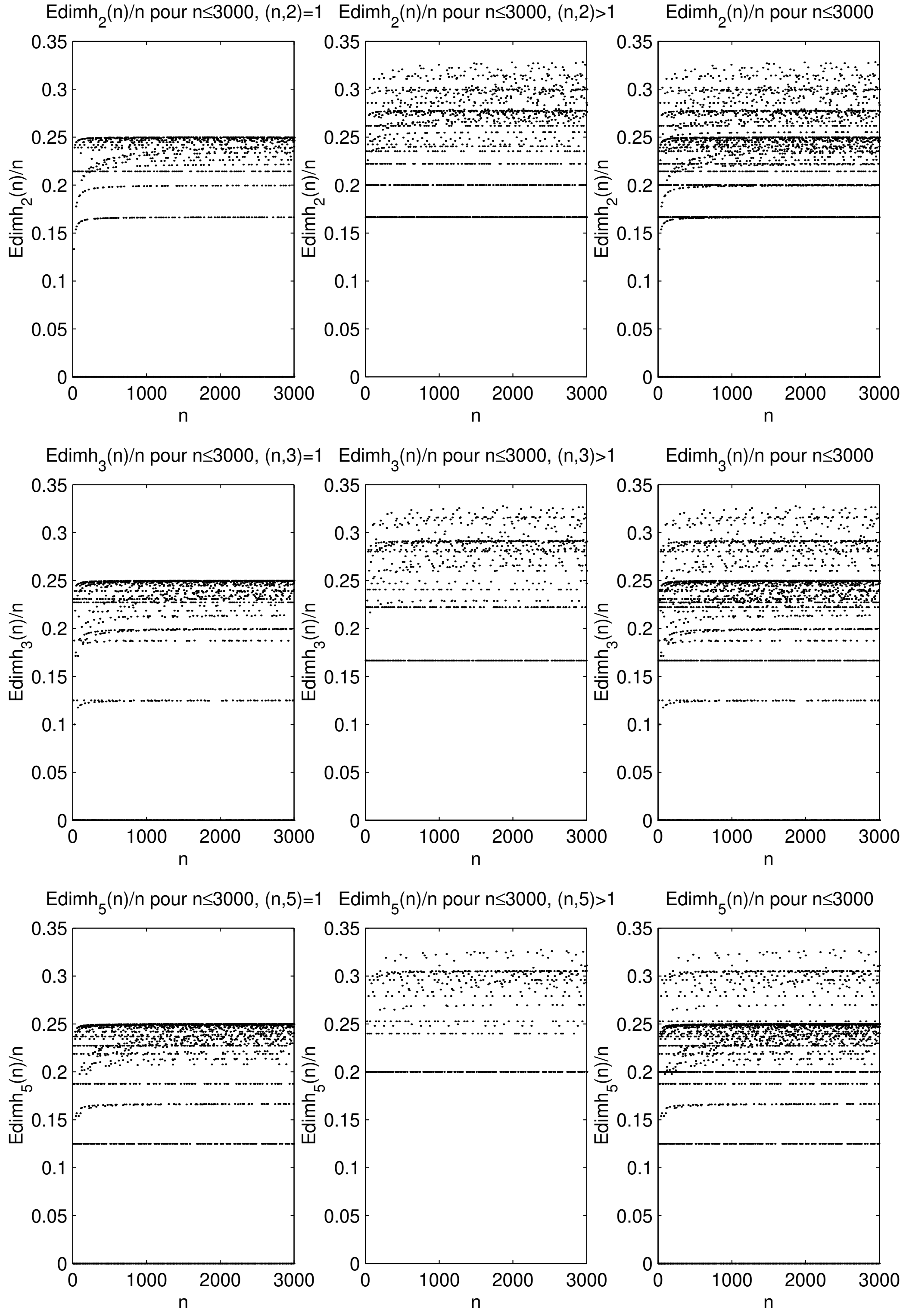

FIG. 4.3 – *Les valeurs de* $\frac{Edimh_q(n)}{n}$ *pour* $q = 2, 3, 5$

$$= \frac{n}{4} - \frac{r}{4} + \frac{r\delta_{p^s}}{4(p^s+1)} + Edimh_q(r)\left(\frac{p^s+1}{3} + \frac{2-3\delta_{p^s}}{3(p^s+1)}\right).$$

Nous obtenons donc

$$Edimh_q(n) = \frac{n}{4} - \frac{r}{4} + \frac{r\delta_{p^s}}{4(p^s+1)} + Edimh_q(r)\left(\frac{p^s+1}{3} + \frac{2-3\delta_{p^s}}{3(p^s+1)}\right). \tag{4.34}$$

4.8.3.3 Comparaison avec n

Nous savons qu'il existe une constante $\epsilon > 0$ telle que $\frac{Edimh_q(r)}{r} \geq \epsilon$ (cf. théorème 4.8.7). Dans cette section nous trouverons une telle constante pour $\frac{Edimh_q(n)}{n}$.

Rappelons que $s > 0$, donc $p^s > 1$.

D'abord, à partir de la formule (4.34) nous obtenons que

$$\frac{Edimh_q(n)}{n} = \frac{1}{4} - \frac{1}{4p^s} + \frac{\delta_{p^s}}{4p^s(p^s+1)} + \frac{Edimh_q(r)}{r}\left(\frac{1}{3} + \frac{1}{3p^s} + \frac{2-3\delta_{p^s}}{3p^s(p^s+1)}\right). \tag{4.35}$$

Premier cas : $p \geq 3$.

Comme dans ce cas $\delta_{p^s} = 0$, l'égalité (4.35) devient

$$\frac{Edimh_q(n)}{n} = \frac{1}{4} - \frac{1}{4p^s} + \frac{Edimh_q(r)}{r}\left(\frac{1}{3} + \frac{1}{3p^s} + \frac{2}{3p^s(p^s+1)}\right),$$

d'où

$$\frac{Edimh_q(n)}{n} \geq \frac{1}{4} - \frac{1}{4p^s} \geq \frac{1}{4}\left(1 - \frac{1}{3}\right) = \frac{1}{6}.$$

D'ailleurs, pour $p \geq 5$ cette constante est encore plus grande, par exemple,

$$\frac{Edimh_q(n)}{n} \geq \begin{cases} \frac{1}{4}\left(1-\frac{1}{5}\right) = \frac{1}{5} = 0.2 & \text{si } p = 5, \\ \frac{1}{4}\left(1-\frac{1}{7}\right) = \frac{3}{14} = 0.214... & \text{si } p = 7, \\ \frac{1}{4}\left(1-\frac{1}{11}\right) = \frac{5}{22} = 0.227... & \text{si } p = 11, \text{ etc...} \end{cases}$$

Deuxième cas : $p = 2$.

Comme dans ce cas $\delta_{p^s} = 1$, l'équation (4.35) devient

$$\frac{Edimh_q(n)}{n} = \frac{1}{4} - \frac{1}{4p^s} + \frac{1}{4p^s(p^s+1)} + \frac{Edimh_q(r)}{r}\left(\frac{1}{3} + \frac{1}{3p^s} - \frac{1}{3p^s(p^s+1)}\right).$$

Comme

$$\frac{1}{3} + \frac{1}{3p^s} - \frac{1}{3p^s(p^s+1)} = \frac{p^s(p^s+2)}{3p^s(p^s+1)} > 0,$$

nous obtenons que

$$\frac{Edimh_q(n)}{n} \geq \frac{1}{4} - \frac{1}{4p^s} + \frac{1}{4p^s(p^s+1)}.$$

Si $s=1$,

$$\frac{Edimh_q(n)}{n} \geq \frac{1}{4} - \frac{1}{8} + \frac{1}{24} = \frac{1}{6}.$$

Si $s \geq 2$,

$$\frac{Edimh_q(n)}{n} \geq \frac{1}{4} - \frac{1}{4p^s} \geq \frac{1}{4} - \frac{1}{4 \cdot 2^2} = \frac{3}{16} > \frac{1}{6}.$$

Dans les deux cas ($p \geq 3$ et $p = 2$) nous obtenons

$$\frac{Edimh_q(n)}{n} \geq \frac{1}{6}. \tag{4.36}$$

Nous avons donc montré le théorème suivant :

Théorème 4.8.8 *Soit q une puissance d'un nombre premier p. Il existe une constante $\epsilon > 0$ telle que $Edimh_q(n) \geq \epsilon n$ pour tout $n \in \mathbb{N}^*$ tel que $(n,q) > 1$.*

Comme nous venons de le voir, on peut choisir pour ϵ une constante supérieure ou égale à $\frac{1}{6}$.

4.8.4 Cas général

Nous avons donc montré le théorème suivant :

Théorème 4.8.9 *Soit q une puissance d'un nombre premier p. Nous avons :*

1. *pour tout $n \in \mathbb{N}^*$, $Edimh_q(n) = 0$ si et seulement si $n \in \mathcal{N}_q$,*
2. *il existe une constante $\epsilon > 0$ telle que $Edimh_q(n) \geq \epsilon n$ pour tout $n \in \mathbb{N}^*$, $n \notin \mathcal{N}_q$.*

Preuve. C'est une conséquence directe des théorèmes 4.6.5, 4.8.7 et 4.8.8. □

Compte tenu du corollaire 4.6.12, $\epsilon n \leq Edimh_q(n) \leq \frac{1}{3}n$ pour tout $n \geq 1$, $n \notin \mathcal{N}_q$. Donc pour les $n \geq 1$ n'appartenant pas à $\mathcal{N}_q$, $Edimh_q(n)$ est de l'ordre de n.

Comme nous l'avons mentionné (cf. section 3.5), notre algorithme pour calculer les groupes de permutations et d'automorphismes d'un code et pour déterminer l'équivalence et l'équivalence par permutation de deux codes (cf. section 3.7) fonctionne bien si le hull du code donné est assez petit pour pouvoir calculer son énumérateur des poids ; par exemple, dans le cas binaire la dimension du hull ne devrait pas excéder 30.

Soit $q = 2$. Si $n \in \mathcal{N}_q$, alors la dimension du hull est nulle pour tout code cyclique binaire de longueur n, donc notre algorithme fonctionnera bien pour ces codes (dans les limites raisonnables, disons, $n \leq 10000$, voir la section 3.10 pour quelques exemples de temps d'exécution). Sinon, pour $n \geq 180$, la dimension moyenne du hull des codes cycliques binaires de longueur n sera supérieure ou égale à $\approx \frac{n}{6} \geq \frac{180}{6} = 30$ (cf. formules 4.32 et 4.36), donc notre algorithme ne fonctionnera pas pour la plupart de ces codes.

Si $\mathcal{N}_q$ était “grand” par rapport à $\mathbb{N}$, il y aurait quand même “beaucoup” de codes cycliques accessibles par notre algorithme. Malheureusement, tout laisse supposer que $\mathcal{N}_q$ est négligeable par rapport à $\mathbb{N}$ (cf. section 4.9).

4.9 Une conjecture sur $\mathcal{N}_q$

Soit $q \geq 2$. Dans cette section nous allons motiver notre conviction que $\mathcal{N}_q$ est "petit" par rapport à $\mathbb{N}$.

Nous donnons d'abord la définition de la notion "presque tous les entiers".

Définition 4.9.1 *[HW68, p. 8] Soit P une propriété d'un entier positif non nul, et soit x un nombre réel. Notons $P(x)$ le nombre des entiers positifs non nuls inférieurs ou égaux à x satisfaisant la propriété P, i.e.*

$$P(x) = |\{0 < n \leq x \mid n \textit{ satisfait la propriété } P\}|.$$

Nous dirons que presque tous les entiers positifs non nuls *satisfont P si*

$$P(x) \sim x \textit{ quand } x \to \infty,$$

autrement dit, si le nombre des entiers positifs non nuls inférieurs ou égaux à x ne satisfaisant pas P est $o(x)$ (cf. notations p. 7).

Par exemple, le nombre $\pi(x)$ des nombres premiers inférieurs ou égaux à x est $o(x)$, donc presque tous les entiers positifs non nuls sont des nombres composés.

Notation 4.9.2 Soit $n \geq 1$. Nous notons $\mathcal{N}_q(n) = |\{m \leq n \mid m \in \mathcal{N}_q\}|$.

La figure 4.4 montre le rapport $\frac{\mathcal{N}_q(n)}{n}$ pour $q = 2, 3, 5$, $n \leq 10^4$ et $n \leq 5 \times 10^6$. Nous voyons que ce rapport décroît. Cela (cf. aussi les commentaires sur l'exemple 4.7.28, p. 165) nous a conduit à la conjecture suivante :

Conjecture 4.9.3 *Soit $q \geq 2$. Alors $\mathcal{N}_q(n) = o(n)$. Autrement dit, $\mathcal{N}_q$ est négligeable par rapport à $\mathbb{N}$, c'est-à-dire presque tous les entiers positifs non nuls n'appartiennent pas à $\mathcal{N}_q$.*

Malheureusement, nous n'avons pas pu l'établir.

Si cette conjecture était vraie, il existerait "peu" d'entiers positifs non nuls n tels que $Edimh_q(n)$ soit "petit", c'est-à-dire pour toute constante $c \geq 0$ nous aurions l'égalité

$$|\{m \leq n \mid Edimh_q(m) \leq c\}| = o(n).$$

Cela signifierait qu'il existerait "peu" d'entiers positifs non nuls n tels que notre algorithme (cf. section 3.7) fonctionne pour la plupart de codes cycliques de longueur n sur $\mathbf{F}_q$.

Malgré cela, pour les petites longueurs n, $\mathcal{N}_q(n)$ est quand même signifiant. La figure 4.5 donne les valeurs de $\mathcal{N}_q(n)$ pour $2 \leq q \leq 1000$ et pour $n = 10000$ et $n = 10^6$. Nous voyons que, avec peu d'exceptions, pour $n = 10000$, $\frac{\mathcal{N}_q(n)}{n}$ se situe entre 0.085 et 0.335, et pour $n = 10^6$, entre 0.06 et 0.24.

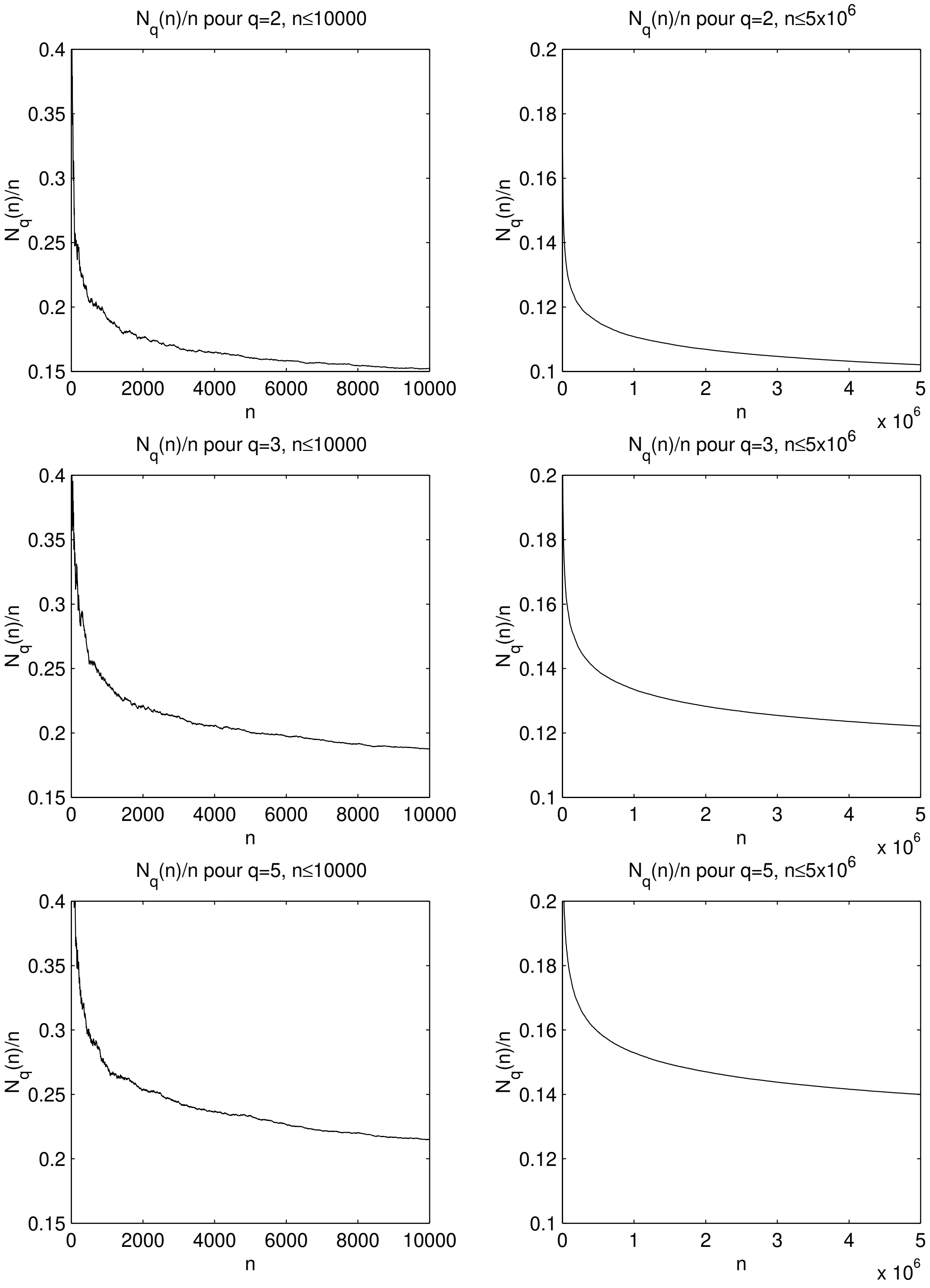

FIG. 4.4 – *Les valeurs de* $\frac{\mathcal{N}_q(n)}{n}$ *pour* $q = 2, 3, 5$, $n \leq 10^4$ *et* $n \leq 5 \times 10^6$

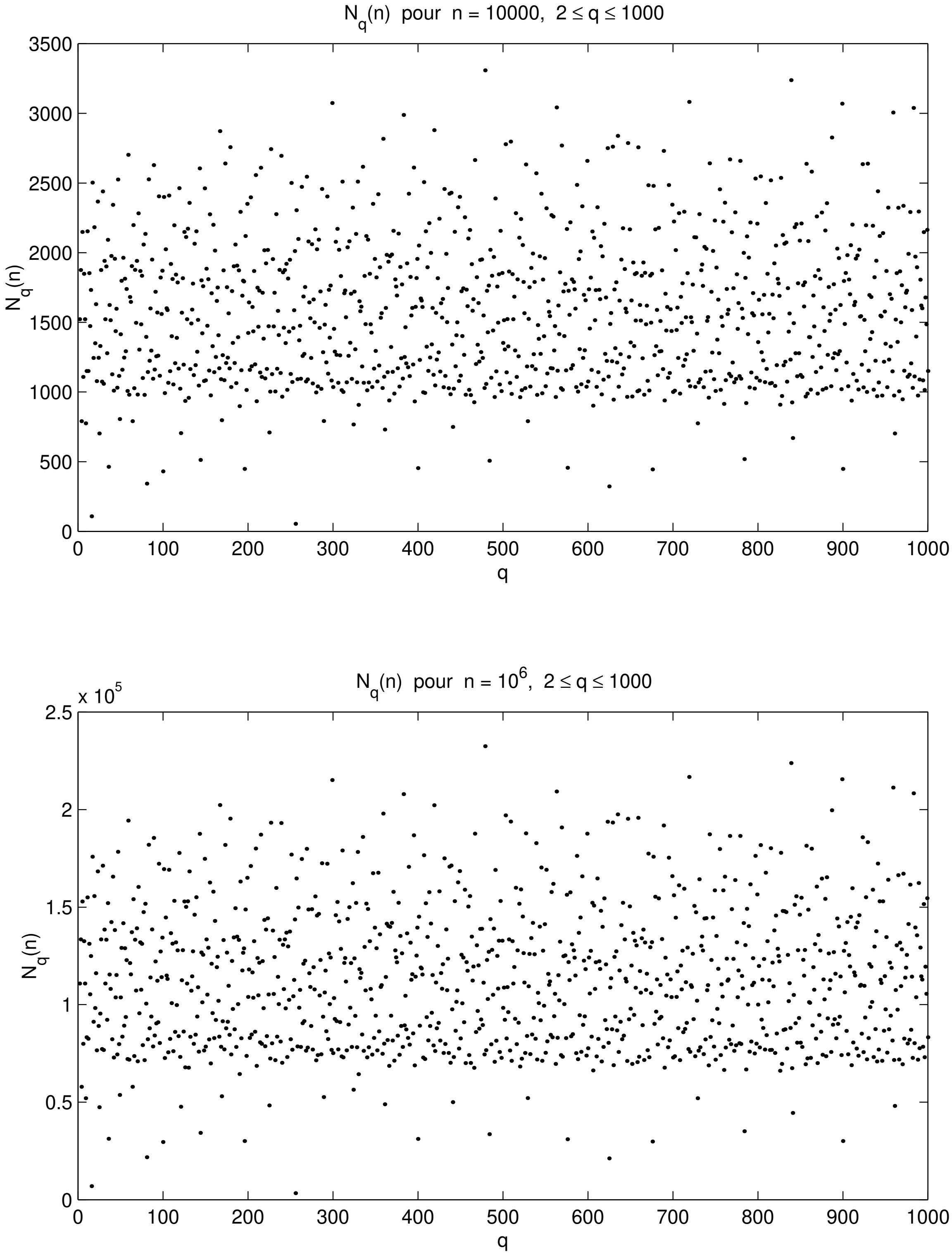

FIG. 4.5 – *Les valeurs de* $\mathcal{N}_q(n)$ *pour* $n = 10000$ *et* $n = 10^6$ *et pour* $2 \leq q \leq 1000$

Conclusions et perspectives

Nous avons présenté dans ce travail un algorithme pour calculer les groupes de permutations et d'automorphismes d'un code et pour déterminer l'équivalence et l'équivalence par permutation de deux codes. Il est basé sur la méthode des partitions de Jeffrey S. Leon et sur l'algorithme de séparation du support de Nicolas Sendrier.

Il utilise la notion d'*invariant* des codes. L'invariant proposé par Sendrier est l'énumérateur des poids du hull des codes, notre algorithme est donc limité par la taille du hull des codes donnés.

La plupart des codes ont un petit hull, mais dans certaines classes de codes (par exemple, les codes cycliques) il existe beaucoup de codes de grand hull.

Est-il possible de trouver un autre invariant qui serait applicable pour les codes dont le hull est de grande taille ? De trouver un tel invariant ou une méthode particulière au moins pour certaines classes de codes ?

Par exemple, la distance minimale du hull est aussi un invariant. Il existe des algorithmes probabilistes pour trouver la distance minimale des grands codes (cf. [CC98, Leo88]). Le problème, c'est que ce sont des algorithmes probabilistes, nous ne sommes donc pas sûrs à cent pour cent d'obtenir un résultat correct. Est-il possible d'utiliser cet invariant dans notre algorithme en s'appuyant sur ces algorithmes probabilistes pour calculer la distance minimale du hull des codes ? Quelle influence cette incertitude peut avoir sur le fonctionnement de notre algorithme et sur la validité de ses résultats ?

Notre algorithme est implanté pour calculer le groupe de permutations d'un code linéaire binaire. Nous l'avons testé et étudié les heuristiques possibles et les performances par rapport à divers paramètres. Il reste un travail très important de programmation (ainsi que la réalisation de nombreux tests) des cas restants : l'équivalence des codes linéaires binaires, le groupe de permutations et l'équivalence par permutation des codes linéaires non binaires, le groupe d'automorphismes et l'équivalence des codes linéaires non binaires. Il faut les programmer, tester et étudier les heuristiques possibles. Il ne restera alors qu'à écrire un mode d'emploi et qu'à offrir les programmes pour utilisation à la communauté scientifique.

Comme la dimension du hull des codes est un paramètre important pour notre algorithme, et comme nous avons remarqué que les codes cycliques possèdent souvent un hull de grande taille, nous avons essayé d'étudier la dimension du hull des codes cycliques. Nous avons montré que la dimension moyenne du hull des codes cycliques de longueur n donnée sur un corps fini $\mathbf{F}_q$ donné est soit nulle (si et seulement si n appartient à l'ensemble des diviseurs des entiers de la forme $q^m + 1$, $m > 0$, notons $\mathcal{N}_q$ cet ensemble), soit de l'ordre de n. Nous avons étudié la structure de $\mathcal{N}_q$ et nous avons divisé $\mathcal{N}_q$ en classes d'équivalence $\mathcal{P}_{q,\alpha}$, $\alpha \geq 0$. Nous conjecturons que $\mathcal{N}_q$ est "petit" par rapport à $\mathbb{N}$. Il faut continuer l'étude de $\mathcal{N}_q$, non seulement pour montrer la conjecture, mais aussi parce que l'étude de la structure de $\mathcal{N}_q$ est un problème intéressant de la théorie des nombres.

On peut commencer, par exemple, par l'étude de l'ensemble des nombres premiers de $\mathcal{N}_q$. En fait, les considérations heuristiques et empiriques nous ont conduit à une conjecture intéressante sur la distribution des nombres premiers, c'est-à-dire sur les proportions des nombres premiers dans $\mathcal{N}_q$ et dans les $\mathcal{P}_{q,\alpha}$, $\alpha \geq 0$ (nous ne l'avons pas présenté dans ce document). Ce serait intéressant de l'étudier plus profondément et d'établir cette conjecture. Après cela, se posera la question de savoir comment passer de l'ensemble des nombres premiers de $\mathcal{N}_q$ à $\mathcal{N}_q$ tout entier.

De même, nous avons quelques résultats exprimant $\mathcal{N}_{q^m}$, $m > 1$, en fonction de $\mathcal{N}_q$ (ils ne sont pas présentés non plus dans ce document). Comment est-il possible d'exploiter ces résultats?

Après avoir étudié la moyenne de la dimension du hull des codes cycliques, l'étape suivante serait de nous intéresser à la distribution des dimensions du hull des codes cycliques. Ou, en continuant sur la voie de la théorie des probabilités, à la variance et aux autres moments de la dimension du hull des codes cycliques.

La moyenne $Edimh_q(n)$ de la dimension du hull des codes cycliques de longueur n sur $\mathbf{F}_q$ n'est définie, évidemment, que si q est une puissance d'un nombre premier. Mais les propositions 4.6.6 et 4.6.11 l'exprime en fonction de $B_{n,q}$ qui peut être défini pour $q \geq 1$ quelconque (cf. proposition 4.6.14). Est-il possible d'étendre cette définition sur $Edimh_q(n)$? Si c'est le cas, quel sens prend $Edimh_q(n)$ pour $q \geq 1$ quelconque?

Enfin, nous pensons qu'il est possible d'étendre l'étude de la dimension du hull pour d'autres classes de codes, par exemple, les codes BCH, les codes affine-invariants, etc...

Bibliographie

[Apo76] Tom M. Apostol. *Introduction to Analytic Number Theory.* Springer-Verlag, New York, 1976.

[BC67] Z.I. Borevitch, I.R. Chafarevitch. *Théorie des nombres.* Gauthier-Villars, Paris, 1967.

[BC82] Gregory Butler and John J. Cannon. *Computing in Permutation and Matrix Groups, I: Normal Closure, Commutator Subgroups, Series.* Math. Comp. 39, 1982, pp. 663–670.

[BC96] Thierry P. Berger et Pascale Charpin. *The permutation group of affine-invariant extended cyclic codes.* IEEE Transactions on Information Theory, vol. 42, n. 6, 1996, p. 2194-2210.

[BCFLS91] L. Babai, G. Cooperman, L. Finkelstein, E.M. Luks, and Á. Seress. *Fast Monte Carlo Algorithms for Permutation Groups.* Proc. 23^{rd} ACM STOC, 1991, pp. 90–100.

[BCFS91] L. Babai, G. Cooperman, L. Finkelstein, and Á. Seress. *Nearly Linear Time Algorithms for Permutation Groups with a Small Base.* Proc. of the 1991 International Symposium on Symbolic and Algebraic Computation (ISSAC '91), Bonn, July 1991, ACM Press, pp. 200–209.

[Ber68] C. Berge. *Principes de combinatoire.* Dunod, Paris, 1968.

[Ber96] Thierry Berger. *Groupes d'automorphismes et de permutations des codes affine-invariants.* Habilitation à diriger des recherches, Université de Limoges, Janvier 1996, 56p.

[BFP89] Cynthia A. Brown, Larry Finkelstein and Paul W. Purdom, Jr. *A New Base Change Algorithm for Permutation Groups.* SIAM J. Computing 18(5), October 1989, pp. 1037–1047.

[BL85] G. Butler and C.W.H. Lam. *A General Backtrack Algorithm for the Isomorphism Problem of Combinatorial Objects.* J. Symb. Comp. 1, 1985, pp. 363–381.

[BLS97] László Babai, Eugene M. Luks, and Ákos Seress. *Fast management of permutation groups, I.* SIAM J. Computing, Vol. 26, No. 5, October 1997, pp. 1310–1342.

[BR74] Alain Bouvier, Denis Richard. *Groupes: Observation, Théorie, Pratique.* Hermann, Paris, 1974.

[BS89] K. Bulota, P. Survila. *Algebra ir skaičių teorija, I.* Mokslas, Vilnius, 1989, 411p.

[BS90] K. Bulota, P. Survila. *Algebra ir skaičių teorija, II.* Mokslas, Vilnius, 1990, 414p.

[But76] Gregory Butler. *The Schreier algorithm for matrix groups.* SYMSAC'76 (Proc. 1976 A.C.M. Symposium on Symbolic and Algebraic Computation, Yorktown Heights, New York, 1976), R.D. Jenks (ed.), A.C.M., New York, 1976, pp. 167–170.

[But82] Gregory Butler. *Computing in Permutation and Matrix Groups, II: Backtrack Algorithm.* Math. Comp. 39, 1982, pp. 671–680.

[But83] Gregory Butler. *Computing normalizers in permutation groups.* J. Algorithms 4, 1983, pp. 163–175.

[Car56] Robert D. Carmichael. *Introduction to the theory of groups of finite order.* Dover Publications, Inc., New York, 1956.

[CC95] Anne Cantaut, Florent Chabaud. *A new algorithm for finding minimum-weight words in a linear code : Application to primitive narrow-sense BCH codes of length 511.* Rapport de recherche N° 2685, INRIA-Rocquencourt, Octobre 1995.

[CC98] Anne Cantaut, Florent Chabaud. *A new algorithm for finding minimum-weight words in a linear code : Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511.* IEEE Transactions on Information Theory, Vol. 44, No. 1, January 1998, pp. 367–378.

[CF92a] Gene Cooperman and Larry Finkelstein. *New Methods for Using Cayley Graphs in Interconnection Networks.* Discrete Applied Mathematics 37/38 (special issue on Interconnection Networks), 1992, pp. 95–118.

[CF92b] Gene Cooperman and Larry Finkelstein. *A Fast Cyclic Base Change for Permutation Groups.* Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC-92), ACM Press, New York, 1992, pp. 224–232.

[CF92c] Gene Cooperman and Larry Finkelstein. *Randomized Algorithms for Permutation Groups.* Centrum Wissenschaft Institut Quarterly (CWI), Netherlands, June, 1992, pp. 107–125.

[CF93] Gene Cooperman and Larry Finkelstein. *Combinatorial Tools for Computational Group Theory.* Proceedings of the DIMACS Workshop on Groups and Computation, L. Finkelstein and W.M. Kantor (eds.), AMS, Providence, RI, DIMACS-AMS 11, 1993, pp. 53–86.

[CF94] Gene Cooperman and Larry Finkelstein. *A Random Base Change Algorithm for Permutation Groups.* J. Symbolic Computation 17, 1994, pp. 513–528.

[CFP89] G. Cooperman, L. Finkelstein and P.W. Purdom. *Fast Group Membership using a Strong Generating Test for Permutation Groups.* Computers and Mathematics (E. Kaltofen and S.M. Watt, eds.), Springer-Verlag, New York, 1989, pp. 27–36.

[CFS91] G. Cooperman, L. Finkelstein and N. Sarawagi. *Applications of Cayley Graphs.* Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-8, Tokyo, 1990), Springer Verlag, Lecture Notes in Computer Science 508, S. Sakata (ed.), 1991, pp. 367–378.

[CH92] John J. Cannon and George Havas. *Algorithms for groups.* Australian Computer Journal, Vol. 24, No. 2, May 1992, pp. 51–60.

[CL94] Pascale Charpin et Françoise Levy-dit-Vehel. *On self-dual affine-invariant codes.* Journal of Combinatorial Theory, Série A, 67, 2, 1994, pp. 223–244.

[Com74] Louis Comtet. *Advanced Combinatorics.* D. Reidel Publishing Company, Dordrecht, Holland, 1974, 343p.

[For96] Scott Fortin. *The graph isomorphism problem.* Technical Report TR 96–20, Department of Computing Science, The University of Alberta, Canada, July 1996.

[Gol71] L. Goldstein. *Density questions in algebraic number theory.* Am. Math. Monthly 78, April 1971, pp. 342–351.

[GPR74] J. Genet, G. Pupion et M. Repussard. *Probabilités, statistiques et sondages.* Librairie Vuibert, Paris, 1974.

[Hal59] Marshall Hall, Jr. *The theory of groups.* The Macmillan Company, New York, 1959.

[Hav91] George Havas. *Coset enumeration strategies.* Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation, Stephen M. Watt, editor, ACM Press, New York, 1991, pp. 191–199.

[Hof82] C.M. Hoffmann. *Group-theoretic algorithms and graph isomorphism.* Lecture Notes in Computer Science 136, Springer Verlag, 1982.

[Huf95] W.C. Huffman. *The automorphism groups of generalized quadratic residue codes.* IEEE Transactions on Information Theory, vol. 41, 1995, pp. 378–386.

[Huf98] W.C. Huffman. *Codes and groups.* Handbook of Coding Theory, V. Pless, W.C. Huffman, and R. Brualdi eds., Elsevier, 1998.

[HW68] G.H. Hardy, E.M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, Ely House, London, 1968.

[IR82] Kenneth Ireland, Michael Rosen. *A Classical Introduction to Modern Number Theory.* Springer-Verlag, New York, 1982.

[Jer86] M. Jerrum. *A compact representation for permutation groups.* J. Algorithms, 7, 1986, pp. 60–78.

[KMS84] D. Kornhauser, G. Miller and P. Spirakis. *Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups and Applications.* Proc. 25^{th} IEEE FOCS, 1984, pp. 241–250.

[Knu75] Donald E. Knuth. *Estimating the Efficiency of Backtrack Programs.* Math. Comp. 29, 1975, pp. 121–136.

[Lau93] R. Laue. *Construction of combinatorial objects — a tutorial.* Bayreuther Math. Schr. 43, 1993, pp. 53–96.

[Leo79] Jeffrey S. Leon. *An algorithm for computing the automorphism group of a Hadamard matrix.* J. Combin. Theory, Series A, vol. 27, 1979, pp. 289–306.

[Leo80] Jeffrey S. Leon. *On an Algorithm for Finding a Base and a Strong Generating Set for a Group Given by Generating Permutations.* Math. Comp. 35, 1980, pp. 941–974.

[Leo82] Jeffrey S. Leon. *Computing Automorphism Groups of Error-Correcting Codes.* IEEE Transactions on Information Theory, Vol. IT–28, No. 3, May 1982, pp. 496–511.

[Leo84] Jeffrey S. Leon. *Computing automorphism groups of combinatorial objects.* In : Computational group theory, Academic Press, London, New York, 1984, pp. 321–335.

[Leo88] Jeffrey S. Leon. *A probabilistic algorithm for computing minimum weights of large error-correcting codes.* IEEE Transactions on Information Theory, Vol. 34, No. 5, September 1988, pp. 1354–1359.

[Leo91] Jeffrey S. Leon. *Permutation Group Algorithms Based on Partitions, I: Theory and Algorithms.* J. Symbolic Computation 12, 1991, pp. 533–583.

[Leo92] Jeffrey S. Leon. *Partition backtrack programs : User's manual.* `http://www.math.uic.edu/~leon/`

[Leo97] Jeffrey S. Leon. *Partitions, Refinements, and Permutation Group Computation.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 28, 1997, pp. 123–158.

[Lev94] Françoise Levy-dit-Vehel. *Divisibilité des codes cycliques : Applications et prolongements.* Thèse de docteur de l'Université Paris VI, Octobre 1994, 197p.

[Lit70] D.E. Littlewood. *A University Algebra.* 2me édition, Dover Publications, Inc., New York, 1970, 324p.

[LN83] R. Lidl, H. Niederreiter. *Finite Fields.* Cambridge University Press, 1983.

[LS98] Pierre Loidreau, Nicolas Sendrier. *Some Weak Keys in McEliece Public-Key Cryptosystem.* IEEE International Symposium on Information Theory, ISIT'98, Boston, USA, 1998.

[Mat61] G.B. Mathews, M.A. *Theory of Numbers.* Chelsea publishing company, New York, 1961.

[McE78] R. J. McEliece. *A public-key cryptosystem based on algebraic coding theory.* DSN Progress Report, Jet Prop. Lab., California Inst. Technol., Pasadena, CA, Janvier 1978, pp. 114–116.

[McK78] Brendan D. McKay. *Computing automorphisms and canonical labellings of graphs.* Lecture Notes in Math. 686, 1978, pp. 223–232.

[McK79] Brendan D. McKay. *Hadamard equivalence via graph isomorphism.* Discrete Math. 27, 1979, pp. 213–214.

[McK81] Brendan D. McKay. *Practical graph isomorphism.* Congressus Numerantium 30, 1981, pp. 45–87.

[Mil75] J.C.P. Miller. *On factorisation, with a suggested new approach.* Mathematics of Computation 29 (129), January 1975, pp. 155–172.

[MS77] F.J. MacWilliams, N.J.A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1977.

[MS78] B.D. McKay and R.G. Stanton. *Isomorphism of two large designs.* Ars Combinatoria 6, 1978, pp. 87–90.

[Nag81] Trygve Nagell. *Introduction to Number Theory.* Chelsea publishing company, New York, 1981.

[Nie86] H. Niederreiter. *Knapsack-type cryptosystems and algebraic coding theory.* Problems of Control and Information Theory, vol. 15, n°2, 1986, pp. 157–166.

[PR97] E. Petrank and R.M. Roth. *Is code equivalence easy to decide ?* IEEE Transactions on Information Theory, 43(5), September 1997, pp. 1602–1604.

[PSW80] Carl Pomerance, J.L. Selfridge et Samuel S. Wagstaff, Jr. *The pseudoprimes to* $25 \cdot 10^9$. Mathematics of Computation 35(151), July 1980, pp. 1003–1026.

[PW72] W. Wesley Peterson, E. J. Weldon, Jr. *Error-Correcting Codes.* Second edition, The MIT Press, Cambridge and London, 1972.

[PW95] Odile Papini, Jacques Wolfmann. *Algèbre Discrète et Codes Correcteurs.* Springer-Verlag, Berlin, 1995.

[PZ61] Charles Pisot, Marc Zamansky. *Mathématiques générales. Algèbre-analyse.* Dunod, Paris, 1961.

[Rab80] Michael O. Rabin. *Probabilistic algorithm for testing primality.* Journal of Number Theory 12, 1980, pp. 128–128.

[Ros84] Kenneth H. Rosen. *Elementary Number Theory and its Applications.* Addison-Wesley Publishing Company, 1984.

[Sen91] Nicolas Sendrier. *Codes correcteurs d'erreurs à haut pouvoir de correction.* Thèse de docteur de l'Université Paris VI, Décembre 1991.

[Sen95] Nicolas Sendrier. *On the dimension of the hull.* INRIA-Rocquencourt, Rapport de recherche n°2682, Octobre 1995.

[Sen96] Nicolas Sendrier. *Un algorithme pour trouver la permutation entre deux codes binaires équivalents.* INRIA-Rocquencourt, Rapport de recherche n°2853, Avril 1996.

[Sen97a] Nicolas Sendrier. *On the dimension of the hull.* SIAM Journal on Applied Mathematics, 10(2), May 1997, pp. 282–293.

[Sen97b] Nicolas Sendrier. *Finding the permutation between equivalent binary codes.* In : IEEE Conference, ISIT'97, Ulm, Germany, June 1997.

[Sen99a] Nicolas Sendrier. *The support splitting algorithm.* INRIA-Rocquencourt, Rapport de recherche n°3637, Mars 1999.

[Sen99b] Nicolas Sendrier. *Finding the permutation between equivalent linear codes : the support splitting algorithm.* IEEE Transactions on Information Theory (à paraître).

[SeS99] Nicolas Sendrier, Gintaras Skersys. *Permutation groups of error-correcting codes.* Proceedings of Workshop on Coding and Cryptography, INRIA, Paris, 1999, p. 33–41.

[Sha78] Daniel Shanks. *Solved and Unsolved Problems in Number Theory.* Chelsea publishing company, New York, 1978.

[Sim70] Charles C. Sims. *Computational methods in the study of permutation groups.* In : Computational Problems in Abstract Algebra, J. Leech, ed., Pergamon Press, Oxford and New York, 1970, pp. 169–183.

[Sim71a] Charles C. Sims. *Determining the conjugacy classes of a permutation group.* Proc. of the Symposium on Computers in Algebra and Number Theory, Amer. Math. Soc., New York, 1971, pp. 191–195.

[Sim71b] Charles C. Sims. *Computation with permutation groups.* Proc. of the Second Symposium on Symbolic and Algebraic Manipulation, Assoc. for Computing Mach., New York, 1971, pp. 23–28.

[Sim78a] Charles C. Sims. *The role of algorithms in the teaching of algebra.* In : Topics in Algebra, Lecture Notes in Math., Vol. 697, Springer-Verlag, Berlin, 1978, pp. 95–107.

[Sim78b] Charles C. Sims. *Some group-theoretic algorithms.* In : Topics in Algebra, Lecture Notes in Math., Vol. 697, Springer-Verlag, Berlin, 1978, pp. 108–124.

[Sim78c] Charles C. Sims. *A method for constructing a group from a subgroup.* In : Topics in Algebra, Lecture Notes in Math., Vol. 697, Springer-Verlag, Berlin, 1978, pp. 125–130.

[Ske96] Gintaras Skersys. *Détermination des codes linéaires binaires équivalents à un code cyclique.* Mémoire de DEA, Université de Limoges, Juillet 1996.

[Ten90] Gérald Tenenbaum. *Introduction à la théorie analytique et probabiliste des nombres.* Institut Elie Cartan, Université de Nancy I, 1990.

[Wil89] Herbert S. Wilf. *Algorithmes et Complexité.* Masson, Paris, 1989.

Index

Résumé : Nous présentons dans ce travail un algorithme pour calculer les groupes de permutations et d'automorphismes d'un code correcteur d'erreurs et pour déterminer l'équivalence et l'équivalence par permutation des codes. Il est basé sur la méthode des partitions de Jeffrey S. Leon et sur l'algorithme de séparation du support de Nicolas Sendrier.

La méthode des partitions de Leon nous semble très technique, nous avons donc essayé de la clarifier en la présentant d'une manière différente de celle de Leon.

Notre algorithme est limité par la dimension du hull (l'intersection du code avec son dual) des codes concernés qui doit être inférieure à une vingtaine. Si c'est le cas, notre algorithme fonctionne efficacement, par exemple, pour les codes linéaires binaires de longueur jusqu'à 10000.

De plus, nous avons étudié la dimension du hull des codes cycliques. Nous avons montré que la dimension moyenne du hull des codes cycliques de longueur n donnée sur un corps fini $\mathbf{F}_q$ donné est soit nulle, soit de l'ordre de n, et elle est nulle si et seulement si n est un diviseur d'un entier de la forme $q^m + 1$, $m > 0$. Nous avons étudié l'ensemble des diviseurs des entiers de cette forme.

Calculation of the automorphism group of the codes. Determination of the equivalence of the codes

Abstract : We present an algorithm for computing the permutation and automorphism groups of an error-correcting code, and for determining the equivalence and the permutation-equivalence of two codes. It is based on the partition method of Jeffrey S. Leon and on the support splitting algorithm of Nicolas Sendrier.

It seems to us that the partition method of Leon is very technical, so we presented it in a different manner than Leon and tried to clarify it.

Our algorithm is limited by the dimension of the hull (the intersection of the code and its dual code) of the concerned codes. This dimension must be less than about twenty. If it is so, our algorithm works efficiently, for example, for the linear binary codes of length up to 10000.

Furthermore, we studied the dimension of the hull of the cyclic codes. We proved that the average dimension of the hull of the cyclic codes of given length n over a given finite field $\mathbf{F}_q$ is either zero or of the same order as n, and it is zero if and only if n is a divisor of an integer of the form $q^m + 1$, $m > 0$. We studied the set of the divisors of the integers of this form.

Discipline : Mathématiques et applications.

Mots-clefs : Codes correcteurs d'erreurs, groupe de permutations d'un code, groupe d'automorphismes d'un code, équivalence des codes, équivalence par permutation des codes, base et ensemble fort de générateurs (BSGS), recherche arrière ("backtrack search"), algorithme, méthode des partitions de J. Leon, raffinement, algorithme de séparation du support de N. Sendrier, signature, hull, dimension du hull des codes cycliques.

Laboratoire : Laboratoire d'arithmétique, de calcul formel et d'optimisation (LACO) UPRESA CNRS N°6090, 123 avenue Albert Thomas, 87060 Limoges CEDEX, France