MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Algorithme de décodage de codes correcteurs d'erreurs : exemple des codes de Reed Muller

Pierre-Louis CAYREL et Rafael FOURQUET

Université Paris VIII
Département de Mathématiques
2, rue de la Liberté
93526 - SAINT-DENIS cedex 02, France

Contact : cayrelpierrelouis@gmail.com, rafakr@gmail.com

Résumé
En 1946, Richard Hamming travaille sur un modèle de calculateur à carte perforée de faible fiabilité. Si, durant la semaine, des ingénieurs peuvent corriger les erreurs, les périodes chômées comme la fin de semaine voient les machines s'arrêter invariablement sur des bogues. La frustration de Hamming le conduit à inventer le premier code correcteur véritablement efficace. Cette période correspond à la naissance de la théorie de l'information. Claude Shannon formalise cette théorie comme une branche des mathématiques. Hamming développe les prémisses de la théorie des codes et décrit sa solution comme un exemple. Après avoir introduit la théorie des codes en décrivant ces principales définitions, nous présenterons l'algorithme de Kabatianski-Tavernier pour le décodage des codes de Reed-Muller d'ordre 1.

Abstract
In 1946, Richard Hamming was working on a model of calculator using punched card of low reliability. If, during the week, engineers can correct errors, during holiday periods such as weekend saw the machines invariably stop because of bugs. Hamming frustration led him to invent the first truly effective error correcting code. This period corresponds to the birth of information theory. Claude Shannon formalized this theory as a branch of mathematics. Hamming develops the premise of the coding theory and describes its solution as an example. After introducing the coding theory by describing the key definitions, we present the Kabatianski-Tavernier's algorithm for decoding Reed-Muller codes of order 1.

Mots-clés : Codes correcteurs d'erreurs, codes de Reed-Muller, fonctions Booléennes

Keywords: Error correcting codes, Reed-Muller codes, boolean functions

---

1. Introduction à la théorie des codes

1.1. Codes correcteurs d'erreurs inéaires

Les codes correcteurs d'erreurs sont un outil visant à améliorer la fiabilité des transmissions sur un canal bruité. La méthode qu'ils utilisent consiste à envoyer sur le canal plus de données que la quantité d'information à transmettre. Une redondance est ainsi introduite. Si cette redondance est structurée de manière exploitable, il est alors possible de corriger d'éventuelles erreurs introduites par le canal. On peut alors, malgré le bruit, retrouver l'intégralité des informations transmises au départ.
Une grande famille de codes correcteurs d'erreurs est constituée des codes par blocs. Pour ces codes l'information est d'abord coupée en blocs de taille constante et chaque bloc est transmis indépendamment des autres, avec une redondance qui lui est propre. La plus grande sous-famille de ces codes rassemble ce que l'on appelle les codes linéaires.

1.2. Définitions

Définition 1.1 *Soit un alphabet* A, *soit* A^* *l'ensemble des vecteurs formés à partir des éléments de* A. *Un code* $\mathcal{C}$ *est un sous-ensemble de vecteurs de* A^*. $\mathcal{C}$ *sera dit binaire si* $A = \{0, 1\}$, *un élément de* $\mathcal{C}$ *est appelé mot de code.*

Définition 1.2 *On note* $\mathbb{F}_2$ *le corps à deux éléments* $\{0; 1\}$. *Soient* u *et* v *deux vecteurs binaires de taille* n ($u = (u_i)_{i=1,\dots n}$ *avec* $u_i = 0$ *ou* 1, *idem pour* v). *On définit :*
- *le poids de* u (*noté* $wt(u)$) *comme le nombre de 1 dans* u. *On définit ainsi une norme sur* $\mathbb{F}_2^n$,
- *le support de* u *noté* $\mathrm{supp}(u)$ *comme l'ensemble des indices* i *tels que* $u_i = 1$,
- *la distance de Hamming de* u *et* v (*notée* $d(u, v)$) *comme le cardinal de l'ensemble des indices* i *tels que* u_i *soit différent de* v_i. *La fonction* $d(.,.)$ *ainsi définie vérifie les propriétés d'une distance sur* $\mathbb{F}_2^n$. *On a* $d(u, v) = wt(u + v)$.

La distance minimale (notée d*) d'un code est le minimum des distances de Hamming entre les mots du code pris distincts deux à deux.*
C'est aussi le poids du mot (non nul) de plus petit poids du code.

Définition 1.3 *On notera* $[n, k, d]$ *un code de longueur* n *de dimension* k *et de distance minimale* d.

Définition 1.4 *Soit* $\mathcal{C}$ *un code linéaire* $[n, k, d]$ *sur* $\mathbb{F}_q$, *une matrice génératrice* $\mathcal{G}$ *est une matrice* $(k \times n)$ *dont les lignes forment une base de* $\mathcal{C}$. *Si* c *appartient à* $\mathcal{C}$ *alors* c *est une combinaison linéaire de lignes de* $\mathcal{G}$. *Si* $\mathcal{G} = (I_k | A_{k\times(n-k)})$ *avec* I_k *la matrice identité de dimension* k *et* $A_{k\times(n-k)}$ *une matrice* $(k, n-k)$, *on dit que* $\mathcal{G}$ *est sous forme systématique.*

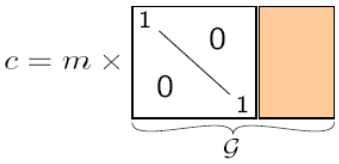

FIGURE 1 – $\mathcal{G}$: Matrice génératrice sous forme systématique

Définition 1.5 *Soit* $\mathcal{C}$ *un code (non nécessairement linéaire) dans* $\mathbb{F}^n$. *Le code dual de* $\mathcal{C}$, *noté* $\mathcal{C}^\perp$, *est le code :*

$$\mathcal{C}^\perp = \{x \in \mathbb{F}^n | x * c = 0, \text{ pour tout } c \in \mathcal{C}\},$$

où $x * c$ *dénote le produit usuel.*

Une matrice génératrice $\mathcal{H}$ du code dual $\mathcal{C}^{\perp}$ d'un code linéaire $\mathcal{C}$ est appelée matrice de parité de $\mathcal{C}$. On a

$$\mathcal{C} = \ker \mathcal{H}.$$

Si la matrice génératrice $\mathcal{G}$ est sous forme systématique alors :

$$\mathcal{H} = \left(-A^{T}_{(n-k)\times k} | I_{n-k}\right),$$

où A^{T} est la transposée de A.

Définition 1.6 *Deux codes sont dits équivalents si leurs deux matrices génératrices se déduisent l'une de l'autre par permutation de colonnes.*

Pour encoder un message m on le multiplie par une matrice génératrice $\mathcal{G}$. On obtient un mot de code c. Ensuite après passage par un canal bruité, une erreur e s'ajoute au mot de code c on obtient $c' = c + e$.

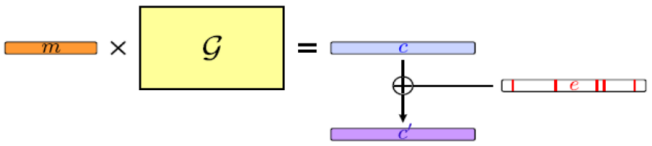

FIGURE 2 – Décodage

En calculant le produit $\mathcal{H}c'$ on obtient un vecteur s qu'on appelle le syndrome de l'erreur e.

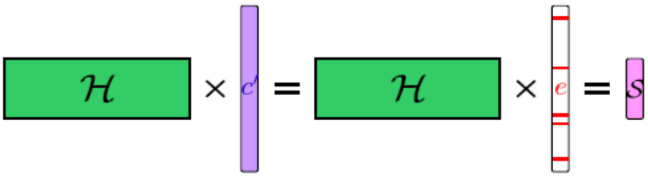

FIGURE 3 – Syndrome

Décoder dans un code $\mathcal{C}$ désigne l'action d'associer un mot du code (un élément du sous-espace vectoriel) à un mot de l'espace vectoriel. On cherche le plus souvent à décoder en associant à un mot le mot de code duquel il est le plus proche. Cependant il faut d'abord décider du sens que l'on veut donner à l'expression le plus proche.
Dans le cas du décodage d'un message transmis le long d'un canal bruité, on s'intéresse essentiellement au *décodage au maximum de vraisemblance*. Cela consiste à toujours associer le mot de code qui a la plus grande probabilité d'avoir donné ce mot en étant transmis sur le canal.

Définition 1.7 *La majoration suivante est vérifiée pour tous les codes linéaires. Elle se nomme borne du singleton : $n - k \geq d - 1$. Si la borne de Singleton est atteinte, le code est dit **MDS**.*

1.3. Codes détecteurs d'erreurs et code de parité

Les codes détecteurs d'erreurs consistent à détecter des erreurs lors d'une transmission sur un canal bruité. On peut protéger autant de symboles que l'on veut avec un symbole. Ces codes sont très simples et économes donc très utilisés (calcul de CRC dans les fichiers informatiques, numéro RIB, numéro de sécurité sociale, transmission modems). Le code de parité est un code $[k+1, k, 2]$, dont le principe est d'ajouter un bit au mot de sorte que le nombre de bits à 1 du mot soit pair. Il suffit alors de vérifier que le poids du mot reçu est pair pour savoir si une erreur de transmission a été commise. On est alors en mesure de dire s'il y a eu une erreur mais on ne peut pas la corriger. De plus on ne peut détecter qu'une seule erreur. Ce codage est très rapide mais ne permet aucune correction d'erreur. Il est particulièrement utilisé dans les modems, puisque chaque octet en contient un ce qui permet au modem de redemander un octet erroné, avec une perte de temps minimale.

Usuellement, on considère que le mot de code émis est celui se trouvant le plus près du mot reçu, ce qui revient à supposer que le minimum de lettres a été modifié. Ce procédé conduit à une erreur de décodage chaque fois que l'erreur est supérieure à la capacité corrective du code. La question naturelle est celle de la valeur de t correspondant au nombre maximum d'erreurs corrigibles.

Une interprétation géométrique donne un élément de réponse. les boules fermées de rayon t centrées sur les mots de code doivent être disjointes. La capacité de correction d'un code correspond au plus grand entier t vérifiant cette propriété, c'est aussi le plus grand entier strictement plus petit que $\frac{d}{2}$ où d représente la distance minimale du code.

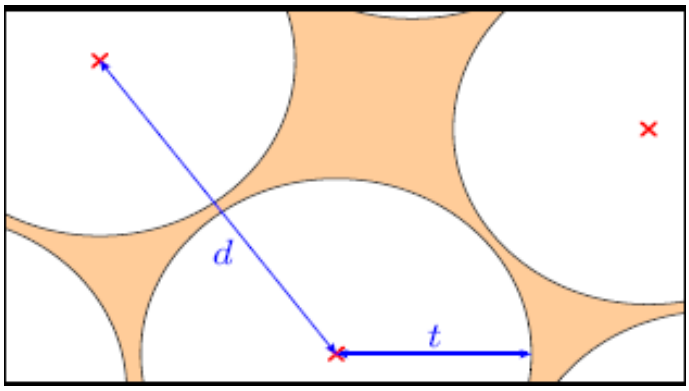

FIGURE 4 – Sphères de rayon t centrées sur les mots du code

2. Codes de Reed Muller

2.1. Définition

On rappelle brièvement la définition suivante des codes de Reed-Muller d'ordre 1. Soit $m \geq 2$ un entier. On note $\mathcal{B}_m$ l'espace de toutes les fonctions de $\mathbb{F}_2^m$ dans $\mathbb{F}_2$ (fonctions *Booléennes*). On munit cet espace de la base des évaluations $(e_z)_{z\in\mathbb{F}_2^m}$ où $e_z(x) = \begin{cases} 1 & \text{si } x = z \\ 0 & \text{sinon} \end{cases}$

Dans cette base toute fonction f s'écrit sous la forme

$$f = \sum_{x\in\mathbb{F}_2^m} f(x)e_x.$$

Autrement dit on d'écrit f par ses images. Le code de Reed-Muller d'ordre 1 correspondant est le sous-espace $RM(1; m)$ de $\mathcal{B}_m$ constitué par les fonctions affines. Une fonction affine s'écrit sous la

forme

$$f(x) = u_0 + u_1 x_1 + \ldots + u_m x_m.$$

Le sous-espace $RM(1;m)$ est donc un sous-espace de dimension $k = m+1$. On peut utiliser ce code de la façon suivante : on part d'un mot brut $(u_0;u_1;\ldots;u_m)$ de longueur $k = m+1$, à ce mot correspond la fonction affine $f \in RM(1;m)$ définie par l'égalité donnée ci-dessus. Le mot encodé est alors donné par la décomposition de f dans la base $(e_z)_z$ c'est-à-dire le mot $f(x)_{x\in\mathbb{F}_2^m}$.

2.2. Les paramètres

Le code $RM(1;m)$ a donc pour dimension $k = m+1$, pour longueur $n = 2^m$. On peut d'écrire facilement le poids des mots du code. En effet, si la fontion affine f est nulle, le poids du mot de code correspondant est 0. Si la fonction affine f est 1, le poids du mot correspondant est 2^m.
Dans tous les autres cas, l'ensemble des zéros de f est un hyperplan affine qui possède donc 2^{m-1} points, et le poids du mot correspondant est donc $2^m - 2^{m-1} = 2^{m-1}$. La distance minimale du code est donc $d = 2^{m-1}$. On en conclut que ce code peut corriger t erreurs avec

$$t = \lfloor \frac{d-1}{2} \rfloor = \lfloor 2^{m-2} - \frac{1}{2} \rfloor = 2^{m-2} - 1.$$

3. Décodage des codes de Reed Muller

3.1. Transformée de Walsh

De nombreuses caractéristiques liées à l'étude des fonctions Booléennes (en cryptographie et en codes) peuvent être exprimées au moyen de la distance de Hamming. De ce point de vue la *transformée de Fourier discrète* est un outil particulierement utile et efficace : étant donnée une fonctions Booléenne f, la connaissance de la transformée de Fourier discrète de f est équivalente à la connaissance de toutes les distances $d(f,l)$, où l est linéaire (ou affine). Elle permet donc en particulier de determiner la fonction affine la plus proche d'une fonction f, et en ce sens de décoder f dans $RM(1;m)$. La *transformée de Walsh* de f, notee $\widehat{f}$, est la transformée de Fourier de la fonction $(-1)^f$, et est plus pratique à utiliser dans notre contexte. Elle est definie par :

$$\begin{aligned} \mathbb{F}_2^m &\to \mathbb{Z} \\ a &\mapsto \widehat{f}(a) = \sum_{x\in\mathbb{F}_2^m} (-1)^{f(x)+a\cdot x} \end{aligned}$$

où $a\cdot x$ désigne le produit scalaire de a et de x, c'est-à-dire $a\cdot x = a_1x_1 + \cdots + a_mx_m$. Via ce produit scalaire, on peut identifier $\mathbb{F}_2^m$ avec l'ensemble de fonctions linéaires de $RM(1,m)$, en définissant, pour $a \in \mathbb{F}_2^m$, la fonction linéaire $a(x) := a\cdot x$. La relation entre la transformée de f en a et la distance entre f et a est alors donnée par :

$$\widehat{f}(a) + 2d(f,a) = 2^m$$

En effet, $\widehat{f} = 1\times|\{x \in \mathbb{F}_2^m \mid a(x) = f(x)\}| - 1\times|\{x \in \mathbb{F}_2^m \mid a(x) \neq f(x)\}| = 2^m - 2\times|\{x \in \mathbb{F}_2^m \mid a(x) \neq f(x)\}| = 2^m - 2d(f,a)$. Le gain lié à l'utilisation de cette transformée, dans le cadre d'un décodage par exemple, provient du fait qu'il existe un algorithme (*Fast Fourier Transform*) de complexité $\mathcal{O}(m2^m)$ calculant toutes les valeurs de $\widehat{f}$. Le principe de cet algorithme est récursif, de type "diviser pour reigner". Il consiste à décomposer la somme contenue dans la définition selon que l'une des coordonnées (en pratique on considère x_1 ou x_m) vaut 1 ou 0 :

$$\begin{aligned} \widehat{f}(a) &= \sum_{x\in\mathbb{F}_2^m, x_m=0} (-1)^{f(x)+a\cdot x} + \sum_{x\in\mathbb{F}_2^m, x_m=1} (-1)^{f(x)+a\cdot x} \\ &= \sum_{x\in\mathbb{F}_2^{m-1}} (-1)^{f(x,0)+(a_1,\ldots,a_{m-1})\cdot x} + \sum_{x\in\mathbb{F}_2^{m-1}} (-1)^{f(x,1)+(a_1,\ldots,a_{m-1})\cdot x + a_m} \\ &= \widehat{f(\cdot,0)}((a_1,\ldots,a_{m-1})) + (-1)^{a_m}\widehat{f(\cdot,1)}((a_1,\ldots,a_{m-1})) \end{aligned}$$

où $f(\cdot,0)$ (resp. $f(\cdot,1)$ désigne la restriction de f au sous-espace de $\mathbb{F}_2$ défini par $x_m = 0$ (resp. $x_m = 1$). Donc une fois calculées $\widehat{f(\cdot,0)}$ et $\widehat{f(\cdot,1)}$ il reste 2^{m-1} additions et 2^{m-1} soustractions pour obtenir $\widehat{f}$. En continuant la décomposition (m fois en tout), on obtient donc $\widehat{f}$ en $m2^m$ additions/soustractions. D'un point de vue pratique, on peu obtenir $\widehat{f}$ à l'aide d'un tableau F de taille 2^m, en ordonnant $\mathbb{F}_2^m$ de manière lexicographique :
- pour tout $x \in \mathbb{F}_2^m$, affecter $(-1)^{f(x)}$ à $F[x]$
- pour $i = 1...m$ faire :
  - pour tout $x \in \mathbb{F}_2^{i-1}$ et tout $s \in \mathbb{F}_2^i$ faire :
    - affecter $F[x,0,s] + F[x,1,s]$ à $F[x,0,s]$ et $F[x,0,s] - F[x,1,s]$ à $F[x,1,s]$

Dans l'exemple suivant, les barres verticales représentent les différentes instances de l'espace $\mathbb{F}_2^i$:

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
$f(x)$	1	0	0	1	0	0	1	1
$(-1)^{f(x)}$	−1	1	1	−1	1	1	−1	−1
$i = 1$	0	−2	0	2	2	0	−2	0
$i = 2$	0	0	0	−4	0	0	4	0
$i = 3$	0	0	4	−4	0	0	−4	−4

Nous obtenons ainsi un décodage à maximum de vraissemblance dans $RM(1;m)$: calculer $\widehat{f}$, puis rechercher les maximums, en valeur absolue, de cette transformée : si $\widehat{f}(a)$ (resp. $-\widehat{f}(a)$) est maximal, alors $d(f,a)$ (resp. $d(f,a+1)$) est minimal.

3.2. Décodage par liste dans $RM(1;m)$

Le problème du décodage par liste des codes de Reed-Muller d'ordre 1 peut s'énoncer ainsi : soit $f \in \mathcal{B}_m$ une fonctions Booléenne et soit T un entier. Il s'git alors de déterminer l'ensemble des mots du code qui sont à distance au plus T de f, c'est-à-dire l'ensemble des $a \in RM(1;m)$ tels que $d(f,a) \leq T$.

Ce type de décodage présente un intérêt tant en théorie des codes, lorsque par exemple le poids de l'erreur est supérieur à la capacité de correction (auquel cas il peut y avoir plusieurs solutions, ou bien la "bonne" solution est plus éloignée du vecteur bruité que les solution renvoyées par un décodage à maximum de vraissemblance), qu'en cryptographie (pour obtenir par exemple l'ensemble des approximations linéaires d'une fonction cryptographique). La distance maximale autorisée T est un paramètre fixé à l'avance. Nous allons reformuler le problème en normalisant cette distance : nous allons fixer un réel ε tel que $0 < \varepsilon \leq 1/2$ en demandant que la fraction maximale d'erreurs admissibles dans le mot de code bruité soit égale à $1/2 - \varepsilon$. En notant $\mathcal{L}_\varepsilon(f)$ la liste des solutions, nous avons alors :

$$\mathcal{L}_\varepsilon(f) = \{a \in RM(1;m) \mid d(f,a) \leq 2^m(1/2-\varepsilon)\}$$

Remarque : la transformée de Walsh permet directement d'effectuer un décodage en liste dans $RM(1;m)$.

Un problème se pose : quel peut être la taille de la liste obtenue ? Si celle-ci n'est pas raisonnablement petite, ce type de décodage présente moins d'intérêt. Pour répondre à cette question, nous disposons de la borne de Johnson. Dans le cas de $RM(1;m)$, celle-ci stipule que quel que soit f et quel que soit ε, la taille de la liste $\mathcal{L}_\varepsilon(f)$ est inférieure ou égale à $\frac{1}{4\varepsilon^2}$.

Nous allons maintenant présenter l'algorithme "sums" de Kabatianski-Tavernier ([2,3]), qui est de complexité $O(2^m \log(1/\varepsilon))$. Celui-ci est souvent (en fonctions de la valeur de ε) plus efficace que la transformée de Walsh, et surtout présente l'avantage d'avoir une version probabiliste, ce qui permet de l'appliquer pour de grande valeurs de m (avec une complexité qui passe à $O(m^2/\varepsilon^6)$), ce qu'il n'est pas possible de faire avec une transformée de Walsh. Cet algorithme peut être vu en fait comme une transformation de Walsh sélective (FFT), pendant laquelle, au fur et à mesure des étapes, on élimine le calcul en les positions qui ne peuvent pas aboutir à une solution. En d'autres termes, lors d'une FFT, toutes les 2^m distances $d(f,a)$ pour $a \in RM(1;m)$

sont calculées, alors que l'on a juste besoin de connaître le résultat de leur comparaison avec le rayon de décodage T. Nous allons donc extraire de l'information à chaque étape i pour invalider certains ensembles de fonctions.
Pour formaliser cela, nous définissons, pour $a = a_0 + a_1x_1 + \cdots + a_mx_m \in RM(1;m)$, le i-ième préfixe a^i de a par $a^i(x_1,\ldots,x_i) = a_1x_1+\cdots+a_ix_i$: a^i est une fonction linéaire en i variables telle que $a_i(x_1,\ldots,x_i) = a(x_1,\ldots,x_i,0,\ldots,0)+a_0$. Le principe de l'algorithme sums consiste à définir à chaque étape i un critère permettant d'éliminer un certain nombres de fonctions linéaires en i variables, celles dont on est sûrs qu'elle ne peuvent être le préfixe d'une solution du problème. Le calcul de la FFT n'est alors plus poursuivi en les positions correspondantes lors des étapes suivantes. Le critère d'acceptaion/émlimination d'un candidat préfixe est fortement lié à l'écriture sous forme de somme de la transformée de Walsh. Soit $a \in RM(1;m)$. Nous avons :

$$d(f,\{a,a+1\}) \leq 2^m(1/2-\varepsilon) \iff 2^{m-1} - \frac{1}{2}\left|\widehat{f}(a^m)\right| \leq 2^{m-1} - 2^m\varepsilon$$
$$\left|\widehat{f}(a^m)\right| \geq 2^{m+1}\varepsilon.$$

La définition de $\mathcal{L}_\varepsilon(f)$ peut alors s'écrire :

$$\mathcal{L}_\varepsilon(f) = \left\{a \in RM(1;m) \mid \left|\widehat{f}(a^m)\right| \geq 2^{m+1}\varepsilon\right\}$$

Pour définir notre critère, nous allons décomposer la somme impliquée dans cette définition. Soit a une fonction linéaire (alors $a = a^m$) et $0 \leq i \leq m$, nous avons :

$$\begin{aligned}
\widehat{f}(a^m) &= \sum_{x\in\mathbb{F}_2^m} (-1)^{f(x)+a^m\cdot x} \\
&= \sum_{s\in\mathbb{F}_2^{m-i}} \sum_{x\in\mathbb{F}_2^i} (-1)^{f(x,s)+a(x,s)} \\
&= \sum_{s\in\mathbb{F}_2^{m-i}} (-1)^{a(0,s)} \sum_{x\in\mathbb{F}_2^i} (-1)^{f(x,s)+a^i(x)} \\
&= \sum_{s\in\mathbb{F}_2^{m-i}} (-1)^{a(0,s)} \widehat{f(\cdot,s)}(a^i)
\end{aligned}$$

Maintenant, si $a \in RM(1;m)$ appartient à $\mathcal{L}_\varepsilon(f)$, nous avons $\left|\widehat{f}(a^m)\right| \geq 2^{m+1}\varepsilon$ d'une part, et d'autre part :

$$\begin{aligned}
\left|\widehat{f}(a^m)\right| &= \left|\sum_{s\in\mathbb{F}_2^{m-i}} (-1)^{a(0,s)} \widehat{f(\cdot,s)}(a^i)\right| \\
&\leq \sum_{s\in\mathbb{F}_2^{m-i}} \left|(-1)^{a(0,s)} \widehat{f(\cdot,s)}(a^i)\right| = \sum_{s\in\mathbb{F}_2^{m-i}} \left|\widehat{f(\cdot,s)}(a^i)\right|
\end{aligned}$$

Le point crucial de cette inégalité est le fait que la somme de droite ne dépend que du préfixe a^i. On déduit des deux inégalités précédentes que $\sum_{s\in\mathbb{F}_2^{m-i}} \left|\widehat{f(\cdot,s)}(a^i)\right| \geq 2^{m+1}$: c'est le critère d'acceptation d'un candidat-préfixe. À la i-ième étape de l'algorithme, nous allons donc construire une liste intermédiaire $\mathcal{L}^i$ des candidats préfixes :

$$\mathcal{L}^i_\varepsilon(f) = \left\{a^i \in \mathbb{F}_2^i \mid \sum_{s\in\mathbb{F}_2^{m-i}} \left|\widehat{f(\cdot,s)}(a^i)\right| \geq 2^{m+1}\right\}$$

Bien sûr, comme lors d'une FFT, nous utilisons le résultat des calculs précédents pour déterminer cette liste, que nous construisons à partir de la précédente. Il faut d'abord remarquer que a^i ne

peut être dans $\mathcal{L}_\varepsilon^i(f)$ que si son préfixe a^{i-1} est dans $\mathcal{L}_\varepsilon^{i-1}(f)$. Ainsi nous n'allons tester le critère que sur les successeurs des éléments de $\mathcal{L}_\varepsilon^{i-1}(f)$, c'est-à-dire sur les éléments a^{i-1} et $a^{i-1}+x_i$ pour tout $a^{i-1} \in \mathcal{L}_\varepsilon^{i-1}(f)$. Ensuite nous pouvons (comme dans la FFT) calculer chaque terme $\widehat{f(\cdot,s)}(a^i)$ impliqué dans le critère par une simple somme/différence :

$$\begin{aligned}\widehat{f(\cdot,s)}(a^i) &= \sum_{x\in\mathbb{F}_2^i, x_i=0} (-1)^{f(x,s)+a^i(x)} + \sum_{x\in\mathbb{F}_2^i, x_i=1} (-1)^{f(x,s)+a^i(x)} \\ &= \sum_{x\in\mathbb{F}_2^{i-1}} (-1)^{f(x,0,s)+a^{i-1}(x)} + (-1)^{a_i} \sum_{x\in\mathbb{F}_2^{i-1}} (-1)^{f(x,1,s)+a^{i-1}(x)} \\ &= \widehat{f(\cdot,0,s)}(a^{i-1}) + (-1)^{a_i}\widehat{f(\cdot,1,s)}(a^{i-1})\end{aligned}$$

Or justement ces deux termes ont été calculés à l'étape précédente.
Qu'en est-il de la taille des listes ? Nous avons vu que la liste finale est borné par $1/4\varepsilon^2$. Mais si les listes intermédiaires étaient trop grosses (en d'autres termes si le critère introduit n'est pas efficace), cet algorithme ne serait pas meilleur qu'une FFT (voire pire). Heureusement, il est montré dans [2] que les listes intermédiaires sont soumises à la même borne. Nous n'avons pas la place de présenter la taille des listes que l'on obtient expérimentalement, mais voici le comportement moyen dans la quasi-totalité des cas : soit i_ε tel que $2^{i_\varepsilon} \leq 1/(4\varepsilon^2) \leq 2^{i_\varepsilon+1}$. Alors, jusqu'à la i_ε, les listes intermédiaires sont exhaustives. Ensuite, celles-ci diminuent très rapidement pour atteindre la taille de la liste finale. Ce comportement est d'ailleurs prédit par un résultat de Helleseth, Kløve et Levenshtein ([1]). Comme le calcul du critère est tout de même relativement couteux, on modifie en fait l'algorithme sums de la manière suivant : pendant les i_ε premières étapes, ne pas appliquer le critère (cela revient exactement à faire i_ε étapes de FFT), poser $\mathcal{L}_\varepsilon^{i_\varepsilon} = \mathbb{F}_2^i$, et n'appliquer le critère que sur les étapes suivantes.

Il existe une version très efficace de cet algorithme permettant le décodage dans les codes $RM(2;m)$, c'est-à-dire l'ensemble des fonctions Booléennes quadratiques.

4. Conclusion

La théorie des codes correcteurs d'erreurs est présente dans l'ensemble des transmissions numériques de nos jours. Il est indispensable d'avoir des algorithmes de décodages efficaces et les recherches actuellement vont vers une accélération et une amélioration des algorithmes existants. Nous avons présenté ici les notions importantes de théorie des codes correcteurs d'erreurs et mis en avant les derniers résultats concernant le décodage des codes de Reed-Muller d'ordre 1.

Bibliographie

1. T. Helleseth, T. Kløve, et V. Levenshtein. Bounds on the error-correcting capability of codes beyond half the minimum distance. In D. Augot, P. Charpin, et G. Kabatianski, editors, *Proceedings of the 3rd International Workshop on Coding and Cryptography, WCC 2003*, pages 243–251, 2003.
2. G. Kabatiansky et C. Tavernier. List decoding of Reed-Muller codes. In *Ninth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT'2004*, pages 230–235, juin 2004. http ://ced.tavernier.free.fr/Balgaria.pdf.
3. G. Kabatiansky et C. Tavernier. List decoding of first order Reed-Muller codes II. In *Tenth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT'2006*, pages 131–134, septembre 2006. http ://ced.tavernier.free.fr/Kabat.pdf.