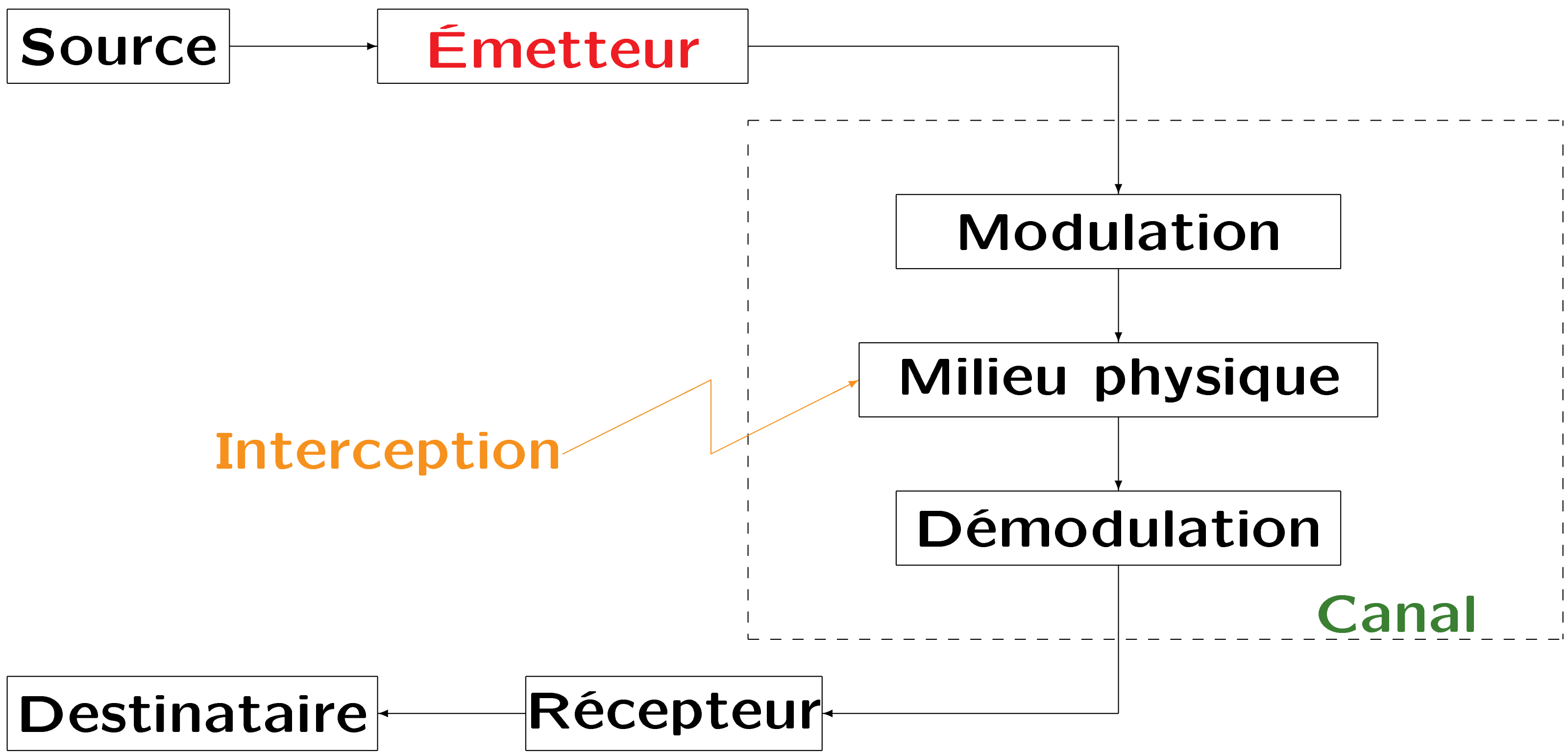
Reconnaissance d'un code en bloc

Mathieu Cluzeau

INRIA-projet CODES
Domaine de Voluceau
78153 Le Chesnay - FRANCE

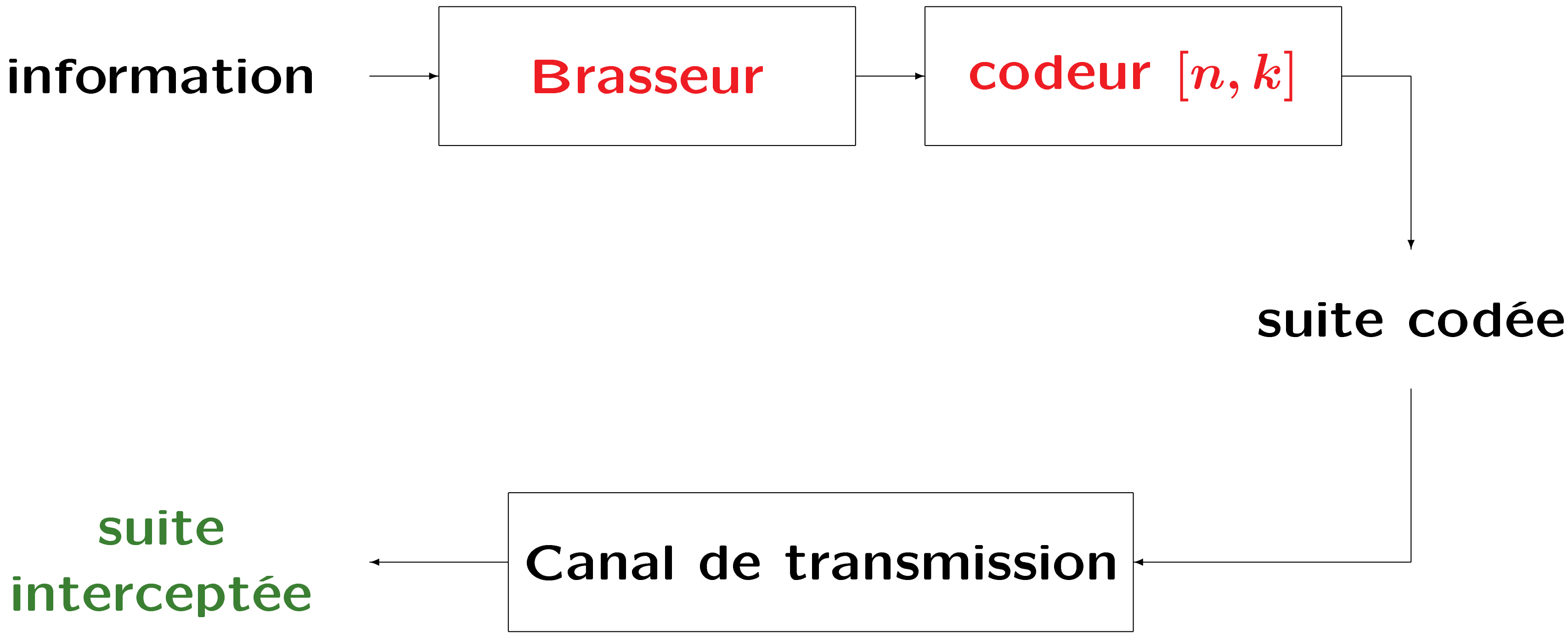
19 Avril 2007

Un schéma classique de transmission



Problème de reverse-engineering : émetteur ?

Motivations : reverse-engineering d'un schéma de transmission



Problème : trouver les différents éléments du système de communication à partir de la suite interceptée.

Plan de l'exposé

- Quelques rappels sur les codes LDPC
  - Une famille spécifique de codes en blocs : les codes LDPC
  - L'algorithme de Gallager
- Utilisation d'un test statistique pour reconstruire un code
  - méthode de base et algorithme de Canteaut-Chabaud
  - un premier algorithme de reconstruction
- Utilisation d'un algorithme de décodage itératif
  - nouvel algorithme de décodage itératif
  - résultats

Rappels sur les codes LDPC

Les codes linéaires binaires

Définition : Un code linéaire $\mathcal{C}$ de longueur n est un sous-espace vectoriel de $\mathbb{F}_2^n$

$$w_H(c) = |\mathrm{Supp}(c)|$$

$$d_{\min}(\mathcal{C}) = \min_{c,c' \in \mathcal{C}, c \neq c'} d(c, c').$$

Pour un code linéaire :

$$d_{\min}(\mathcal{C}) = \min_{c \in \mathcal{C}, c \neq 0} wt_H(c).$$

Théorème : Si un code $\mathcal{C}$ a une distance minimale d_{min}, alors il permet de corriger toute erreur de poids inférieur ou égal à $t = \lfloor \frac{d_{min}-1}{2} \rfloor$. L'entier t est la capacité de correction du code $\mathcal{C}$.

Code dual

Définition : Une matrice génératrice du code $\mathcal{C}$ est une matrice de taille $k \times n$ dont les k lignes forment une base du code $\mathcal{C}$.

Le dual d'un code $\mathcal{C}$ est l'orthogonal de $\mathcal{C}$ pour le produit scalaire usuel :

$$\mathcal{C}^{\perp} = \{h \in \mathbb{F}_2^n | \forall c \in \mathcal{C},\ hc^T = 0\}.$$

On a $\dim(\mathcal{C}^{\perp}) = n - \dim(\mathcal{C})$.

Une matrice de parité d'un code $\mathcal{C}$ est une matrice génératrice du code dual.

Forme systématique :

$$\left(\begin{array}{c|c} I_k & Z \end{array} \right)$$

Les codes LDPC

LDPC pour Low Density Parity Check codes
Caractérisés par une matrice de parité très creuse

LDPC (i, j)

- **toute colonne de la matrice de parité a un poids i;**
- **toute ligne a un poids j.**

Par exemple, un code LDPC $(3, 6)$ de longueur n aura $\frac{n}{2}$ équations de parités de poids 6. Chaque bit d'un mot vérifie 3 équations de parité.

Décodage des codes LDPC

On a transmis un mot $c = (c_1, \ldots, c_n)$ sur le canal et on a reçu un mot que l'on appellera l'observation.

On va associer à chaque position du mot une probabilité. $\longrightarrow$ On notera, pour alléger les notations, c_t la probabilité que la variable aléatoire associée à la position t du mot de code soit égal à 1.

Algorithme de décodage des codes LDPC

$\longrightarrow$ **Une équation de parité nous permet de remettre à jour la probabilité des bits y participant en fonction des bits voisins :**

$$EQ \quad : \quad c_{t_0} + \sum_{j=1}^{d-1} c_{t_j} = 0,$$

$$\Pr[c_{t_0}|EQ] = \frac{1 - \prod_{i=1}^{d-1}(1 - 2\Pr[c_{t_j}])}{2}.$$

$\longrightarrow$ **génération de la probabilité a posteriori (APP) du bit a_t en tenant compte de toutes les équations de parité**

Calcul de la probabilité a posteriori

En supposant que le canal est sans mémoire et qu'il y a équiprobabilité des bits à l'émission, on a :

$$
\begin{aligned}
Pr[x=1|y_1,y_2] &= \frac{Pr[x=1|y_1]Pr[x=1|y_2]}{Pr[x=1|y_1]Pr[x=1|y_2]+Pr[x=0|y_1]Pr[x=0|y_2]} \\
&\overset{\text{def}}{=} Pr[x=1|y_1]\otimes Pr[x=1|y_2]
\end{aligned}
$$

L'APP d'un bit s'obtient en combinant ($\otimes$) toutes les probabilités entrantes (OBS et EXT)

$$
\begin{aligned}
APP[c_t] &= \left(\bigotimes_{\mathrm{i=0}}^{\mathrm{nb_{eq}-1}} Extr_i[c_t]\right)\otimes Obs[c_t] \\
&\propto Obs[c_t]\prod_{i=0}^{nb_{eq}-1} Extr_i[c_t].
\end{aligned}
$$

Schéma : Calcul de l'APP

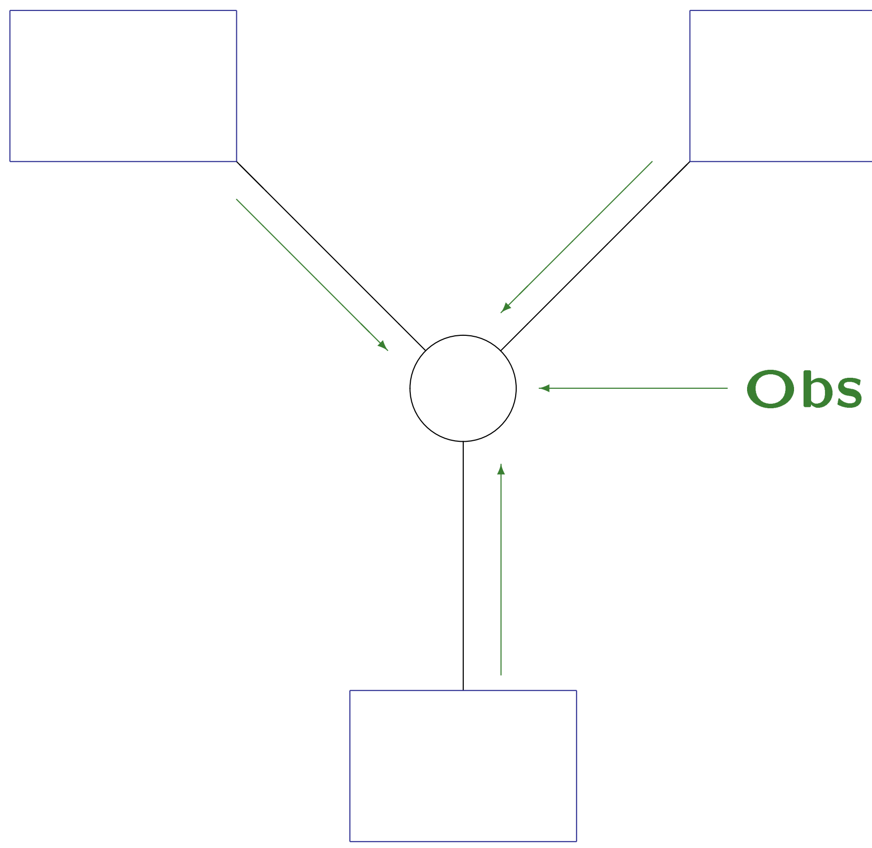
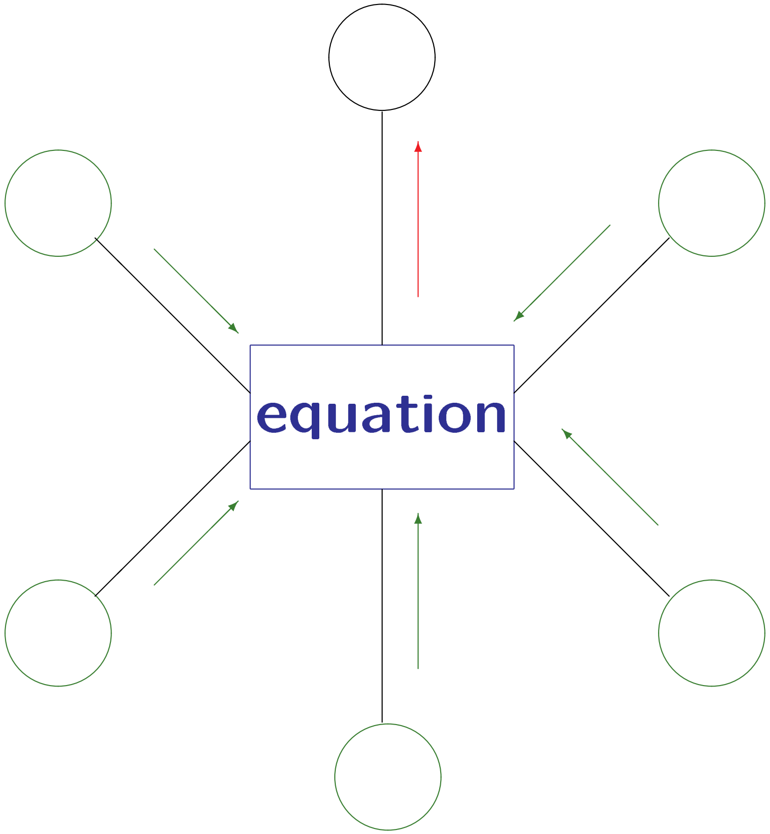


Schéma : Extr



Algorithme de Gallager

- **Initialisation :**
  Pour tout $t \in \{1, \ldots, n\}$,
  - **Calculer** $\mathrm{Obs}(c_t) = \begin{cases} 1-\tau & \text{si l'observation est un 1.} \\ \tau & \text{si l'observation est un 0.} \end{cases}$
  - $\forall e$, $\mathrm{Extr_e}(c_t) = \frac{1}{2}$.

- **Itérer : Pour tout $t \in \{1, \ldots, n\}$, calculer :**

  - $\forall e,\ \mathrm{A_e}(c_t) = \mathrm{Obs}(c_t) \otimes \left[\bigotimes_{i \neq e} \mathrm{Extr_i}(c_t) \right]$.
  - $\forall e,\ \mathrm{Extr_e}(c_t) = \dfrac{1}{2} \left[1 - \prod_{i \in I_e \setminus \{t\}} (1 - 2\mathrm{A_e}(c_i)) \right]$,

avec $I_e = \{i | a_i$ appartient à l'équation $e\}$.

Algorithme de Gallager

- **Terminaison :**
  Pour tout $t \in \{1, \ldots, n\}$,
  - **calculer**
  $$\mathrm{APP}(c_t) = \mathrm{Obs}(c_t) \otimes \left[\bigotimes_i \mathrm{Extr_i}(c_t) \right].$$
  - **Si** $\mathrm{APP}(c_t) > \frac{1}{2}$ **alors** $c_t = 1$, **sinon** $c_t = 0$.

Utilisation d'un test statistique pour reconstruire un code

Problème général

Retrouver le code $\mathcal{C}$ utilisé

Le problème général :
Données : une matrice X, deux entiers k et w,

Propriété : Existe-t-il une matrice E^* vérifiant $rk(X + E^*) \leq k$ et $wt(E^*) \leq w$ est un problème NP-complet.

Cependant si le taux d'erreur est faible et/ou si la taille de bloc n'est pas trop grande, il est possible de retrouver le code.

Principe de base

Canal : canal binaire symétrique de probabilité d'erreur τ.

On découpe la séquence observée en M mots de code bruités $\widetilde{m}$:

$$\mathrm{R} = \begin{pmatrix} \widetilde{m}_1 \\ \widetilde{m}_2 \\ \vdots \\ \widetilde{m}_M \end{pmatrix}.$$

Pour $h \in \mathcal{C}^{\perp}$ (une équation de parité),

$$\Pr[h\widetilde{m}^T = 0] = \frac{1}{2} + \frac{(1-2\tau)^{wt_H(h)}}{2},$$

$\longrightarrow wt_H(hR^T)$ faible quand $h \in \mathcal{C}^{\perp}$.

Algorithme de recherche de mots de poids faible

$[n, k]$-code de matrice génératrice G.

- Choisir aléatoirement un ensemble d'information I :

$$P^{-1}G = G_I = (I_k|Z).$$

- Séparer I en 2 parties, choisir un sous-ensemble J de s positions.
  Trouver des mots m tels que

$$wt_H(m_{|_{I_1}}) = wt_H(m_{|_{I_2}}) = \mathrm{p} \text{ et } wt_H(m_{|_J}) = 0.$$

  On a alors : $m = hG_I = hP^{-1}G$.

- Si $wt_H(hZ) + 2p \leq \mathrm{T}$, retourner hP^{-1}.

Recommencer avec un autre ensemble d'information en changeant seulement une position.

Algorithme de Valembois [Valembois 00]

But : Construire une base de $\mathcal{C}^{\perp}$.

Deux étapes :

- Trouver des mots h tels que hR^T ait un poids de Hamming faible (algorithme de Canteaut-Chabaud).

- Décider si ils appartiennent a $\mathcal{C}^{\perp}$ ou pas selon $(wt_H(hR^T), wt_H(h))$.

  $\longrightarrow$ Test d'hypothèse [Valembois 00] :

  On décide que $h \in \mathcal{C}^{\perp}$ pour des grandes valeurs de

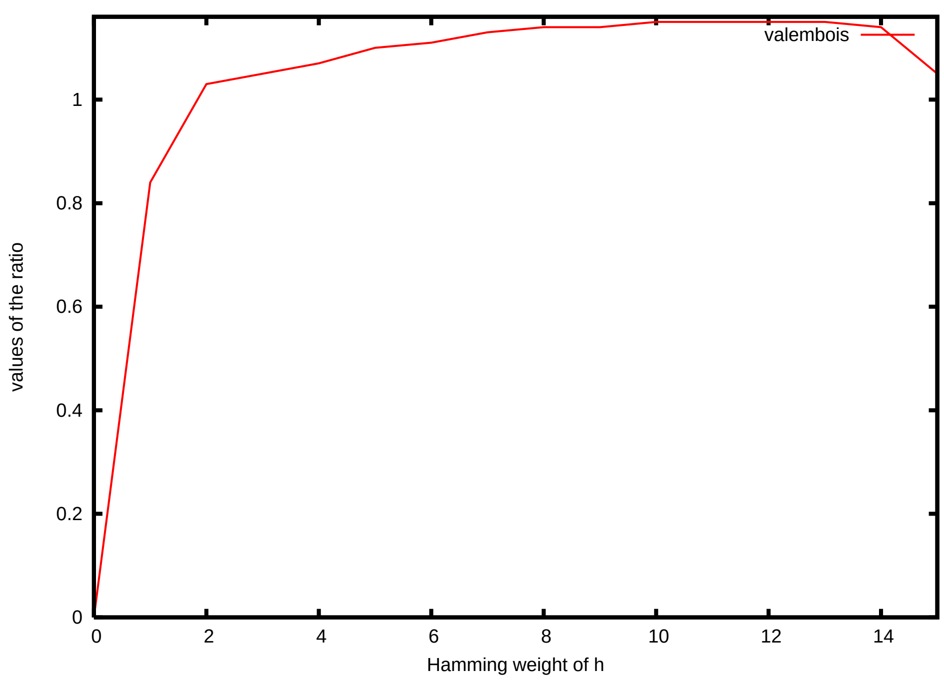
$$\frac{\Pr[R|h \in \mathcal{C}^{\perp}]}{\Pr[R]} = \left(1 + (1-2\tau)^{wt_H(h)}\right)^M \times \left(\frac{1-(1-2\tau)^{wt_H(h)}}{1+(1-2\tau)^{wt_H(h)}}\right)^{wt_H(hR^T)}$$

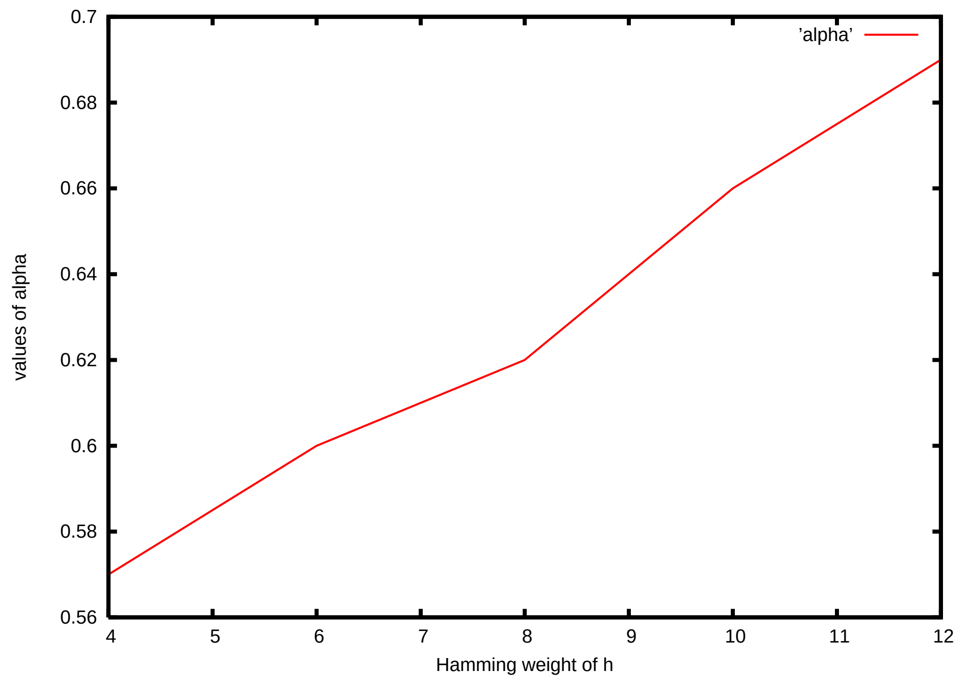
Analyse du test pour le code BCH[15,7,5]

Valeurs en fonction de $wt_H(h)$ du rapport

$$\alpha = \frac{\sharp\left\{h \notin \mathcal{C}^{\perp} \text{ tels que } wt_H(hR^T) \leq 3\right\}}{\sharp\left\{h \in \mathcal{C}^{\perp} \text{ tels que } wt_H(hR^T) \leq 3\right\}}.$$

$\longrightarrow$ **Les h de poids faible sont plus sûrs.**





Un nouveau test statistique [Cluzeau 06]

On décide que $h \in \mathcal{C}^{\perp}$ si et seulement si $wt_H(hR^T) \leq T$.

Théorème : Pour avoir

$$\begin{aligned}
\frac{\sharp\left\{h \notin \mathcal{C}^{\perp} \text{ tels que } wt_H(hR^T) \leq T\right\}}{\sharp\left\{h \in \mathcal{C}^{\perp} \text{ tels que } wt_H(hR^T) \leq T\right\}} &= \alpha, \\
\Pr\left[wt_H(hR^T) \geq T | h \in \mathcal{C}^{\perp}\right] &= \beta,
\end{aligned}$$

on a besoin de

$$M = \left(\frac{b\sqrt{1-x^2}+a}{x}\right)^2 \text{ mots de code bruités}$$

$$\text{et } T = \frac{1}{2}\left(M - a\sqrt{M}\right)$$

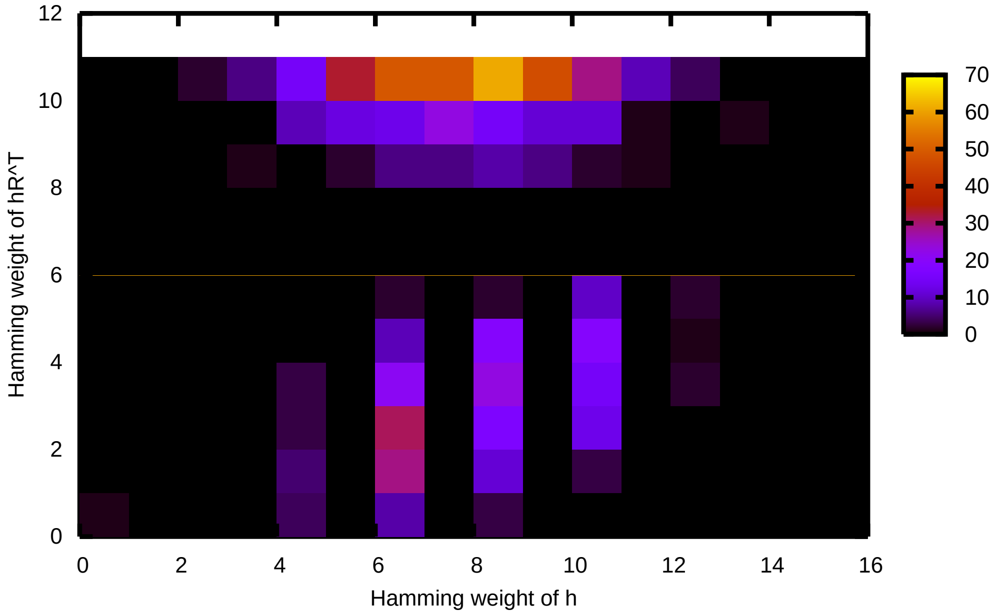
avec $a = -\phi^{-1}\left(\frac{\alpha(1-\beta)}{2^k-1}\right)$, $b = \phi^{-1}(1-\beta)$ et $x = (1-2\tau)^{wt_H(h)}$.

Exemple pour un code BCH [15,7,5], $\tau = 0.01$

Avec $\alpha = 0.01$ et $\beta = 0.1$:

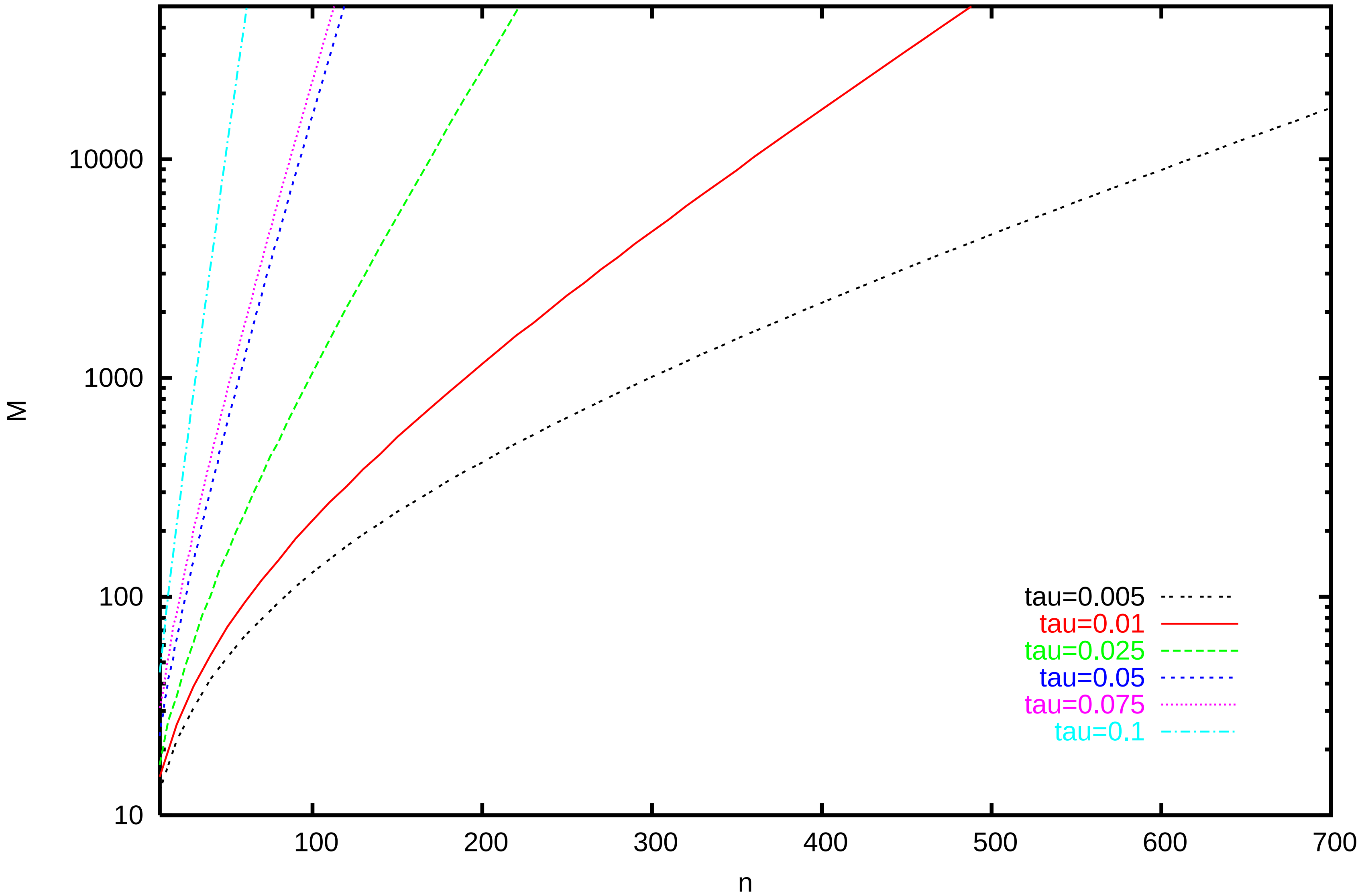
$\longrightarrow \; M = 32, \; T = 5.$

On sépare bien les mots de $\mathcal{C}^{\perp}$ des autres.

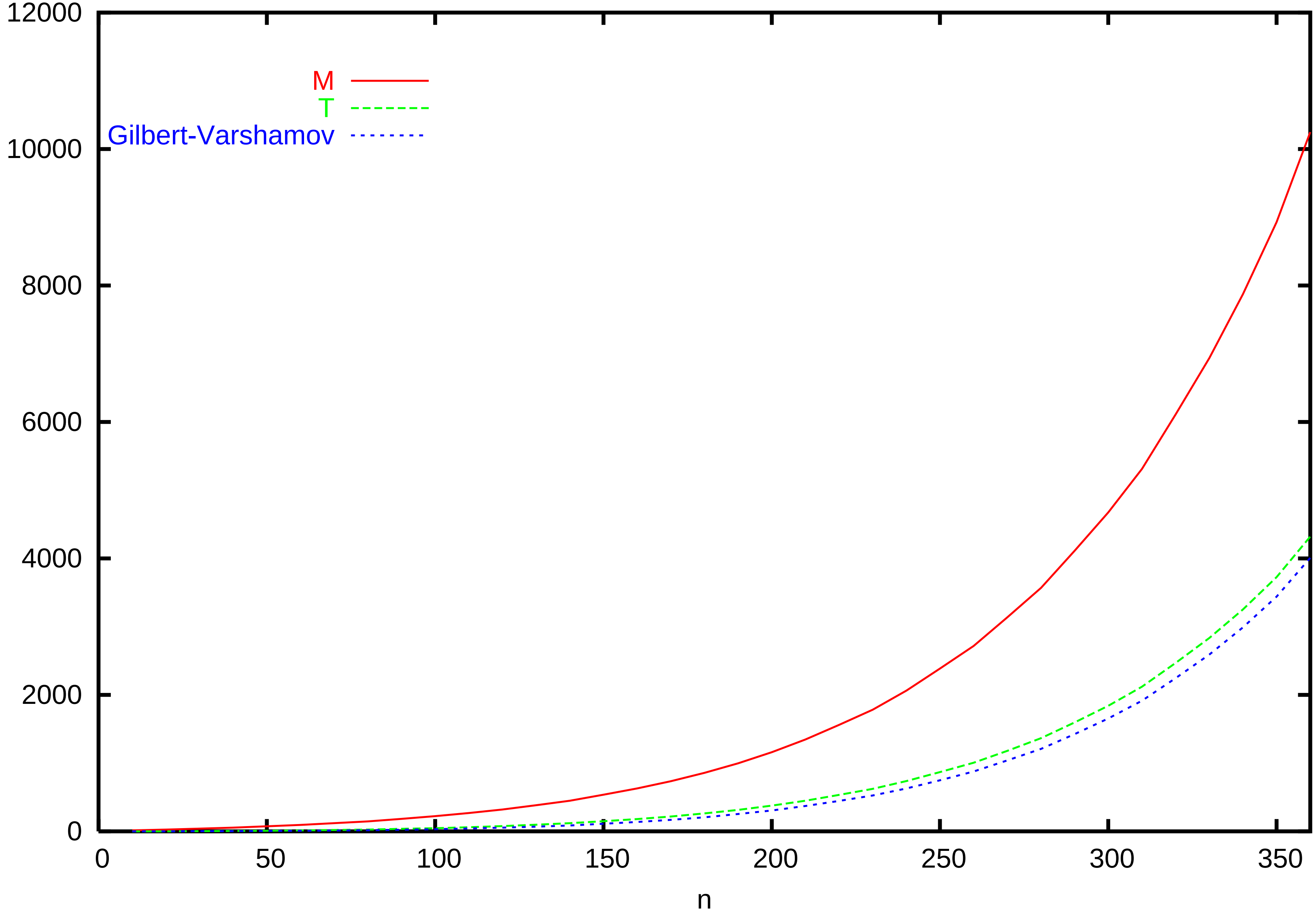


Ce nouveau test est mieux approprié à notre contexte.

Nombre de mots de code bruités nécessaires pour $\mathcal{C}$ de taux $\frac{1}{2}$



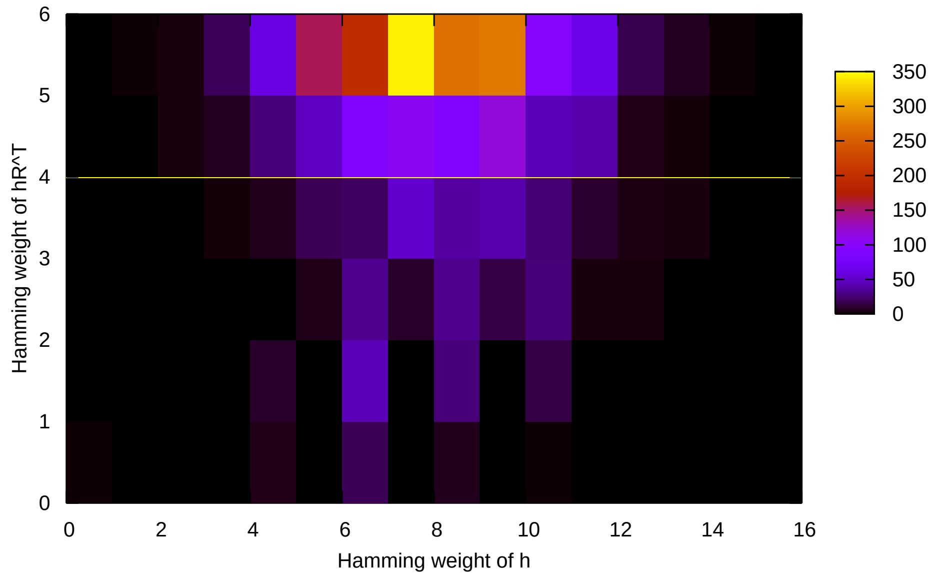
Pour $\mathcal{C}$ de taux $\frac{1}{2}$ et une probabilité d'erreur de $\tau = 0.01$



Pour des valeurs plus faibles de M

On prend moins de mots de code bruités pour réduire la complexité de la recherche de mots de poids faible.

Exemple pour notre code BCH, $\tau = 0.01$.
On prend maintenant $\alpha = 0.7$ et $\beta = 0.1 \longrightarrow M = 17, \; T = 3$.



On retourne aussi des mots h qui n'appartiennent pas à $\mathcal{C}^{\perp}$.

Idées pour améliorer le test statistique

Le test statistique précédent retourne des candidats pour être dans $\mathcal{C}^\perp$.

- On peut essayer de chercher des candidats avec un nombre de mots M_1 plus faible et tester sur un échantillon plus large (en cours).

- On souhaite améliorer l'information sur ces équations de parité.
  $\longrightarrow$ On utilise un algorithme de décodage itératif qui utilise ces équations de parité probables ([Cluzeau 06]).

Recherche avec M_1 mots et test avec M

On utilise un premier échantillon de M_1 mots et on retourne (algorithme de Canteaut-Chabaud) tous les mots h tels que $wt_H(hX_1^T) < T_1$.

On teste ces mots h sur l'ensemble des M mots reçus et on retient seulement ceux qui vérifient $wt_H(hX^T) < T$.

Idée : Prendre T_1 relativement élevé pour trouver suffisamment de mots de $\mathcal{C}^\perp$.

Prendre T suffisamment faible pour ne pas garder trop de mots qui ne sont pas dans $\mathcal{C}^\perp$.

Utilisation d'un algorithme de décodage itératif avec des équations pondérées

Décodage itératif avec des équations de parité pondérées

Initialisation :

Pour tous les mots reçues $\widetilde{m}_i$:

$$\mathrm{A_e}(m_t^{(i)}) = \mathrm{Obs}(m_t^{(i)}) = \begin{cases} 1-\tau & \text{si } (\widetilde{m}_i)_t = 1. \\ \tau & \text{si } (\widetilde{m}_i)_t = 0. \end{cases}$$

Itérer :

Pour tous les mots $\widetilde{m}_i$ et pour tout e, calculer :

- $\widetilde{p}_{e,i} = \dfrac{1 + \prod_{t \in \mathrm{Supp}(e)} (1 - 2\mathrm{A_e}(m_t^{(i)}))}{2}$ et $p_{e,i} = \dfrac{\sum_{j \neq i} \widetilde{p}_{e,j}}{M-1}$.
- $\mathrm{Extr_e}(m_t^{(i)}) = \left(\dfrac{1}{2} - p_{e,i}\right) \displaystyle\prod_{j \in \mathrm{Supp}(e) \setminus \{t\}} \left(1 - 2\mathrm{A_e}(m_j^{(i)})\right) + \dfrac{1}{2}$.
- $\mathrm{A_e}(m_t^{(i)}) = \mathrm{Obs}(m_t^{(i)}) \otimes \left[\displaystyle\bigotimes_{j \neq e} \mathrm{Extr_j}(m_t^{(i)}) \right]$.

Décodage itératif avec des équations de parité pondérées

Terminaison :

Pour tous les mots $\widetilde{m}_i$, pour tout t :

- **Calculer**

$$\mathrm{APP}(m_t^{(i)}) = \mathrm{Obs}(m_t^{(i)}) \otimes \left[\bigotimes_j \mathrm{Extr}_\mathrm{j}(m_t^{(i)}) \right].$$

- **Si** $\mathrm{APP}(m_t^{(i)}) > \frac{1}{2}$ **alors** $m_t^{(i)} = 1$, **sinon** $m_t^{(i)} = 0$.

Pour toute équation e :

- **Calculer**

$$p_e = \frac{\sum_{j=1}^{M} \widetilde{p}_{e,j}}{M}.$$

Nouvel algorithme pour reconstruire un code linéaire [Cluzeau 06]

1. Trouver des mots de poids faible dans le code engendré par R^T à l'aide de l'algorithme de Canteaut-Chabaud. La valeur du seuil sur $wt_H(hR^T)$ est donnée par le test statistique.

2. Éliminer les équations de parité qui font apparaître des cycles courts (dans nos tests, on supprime les cycles de longueur inférieure à 4).

3. Utiliser ces équations de parité probables pour corriger des erreurs à l'aide de l'algorithme décrit précédemment.

4. Trouver les $(n-k)$ équations de parité les plus probables.

Exemple : pour un code BCH code [15,7,5], $\tau = 0.05$

- $\alpha = 0.01 \longrightarrow M = 298$ et $T = 117$.
  248 mots de poids faible, tous dans le code dual.
  nous avons reconstruit notre code.

- $\alpha = 1 \longrightarrow M = 170$ et $T = 69$.
  387 mots de poids faible, 132 faux.
  Élimination des cycles courts $\rightarrow$ 13 équations.
  Après décodage, il reste quelques erreurs ($p_{err} = 0.027$).
  Pour les équations justes p_e est compris entre 0.80 et 0.94, alors que pour les fausses p_e est entre 0.5 et 0.62.
  $\longrightarrow$ nous avons reconstruit notre code.

Pour un code LDPC (3,6) de longueur 250, $\tau = 0.005$

Avec $wt_H(h) = 63$, $\alpha = 0.01$, $\beta = 0.1$:
$\longrightarrow$ $M = 703$ et $T = 179$.

τ	algo CC	p_{err}	dimension du code dual reconstruit
0.005	10 s	0.003	63
0.005	30 s	0.003	74
0.005	1 min	0.001	105
0.005	5 min	0.0003	115
0.005	10 min	0.0001	119
0.005	30 min	0	125

Pour un code LDPC, il y a de nombreux mots de poids très faible dans $\mathcal{C}^\perp$.
$\longrightarrow$ on peut prendre un seuil T plus faible, dans l'espoir de détecter plus rapidement ces mots.

Pour un code LDPC (3,6) de longueur 250, $\tau = 0.005$

$M = 518$ et $T = 30$

τ	algo CC	p_{err}	dimension du code dual reconstruit
0.005	30 s	0.0008	103
0.005	2 min	0	119
0.005	3 min	0	121
0.005	10 min	0	125

$M = 518$ et $T = 15$

τ	algo CC	p_{err}	dimension du code dual reconstruit
0.005	1 min	0.004	45
0.005	5 min	0.002	79
0.005	10 min	0.002	63

Pour un code LDPC (3,6) de longueur 1000

- Pour $\tau = 0.001$, $M = 1986$ et $T = 413$.

- Pour $\tau = 0.002$, $M = 5485$ (trop grand). On prend $M = 1619$ et $T = 286$ (*i.e.*, on prend seulement $wt_H(h) \leq 100$).

τ	algo CC	p_{err}	dimension du code dual reconstruit
0.001	1 min	0.0004	315
0.001	2 min	0.0002	383
0.001	5 min	0.00008	441
0.001	10 min	0.00002	475
0.001	30 min	0	491
0.001	1h	0	500
0.002	10 min	0.002	4
0.002	20 min	0.002	10
0.002	1h	0.002	12
0.002	12h	0.0019	48

Pour un code aléatoire de longueur 100 et de dimension 50

Avec $\alpha = 0.01$, $\beta = 0.1$,

- **Pour** $\tau = 0.005$, $M = 158$ **et** $T = 22$.
- **Pour** $\tau = 0.01$, $M = 274$ **et** $T = 62$.
- **Pour** $\tau = 0.02$, $M = 797$ **et** $T = 271$.

τ	algo CC	p_{err}	dimension du code dual reconstruit
0.005	1 s	0.0047	50
0.01	1 s	0.008	50
0.02	30 s	-	0
0.02	5 min	0.02	28
0.02	10 min	0.019	50

Conclusions

Algorithme de reconstruction effectif pour les codes linéaires de taille raisonnable.

Travaux en cours (ou à venir) :

- pousser encore les simulations pour réussir à atteindre des tailles de blocs très importante (entrelacement) ;
- adapter l'algorithme de décodage à d'autres types de canaux, en particulier lorsqu'on dispose d'une information souple provenant de la démodulation ;
- utiliser l'algorithme de décodage itératif avec des équations de parité probables dans d'autres contextes, comme par exemple en cryptographie.