Reconstruction de la permutation d'un turbo-code

Jean-Pierre Tillich, **Audrey Tixier**, Nicolas Sendrier

Inria Paris-Rocquencourt

Journées C2 - mars 2014

Plan

Problème de la reconnaissance de code

- Le problème :
  - $\mathcal{E}$ une famille de codes
  - $\mathcal{C}$ un code tiré uniformément dans $\mathcal{E}$
  - $c_1, \ldots, c_M$ des mots de code tirés uniformément dans $\mathcal{C}$
- Les données :
  - $\hat{c}_1, \ldots, \hat{c}_M$ les mots de code bruités
- Les objectifs :
  - 1. Retrouver $\mathcal{C}$
  - 2. Décoder $\hat{c}_1, \ldots, \hat{c}_M$

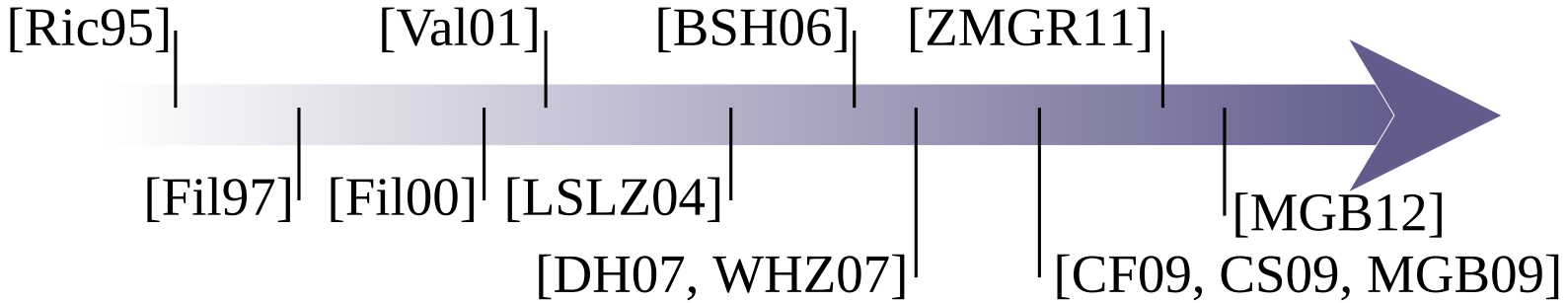
Motivations

- Conception de récepteurs intelligents
  - Domaine des télécommunications
  - Récepteurs auto-adaptifs en fonction des données reçues
- Compréhension du codage de l'ADN
  - Comment modéliser la redondance présente dans l'ADN ?
- Dans un milieu non-coopératif, on dispose de données codées par un code inconnu
  - Pouvoir décoder ces données

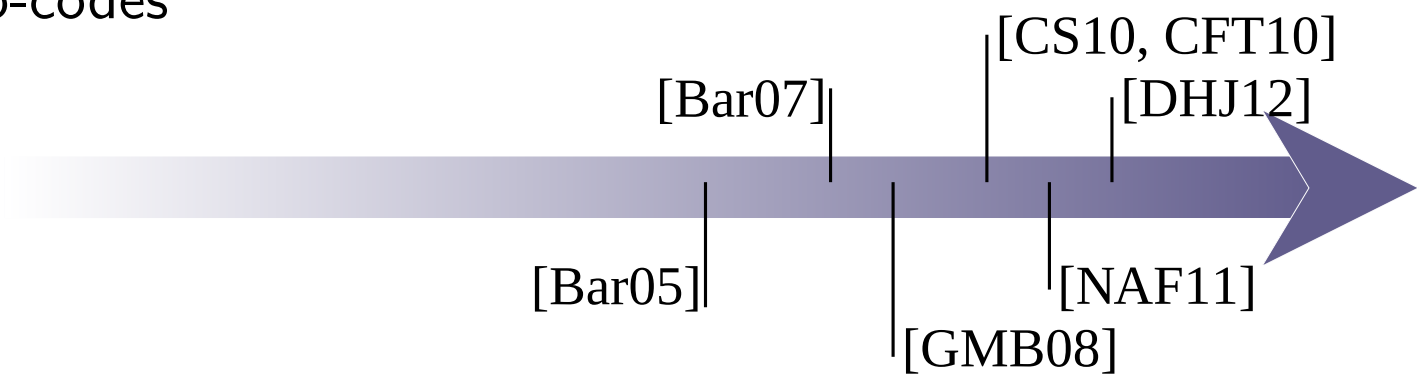
Historique

[Val01] : Problème NP Complet pour les codes linéaires

- Codes convolutifs



- Turbo-codes



- Codes LDPC

Codes convolutifs

- Un code convolutif (n, k) possède une matrice génératrice infinie de la forme

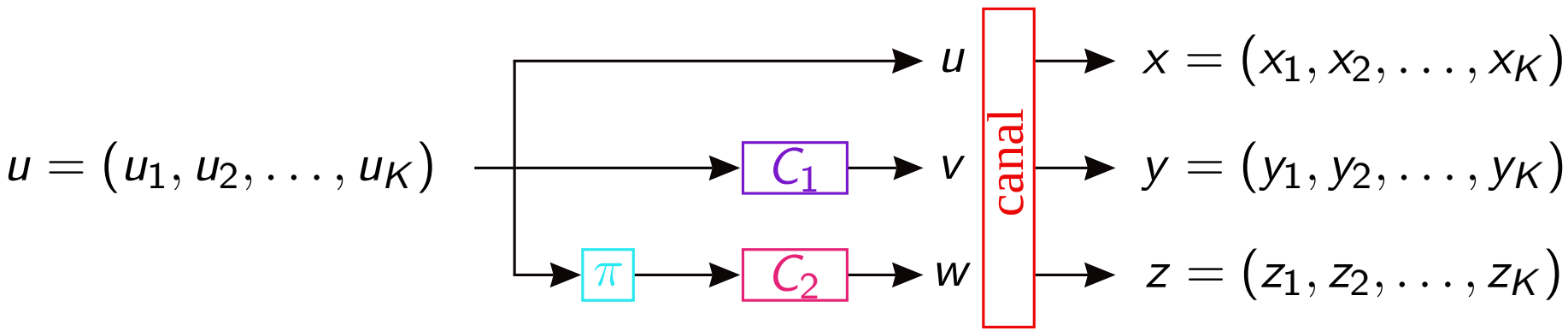
$$G = \begin{pmatrix} G_1 & G_2 & \dots & G_s & & \\ & G_1 & G_2 & \dots & G_s & \\ & & G_1 & G_2 & \dots & G_s \\ & & & \dots & \dots & \dots \end{pmatrix}$$

  Où G_i est une matrice binaire de taille $k \times n$

- $\underbrace{(u_1, u_2, \dots, u_K)}_{\textit{information}} \times G = \underbrace{(v_1, v_2, \dots, v_N)}_{\textit{mot de code}}$

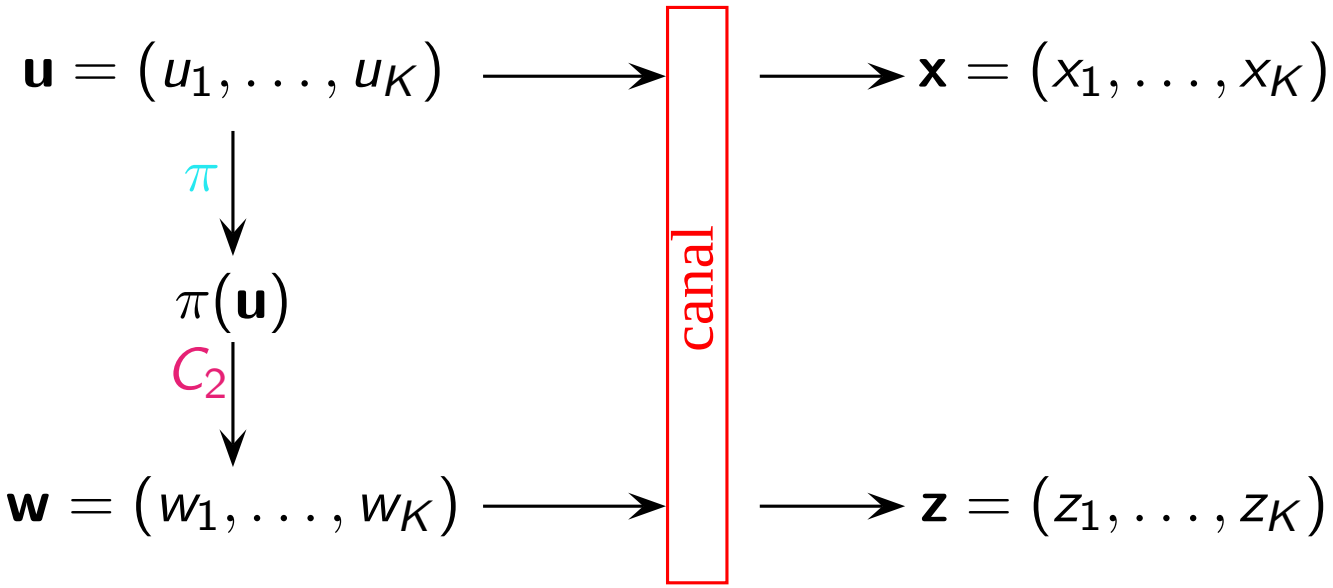
Turbo-codes

- Schéma de codage



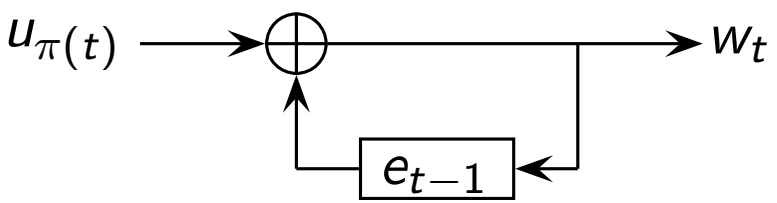
- Notations :
  - C_1 et C_2 des codes convolutifs
  - π une permutation de longueur K

Exemple



Avec C_2 tel que :

$$\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$$



Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0				
mot 2	1	1	?	1	0	0	?	1	?	?	0				
mot 3	1	0	?	1	0	?	1	0	1	1	0				
mot 4	?	?	1	0	1	?	0	?	1	0	0				

$$\pi = (\ ,\ ,\ ,\ ,\ )$$

Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0				
mot 2	1	1	?	1	0	0	?	1	?	?	0				
mot 3	1	0	?	1	0	?	1	0	1	1	0				
mot 4	?	?	1	0	1	?	0	?	1	0	0				

$$\pi = (\ ,\ ,\ ,\ ,\ )$$

Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0				
mot 2	1	1	?	1	0	0	?	1	?	?	0				
mot 3	1	0	?	1	0	?	1	0	1	1	0				
mot 4	?	?	1	0	1	?	0	?	1	0	0				

$$\pi = (3, \ , \ , \ , \ )$$

Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0	1			
mot 2	1	1	?	1	0	0	?	1	?	?	0	0			
mot 3	1	0	?	1	0	?	1	0	1	1	0	?			
mot 4	?	?	1	0	1	?	0	?	1	0	0	1			

$$\pi = (3, \ , \ , \ , \ )$$

Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0	1			
mot 2	1	1	?	1	0	0	?	1	?	?	0	0			
mot 3	1	0	?	1	0	?	1	0	1	1	0	?			
mot 4	?	?	1	0	1	?	0	?	1	0	0	1			

$$\pi = (3, 5, \ , \ , \ )$$

Exemple

- Code C_2 tel que : $\begin{cases} w_t = u_{\pi(t)} \oplus e_{t-1} \\ e_t = w_t \\ e_0 = 0 \end{cases}$

	x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	e_0	e_1	e_2	e_3	e_4
mot 1	1	1	1	0	0	1	1	1	0	?	0	1	1	1	0
mot 2	1	1	?	1	0	0	?	1	?	?	0	0	0	1	0
mot 3	1	0	?	1	0	?	1	0	1	1	0	?	1	0	1
mot 4	?	?	1	0	1	?	0	?	1	0	0	1	0	0	1

$$\pi = (3, 5, 4, 1, 2)$$

$\longrightarrow$ Reconstruction de la permutation pas à pas

Reconstruction de la permutation

- Information contenue dans une liste de mots de code

$$\mathbf{p}(\pi(t) = j | \pi(1), \ldots, \pi(t-1), \mathbf{x}^1, \mathbf{z}^1_{1..t}, \ldots, \mathbf{x}^M, \mathbf{z}^M_{1..t})$$

Proposition

Avec λ une constante de normalisation

$$\mathbf{p}(\pi(t) = j | \pi(1), \ldots, \pi(t-1), \mathbf{x}^1, \mathbf{z}^1_{1..t}, \ldots, \mathbf{x}^M, \mathbf{z}^M_{1..t})$$

$$= \lambda \Pi_{s=1}^{M} \mathbf{p}(\pi(t) = j | \pi(1), \ldots, \pi(t-1), \mathbf{x}^s, \mathbf{z}^s_{1..t})$$

- Information contenue dans chaque mots (mots indépendants)

$$\mathbf{p}(\pi(t) = j | \pi(1), \ldots, \pi(t-1), \mathbf{x}^s, \mathbf{z}^s_{1..t})$$

Reconstruction de la permutation

Notations :

- e_t l'état du codeur à l'instant t
- $\mathbf{x}^{\pi}_{1..t-1} = (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(t-1)})$
- $\sum_{\alpha \xrightarrow{ab} \beta}$: la somme sur tous les triplets (a, b, β) tels que le codeur passe de l'état α à l'état β en entrant a, la redondance émise correspondante est b

Proposition

Avec γ une constante de normalisation

$$\mathbf{p}(\pi(t) = j | \pi(1), \ldots, \pi(t-1), \mathbf{x}, \mathbf{z}_{1..t}) =$$

$$\frac{\gamma}{\mathbf{p}(x_j)} \times \sum_{\alpha} \sum_{\alpha \xrightarrow{ab} \beta} \mathbf{p}(x_j | u_j = a) \mathbf{p}(z_t | w_t = b) \mathbf{p}(e_{t-1} = \alpha | \mathbf{x}^{\pi}_{1..t-1}, \mathbf{z}_{1..t-1})$$

Résultats pratiques

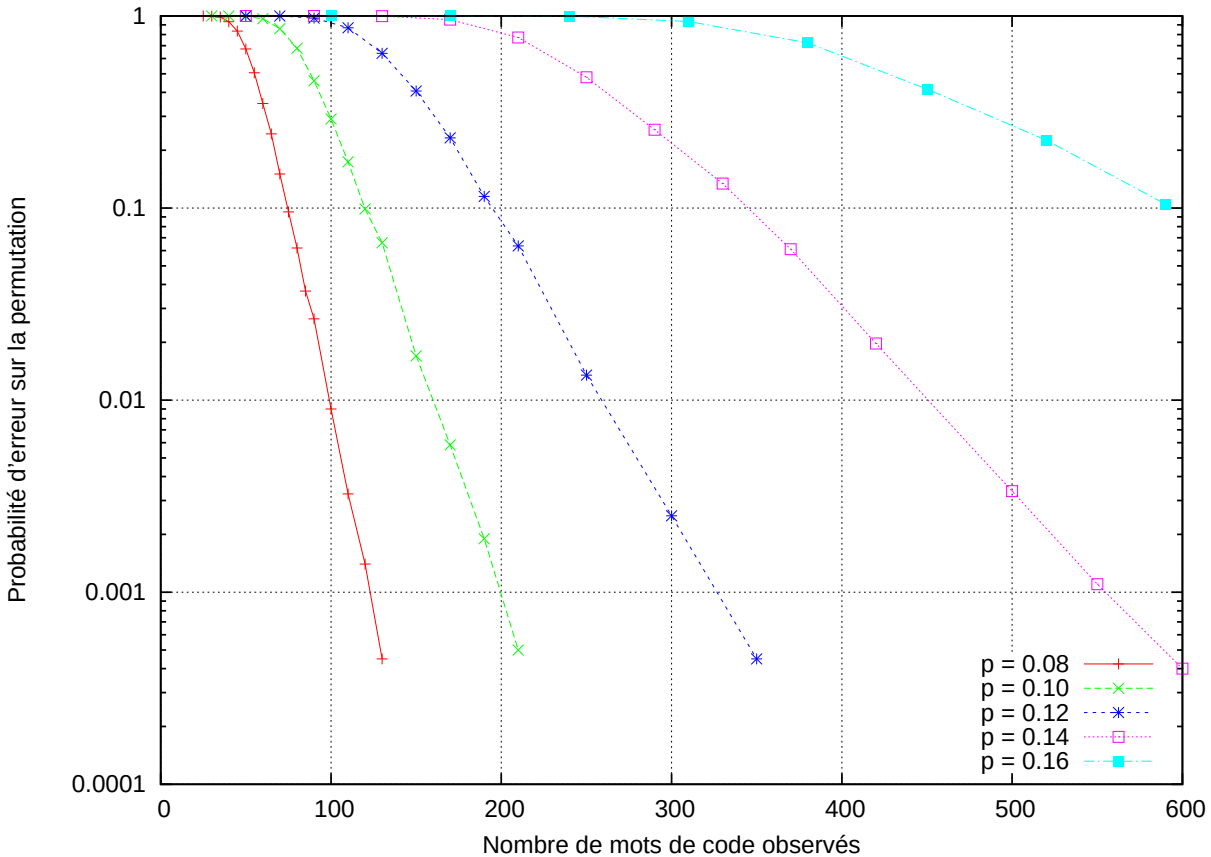
- Canal Gaussien (AWGN) d'écart type σ, $C_2 : (1, \frac{1+D^2}{1+D+D^2})$

K	σ	M	M[1]	temps de calcul en secondes	temps de calcul en secondes [1]
64	0.43	20	50	0.02	0.2
64	0.6	34	115	0.02	0.3
64	1	243	1243	0.17	12
512	0.6	46	170	1.87	11
512	0.8	111	600	7	37
512	1	346	2 800	17	173
512	1.1	660	3 837	20	357
512	1.3	1820	29 500	64	4 477
10 000	0.43	40	300	797	8173

[1] : Cluzeau, Finiasz, Tillich, "Methods for the Reconstruction of Parallel Turbo Codes," ISIT 2010, pp. 2008–2012.

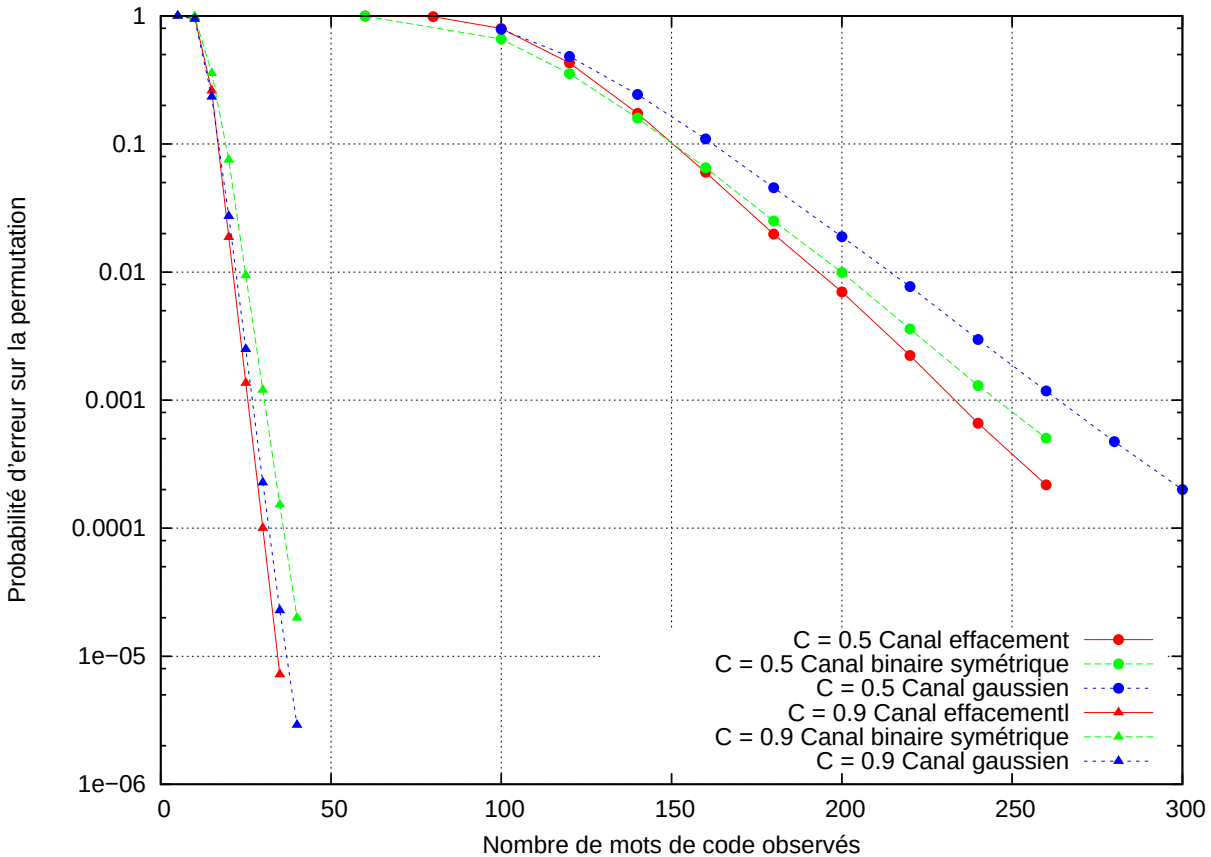
Résultats pratiques

- Canal Binaire Symétrique, $C_2 : (1, \frac{1+D^2}{1+D+D^2})$ et $K = 64$



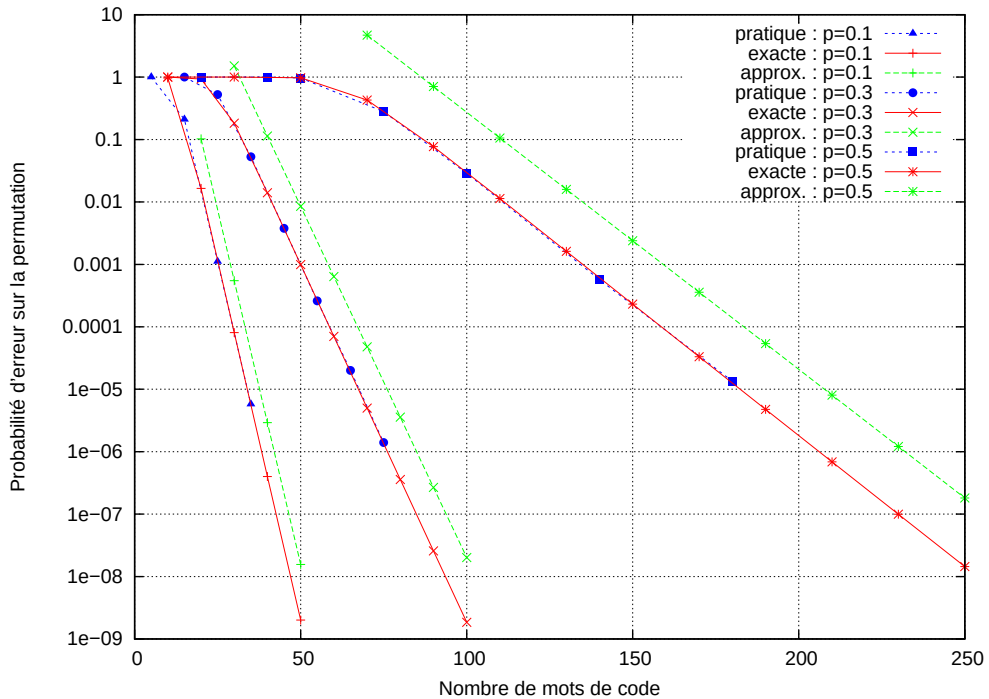
Résultats pratiques

- A capacité de canal fixée, $C_2 : (1, \frac{1+D^2}{1+D+D^2})$ et $K = 64$



Analyse théorique

- Hypothèses
  - Canal à effacement de probabilité p
  - $C_2 : (1, \frac{1}{1+D})$
- Probabilité de ne pas retrouver la permutation



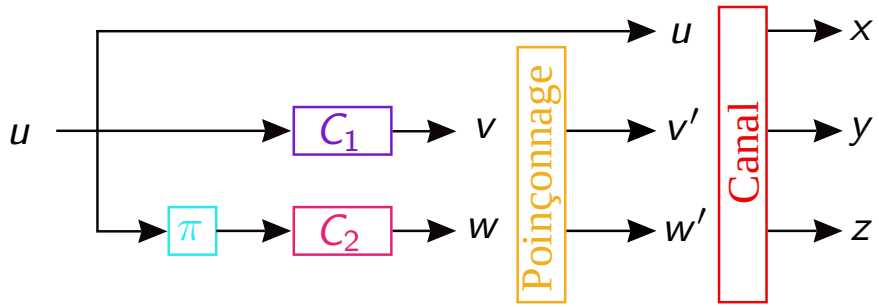
$$P_{erreur} = \mathcal{O}\left(K^2\left(\frac{2p + ((\sqrt{2}-1)p+1)^2(1-p)^2}{2(1+p^2-p)}\right)^M\right)$$

Conclusion

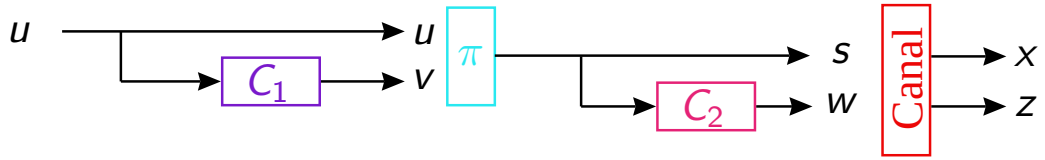
- Algorithme de reconstruction de la permutation d'un turbo-code
  - Peu complexe : $\mathcal{O}(MK^2)$
  - Utilise l'information de façon optimale
  - Avec de bons résultats dans le cas bruité

Conclusion

- Généralisable
  - Aux turbo-codes parallèles poinçonnés (faiblement)



  - Aux turbo-codes séries



- Perspective
  - Trouver une méthode pour le cas fortement poinçonné