Décodage itératif des codes correcteurs d'erreurs courts en bloc linéaires basé sur des treillis produits sectionnalisés

Senad Mohamed-Mahmoud[1,2], Jean-Claude Carlach[1], Patrick Adde[2] et Michel Jezequel[2]

[1]Orange-Labs
4 rue du Clos Courtel, 35512, Cesson-Sévigné, France

[2]Institut Telecom; Telecom Bretagne;
CNRS Lab-STICC UMR 3192
29238 Brest Cedex 3

senad.mohamedmahmoud@orange.com

Résumé − Cet article présente un nouvel algorithme d'estimation des probabilités a *posteriori* des bits d'un mot de code en bloc de longueur n de quelques centaines de bits au maximum, $n < 1000$. Le décodage quasi-optimal de ces codes courts est toujours un problème ouvert car les algorithmes de type Belief-Propagation(BP)[1] se heurtent notamment au problème des cycles courts dans leurs graphes de Tanner d'où des performances insuffisantes pour des codes de courtes longueurs mais de grandes distances minimales relatives (d_{min}/n). Nous proposons donc dans cet article un type d'algorithme hybride entre l'algorithme BP et l'algorithme de Viterbi[2]. Les données initiales sont les probabilités a *priori* des symboles ou bits reçus et les deux matrices génératrice **G** et de contrôle **H** du code. Cet algorithme utilise les techniques de décodage classiques sur treillis comme les algorithmes BCJR[3] ou SOVA[4]. Mais afin d'éviter la complexité trop grande du décodage sur le treillis global du code, l'algorithme utilise des treillis produits et sectionnalisés de complexité réduite construits à partir des treillis élémentaires représentant des lignes des matrices **G** et **H**. A chaque itération les treillis échangent des informations extrinsèques sur les groupes de bits formant les étiquettes des branches de leurs sections. Les résultats de simulation ainsi que la complexité de cette méthode sont montrés à la fin de cet article pour les codes de Hamming$(8, 4, 4)$ et de Golay$(24, 12, 8)$ dans le cas d'un canal gaussien.

Abstract − This article presents a new algorithm to estimate the a *posteriori* bit probabilities of a codeword belonging to a linear block code whose length n is a few hundreds of symbols ($n < 1000$). The quasi-optimal soft-decoding with low-complexity of the short block codes is still an open problem because the best known low-complexity algorithms such as Belief-Propagation(BP)[1] are impaired by short cycles into the Tanner graphs of the codes which gives low performance error-rates for short codes but with high relative minimum distance (d_{min}/n). So we propose in this article a new type of algorithm which is an hybrid between the BP algorithm and the Viterbi algorithm[2]. The initial data are the a *priori* symbols or bits probabilities of a received noisy vector and the generator or parity-check matrix of the code. This algorithm uses classical soft-decoding techniques based on trellises, i.e. BCJR[3] or SOVA[4]. But in order to avoid the huge decoding complexity on the full code trellis, the algorithm uses low complexity sectionalized product-trellis which are constructed based on elementary trellis representing the lines of generator matrix **G** and parity matrix **H** of the code. At each iteration the trellises exchange extrinsic information on the group of bits labeling the section branches. The complexity and the simulations results of this algorithm are shown at the end of this article for the $(8, 4, 4)$ Hamming code and the Golay$(24, 12, 8)$ code for a gaussian channel.

1 Introduction

L'algorithme de décodage de Viterbi[2] minimise la probabilité d'erreur de décodage d'un mot de code sachant les probabilités ou les LLR (Log-Likelihood Ratio) des symboles du mot reçu, tandis que l'algorithme BCJR[3] minimise la probabilité d'erreur par symbole de ce mot de code. Ces deux méthodes nécessitent la construction d'un treillis global du code et ont donc une complexité d'ordre $\mathcal{O}(2^{n-k})$ d'après Wolf[5], où $(n-k)$ est le nombre de bits de redondance du code et k est le nombre de bits d'informations.

L'algorithme présenté ici évite donc de calculer un treillis global de complexité trop grande et utilise un ensemble de treillis-produits de faibles complexités construits à partir des treillis élémentaires représentant des lignes des matrices **G** et **H**. Chaque treillis-produit est considéré comme un code de base et leur ensemble une concaténation de codes de base qui s'échangent, d'une manière itérative, des informations extrinsèques comme dans le décodage des codes LDPC[1] ou des turbocodes[6].

Après cette introduction, cet article est structuré en 4 sections. La section 2 décrit le produit de deux treillis élémentaires, présente la sectionnalisation d'un treillis et explicite l'algorithme proposé sur l'exemple du code Hamming$(8, 4, 4)$. La section 3 présente les résultats de

simulation de l'algorithme pour les codes de Hamming $(8, 4, 4)$ et de Golay$(24, 12, 8)$. La section 4 étudie la complexité de l'algorithme proposé. Finalement, la section 5 conclut cet article en présentant quelques perspectives.

2 Description de l'algorithme

Nous explicitons notre algorithme de décodage sur le code de Hamming de paramètres $(n = 8, k = 4, d_{min} = 4)$ dont les matrice génératrice $\mathbf{G}$ et de contrôle $\mathbf{H}$ sont données ci-dessous sous forme systématique:

$$\mathbf{G} = [\mathbf{Id_4}, \mathbf{P}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{H} = \left[\mathbf{P}^{tr}, \mathbf{Id_4}\right] = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

où Id_4 est la matrice identité de dimensions 4, et tr est l'opérateur de transposition.
Un treillis élémentaire associé à une ligne des matrices $\mathbf{G}$ et $\mathbf{H}$ est constitué par la suite de sections correspondant respectivement aux valeurs binaires 0 et 1 tel que ci-dessous:

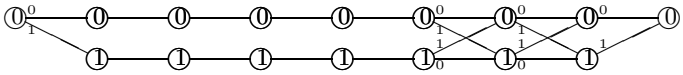

FIG. 1: Treillis élémentaire T_0 associé à 10000111.

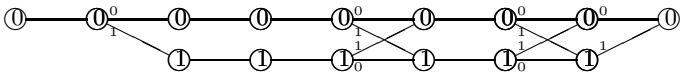

FIG. 2: Treillis élémentaire T_1 associé à 01001011.

2.1 Produit de deux treillis

Soient deux treillis T_0 et T_1, de même nombre n de sections d'indice $t \in \{0, 1, \ldots, n-1\}$ tels que: $T_0 = \{(s_0^t, e_0^t, s_0^{t+1})\}$ et $T_1 = \{(s_1^t, e_1^t, s_1^{t+1})\}$. Chaque section d'un treillis T_i, $i \in \{0, 1\}$, est un ensemble de branches d'étiquettes e_i^t reliant un état de départ s_i^t à un état d'arrivée s_i^{t+1}, $T_i = \{(s_i^t, e_i^t, s_i^{t+1})\}$.

Le produit $T_0 \otimes T_1$ est défini par [7]:

$$T_0 \otimes T_1 = \{((s_0^t, s_1^t), (e_0^t, e_1^t), (s_0^{t+1}, s_1^{t+1}))|e_0^t = e_1^t\}.$$

Ce produit de 2 treillis effectue donc le produit cartésien de chacune des 3 composantes des triplets-branches mais en supprimant les branches si $e_0^t \neq e_1^t$. A titre d'exemple, le treillis $T_0 \otimes T_1$ résultant du produit des deux treillis élémentaires T_0 et T_1 décrits précédemment est représenté ci-dessous:

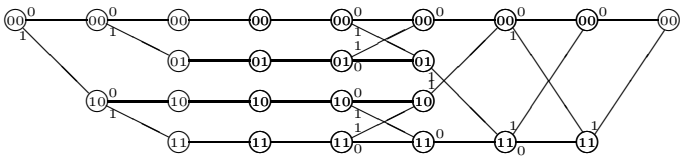

FIG. 3: Produit $T_0 \otimes T_1$ des deux treillis T_0 et T_1 précédents associés respectivement aux générateurs 10000111 et 01001011.

2.2 Sectionnalisation des treillis

La sectionnalisation[7] d'un treillis consiste à fusionner des sections successives pour en former une seule. Chaque section résultant d'une fusion a donc pour étiquettes de branches la concaténation des étiquettes des branches de transition entre les états de départ et les états d'arrivée des sections fusionnées. Pour exprimer la sectionnalisation sous forme mathématique, considérons deux sections successives S_0 et S_1 d'un même treillis T telles que: $S_0 = \{(a_0, e_0, b_0)\}$ et $S_1 = \{(a_1, e_1, b_1)\}$ où a_i, b_i et e_i, $i \in \{0, 1\}$, sont respectivement les ensembles d'états de départ, d'états d'arrivée et d'étiquettes des branches constituant une section S_i. La sectionnalisation $(S_0|S_1)$ ou fusion des sections S_0 et S_1 est définie telle que:

$$(S_0|S_1) = \{(a_0, b_0), (e_0, e_1), (a_1, b_1)|b_0 = a_1\}$$

ce qui se simplifie par la suppression des produits cartésiens ci-dessus pour lesquels $b_0 \neq a_1$: $(S_0|S_1) = \{a_0, (e_0, e_1), b_1\}$.

2.3 Structure globale de l'algorithme

La modulation utilisée est la modulation de phase à 2 états (MDP2) qui associe le bit 0 (resp. 1) à la valeur modulée -1.0 (resp.$+1.0$). Le canal de transmission est le canal gaussien (AWGN: Additive White Gaussian Noise) de bruit gaussien centré d'écart type σ. Les valeurs de métriques de fiabilité reçues correspondantes aux bits c_i et aux valeurs reçues y_i sont notées $LLR_{apr}(c_i) = \frac{2}{\sigma^2} * y_i$. Le paramètre n_{lignes} est le nombre de lignes des matrices $\mathbf{G}$ et $\mathbf{H}$ associées à un treillis-produit, et $n_{sections}$ le nombre de sections regroupées par la sectionnalisation. Le nombre n_p de treillis-produits est donc ici $n_p = 2 * k/n_{lignes}$. Nous nous plaçons bien sûr dans un premier temps dans le cas le plus simple où n_{lignes} divise k. Les treillis-produits $T_{p_j}^G$ (resp. $T_{p_j}^H$), d'indices $j = 0, 1, \ldots, n_p/2 - 1$, issus des lignes de la matrice $\mathbf{G}$ (resp. la matrice $\mathbf{H}$), sont donnés par:

$$T_{p_j}^G = \otimes_{i=0}^{n_{lignes}-1} T_{i+j*n_{lignes}}^G.$$

$$T_{p_j}^H = \otimes_{i=0}^{n_{lignes}-1} T_{i+j*n_{lignes}}^H.$$

Où T_m^G, $m \in \{0, 1, \ldots, k-1\}$, (resp. T_m^H) représente le treillis élémentaire associé à la m$^{\text{ème}}$ ligne de la matrice $\mathbf{G}$ (resp. $\mathbf{H}$) et $\otimes$ designe le produit des treillis. La Figure 4 ci-après décrit la structure graphique globale correspondant au code de Hamming$(8, 4, 4)$.

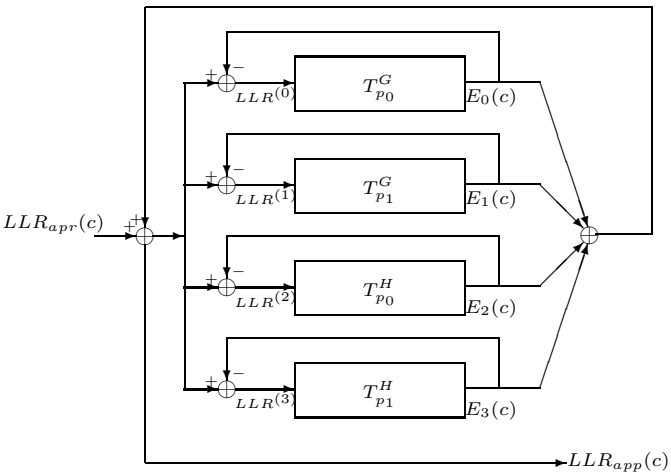

FIG. 4: Structure de décodage pour le code de Hamming$(8, 4, 4)$.

3 Résultats de simulation

Dans cette section, nous présentons les résultats de simulation obtenus par cette méthode de décodage pour les codes de Hamming$(8, 4, 4)$ et de Golay$(24, 12, 8)$.
Les courbes de taux d'erreur binaire (BER) présentées sur les Figures 5 et 6 montrent les performances de la méthode de décodage obtenues pour les codes de Hamming$(8, 4, 4)$ et de Golay$(24, 12, 8)$ en canal gaussien. Nous avons tracé sur les mêmes figures les performances obtenues avec un décodage optimal ou ML-exhaustif (pour Maximum Likelihood) pour les mêmes codes.
Les performances du code de Hamming$(8, 4, 4)$ sont obtenues avec les paramètres $n_{lignes} = 2$ et $n_{section} = 1$ et celles du code de Golay$(24, 12, 8)$ sont obtenues avec les paramètres $n_{lignes} = 4$ et $n_{section} = 6$.

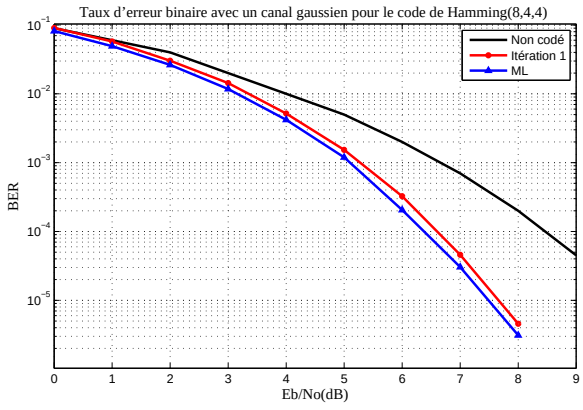

FIG. 5: Performance de l'algorithme de décodage pour le code de Hamming$(8, 4, 4)$ avec les paramètres $n_{lignes} = 2$ et $n_{sections} = 1$.

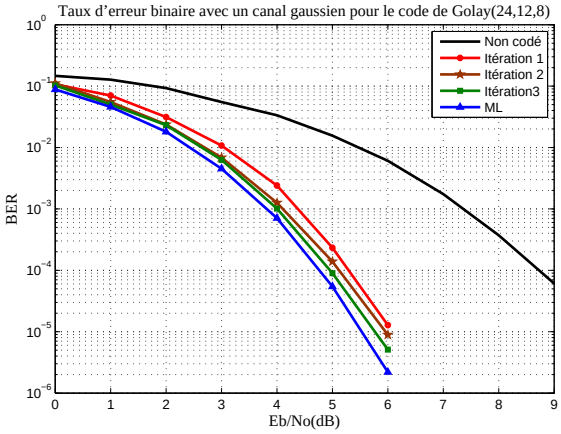

FIG. 6: Performance de l'algorithme de décodage pour le code de Golay$(24, 12, 8)$ avec les paramètres $n_{lignes} = 4$ et $n_{sections} = 6$.

Le tableau 1 ci-dessous donne les performances pour certaines configurations de décodage $(n_{lignes}, n_{sections})$ pour le code de Golay$(24, 12, 8)$. La valeur numérique associée à chaque configuration est l'écart de performance (en dB) calculé par rapport au ML (Maximum Likelihood) pour un taux d'erreur binaire de 10^{-4} et 3 itérations.

n_{lignes} \ $n_{sections}$	2	4	6
2 (12 treillis)	1.1 dB	0.8 dB	0.6 dB
3 (8 treillis)	0.8 dB	0.6 dB	0.4 dB
4 (6 treillis)	0.6 dB	0.4 dB	0.2 dB

TAB. 1: Comparaison entre quelques configurations de décodage $(n_{lignes}, n_{sections})$ pour le code Golay$(24, 12, 8)$.

D'après le tableau 1, la performance de l'algorithme proposé dépend des paramètres n_{lignes} et $n_{sections}$ utilisés. Les performances sont d'autant meilleures que ces paramètres sont grands. Mais leur choix reste contraint par le compromis performance/complexité. Par exemple les deux configurations $(3, 2)$ et $(2, 4)$ donnent les mêmes performances mais le choix entre elles sera déterminé par leurs complexités que nous étudierons dans le paragraphe 4.
Un treillis-produit représente un super-code pour le code en bloc C, et donc comporte plus de chemins que le treillis du code C. L'augmentation de la taille du treillis-produit (en augmentant n_{lignes}), réduit l'espace des mots du super-code et donc augmente le pourcentage des mots du code C dans cet espace. Les probabilités extrinsèques fournies par ce treillis-produit sont calculées à l'aide des mots du super-code. Elles sont donc d'autant plus pertinentes que le pourcentage des mots du code C dans l'espace des mots du super-code est grand. Ceci explique la croissance des performances avec l'augmentation de la taille du treillis-produit.

L'utilisation des matrices génératrice **G** et de contrôle **H** du code crée de la diversité et améliore les performances. En effet, l'utilisation de la matrice **G** seule (resp. **H**) conduit à des treillis-produits ayant des sections nulles dans la partie systématique (resp. partie redondance) du treillis. Le treillis-produit ne fournit donc aucune information aux autres treillis-produits sur les bits étiquetant des sections nulles, et donc certains bits du mot reçu profitent moins du décodage itératif. Par contre, en utilisant les deux matrices **G** et **H**, si un bit n'est pas couvert par les treillis-produits relevant de l'une des matrice, il le sera par les treillis-produits relevant de l'autre matrice. Dans ce cas, tous les bits sont couverts et profitent du décodage itératif pour augmenter leur probabilités d'être décidés correctement.

4 Complexité de décodage

La complexité de la méthode de décodage proposée est la complexité d'un décodeur élémentaire multipliée par le nombre total de ces décodeurs n_p et le nombre maximal d'itérations I. Un décodeur élémentaire effectue un algorithme BCJR sur un treillis-produit à n_e états ($n_e = 2^{n_{lignes}}$) et de longueur $L = n/n_{sections}$. Le nombre de branches n_b incidentes à un état d'une section d'un treillis-produit est égal à $n_b = 2^{n_{sections}}$. La complexité totale de l'algorithme proposé est alors donnée par: $n_p * ((L-2) * n_e * n_b + 2 * n_b) * I$. La complexité depend donc en grande partie du nombre de treillis élémentaires n_{lignes} associés à un treillis-produit et le nombre de sections groupées $n_{sections}$ sur chaque treillis-produit. Le choix de ces deux paramètres est donc important pour assurer un bon compromis performance/complexité. Une manière pour réduire la complexité sans dégrader les performances est de ne pas faire les calculs pendant les phases forward/backward de l'algorithme BCJR sur les sections nulles d'un treillis-produit. Une section nulle S_i est une section où chaque états s de l'étage i est connecté à un état homologue s de l'étage $i+1$ par l'ensemble des branches possibles ($2^{n_{sections}}$ branches). Deux états de cette section ne sont pas connectés entre eux s'ils sont différents. Pour le code de Golay$(24, 12, 8)$, cela permet de réduire la complexité globale de décodage d'environ 17 à 25% selon la configuration utilisée. Le tableau 2 donne la complexité pour quelques configurations et 3 itérations.

Algorithme de décodage	opérations $(+,\times)$
$(n_{lignes} = 2, n_{sections} = 6)$	34 020
$(n_{lignes} = 3, n_{sections} = 6)$	40 224
$(n_{lignes} = 4, n_{sections} = 6)$	64 002
algorithme ML-exhaustif	98 304

TAB. 2: Complexités de quelques configurations de décodage $(n_{lignes}, n_{sections})$ pour le code Golay$(24, 12, 8)$.

5 Conclusion et perspectives

Dans cet article, un nouvel algorithme de décodage itératif pour les codes en bloc a été présenté. Les performances de cette technique de décodage ont été présentées et comparées à celles du décodage optimal pour les codes de Hamming $(8, 4, 4)$ et de Golay$(24, 12, 8)$. La complexité de ce schéma de décodage depend des treillis-produit utilisés et la façon dont ils sont sectionnalisés.

References

[1] D.J.C. MacKay and R.M. Neal, *Good codes based on very sparse matrices*, in Cryptography and Coding 5th IMA Conf., (Springer),pp.100-111, 1995.

[2] G.D. Forney, *The Viterbi Algorithm*, Proceedings of the IEEE, Vol. 61, no.3, pp. 268-278, March 1973.

[3] L.R. Bahl, J. Cocke, F. Jelinek and R. Raviv, *Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate*, IEEE Transactions on Information Theory, Vol. IT-20, pp. 284-287, March 1974.

[4] C. Berrou, P. Adde, E. Angui and S. Faudeil, *A Low Complexity Soft-Output Viterbi Decoder Architecture*, ICC'93,Geneva, Switzerland, Vol. 2, pp. 737-740, May 1993.

[5] J.K. Wolf, *Efficient Maximum Likelihood Decoding of Linear Block Codes using a Trellis*, IEEE Transactions on IT, Vol. IT-24, pp. 76-80, January 1978.

[6] C. Berrou, A. Glavieux and P. Thitimajshima, *Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes*, ICC'93, Geneva, Switzerland, Vol. 2, pp. 1064-1070, May 1993.

[7] A.R. Calderbank, G.D. Jr. Forney and A. Vardy, *Minimal Tail-biting Trellises: the Golay Code and More*, IEEE Transactions on IT, Vol. IT.45, no. 5, pp.1435-1455, July 1999.