Post-traitement OSD pour le décodage BP basé sur des ensembles de LLRs complémentaires

Joachim ROSSEEL[1,2] Valérian MANNONI[1] Valentin SAVIN[1] Inbar FIJALKOW[2]

[1]CEA-Leti, Université Grenoble Alpes, F-38000 Grenoble, France

[2]ETIS UMR 8051, CY Cergy Paris Univ., ENSEA, CNRS F-95000, France

Résumé – Cet article traite du post-traitement par OSD (*Ordered Statistics Decoding*) appliqué aux sorties souples de l'algorithme BP (*Belief Propagation*). Notre approche consiste dans un premier temps à modéliser la somme pondérée des LLRs (*Log Likelihood Ratios*) *a posteriori* calculés au cours des itérations du BP en un seul neurone. Ce neurone est alors entraîné avec la fonction de coût focale afin d'obtenir pour chaque échec du BP un ensemble de LLRs accumulés qui soient adaptés au post-traitement par OSD. Nous proposons ensuite une procédure de sélection récursive d'ensembles de LLRs pour un post-traitement OSD multiple. Cette sélection est réalisée à partir des ensembles des LLRs *a posteriori* calculés à chaque itération du BP ainsi que des LLRs accumulés et optimisés pour l'OSD, selon leurs probabilités conjointes d'échec avec le post-traitement OSD. Un OSD est alors appliqué sur chaque ensemble de LLRs de la sélection. Nos résultats montrent que cette nouvelle méthode de décodage fournit un moyen efficace d'atteindre la performance du décodeur par maximum de vraisemblance pour des codes LDPC (*Low Density Parity Check*) courts.

Abstract – This article deals with Ordered Statistics Decoding (OSD) applied to the soft outputs of the Belief Propagation (BP) algorithm. We first model the weighted sum of the *a posteriori* LLRs across BP decoding iterations into a neuron. The neuron is then trained with the focal loss to compute for each BP decoding failure a set of accumulated Log Likelihood Ratios (LLRs) suited for OSD post-processing. Then, we then propose a recursive selection procedure of LLR sets, for multiple OSD post-processing. This selection is carried out from the sets of *a posteriori* LLRs calculated at each BP iteration, and from the accumulated LLRs optimized for the OSD, based on their joint probabilities of failure with OSD post-processing. An OSD is then applied to each set of LLRs belonging to the selection. Our results show that this new decoding method provides an effective way to bridge the gap to maximum likelihood decoding for short Low Density Parity Check (LDPC) codes.

1 Introduction

L'émergence des systèmes de communication pour l'Internet des Objets, avec notamment la transmission de paquets courts, a revitalisée la recheche et la pratique de codes correcteurs d'erreurs efficaces pour des tailles de messages allant de quelques dizaines à quelques centaines de bits. Bien que des progrès ont été réalisés dans la compréhension des limites du codage en taille courte [1], la conception de codes courts efficaces et d'algorithmes de décodage associés posent encore de nombreux défis [2].

Les codes LDPC (*Low Density Parity Check*) [3] sont une classe de codes correcteurs d'erreurs définie par leur matrice de parité creuse et représentée par leur graphe de Tanner.Ils sont connus pour leurs excellentes performances asymptotiques (taille de mot de code très importante), proche de la limite de Shannon, lorsqu'ils sont décodés avec l'algorithme par propagation de croyance (*Belief Propagation* ou BP) [4]. Cependant, l'apparition de cycles pour les codes courts dégradent considérablement la performance du BP.

Afin d'améliorer la capacité de correction des codes LDPC courts, le BP peut être associé à un décodeur par statistiques ordonnées (*Ordered Statistics Decoding* ou OSD) [5]. Une première méthode de combinaison, introduite par [6], consiste à appliquer un OSD d'ordre faible à la fin de chaque itération du BP. Bien que les performances soient effectivement améliorées, l'utilisation de l'OSD à chaque itération augmente la complexité de manière significative. Pour limiter cette complexité, l'OSD peut aussi être utilisé comme une étape de post-traitement, exploitant la sortie souple du BP uniquement lorsque ce dernier n'arrive pas à converger vers un mot de code. Dans [7], une nouvelle mesure de fiabilité des noeuds de donnée pour l'OSD est calculée à partir de l'accumulation des LLRs *a posteriori* au cours des itérations du BP. Le post-traitment OSD est également effectué dans [8], après une diversité de décodage composée de plusieurs BPs modélisés par des réseaux de neurones récurrents (*BP-Reccurent Neural Networks* ou BP-RNNs) et entraînés pour traiter des classes d'erreurs spécifiques.

Dans ce papier, nous voulons améliorer le décodage des codes LDPC courts avec un post-traitement OSD des sorties du BP. Notre appoche consiste alors à modéliser une somme pondérée basée sur l'approche de [7] en un neurone, puis à l'optimiser afin de calculer pour chaque mot destiné au post-traitement l'ensemble des LLRs accumulés et adaptés à l'OSD. Cette optimisation est notamment réalisée grâce à une fonction de coût focale [9]. Nous proposons ensuite de construire récursivement une sélection d'ensembles de LLRs à partir des ensembles de LLRs *a posteriori* calculés pour chaque itération de décodage BP et de l'ensemble des LLRs accumulés et optimisés pour l'OSD. La sélection se base sur la complementarité de ces ensembles, c'est à dire selon leurs probabilités conjointes d'échec avec le post-traitement OSD. Un OSD est alors appliqué sur chaque ensemble de LLRs de la sélection.

L'article est organisé comme suit. La section 2 introduit les notations, rappelle l'algorithme OSD et décrit l'approche

Ce travail a été réalisé dans le cadre du projet ANR AI4CODE (ANR-21-CE25-0006).

de [7]. L'entraînement du neurone modélisant le LLR accumulé avec la fonction de coût focale, et la procédure pour sélectionner des ensembles de LLRs complémentaires sont ensuite expliqués en section 3. Enfin, la section 4 présente les résultats de simulation, et la section 5 conclut cet article.

2 Post-traitement OSD sur les LLRs *a posteriori* du BP

Nous considérons un code LDPC représenté par son graphe de Tanner (bipartite) avec N noeuds de données, M noeuds de parité, denotés respectivement par $n \in \{1, \dots, N\}$ et $m \in \{1, \dots, M\}$, et $K = N - M$ bits d'information. Le décodage par BP consiste en un échange itératif de messages le long des arêtes du graphe de Tanner, où chaque message fournit une estimation LLR du bit ou noeud de donnée incident [4].

D'autre part, nous considérons l'OSD [5], un algorithme de décodage capable d'approcher la performance de décodage par maximum de vraisemblance (*Maximum Likelihood* ou ML) pour des codes linéaires de tailles moyennes, avec une complexité polynomiale. Il peut être utilisé comme un décodeur à part entière, exploitant la sortie souple du canal, ou comme une étape de post-traitement exploitant la sortie souple d'un décodeur à décision souple tel que le BP. Dans l'OSD, les noeuds de données sont tout d'abord triés selon leur *fiabilité*, mesurée par la valeur absolue de la décision souple correspondante. Dans le cas d'un post-traitemenent du BP, la décision souple, et donc la fiabilité, dépend des LLRs *a posteriori* du BP. La matrice de parité du code est ensuite mise sous une forme systématique, notée H_{syst}, de sorte à avoir les K premières colonnes correspondant aux noeuds de données les plus fiables possibles.

Dans l'OSD-0, la décision dure est prise sur les noeuds de donnée les plus fiables, et les noeuds de donnée les moins fiables sont déterminés en résolvant un système linéaire donné par H_{syst}. Le décodage est ainsi réussi si les bits les plus fiables sont exempts d'erreur. Pour traiter les cas où ces noeuds de données contiennent des erreurs, l'OSD-o considère tous les choix possibles d'au plus o erreurs parmi les noeuds de données les plus fiables. Pour chaque choix, la décision dure initiale des noeuds de données correspondants est inversée, et les bits les moins fiables sont déterminés à nouveau en résolvant le système linéaire donné par H_{syst}. Cette procèdure produit une liste de $\sum_{p=0}^{o} \binom{K}{p}$ mots de code, notée $\mathcal{S}$. Le mot de code le plus vraisemblable dans $\mathcal{S}$ est ensuite sélectionné grâce au critère par maximum de vraisemblance :

$$\hat{\mathbf{c}} = \arg\max_{\hat{\mathbf{c}}_\mathbf{s} \in S} P(\hat{\mathbf{c}}_\mathbf{s}|\mathbf{y}) \tag{1}$$

où $\hat{\mathbf{c}}_\mathbf{s}$ dénote un mot de code déterminé par l'OSD-o et $\mathbf{y} = \{y_1, \dots, y_N\}$ le mot reçu en sortie du canal.

Afin d'améliorer les performances du post-traitement OSD des sorties du BP, [7] propose une nouvelle mesure de la fiabilité des noeuds de donnée pour l'OSD-o, basée sur l'accumulation des LLRs *a posteriori* au cours des itérations du BP. Plus précisement, si le BP n'obtient pas un mot de code après un nombre d'itérations maximale I, la fiabilité $\hat{L}_n^{(\text{S})}$ d'un noeud de donnée n est calculée comme suit :

$$\hat{L}_n^{(\text{S})} = \sum_{i=0}^{I} \hat{L}_n^{(i)}, n \in [1, N] \tag{2}$$

où $\hat{L}_n^{(0)}$ correspond au LLR observé du bit n, et $\hat{L}_n^{(i)}$ à son LLR *a posteriori* calculé par le BP à l'itération i, pour $i \in [1, I]$.

Intuitivement, cette mesure de fiabilité décrite par l'équation (2) permet d'obtenir une fiabilité plus forte sur les noeuds de donnée dont le signe des $\hat{L}_n^{(i)}$ restent le même au cours des itérations du BP, que sur ceux dont le signe des $\hat{L}_n^{(i)}$ changent. L'ensemble $\hat{\mathbf{L}}^{(\text{S})} := \{\hat{L}_1^{(\text{S})}, \dots, \hat{L}_N^{(\text{S})}\}$ assure par conséquent que les noeuds de donnée les plus fiables et fixés par l'OSD correspondent à des bits dont le BP est « certain ».

3 Post-traitement OSD mutiples

3.1 Calcul d'un LLR accumulé et optimisé pour le post-traitement OSD

On s'intéresse dans cette section à determiner un LLR accumulé au cours des itérations et qui soit plus adapté au post-traitement OSD. Pour cela, nous introduisons des poids $w^{(i)} \geq 0$, $i \in [0, I]$, dans l'équation (2). Nous proposons ainsi de calculer la fiabilité d'un noeud de donnée n par le LLR accumulé des I itérations suivant :

$$\hat{L}_n^{(\text{NS})} = \sum_{i=0}^{I} w^{(i)} \hat{L}_n^{(i)}, n \in [1, N] \tag{3}$$

Cette équation est ensuite modélisée par un simple neurone. Pour créer le jeu d'entraînement du neurone, nous générons dans un premier temps un jeu $\mathcal{T}_{\text{BP}}$ de mots de code bruités, en considérant un canal AWGN à entrée binaire, avec un alphabet d'entrées BPSK (± 1), et un bruit de variance σ^2 fixée. $\mathcal{T}_{\text{BP}}$ est ensuite décodé par le BP. Comme le canal AWGN et le BP sont symétriques, uniquement le mot de code tout zéro est transmis. Les mots bruités de $\mathcal{T}_{\text{BP}}$ pour lesquels le BP n'arrive pas à converger vers un mot de code au bout de I itérations forment un sous-ensemble de $\mathcal{T}_{\text{BP}}$, noté $\mathcal{T}_{\text{BP-OSD}}$. Les ensembles $\hat{\mathbf{L}}_\mathbf{n} := \{\hat{L}_n^{(0)}, \dots, \hat{L}_n^{(I)}\}$, avec $n \in [1, N]$, de chaque mot bruité appartenant à $\mathcal{T}_{\text{BP-OSD}}$ constituent alors le jeu d'entraînement du neurone. Ce jeu d'entraînement permet ainsi au neurone d'être entraîné uniquement dans les cas où le BP ne converge pas vers un mot de code.

Pour optimiser les poids du neurone, nous proposons d'utiliser la fonction de coût focale introduite dans [9] :

$$\begin{aligned}\text{FL}\big(b_n, \hat{L}_n^{(\text{NS})}\big) = &-b_n \sigma\big(\hat{L}_n^{(\text{NS})}\big)^\gamma \log\Big(1 - \sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big) \\ &- (1 - b_n)\Big(1 - \sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big)^\gamma \log\Big(\sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big)\end{aligned} \tag{4}$$

où b_n est la valeur attendue du bit n, $\sigma(x) = (1 + e^{-x})^{-1}$ est la fonction sigmoïde convertissant $\hat{L}_n^{(\text{NS})}$ en la probabilité que le bit n soit égal à zéro, et $\gamma \geq 0$ est un hyper-paramètre ajustable. En supposant que le mot tout zéro est transmis, soit $b_n = 0$ pour $n \in [1, N]$, (4) se simplifie en :

$$\text{FL}\big(\hat{L}_n^{(\text{NS})}\big) = -\Big(1 - \sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big)^\gamma \log\Big(\sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big) \tag{5}$$

Le facteur focal $\Big(1 - \sigma\big(\hat{L}_n^{(\text{NS})}\big)\Big)^\gamma$ permet au neurone de se focaliser sur les $\hat{L}_n^{(\text{NS})}$ négatifs avec les amplitudes les plus importantes en valeur absolue lors de l'entraînement, et donc d'optimiser les poids afin de diminuer leurs fiabilités. Or, ces

$\hat{L}_n^{(\text{NS})}$ négatifs correspondent aux noeuds de données erronées et potentiellement sélectionnés parmi les K plus fiables lors du décodage par OSD. Par conséquent, minimiser la fonction de côut focale revient à minimiser le nombre de bits erronés parmi les K plus fiables, et ainsi permet d'améliorer la performance du post-traitement OSD. Nous avons observé que le LLR accumulé $\hat{\mathbf{L}}^{(\text{NS})}$ optimisé avec une entropie croisée binaire (soit $\gamma = 0$) est moins performant avec le post-traitement OSD qu'un $\hat{\mathbf{L}}^{(\text{NS})}$ optimisé avec un coût focal $\gamma > 0$. De plus, une dépendance des poids en fonction de n dans l'équation (3) a été testé, mais nous n'avons pas remarqué de meilleures performances.

3.2 Sélection d'ensembles de LLRs

Afin de se rapprocher de la performance de décodage ML, nous proposons ici une méthode de décodage où un post-traitement OSD-o est appliqué sur chaque ensemble de LLRs d'une sélection $\mathcal{L}_Z$ de Z ensembles de LLRs, avec $Z \in [1, I+2]$. Pour effectuer cette sélection, la complémentarité des $\hat{\mathbf{L}}^{(\mathbf{i})} := \left\{\hat{L}_1^{(i)}, \ldots, \hat{L}_N^{(i)}\right\}$, $i \in [0, I]$, et de $\hat{\mathbf{L}}^{(\text{NS})}$ avec l'OSD est évaluée via le nombre d'erreurs restantes après un post-traitement OSD-o. Nous générons dans un premier temps un jeu de test $\mathcal{T}_{\text{BP-OSD}}$, en suivant la méthode décrite dans la section précédente. La performance de décodage du post-traitement OSD-o est ensuite évaluée sur $\mathcal{T}_{\text{BP-OSD}}$ avec chaque $\hat{\mathbf{L}}^{(\mathbf{i})}$ et avec $\hat{\mathbf{L}}^{(\text{NS})}$. Nous dénotons par $\mathcal{F}^{(i)} \subset \mathcal{T}_{\text{BP-OSD}}$ (resp. $\mathcal{F}^{(\text{NS})} \subset \mathcal{T}_{\text{BP-OSD}}$) le sous-ensemble des mots bruités sur lequel $\hat{\mathbf{L}}^{(\mathbf{i})}$ (resp. $\hat{\mathbf{L}}^{(\text{NS})}$) a engendré un échec lors du décodage par l'OSD-o. Par la suite, nous construisons récursivement une liste *ordonnée* des ensembles de LLRs, dénotée $\mathcal{L}$. Cette liste est intialisée avec $\hat{L}^{(\text{NS})}$, soit $\mathcal{L} = \left\{\hat{\mathbf{L}}^{(\text{NS})}\right\}$, étant donné que $\hat{\mathbf{L}}^{(\text{NS})}$ est optimisé pour le post-traitement OSD. Pour ajouter un nouvel ensemble de LLRs $\hat{\mathbf{L}}^{\text{new}}$ à $\mathcal{L}$, nous proposons d'appliquer la régle suivante :

$$\hat{L}^{\text{new}} = \underset{\hat{\mathbf{L}}^{(\mathbf{i})} \in \{\hat{\mathbf{L}}^{(\mathbf{0})}, \ldots, \hat{\mathbf{L}}^{(\mathbf{I})}\} \setminus \mathcal{L}}{\arg\min} \left| \mathcal{F}_{\mathcal{L}} \cap \mathcal{F}^{(i)} \right|, \tag{6}$$

où $\mathcal{F}_{\mathcal{L}} := \mathcal{F}^{(\text{NS})}$ si $\mathcal{L} = \left\{\hat{\mathbf{L}}^{(\text{NS})}\right\}$, $\mathcal{F}_{\mathcal{L}} := \mathcal{F}^{(\text{NS})} \cap \left(\cap_{\hat{\mathbf{L}}^{(\mathbf{i})} \in \mathcal{L}} \mathcal{F}^{(i)} \right)$ sinon. La régle ci-dessus est appliquée $I+1$ fois, jusqu'à ce que $\mathcal{L}$ contienne tous les ensembles de LLRs. Si l'argument minimum de (6) n'est pas unique, un choix arbitraire est fait parmis ces valeurs.

Pour $Z \leq I+2$, $\mathcal{L}_Z$ est définie comme étant la sous liste des Z premiers ensembles de LLRs de $\mathcal{L}$. $\mathcal{L}_Z$ est ainsi une liste ordonnée d'ensembles de LLRs, représentant Z niveaux de fiabilités complémentaires vis-à-vis du post-traitement OSD. Le post-traitement OSD-o étant appliqué après chaque élement de $\mathcal{L}_Z$, une régle ML telle que définie dans (1) est utilisée pour choisir le mot de code final parmi les Z mots de code candidats. Le décodage est alors réussi si le mot de code final est égal au mot de code transmis. Nous notons cette méthode de décodage par BP-$\mathcal{L}_Z$-OSD-o.

4 Résultats numériques

4.1 Paramètres de simulations

Deux codes LDPC ont été considérés dans nos simulations. Les paramètres de ces codes sont donnés dans Table 1, où $R_c := K/N$ dénote le rendement de codage, d_v le degré des noeuds de données, et d_c le degré des noeuds de parité.

TABLE 1 : Paramètres des codes LDPC

	$\mathbf{N}$	$\mathbf{K}$	$\mathbf{R_c}$	$\mathbf{d_v}$	$\mathbf{d_c}$
Code **CCSDS** [10]	128	64	0.5	3-5	8
Code **Tanner** [11]	155	64	0.41	3	5

Le nombre d'itérations I du décodage du BP a été fixé à 25. Pour limiter le nombre de combinaisons lors du post-traitement OSD-o, nous évaluons ce dernier pour $o = 0, 1, 2$. Concernant le neurone présenté en section 3.1, un premier jeu $\mathcal{T}_{\text{BP-OSD}}$ comprenant 10000 mots bruités a été généré pour construire le jeu d'entraînement. Les poids, initialisés à 1, ont ensuite été optimisés sur 50 éqoques. De plus, en mesurant le taux d'erreur par bloc (*Block Error Rate* ou BLER) de l'OSD-o avec $\hat{\mathbf{L}}^{(\text{NS})}$ selon l'hyperparamètre γ, nous avons déterminé empiriquement $\gamma = 10$ comme étant un bon choix. Enfin, le neurone est entraîné pour chaque valeur de SNR allant de $2.5\,\text{dB}$ à $4.5\,\text{dB}$ (resp. $1.5\,\text{dB}$ à $3.5\,\text{dB}$), avec un pas de $0.5\,\text{dB}$ pour le code CCSDS (resp. le code de Tanner). Pour chaque valeur de SNR, un jeu de test $\mathcal{T}_{\text{BP-OSD}}$ comprenant 10000 mots bruités est aussi généré, et une liste $\mathcal{L}$ est ensuite construite selon la méthode décrite en section 3.2.

4.2 Evaluation des performances BLER

Nous comparons ici les performances des différentes stratégies proposées dans ce papier pour les 2 codes, à travers l'évaluation du BLER. Les gains sont évalués pour un BLER de 10^{-4}.

Sur la figure 1, la performance d'un post-traitement OSD-o avec $\hat{\mathbf{L}}^{(\text{NS})}$ est comparée avec celle d'un post-traitement OSD-o utilisant $\hat{\mathbf{L}}^{(\text{S})}$ pour le code CCSDS. Nous observons avec $\hat{\mathbf{L}}^{(\text{NS})}$ une performance légérement meilleure que $\hat{\mathbf{L}}^{(\text{S})}$ pour $o = 1$ (similaire pour $o = 0$). Pour un ordre o fixé à 2, $\hat{\mathbf{L}}^{(\text{NS})}$ permet d'obtenir un gain de $0.16\,\text{dB}$ par rapport à $\hat{\mathbf{L}}^{(\text{S})}$. Donc, plus l'ordre de l'OSD augmente, mieux $\hat{\mathbf{L}}^{(\text{NS})}$ est adapté au le post-traitement OSD.

Dans la suite, nous considérons un budget maximum de 3 post-traitements OSD-o. Pour commencer, l'ensemble $\mathcal{L}_3$ est déterminé à partir de $\mathcal{L}$ comme décrit en section 3.2. Les performances du post-traitement OSD-o avec $\mathcal{L}_3$ et $\hat{\mathbf{L}}^{(\text{NS})}$ seul sont affichées sur la figure 2. Nous constatons que l'application d'un OSD-o après chaque ensemble de LLRs de $\mathcal{L}_3$ apporte un gain croissant avec o par rapport au $\hat{\mathbf{L}}^{(\text{NS})}$: le gain est respectivement de $0.12\,\text{dB}$, $0.22\,\text{dB}$, et $0.27\,\text{dB}$ pour $o = 0, 1, 2$. Enfin, nous remarquons que BP-$\mathcal{L}_3$-OSD-2 obtient une performance située à seulement $0.17\,\text{dB}$ de la performance ML .

Enfin, le post-traitement OSD-o est effectué après une diversité de 3 BP-RNNs, notée $\mathcal{D}_3$. La méthode de construction de cette diversité de décodeurs est détaillée dans [8]. Pour chaque BP-RNN de $\mathcal{D}_3$, un neurone seul est optimisé avec les paramètres décrits en section 4.1. Trois fiabilités $\hat{\mathbf{L}}^{(\text{NS})}$ sont ainsi calculées puis évaluées avec un post-traitement OSD-o. Une régle ML détermine le mot de code final. Nous notons cette approche de diversité $\mathcal{D}_3$-$\hat{\mathbf{L}}^{(\text{NS})}$-OSD-$o$. Les résultats correspondant sont illustrés sur la figure 2. Nous obversons que BP-$\mathcal{L}_3$-OSD-0 apporte une légère amélioration par rapport à $\mathcal{D}_3$-$\hat{\mathbf{L}}^{(\text{NS})}$-OSD-0. Le gain s'accentue à $0.1\,\text{dB}$ pour $o = 1$ puis

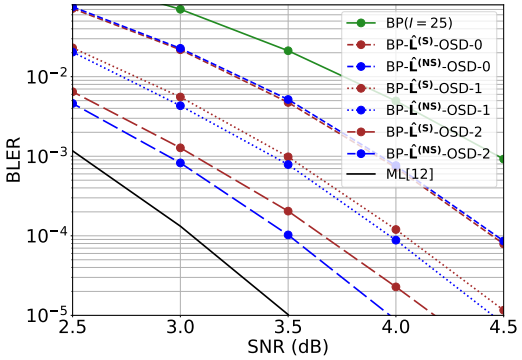

FIGURE 1 : BLER pour le Code CCSDS, $\hat{L}^{(\mathrm{S})}$ vs $\hat{L}^{(\mathrm{NS})}$.

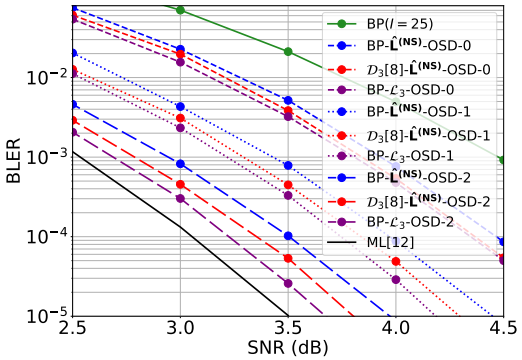

FIGURE 2 : BLER pour le Code CCSDS.

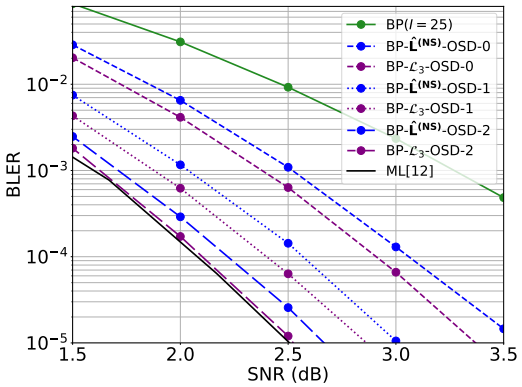

FIGURE 3 : BLER pour le Code de Tanner.

à $0.12\,\mathrm{dB}$ pour $o = 2$. Par conséquent, utiliser le décodeur BP seul et construire une liste d'ensembles de LLRs complémentaires est une meilleure stratégie en terme de performance avec le post-traitement OSD-o.

Les résultats de simulations obtenus avec le code de Tanner sont présentés sur la figure 3, avec la même méthodologie. Le décodage par BP-$\mathcal{L}_3$-OSD-0 présente un gain de $0.16\,\mathrm{dB}$ par rapport à BP-$\hat{\mathbf{L}}^{(\mathrm{NS})}$-OSD-0. Ce gain reste similaire lorsque l'ordre de l'OSD vaut 1 ou 2. Enfin, nous observons que BP-$\mathcal{L}_3$-OSD-2 atteint quasiment la performance de décodage ML.

5 Conclusion

Dans cet article, nous nous sommes intéressés à améliorer la performance du post-traitement OSD des sorties du BP pour des codes LDPC courts. Nous avons ainsi dans un premier temps optimisé un LLR accumulé pour le post-traitement OSD grâce à une méthode d'entraînement utilisant la fonction de coût focale. Ensuite, nous avons construit une liste ordonnée d'ensembles de LLRs complémentaires vis-à-vis de l'OSD, comprenant les LLRs accumulés et optimisés. Cette liste nous a ainsi permis de proposer un nouvelle stratégie de décodage, où un post-traitement OSD est appliqué après chaque ensemble de LLRs appartenant à la liste.

Cette méthodologie peut être répliquée pour des codes LDPC plus longs, étant donné que la modélisation de la somme pondérée en neurone ne dépend pas de la taille du code. Ce travail ouvre aussi de nouvelles perspectives sur l'utilisation de la fonction de coût focale pour le décodage avec des réseaux de neurones.

Références

[1] Y. Polyanskiy, H. V. Poor, and S. Verdú. Channel coding rate in the finite blocklength regime. *IEEE Trans. on Inf. Theory*, 56(5).

[2] M. C. Coşkun, G. Durisi, T. Jerkovits, G. Liva, W. Ryan, B. Stein, and F. Steiner. Efficient error-correcting codes in the short blocklength regime. *Physical Com.*, 34.

[3] R. G. Gallager. Low density parity check codes. MIT Press, Cambridge, 1963. Research Monograph series.

[4] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. on Inf. Theory*, 47(2).

[5] M.P.C. Fossorier and Shu L. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Trans. on Inf. Theory*, 41(5), 1995.

[6] M.P.C. Fossorier. Iterative reliability-based decoding of low-density parity check codes. *IEEE Journal on Selected Areas in Communications*, 19(5).

[7] M. Jiang, C. Zhao, E. Xu, and L. Zhang. Reliability-based iterative decoding of LDPC codes using likelihood accumulation. *IEEE Communications Letters*, 11(8).

[8] J. Rosseel, V. Mannoni, I. Fijalkow, and V. Savin. Decoding short LDPC codes via BP-RNN diversity and reliability-based post-processing. *IEEE Trans. on Communications*, 70(12).

[9] TY. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proc. of the IEEE Int. Conference on Computer Vision*, 2017.

[10] Short block length LDPC codes for TC synchronization and channel codding (CCSDS 231.1-O-1). Consultative Committee for Space Data Systems (CCSDS), Technical Report, April 2015.

[11] R. Tanner, D. Sridhara, and T. Fuja. A class of group-structured LDPC codes. In *Proc. ISTA*.

[12] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn. Database of channel codes and ML simulation results. `www.uni-kl.de/channel-codes`, 2019.