Décodeurs BP-RNNs mis en parallèle et spécialisés dans le décodage de codes LDPC courts

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

Joachim.Rosseel@cea.fr, Valerian.Mannoni@cea.fr, Valentin.Savin@cea.fr, Inbar.Fijalkow@ensea.fr

Résumé – Cet article traite du décodage des codes LDPC (*Low Density Parity Check*) courts par l'algorithme BP (*Belief Propagation*). Ce dernier pouvant être modélisé à l'aide d'un réseau de neurones récurrent (BP-RNN), nous introduisons une nouvelle méthode d'entraînement qui a pour objectif de spécialiser le décodeur BP-RNN sur des événements d'erreur partageant des propriétés structurelles similaires. Cette approche est ensuite associée à une nouvelle architecture de décodage composée de plusieurs BP-RNNs spécialisés mis en parallèle, où chaque BP-RNN est entraîné à corriger un type différent d'événement d'erreur. Nos résultats de simulations montrent que les BP-RNNs spécialisés mis en parallèles améliorent efficacement la capacité de décodage des codes LDPC de taille courte.

Abstract – This article deals with the decoding of short block length Low Density Parity Check (LDPC) codes. It has already been demonstrated that Belief Propagation (BP) can be adjusted to the short coding length, thanks to its modeling by a Recurrent Neural Network (BP-RNN). To strengthen this adaptation, we introduce a new training method for the BP-RNN. Its aim is to specialize the BP-RNN on error events sharing the same structural properties. This approach is then associated with a new decoder composed of several parallel specialized BP-RNN decoders, each trained on correcting a different type of error events. Our results show that the proposed specialized BP-RNNs working in parallel effectively enhance the decoding ability for short block length LDPC codes.

1 Introduction

L'émergence des systèmes de communication pour l'Internet des Objets, avec notamment la transmission de paquets courts, a redynamisé l'intérêt de la communauté pour la recherche de codes correcteurs d'erreurs efficaces et définis pour des messages allant de quelques dizaines à quelques centaines de bits. La conception de codes courts et d'algorithmes de décodage associés efficaces soulève encore de nombreux défis [2], bien que des progrès importants aient été réalisés au cours de ces dernières années, notamment dans la compréhension des limites du codage pour des longueurs de blocs courtes [3],

Les codes LDPC (*Low Density Parity Check*) [4] sont une classe de codes correcteurs d'erreurs définie par leur matrice de parité creuse et représentée par leur graphe de Tanner [5]. 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) [6]. En effet, dans le cas asymptotique, soit en l'absence de cycle court dans le graphe de Tanner, le BP permet d'estimer parfaitement le maximum a posteriori de chaque bit codé. Cependant, l'apparition de ces cycles pour les codes courts dégradent considérablement les performances du BP.

Pour réduire l'impact des cycles courts, une variante pondérée du décodage BP a été introduit dans [7], dans laquelle les poids sont optimisés à l'aide d'un réseau neuronal récurrent (*Reccurent Neural Network* ou RNN). Il a alors été montré que le décodeur résultant, appelé BP-RNN, est plus performant que le décodeur BP habituel pour les codes Bose-Chaudhuri-Hocquenghem (BCH) courts (appartenant à la classe des codes *High Density Parity Check* ou HDPC). Par la suite, plusieurs variantes du décodage BP à base de réseau de neurones ont été proposées dans la littérature [8–10].

Dans cet article, nous adressons la problématique du décodage de codes LDPC courts par BP-RNN. Notre approche consiste dans un premier temps en une classification des événements d'erreur, selon certaines propriétés structurelles. Ensuite, une architecture parallèle, comprenant plusieurs BP-RNNs fonctionnant en parallèle, est décrite. Chaque BP-RNN dans cet architecture parallèle est spécialisé pour traiter une classe d'erreur spécifique. Enfin, nous discutons de l'entraînement des décodeurs parallèles BP-RNNs.

L'article est organisé comme suit. La section 2 introduit les notations et rappelle l'algorithme de décodage BP-RNN. La section 3 définit la classification des événements d'erreur, l'architecture parallèle des décodeurs BP-RNNs, et l'entraînement des décodeurs BP-RNNs. Enfin, la section 4 présente les résul-

. Ce travail a été réalisé dans le cadre du projet ANR AI4CODE (ANR-21-CE25-0006). Une partie de ce travail a été publié dans [1].

tats de simulation, et la section 5 conclut cet article.

2 Décodage par BP neuronal

Nous considérons un code LDPC représenté par son graphe de Tanner, avec N noeuds de donnée et M noeuds de parité, notés respectivement par $n \in \{1,\ldots,N\}$ et $m \in \{1,\ldots,M\}$. Nous notons également par $\mathcal{N}(m)$ l'ensemble des noeuds de donnée connectés au noeud de parité m, et par $\mathcal{M}(n)$ l'ensemble des noeuds de parité connectés au noeud de donnée n.

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 du noeud de donnée incident [6]. L'algorithme de décodage BP-RNN est une des variantes pondérées du décodage BP, dans lequel les messages échangés sont multipliés par des poids appris par le RNN [7]. Le BP-RNN est divisé en trois couches neuronales: la couche *check-pass* suivi des couches *data-pass* et *a posteriori*, où chacune modélise une étape réalisée lors d'une itération de décodage BP.

Les formules ci-dessous détaillent le calcul des messages dans chaque couche. Nous appelons $\beta_{m\to n}$ et $\alpha_{n\to m}$ les messages calculés respectivement par les couches *check-pass* et *data-pass* (où (m,n) est une arête du graphe de Tanner). Les LLRs (*Log Likelihood Ratios*) observés sont notés par $L_{\mathrm{ch},n}$, et sont utilisés pour initialiser les $\alpha_{n\to m}$ avant la première itération de décodage, tandis que les $\tilde{L}_n$ correspondent aux LLRs a posteriori calculés par la couche *a posteriori*.

$$\beta_{m\to n} = 2\tanh^{-1}\left(\prod_{n'\in\mathcal{N}(m)\setminus n}\tanh\left(\frac{\alpha_{n'\to m}}{2}\right)\right) \quad (1)$$

$$\alpha_{n\to m} = L_{\mathrm{ch},n} + w_{n\to m}\sum_{m'\in\mathcal{M}(n)\setminus m}\beta_{m'\to n}, \quad (2)$$

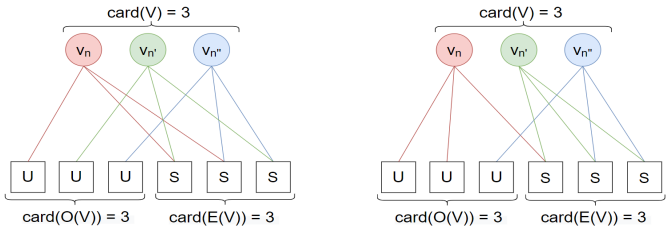
$$\tilde{L}_n = L_{\mathrm{ch},n} + \sum_{m\in\mathcal{M}(n)}\tilde{w}_{m\to n}\beta_{m\to n} \quad (3)$$

Dans (2), les poids sont indiqués par $w_{n\to m}$, où l'indice représente le neurone correspondant $n \to m$ dans la couche *data-pass*. Dans (3), les poids sont indiqués par $\tilde{w}_{m\to n}$, où l'indice représente à la fois le neurone n correspondant dans la couche *a posteriori*, et l'arête neuronale provenant du neurone $m \to n$ dans la couche *check-pass*. On observe que les poids sont appliqués uniquement sur les messages entrant dans les couches *data-pass* (2) et a posteriori (3).

Pour entraîner le BP-RNN, nous utilisons la fonction de coût suivante, en supposant sans perte de généralité que le mot de code tout zéro est transmis:

$$\mathrm{Loss}(\tilde{L}) = \frac{-1}{N}\sum_{n=0}^{N-1}\log(\sigma(\tilde{L}_n)) \quad (4)$$

où $\sigma(x) = (1+\exp(-x))^{-1}$ est la fonction sigmoïde convertissant les LLRs $\tilde{L}_n$ en probabilités. La fonction de coût est minimisée durant l'entraînement du réseau de neurones, réduisant ainsi le taux d'erreur binaire du décodeur entraîné.



(a) Graphe induit possible n°1 (b) Graphe induit possible n°2

FIG. 1: Exemple de deux ensembles V, avec $\mathbf{card}(O(V)) = \mathbf{card}(E(V)) = 3$. Si les noeuds de donnée dans V sont en erreur, les noeuds de parité denotés par U sont non satisfaits, tandis que ceux notés par S sont satisfaits.

3 Spécialisation des décodeurs BP-RNNs selon une classification des événements d'erreur

3.1 Classification des événements d'erreur par type d'absorption

On s'intéresse dans cette section à la classification des événements d'erreur, possédant un certain nombre d'erreurs et des propriétés structurelles similaires dans le sous-graphe de Tanner induit. Soit V un ensemble de noeuds de donnée de cardinal ν; et C l'ensemble des noeuds de parité connectés à au moins un noeud de donnée dans V. Nous appelons $O(V) \subset C$ l'ensemble des noeuds de parité ayant un nombre impair de connections à V, et $E(V) := C\backslash O(V)$ l'ensemble des noeuds de parité ayant un nombre un nombre pair de connections à V.

Nous définissons alors le type absorbant d'un ensemble de noeuds de donnée V comme ν-(ω,ε), où $\nu := \mathbf{card}(V)$, $\omega := \mathbf{card}(O(V))$, et $\varepsilon := \mathbf{card}(E(V))$. Nous pouvons remarquer que le type absorbant indique le nombre total de noeuds de parité (non satisfait, satisfait), dans le cas où les noeuds de donnée de V sont tous erronés.

Il est à noter que des ensembles V ayant un même type absorbant peuvent induire des sous-graphes de Tanner différents (plus précisément *non isomorphes*). Un tel exemple est illustré dans la Fig. 1, pour deux ensembles de noeuds de donnée de type absorbant 3-(3,3). En effet, pour l'ensemble V du cas (b), nous pouvons voir que le noeud de donnée n est connecté à deux noeuds de parité non satisfaits et un satisfait, tandis que sur le cas (a), ce même noeud n est connecté à un noeud de parité non satisfait et deux satisfaits.

Par conséquent, nous définissons la classe ν-(ω,ε,s) comme comprenant tous les événements d'erreur dont l'ensemble des noeuds de données V possède un type absorbant ν-(ω,ε) et dont les sous-graphes induits sont isomorphes (où s désigne l'indice d'un sous-graphe possible). Les ensembles de noeuds de donnée illustrés dans la Fig. 1 sont associés aux classes 3-(3,3,1) et 3-(3,3,2) respectivement.

3.2 Décodeurs BP-RNNs mis en parallèle

Pour améliorer la performance du décodage des codes LDPC courts, nous proposons de *spécialiser* (*i.e.*, entraîner) un décodeur BP-RNN pour chaque classe d'erreur ν-(ω,ε,s), selon la classification présentée dans la sous-section précédente. Le nombre de classes d'erreur différentes; et donc de décodeurs BP-RNNs à entraîner, dépend de la structure du code LDPC; et aussi de la valeur de ν. Pour un code LDPC court donné, nous déterminons ainsi toutes les classes d'erreur possibles, en considérant tous les sous-ensembles de noeuds de donnée V, de cardinal ν. De plus, plusieurs valeurs de ν sont à considérer selon la capacité de correction du code LDPC court, augmentant ainsi le nombre de décodeurs BP-RNNs à entraîner.

Une fois qu'un décodeur BN-RNN a été entraîné pour chaque classe d'erreur (la procédure d'entraînement sera détaillée dans la prochaine sous-section), nous considérons une architecture de décodage parallèle, où l'ensemble des décodeurs entraînés sont exécutés en parallèle. Si aucun des décodeurs parallèles ne produit de mot de code, ce qui est vérifié en calculant le syndrome, le décodage échoue. Sinon, parmi les mots de code décodés (syndrome nul), nous sélectionnons celui qui a été trouvé le plus souvent, dans le cas où différents décodeurs produisent différents mots de code. En cas d'égalité, le mot de code sélectionné est choisi aléatoirement. Le décodage est alors réussi si le mot de code choisi est égal à celui transmis.

3.3 Entraînement des BP-RNNs mis en parallèle

Nous proposons dans cette section une construction du jeu d'entraînement, utilisé pour entraîner le décodeur BP-RNN sur une classe d'erreur donnée. Nous supposons que les bits codés sont mappés à des symboles BPSK ± 1, et transmis à travers un canal à bruit additif blanc Gaussien. Ce modèle de bruit et le décodeur BP-RNN étant symétriques [7], nous pouvons supposer que le mot de code tout-zéro est transmis, correspondant à un signal entièrement modulé $+1$. Ainsi, avec ce modèle, les symboles reçus sont donnés par

$$y_n = 1 + z_n, \quad \forall n = 1, \ldots, N \tag{5}$$

où z_n note une variable aléatoire à valeur réelle suivant une loi normale de moyenne nulle, et de variance égale à σ^2.

Pour générer un événement d'erreur aléatoire dans une classe d'erreur donnée ν-(ω,ε,s), nous considérons un ensemble de noeuds de donnée V choisi au hasard parmi ceux de la classe d'erreur, puis nous générons les symboles reçus y_n avec

$$z_n \sim \begin{cases} \mathcal{N}(0,\sigma^2,-\infty,-1), & \text{si } n \in V \\ \mathcal{N}(0,\sigma^2,-1,\infty), & \text{sinon} \end{cases} \tag{6}$$

où $\mathcal{N}(0,\sigma^2,a,b)$ note la distribution normale tronquée, de moyenne nulle et de variance σ^2, prenant ses valeurs dans l'intervalle (a,b). Le jeu d'entraînement est obtenu en répétant la procédure ci-dessus plusieurs fois, pour chaque ensemble V dans la classe d'erreur traitée. De cette façon, le jeu d'entraînement est représentatif de la classe d'erreur, ce qui permet au décodeur BP-RNN de se spécialiser dans le décodage des événements d'erreur de la classe correspondante.

4 Résultats numériques

4.1 Paramètres de simulation

Deux codes LDPC avec des noeuds de donnée de degré régulier ont été considérés dans nos simulations. Les paramètres de ces codes sont fournis dans Tab 1, où N est la taille du code, K le nombre de bits d'informations, $R_c := K/N$ le rendement de codage, d_v le degré des noeuds de donnée, d_c le degré des noeuds de parité, et $n_{4\text{-}cycles}$ le nombre de cycle de taille 4. Les graphes de Tanner de ces deux codes ont été construits en utilisant l'algorithme à croissance d'arêtes progressive (*Progressive Edge Growth* ou PEG) [11].

TAB. 1: Paramètres des codes LDPC

	$\mathbf{N}$	$\mathbf{K}$	$\mathbf{R_c}$	$\mathbf{d_v}$	$\mathbf{d_c}$	$\mathbf{n_{4\text{-}cycles}}$
Code-1	64	46	0.71	3	10-11	47
Code-2	64	32	0.5	3	6	0

Le nombre d'itérations de décodage a été fixé à dix, pour l'ensemble des décodeurs. Au niveau de la classification des événements d'erreur, nous avons fixé $\nu = 2{,}3$ pour le Code-1 et $\nu = 2{,}3{,}4$ pour le Code-2. Ces choix de ν nous permettent de traiter des tailles d'événements d'erreur adaptées aux capacités de correction des nos codes LDPC. En appliquant notre procédure de classification présentée en Section 3.1, nous obtenons treize (resp. dix-sept) classes pour le Code-1 (resp. Code-2). Ainsi, un total de treize (resp. dix-sept) décodeurs BP-RNNs ont été utilisés en parallèle. Chaque BP-RNN a été entraîné de manière indépendante, avec la méthode de construction du jeu d'entraînement décrite dans la Section 3.3. De plus, nous avons entraîné un BP-RNN seul en suivant la procédure de [7], pour avoir une référence de comparaison.

Enfin, en pratique, il est souhaitable de réduire le nombre de décodeurs fonctionnant en parallèle. A cette fin, nous avons calculé la combinaison de P décodeurs parmi les treize (resp. dix-sept) BP-RNNs mis en parallèle pour le Code-1 (resp. Code-2) donnant la meilleure performance en terme de taux d'erreur paquet (*Frame Error Rate* ou FER). P est ainsi un paramètre permettant d'évaluer un compromis complexité/performances. Dans ce travail, P a été fixé à huit, puis de nouvelles simulations indépendantes ont été exécutées pour les deux codes, nous permettant ainsi d'évaluer le FER pour uniquement huit décodeurs en parallèle.

4.2 Évaluation des performances FER

Nous comparons ici la performance des différentes stratégies discutées dans la section précédente, en terme de FER. Les gains sont évalués à un FER de 10^{-4}.

Les résultats de simulations du Code-1 sont montrés dans la Fig. 2. Le BP-RNN entraîné comme dans [7] apporte un gain de $0.18\,\mathrm{dB}$ par rapport au BP standard, tandis qu'en utilisant notre architecture parallèle de BP-RNNs spécialisés, le gain est accentué à $0.42\,\mathrm{dB}$. De plus, il est important de remarquer que

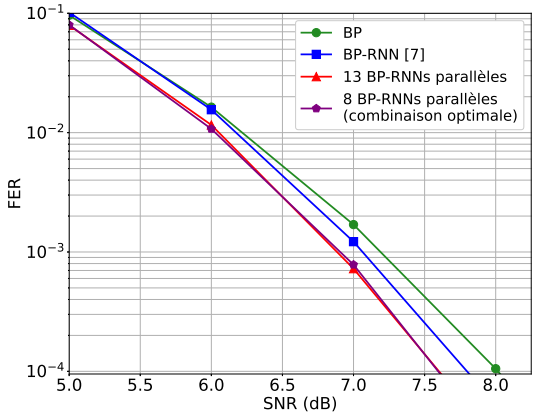


FIG. 2: FER pour le Code-1.

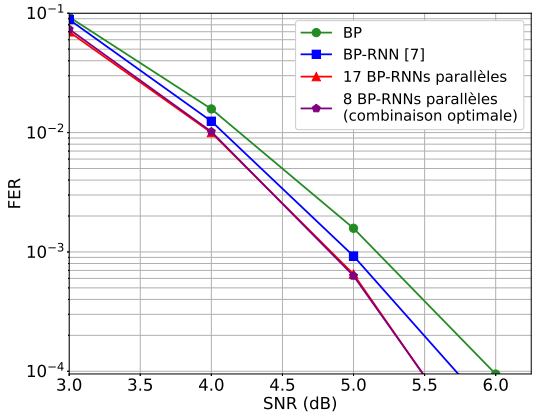


FIG. 3: FER pour le Code-2.

la combinaison de huit décodeurs permet d'obtenir les mêmes gains, tout en réduisant la complexité calculatoire.

Les résultats de simulations du Code-2 sont montrés sur la Fig. 3. Bien que le rendement de codage soit réduit, des gains similaires au code précédent sont observés. En effet, l'architecture proposée permet d'obtenir un gain de 0.5 dB par rapport au BP. Enfin, dans cet exemple, il nous suffit de moins de la moitié des décodeurs originaux (nos huit décodeurs sélectionnés) pour atteindre les mêmes performances que l'architecture parallèle originale. Ainsi, pour les deux codes, les combinaisons de huit décodeurs nous permettent de conserver les décodeurs se complétant les uns avec les autres, c'est à dire décodant des événements d'erreur différents.

5 Conclusion et Perspectives

Dans cet article, nous nous sommes intéressés à améliorer les performances du décodeur BP-RNN pour des codes LDPC courts. À cette fin, nous avons étudié et classé les événements d'erreur en fonction de leurs sous-graphes induits. Ensuite, nous avons proposé une nouvelle stratégie de décodage en parallèle composée de décodeurs BP-RNNs spécialisés, où chaque BP-RNN a été entraîné pour une classe d'erreur spécifique. Nous avons aussi montré qu'il était possible de garder uniquement un sous-ensemble de ces décodeurs tout en conservant des performances similaires à la structure complète.

Ce travail est un premier pas vers des décodeurs neuronaux spécialisés. D'autres stratégies de spécialisation pourraient approcher la performance de décodage de vraisemblance maximale à courte ou moyenne longueur de code, pour laquelle nous devrons probablement compter sur plusieurs décodeurs au lieu d'un unique.

Références

[1] J. Rosseel, V. Mannoni, V. Savin, and I. Fijalkow, "Error structure aware parallel BP-RNN decoders for short LDPC codes," *11th International Symposium on Topics in Coding (ISTC)*, pp. 1–5, 2021.

[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 Communication*, vol. 34, pp. 66–79, 2019.

[3] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[4] R. G. Gallager, "Low density parity check codes," MIT Press, Cambridge, 1963, research Monograph series.

[5] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[6] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[7] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[8] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross, "Neural belief propagation decoding of crc-polar concatenated codes," in *IEEE Int. Conference on Communications (ICC)*, 2019, pp. 1–6.

[9] X. Xiao, B. Vasić, R. Tandon, and S. Lin, "Designing finite alphabet iterative decoders of LDPC codes via recurrent quantized neural networks," *IEEE Trans. on Communications*, vol. 68, no. 7, pp. 3963–3974, 2020.

[10] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. G. i Amat, "Pruning neural belief propagation decoders," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 338–342.

[11] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Transactions on Information Theory*, vol. 52, no. 51, pp. 386–398, 2005.