Quasi-Cyclic Grandient Learning : une méthode d'optimisation des codes QC-LDPC de la 5G

Emmanuelle BODJI[1,2] Bruno JAHAN[2] Christophe JEGO[3] Emmanuel BOUTILLON[1]

[1]Université Bretagne Sud, Lab-STICC UMR 6285, 27 rue Armand Guillemot CS 7030, 56321 LORIENT Cedex, France

[2]Orange Innovation, 4 Rue du Clos Courtel, 35510 Cesson-Sévigné, France

[3]University of Bordeaux, Bordeaux INP, IMS Laboratory, UMR CNRS 5218, Avenue des facultés, 33405 Talence, France

Résumé – L'utilisation des réseaux neuronaux progresse dans plusieurs domaines, notamment les télécommunications. Cependant, en ce qui concerne les codes correcteurs d'erreurs, le consensus actuel est que les outils d'Intelligence Artificielle (IA) peuvent au mieux faire aussi bien que ceux qui existent déjà, sans vraiment apporter de simplification : l'enthousiasme des débuts s'est estompé. Dans cet article, nous montrons qu'il est possible d'utiliser l'IA comme un oracle pour identifier et améliorer progressivement les faiblesses d'une matrice construction d'un code. Nous avons appliqué cette approche à l'optimisation de la structure de base des codes LDPC standardisés pour la cinquième génération de téléphonie mobile appelée 5G. Pour les petites matrices, nous avons obtenu des gains de performance significatifs allant jusqu'à 1,3 dB en rapport signal-bruit pour un taux d'erreur de trame de 10^{-2}.

Abstract – The use of neural networks is advancing across several fields, notably telecommunications. However, regarding error-correcting codes, the current consensus is that Artificial Intelligence (AI) tools can at best perform on par with existing methods, without offering real simplifications; the initial enthusiasm has waned. In this article, we demonstrate that it is possible to employ AI as an oracle to progressively identify and improve the weaknesses in a code construction matrix. We applied this approach to optimizing the base structure of LDPC codes standardized for the fifth generation of mobile telephony, known as 5G. For small matrices, we achieved significant performance gains of up to 1.3 dB in signal-to-noise ratio at a frame error rate of 10^{-2}.

1 Introduction

Les codes Low-Density Parity Check (LDPC) de la 5G offrent une grande flexibilité remarquable au niveau du rendement et de la taille des codes, grâce à une construction hiérarchique où la matrice de parité finale **H** est obtenue par extension d'un facteur Z_c sur une matrice de base. Ce type de matrices s'appelle Quasi-Cyclic LDPC (QC-LDPC) [1]. Si les codes de la 5G ont des performances excellentes pour les messages de grande taille (supérieurs à 1000 bits), ils sont, en revanche, beaucoup moins performants pour les petites tailles. En effet, dans le standard 5G, les matrices de parité dédiées aux messages de petite taille possèdent un nombre élevé de cycles courts. Ces derniers génèrent des structures topologiques particulières (trapping set, absorbing set [9]) qui affectent fortement la convergence des algorithmes itératifs de décodage appelés "Belief Propagation" (BP) [5]. Dans cet article, nous désignerons ce type de structure sous le terme générique de Structures Topologiques Pénalisantes (STP).

Il existe une abondante littérature sur l'optimisation des codes LDPC pour des blocs de petite taille. Cette littérature contient deux volets complémentaires : la construction de matrices de parité et le décodage des codes. Dans cet article, nous nous intéressons uniquement à la première question. Au niveau du décodage, nous n'utilisons que l'algorithme Normalized Min-Sum [8]. La littérature sur la construction de codes courts se divise en deux branches : les méthodes classiques basées sur la connaissance fine des STP [8] et les méthodes agnostiques. Dans ce dernier champ, nous pouvons par exemple citer [2], qui utilise un algorithme d'optimisation génétique. Nous nous intéressons principalement à des méthodes basées sur l'apprentissage automatique à base de réseaux de neurones. Ces méthodes reposent sur le fait que l'algorithme de décodage peut être vu comme un graphe de calcul dérivable, sur lequel il est possible d'appliquer la rétropropagation de gradient pour apprendre de bons codes. Dans ce domaine encore peu exploré, nous pouvons citer en particulier [4]. Bien que ces recherches aient produit quelques résultats, les matrices de parité **H** obtenues n'ont pas de structure particulière, et encore moins la structure Quasi-Cyclique utilisée dans le standard 5G.

Cet article présente une nouvelle méthode visant à construire de façon agnostique de bons codes QC-LDPC de petite taille. La méthode proposée est dérivée de [4]. Elle est basée sur l'amélioration itérative de la matrice de parité H en utilisant à chaque itération l'information apportée par le gradient rétropropagé sur le graphe de calcul du décodeur itératif. D'un point de vue pratique, ce dernier est assimilé à un réseau de neurones récurrents (RNN). Une agrégation par bloc permet d'optimiser la matrice en conservant sa structure Quasi-Cyclique. En appliquant la méthode proposée, nous avons pu construire de nouvelles familles de matrices ayant les mêmes contraintes que celles de la 5G (pour la rétrocompatibilité), mais ayant des performances significativement supérieures. De plus, effet bénéfique secondaire, les matrices obtenues sont moins denses, et par conséquent, moins complexes à décoder.

Le reste de l'article est structuré en quatre sections : la Section 2 introduit les codes LDPC de la norme 5G. Les travaux antérieurs sur l'application des RNN aux codes LDPC sont abordés dans la Section 3. La Section 4 propose une méthode d'amélioration des codes LDPC pour la 5G. Enfin, la Section 5 présente les résultats expérimentaux concernant les performances de décodage des codes que nous avons construits.

2 Code LDPC et code QC-LDPC de la 5G

L'algorithme de Belief Propagation (ou algorithme de somme-produit) repose sur l'échange de messages entre les nœuds de variable (bits) et les nœuds de parité dans un graphe de Tanner, qui peut être représenté par une matrice de parité $\mathbf{H}$. On considère les notations suivantes : λ_i : le rapport logarithmique de vraisemblance (LLR) provenant du canal pour le bit b_i et $h(i,j) \in \{0,1\}$ est l'élément en position i, j de la matrice $\mathbf{H}$. Une valeur $h(i,j) = 1$ indique une connexion entre le nœud de variable j et le nœud de parité i, $L_{i\to j}$ le message du nœud de parité i au nœud de variable j et enfin, $L_{j\to i}$ le message du nœud de la variable j au nœud de parité i. À l'initialisation, les messages $L_{i\to j}$ sont égaux à λ_i. Puis, chaque itération de décodage est constituée de la mise à jour des nœuds de parité (1), suivie de la mise à jour des nœuds de variable (2), comme indiqué ci-dessous :

$$L_{j\to i} = 2h(i,j)\tanh^{-1}\left(\prod_{i',i'\neq i} h(i',j)\tanh\left(\frac{L_{i'\to j}}{2}\right)\right), \quad (1)$$

$$L_{i\to j} = h(i,j)\left(\lambda_i + \sum_{j',j'\neq j} h(i,j') * L_{j'\to i}\right). \quad (2)$$

Une fois toutes les itérations réalisées, la probabilité a posteriori L_i du bit i est calculée comme suit :

$$L_j = \lambda_j + \sum_i h(i,j)L_{i\to j}. \quad (3)$$

La décision binaire estimée est alors 0 si $L_i > 0$, 1 sinon. A partir de L_j, on détermine la probabilité p_j que le j^{ime} nœud de variable soit égal à 1 comme $p_j = \frac{1}{1+e^{L_j}}$. Cette formulation du BP inclut inutilement tous les coefficients nuls de $\mathbf{H}$, ce qui n'est pas optimal, mais elle reste nécessaire pour la suite de l'article.

La norme 5G contient la définition de deux matrices binaires protographes ou Base Graph (**BG**), désignées par **BG1** et **BG2**. La première, **BG1**, est appliquée aux grands blocs, tandis que la seconde est conçue pour les petits blocs et les faibles rendements de codage. Dans cet article, nous appliquons la méthode d'optimisation sur **BG2**. Cette matrice de base est subdivisée en sous-matrices dont la partie d'information contient kb colonnes, qui correspondent, après expansion d'un facteur Z_c, au $kb \times Z_c$ bits d'information du code (les valeurs possibles de kb sont 6, 8, 9 et 10). Le standard 5G est structuré en sous-ensembles de facteur d'expansion, chaque sous-ensemble est associé à une matrice de base $\mathbf{BM}$. Pour un sous-ensemble de facteurs d'expansion, la matrice $\mathbf{BG2}$ est associée à une matrice de base $\mathbf{BM}$ prenant ses valeurs dans $\{-\infty, 0, 1, \ldots, Z_{max} - 1\}$. Si $BG(i,j) = 0$, $BM(i,j) = -\infty$ alors l'expansion se à partir de la matrice nulle $Z_c \times Z_c$. Si $BG(i,j) = 1$, alors l'expansion se réalise à partir de la matrice identité $Z_c \times Z_c$ ayant subi une rotation circulaire de ses colonnes d'un facteur $BM(i,j) \mod Z_c$. Ainsi, une matrice $\mathbf{BG}$ de taille $m \times n$ génère une matrice $\mathbf{H}$ de taille $(M,N) = (mZ_c, nZ_c)$. La figure 1 illustre la construction d'une matrice $\mathbf{H}$ de taille 3×6 avec $Z_c = 3$.

$$\underset{\mathbf{BG}}{\begin{bmatrix}1 & 0\end{bmatrix}} \xrightarrow{Z_c=3} \underset{\mathbf{BM}}{\begin{bmatrix}2 & -1\end{bmatrix}} \xrightarrow{Z_c=3} \underset{\mathbf{H}}{\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}}$$

FIGURE 1 : Construction de la matrice $\mathbf{H}$.

3 Construction de matrice par auto-encodeur

Comme indiqué dans la section précédente, il est possible de représenter un décodeur LDPC comme un graphe entièrement connecté avec des fonctions dérivables comme indiqué dans la figure 2. Dans ce graphe, toutes les connexions sont associées à un facteur multiplicatif égal à $h(i,j)$ que l'on peut considérer comme le poids d'une synapse d'un réseau de neurones. La qualité d'un décodage peut être caractérisée par la Binary Cross Entropy (BCE) B définie par $B = -\frac{1}{N}\sum_{j=1}^{N}(b_j \log(p_j) + (1-b_j)\log(1-p_j))$ avec b_j le bit émis et p_j la probabilité qu'il soit égal à 1.

La Figure 2 représente de façon simplifiée le processus d'optimisation d'une matrice LDPC proposée dans [7]. Cette méthode est appelée Gradient Quantization Learning Belief Propagation (GQLA-BP). Elle consiste à améliorer itérativement la matrice de parité $\mathbf{H}$ par une succession de matrices $\mathbf{H}_t$, $t = 0, 1, \ldots, t_{max}$. Chaque étape est constituée de deux phases : une phase d'apprentissage (ou recherche des STP de $\mathbf{H}_t$) par l'algorithme du gradient, puis une phase de modification de la matrice pour tenter d'obtenir une matrice $\mathbf{H}_{t+1}$ améliorée.

La phase d'apprentissage est réalisée en utilisant le mot tout à zéro et en injectant N_e erreurs aléatoirement parmi les N bits du mot de code tout à zéro. La valeur de N_e est un paramètre important. En effet, si N_e est trop faible, le message est directement décodé et il n'y a pas d'erreurs de décodage sur lesquelles "apprendre". Réciproquement, si N_e est trop élevé, le décodeur ne converge pas et ajoute plutôt des erreurs : le gradient ne peut alors exhiber aucune tendance. La BCE devient : $B = -\frac{1}{N}\sum_{j=1}^{N}\log(1-p_j)$.

Le calcul du gradient permettant de diminuer la BCE est rétropropagé couche par couche par l'algorithme du gradient pour évaluer comment modifier les poids des synapses (les valeurs $h(i,j)$) pour améliorer la qualité du décodage. En accumulant les valeurs des gradients associées à une même branche pour des batchs de 32 mots bruités, et en ré-itérant le processus jusqu'à ce qu'au moins un élément soit supérieur ou égal à la valeur seuil s (définie initialement). Avec un algorithme classique de gradient, la connexion $H_t(i,j)$ serait mise à jour car $H_{t+1}(i,j) = H_t(i,j) - G_t(i,j)$ pour minimiser la fonction de coût. Les coefficients appartenant à l'ensemble $\{0,1\}$, une telle transformation est impossible. On peut toutefois noter que $H_t(i,j) = 1$ et $G_t(i,j)$ positif et de grande magnitude incitent à changer le coefficient $H_{t+1}(i,j)$ de un ezéroro. Symétriquement, $H_t(i,j) = 0$ et $G_t(i,j)$ négatif et de grande magnitude incitent à changer le coefficient $H_{t+1}(i,j)$ de 0 en 1. Le principe et les détails de cet algorithme appelé Gradient Quantization LeArning (GQLA) sont disponibles dans [7]. Cette méthode s'avère efficace pour les matrices de petites tailles. Toutefois, elle ne prend pas en compte les contraintes

inhérentes à la structure des codes LDPC de la 5G. Pour résoudre ce problème, nous proposons d'intégrer la structure QC du code dans le processus d'optimisation de la matrice.

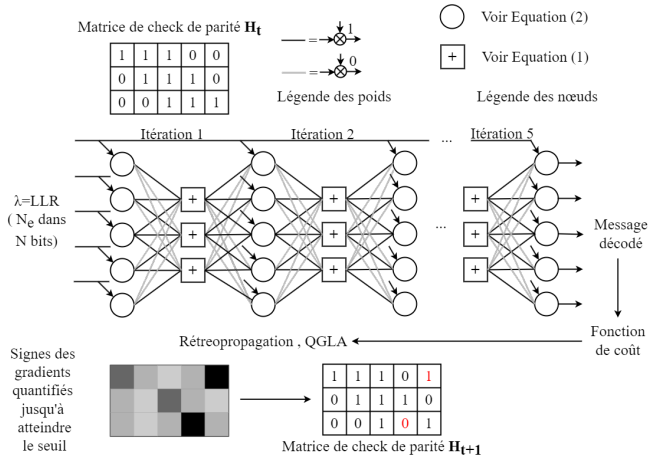

FIGURE 2 : Représentation d'une itération de l'algorithme GQLA-BP

4 Quasi-Cyclic Gradient learning

Dans cette section, nous présentons l'algorithme Quasi-Cyclic Gradient Learning (QCGL). Il s'agit d'une évolution de GQLA qui préserve par construction la nature quasi-cyclique de la matrice de parité.

En partant d'une matrice QC-LDPC $\mathbf{H}$, la mise à jour de $\mathbf{H}_t$ à l'itération t ne préserve pas nécessairement la structure QC du code. Pour la garder, il est nécessaire d'effectuer une mise à jour directement sur le couple $(\mathbf{BG_t}, \mathbf{BM_t})$ en autorisant uniquement les trois types de modifications suivantes entre $(\mathbf{BG}_t(i,j), \mathbf{BM}_t(i,j))$ et $(\mathbf{BG}_{t+1}(i,j), \mathbf{BM}_{t+1}(i,j))$:

$$\begin{cases} (0,-\infty) \rightarrow (1,d) \text{ (Création : ligne 4 Algo. 1).} \\ (1,a) \rightarrow (1,d) \text{ (Modification : ligne 6 Algo.1).} \\ (1,a) \rightarrow (0,-\infty) \text{ (Suppression : ligne 9 Algo. 1)..} \end{cases} \quad (4)$$

La matrice du gradient est estimée sur un batch de taille 32. L'opération est répétée 1000 fois pour agréger suffisamment de résultats pour faire apparaître les STP. La structure quasi-cyclique de la matrice finale $\mathbf{H}$ implique qu'une STP est dupliquée Z_c fois le long des diagonales de 1 des blocs qui la constituent. Une fois la matrice $\mathbf{G}$ obtenue, il est donc naturel de la subdiviser en sous-matrices de taille $Z_c \times Z_c$ et, pour chacune de ses sous-matrices, d'accumuler le résultat du gradient le long des Z_c diagonales. Le résultat est un tenseur $\mathbf{G}_D$ (Somme Diagonale des Gradients), dont l'élément $G_D(i,j,k)$ est égal à la somme des gradients de la k^{ime} diagonale du bloc (i,j) de $\mathbf{G}$, soit

$$G_D(i,j,k) = \sum_{l=0}^{Z_c-1} G(iZ_c+l, jZ_c+((l+k) \bmod Z_c)). \quad (5)$$

En travaillant tout d'abord sur les QC-LDPC de petites tailles dans lesquels nous avions introduit exprès des STP et en observant l'impact de ces STP sur la matrice $\mathbf{G}_D$ générée, nous avons déduit une façon efficace d'utiliser les informations données par $\mathbf{G}_D$. L'algorithme de transformation de $(\mathbf{BG}_t, \mathbf{BM}_t)$ en $(\mathbf{BG}_{t+1}, \mathbf{BM}_{t+1})$ à partir de $\mathbf{G}_D$ est présenté dans Algo. 1. La première étape consiste à identifier la position (i_0, j_0, k_0) de $\mathbf{G}_D$ correspondant à la plus forte incitation à un changement d'état d'une diagonale : passage de 0 à 1 si le gradient est négatif, ou passage de 1 à 0 si le gradient est positif (ligne 2, Algo. 1). Une fois le bloc (i_0, j_0) sélectionné, la modification minimisant la fonction de coût est effectuée. La valeur de S est un seuil d'extinction du nœud qui indique que le gradient n'a pas de forte indication pour choisir une autre diagonale à la place de celle qu'il faut supprimer. L'algorithme se poursuit tant que la fonction de coût diminue. Les performances finales sont alors déterminées par une simulation de Monte-Carlo classique. L'ensemble de cette procédure définit l'algorithme QCGL.

Algorithm 1 Algorithme de mise à jour à l'itération t

1: **Input :** $\mathbf{G}_D$, $\mathbf{BG}_t$, $\mathbf{BM}_t$,
2: $(i_0, j_0, k_0) = \arg\max_{i,j,k}\{|G_D(i,j,k)|\}$;
3: **if** $BG_t(i_0, j_0) = 0$ **then**
4: $\quad BG_{t+1}(i_0, j_0) = 1, BM_{t+1}(i_0, j_0) = k_0$;
5: **else**
6: $\quad BM_{t+1}(i_0, j_0) = \arg\min_{l}\{|G_D(i_0, j_0, k)|\}$;
7:
8: $\quad$ **if** $\exists\, t' < t,\ BM_{t+1}(i_0, j_0) = BM_{t'}(i_0, j_0)$ **then**
9: $\quad\quad BM_{t+1}(i_0, j_0) = 0$; $BM_{t+1}(i_0, j_0) = -\infty$;
10: $\quad$ **end if**
11: **end if**
12: **return** $\mathbf{BG}_{t+1}$, $\mathbf{BM}_{t+1}$, FER_{new}.

L'exécution du décodeur neuronal sur un GPU, bien qu'efficace, requiert d'importantes ressources en mémoire. En pratique, il s'avère que l'approche proposée est limitée à une valeur de Z_c relativement faible (2 ou 4) avec des batchs de taille 32. Pour une valeur de $Z_c = 8$, la taille maximale du batch est réduite à 3 pour un GPU ayant 16 Go de capacité mémoire.

La 5G proposant des facteurs d'expansion de taille maximale $Z_c = 384$, il est impossible d'utiliser notre méthode pour ces tailles. Toutefois, [6] montre comment calculer la matrice de base d'un Z_c à partir de la matrice de base d'un de ses diviseurs. Il existe donc une solution efficace à notre problème d'optimisation de matrice QC ayant de grands facteurs d'expansion.

Pour évaluer l'efficacité de la méthode QCGL et l'utilité des informations du gradient, nous avons comparé son optimisation à une approche aléatoire. L'étape t est alors constituée : a) du choix aléatoire de N_m positions (i_t, j_t) à modifier (N_m est le nombre de modifications à effectuer à chaque étape t), b) du remplacement des valeurs $BM(i,j)$ par les nouvelles valeurs prises aléatoirement dans l'ensemble $\{-\infty, 0, 1, \ldots, Z_c - 1\}$. Une simulation de Monte-Carlo est alors effectuée pour évaluer le nouveau taux d'erreurs par trame. Ainsi, si la modification améliore les performances, elle est conservée, sinon elle est rejetée. Cette procédure est répétée sur une durée équivalente à celle de l'optimisation QCGL.

5 Résultats expérimentaux

Dans cette section, l'évaluation des performances de matrices a été effectuée à l'aide du logiciel open source AFF3CT [3]. Les messages sont générés aléatoirement, encodés et modulés suivant un mapping BPSK. La simulation se déroule pour un

canal Bi-AWGN et le décodeur est basé sur l'algorithme Layered Normalized Min-Sum avec un facteur de normalisation de $\alpha = 0,7$. Le nombre cible de trames erronées est de 100, avec un maximum de 10^7 trames. La performance est évaluée, pour chaque rapport signal à bruit (Signal to Noise Ratio, SNR) en taux de trames erronées (FER pour Frame Error Rate).

Les performances des matrices générées par les deux méthodes d'optimisation, à savoir QCGL et l'approche aléatoire, sont comparées entre elles, ainsi qu'à des matrices du standard 5G. Ces matrices sont respectivement désignées par **QCGL**, **Random** et **Standard** dans la Figure 3. L'expérimentation a été réalisée pour $Z_c = 2$, $kb = 6$, un nombre d'itérations $I = 5$, $N_m = 10$ et une durée $T = 3$ heures (temps équivalent à celui consacré à l'optimisation de la matrice à l'aide du QCGL). La Figure 3 présente les résultats pour les trois types de matrices. Les matrices **Random** et **QCGL** présentent de meilleures performances que celles du standard 5G. En revanche, les performances de **Random** sont moins bonnes que celles de **QCGL**, avec une différence de l'ordre de 0,6 dB pour un 10^{-2} de FER. Ce résultat souligne l'apport significatif de la méthode QCGL dans l'optimisation des matrices LDPC pour la norme 5G. Pour évaluer de manière plus exhaustive le gain de l'approche proposée dans cet article, les performances de la matrice **QCGL** sont comparées à celles de **Standard** pour $Z_c = 2, 4, 8$, $kb = 6, 10$ et $I = 5$. Les résultats de cette seconde expérimentation sont présentés dans la figure 4. Nous pouvons relever un gain important de 1,3 dB à 10^{-2} pour $Z_c = 2$ et $kb = 6$. De plus, le gain pour la matrice **QCGL** lorsque $kb = 10$ est significatif pour les différentes valeurs de Z_c. Sur le plan de la complexité calculatoire lors du décodage, le nombre de connexions dans la matrice **QCGL** diminue de 198 à 188, soit une réduction d'environ 5%. Cette diminution contribue non seulement à une réduction du temps de décodage, mais également à une baisse de la consommation énergétique.

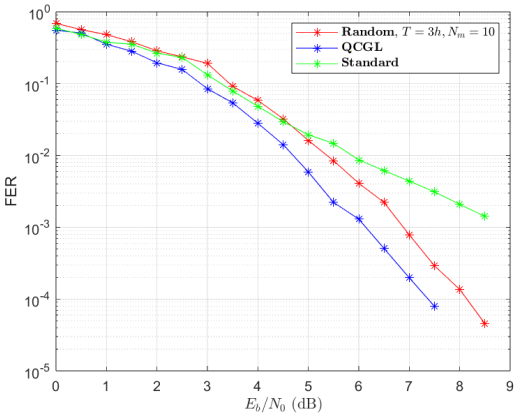

FIGURE 3 : Comparaison des performances des matrices améliorées par la méthode QCGL et une approche aléatoire pour **BG2** avec $Z_c = 2$, $kb = 6$ et $I = 5$.

6 Conclusion

Une première méthode d'optimisation par auto-encodeur de matrices quasi-cycliques à l'aide d'apprentissage par gradient est détaillée dans cet article. La méthode appelée QCGL permet, d'une part, d'améliorer les performances (jusqu'à 1,3 dB de gain à un FER de 10^{-2} pour les petites tailles de matrices)

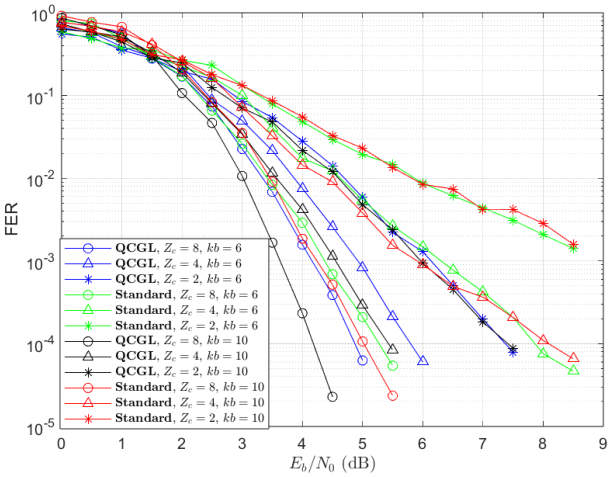

FIGURE 4 : Comparaison des performances des matrices améliorées par la méthode QCGL vis à vis des matrices du standard 5G pour **BG2** avec $Z_c = \{2, 4, 8\}$, $kb = \{6, 10\}$ et $I = 5$.

des matrices de la 5G, et ce, en conservant leur structure pour garantir la rétrocompatibilité. D'autre part, un effet bénéfique secondaire est à souligner. Ainsi, la matrice optimisée par l'approche QCGL possède 5 % de connexions en moins que la matrice 5G de départ, entraînant une diminution proportionnelle de la complexité calculatoire (et donc énergétique) du décodeur. Dans les futures travaux, nous envisageons de lever le verrou de la taille limitée des matrices que l'on peut optimiser. Pour ce faire, nous envisageons deux méthodes complémentaires. La première concerne l'optimisation de la mémoire GPU, la seconde, plus prometteuse, propose de remplacer l'optimisation directe de la matrice par une optimisation séquentielle, partie par partie.

Références

[1] J.H. BAE et A. Ahmed et AL : An overview of channel coding for 5G NR cellular communications. *APSIPA Transactions on Signal and Information Processing*, 8(1), 2019.

[2] Ahmed ELKELESH et Ebada et AL : Decoder-in-the-loop : Genetic optimization-based ldpc code design. *IEEE Access*, 7:141161–141170, 2019.

[3] Adrien CASSAGNE et AL : Aff3ct : A fast forward error correction toolbox ! Elsevier SoftwareX (Volume : 10), October 2019.

[4] L. GUILLAUME et D. Louis-Adrien et AL : Neural belief propagation auto-encoder for linear block code design. *IEEE Transactions on Communications*, 70(11):7250–7264, 2022.

[5] F.R. KSCHISCHANG et B.J. Frey et AL. : Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[6] Huaan LI, Hengzhou XU, Chao CHEN et Baoming BAI : Efficient construction of quasi-cyclic ldpc codes with multiple lifting sizes. *IEEE Communications Letters*, 28(4):754–758, 2024.

[7] D. LOUIS-ADRIEN, L. QUENTIN et L. GUILLAUME : Learning linear block codes with gradient quantization, 2025.

[8] F. MARCO et T. Alessandro et AL : Thresholds of absorbing sets under scaled min-sum ldpc decoding. *IEEE Transactions on Communications*, 67(10):6643–6651, 2019.

[9] Tom RICHARDSON : Error floors of ldpc codes. *In Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 1426–1435. The University ; 1998, 2003.

[10] Mingxu ZHANG, Qin HUANG, Shuai WANG et Zulin WANG : Construction of ldpc codes based on deep reinforcement learning. *In 2018 10th International Conference on WCSP*, pages 1–4, 2018.