Comparaison des décodeurs de Chase, l'OSD et ceux basés sur les algorithmes génétiques

Faissal El Bouanani[1], Hassan Berbia[1], Mostafa Belkasmi[1], Hussain Ben-azza[2]

[1]Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes - Rabat, Maroc

[2]Ecole Nationale Supérieure d'Arts et Métiers - Meknès, Maroc
elbouanani@ensam.ac.ma,berbia@ensias.ma,belkasmi@ensias.ma,hbenazza@yahoo.com

Résumé − Les décodeurs basés sur les algorithmes génétiques (AG) appliqués aux codes BCH ont de bonnes performances par rapport à Chase-2 et l'OSD d'ordre 1 et atteignent les performances de l'OSD-3 pour quelques codes Résidu Quadratiques (RQ). Ces algorithmes restent moins complexes pour les codes linéaires de grandes longueurs; en plus leurs performances peuvent être améliorées en changeant les paramètres, en particulier le nombre d'individus par population et le nombre de générations, ce qui les rend attractifs.

Abstract − The decoders based on the genetic algorithms (AG) applied to BCH codes give good performances compared to Chase-2 and the OSD of order-1 and reach the performances of the OSD-3 for some Residue Quadratic (RQ) codes. These algorithms are less complex for linear block codes of large block length; furthermore their performances can be improved by changing the parameters, in particular the number of individuals by population and the number of generations, which makes them more attractive.

1 Introduction

Le développement des systèmes communication a un grand effet sur les activités de recherche et développement dans le domaine des codes correcteurs d'erreurs. La mise en œuvre des codes passe par l'utilisation d'un couple codeur/décodeur. Le codeur génère les mots de code en ajoutant de la redondance aux vecteurs d'information, tandis que les algorithmes de décodage, cherchent le mot de code transmis le plus probable associé au mot reçu bruité.

Le décodage à décision pondérée est un problème NP-Complet approché par plusieurs méthodes. Les premières solutions ont été basée sur des méthodes algébriques et probabilistes, par exemple, GMD (Generalized Minimum Distance Decoding) [3], l'algorithme de Chase [1], celui de Hartmann Rudolph [2] et l'OSD [8] pour les codes en bloc.

Les techniques d'intelligence artificielle introduites récemment dans ce type de décodage ont donnée des performances assez bonnes. Par exemple, l'utilisation de l'algorithme A* par Han et al. pour décoder les codes BCH et RQ [5], les AG pour décoder les codes linéaires [6] et les réseaux de neurones pour décoder les codes de Hamming et BCH [7].

Dans ce travail, nous comparons les performances et la complexité temporelle des décodeurs à base des AG (DAG), de l'OSD et celui de Chase pour les codes de taux $\frac{1}{2}$: $BCH(127, 64, 21)$, $BCH(63, 30, 14)$, $BCH(31, 16, 7)$, $RQ(103, 52, 19)$ et $RQ(71, 36, 11)$. L'algorithme de Chase, basé sur un décodeur à décision ferme, est appliqué uniquement sur les codes BCH. Ainsi, la comparaison pour les codes RQ est limitée aux algorithmes OSD et DAG.

Cet article est organisé comme suit. La section 2 décrit les algorithmes étudiés. Une brève escription de leurs complexités est présentée dans la section 3. La section 4 présente et analyse les performances et la complexité des différents algorithmes. Enfin, la section 5 conclut ce papier et propose des extensions envisagés.

2 Les algorithmes de décodage étudiés

On considère un code en bloc linéaire C de paramètres (n, k, d) permettant de corriger t erreurs et de matrice génératrice G. On désigne par $(r_i)_{1\leq i\leq n}$ le mot reçu et $(s_i)_{1\leq i\leq n}$ la décision ferme correspondante :

$$s_i = \begin{cases} 1 \ si \ r_i > 0 \\ 0 \ si \ r_i \leq 0 \end{cases} \quad , 1 \leq i \leq n \tag{1}$$

Nous commençons cette section par une brève description des algorithmes utilisés.

2.1 Algorithme Chase-2

En triant les bits r_i dans l'ordre *croissant*, on obtient la nouvelle séquence triée $(r_{p_i})_{1\leq i\leq n}$ où $p_i \epsilon \{1, .., n\}$:

$$\begin{cases} q_i = r_{p_i} \qquad 1 \leq i \leq n \\ |q_1| \leq |q_2| \leq \cdots \leq |q_n| \end{cases} \tag{2}$$

L'algorithme de Chase-2 génère 2^t séquences de test à partir de la séquence $(s_i)_{1\leq i\leq n}$ en inversant j bits parmi les t bits les moins fiables $(0 \leq j \leq t)$. Chaque séquence de test

est ensuite décodée par un décodeur à décision ferme. Le mot retenu est le mot le plus proche du mot reçu parmi les 2^t mots décodés.

2.2 Algorithme OSD

Comme Chase, l'OSD crée des séquences de test à partir de la décision ferme $(s_i)_{1\leq i\leq n}$. Dans cet algorithme, le mot reçu est trié dans l'ordre *décroissant* :

$$\begin{cases} q_i = r_{p_i} & 1 \leq i \leq n \\ |q_1| \geq |q_2| \geq \cdots \geq |q_n| \end{cases} \quad (3)$$

Soit π_1 la permutation associée à ce tri : $q = \pi_1(r)$. En permuttant les colonnes de G par π_1, on obtient :

$$G^{(1)} = \pi_1(G) \quad (4)$$

Ensuite, on passe à une nouvelle matrice $G^{(2)}$ en utilisant une permutation π_2 afin de rendre les k premières colonnes de la matrice génératrice indépendantes :

$$G^{(2)} = \pi_2(G^{(1)}) = \pi_2\pi_1(G) \quad (5)$$

La séquence $(q_i)_{1\leq i\leq n}$ est réordonnée elle aussi par π_2 donnant ainsi la séquence $(x_i)_{1\leq i\leq n}$

$$x = \pi_2(q) = \pi_2\pi_1(r) \quad (6)$$

Soit $G^{(3)}$ la matrice génératrice systématique associée au code généré par $G^{(2)}$:

$$G^{(3)} = (I_k P_{k,n-k}) \quad (7)$$

Soit $(y_i)_{1\leq i\leq k}$ la décision ferme associée à $(x_i)_{1\leq i\leq k}$. L'OSD(m) d'ordre m génère toutes les séquences d'information $z = (z_i)_{1\leq i\leq k}$ à partir $(y_i)_{1\leq i\leq k}$ en inversant j bits parmi les k bits ($0 \leq j \leq m$). A toute séquence z, on associe alors un mot de code $c^{(3)}$:

$$c^{(3)} = zG^{(3)} \quad (8)$$

Le mot de code retenu $\overline{c}^{(3)}$ est le mot de code $c^{(3)}$ le plus proche de la séquence réordonnée $(x_i)_{1\leq i\leq n}$. Par conséquent, le mot de code associé à la matrice génératrice initiale G est :

$$\overline{c} = \pi_1^{-1}\pi_2^{-1}(\overline{c}^{(3)}) \quad (9)$$

2.3 Description de DAG

La technique des AG [4] a été utilisée par Maini et al. [6] pour décoder les codes linéaires. Nous avons modifié cet algorithme en introduisant la notion d'élitisme. Dans cet algorithme, la séquence reçue r est triée dans l'ordre *décroissant* de telle façon que les k premières colonnes de $G' = \pi(G)$ soient linéairement indépendantes. L'étape suivante consiste à générer une population initiale de N_i vecteurs d'information, contenant les k premiers bits d'information de $q = \pi(r)$. La population est triée dans l'ordre croissant de la *fitness* des individus. Un individu représente aussi bien un vecteur d'information que le mot de code qui lui est associé.
La fitness est définie comme étant la distance euclidienne entre l'individu et le mot reçu permuté q. Les N_e premiers individus sont introduits dans la nouvelle génération où N_e est le nombre d'élites (ou têtes de liste). Ensuite, une opération de sélection utilisant la méthode d'ordonnancement linéaire est appliquée afin d'identifier les meilleurs parents $(p^{(1)}, p^{(2)})$ auxquels on appliquera les opérateurs de reproduction. Nous avons appliqué l'ordonnancement linéaire défini par l'équation suivante :

$$poids_i = poids\_\max - \frac{2i.(poids\_\max - 1)}{N_i - 1}, 1 \leq i \leq N_i \quad (10)$$

où $poids_i$ est le poids du $i^{ème}$ individu et $poids\_\max$ représente le poids du premier individu fixé ici à 1.1;

Les opérations de croisement et mutation sont ensuite appliquées aux parents sélectionnés pour donner naissance à des individus $(c_i)_{N_e+1\leq i\leq N_i}$ formant la nouvelle génération. Désignons par *Random* une valeur tirée aléatoirement suivant une loi uniforme comprise entre 0 et 1.

- *Croisement : Si* p_c est la probabilité de croisement, alors pour $N_e + 1 \leq i \leq N_i$:

– **Si** $Random < p_c$, alors $\forall j \in \{1, .., k\}$:

$$p\left(c_{ij} = p_j^{(1)}\right) = \left(1 - p_j^{(1)} + p_j^{(1)}p_j^{(2)}\right) + \frac{p_j^{(1)} - p_j^{(2)}}{1 + e^{\frac{-4q_j}{N_0}}} \quad (11)$$

Dans ce cas, les bits de l'individu c_i seront *mutés* avec une probabilité p_m :

$$c_{ij} \leftarrow 1 - c_{ij} \quad si\ Random < p_m \quad (12)$$

– **Sinon** :

$$c_i = \begin{cases} p^{(1)}\ si\ Random < 0,5 \\ p^{(2)}\ sinon \end{cases} \quad (13)$$

Ces étapes sont répétées pour un nombre fixe de générations N_g. Le mot décodé retenu $\widehat{r}$ est le plus proche de $\pi(r)$, et le mot de code décidé associé à G est $\pi^{-1}(\widehat{r})$.

En [6], Maini et al. ont appliqué DAG sur un seul code qui est le code étendu RQ(104,52) avec les paramètres $N_g = 100$, $N_i = 300, p_c = 0.7$ et $p_m = 0.03$. L'un des objectifs de ce travail est de vérifier l'efficacité de DAG pour d'autres codes RQ et BCH.

3 Complexité des algorithmes

La table 1 montre la complexité des trois algorithmes.

TAB. 1 – La complexité des trois algorithmes

Chase-2	OSD-m	DAG
$O\left(2^t n^2 \log_2 n\right)$	$O\left(n^{m+1}\right)$	$O\left(N_i N_g \left[kn + \log N_i\right] + k^2 n\right)$

L'algorithme de Chase-2 s'accroît exponentiellement avec la capacité de correction t du code. Sa complexité est 2^t fois la complexité d'un décodage à entrée et sortie fermes [1]. De même, la complexité de l'OSD d'ordre m est exponentielle en m [8]; en plus, ce décodage devient plus complexe dans le cas de grandes longueurs du code. Quant à

DAG, la complexité est polynomiale en k, n, N_g, N_i et $\log N_i$ [6], ce qui le rend moins complexe par rapport aux deux autres algorithmes.

4 Performances des algorithmes

Nous avons commencé cette étude par une implémentation de DAG. Tous les résultats énoncés sont issus de simulations d'un système de transmission utilisant un canal à bruit blanc gaussien additif et une modulation MDP-2. Avant l'étude des performances de DAG, une étape préliminaire était d'optimiser p_c, p_m, N_g, N_i et $poids\_\max$. Excepté la figure 1, les performances de DAG ont été obtenues pour $p_c = 0.97$, $p_m = 0.03, N_g = 100$, $N_i = 300$ et $N_e = 1$. L'arrêt de la simulation correspond à un nombre minimum de 30 blocs erronés. Les performances sont données en terme de BER en fonction de $SNR = \frac{E_b}{N_0}$.

4.1 Effet du nombre d'individus

L'agrandissement du nombre d'individus global $N_g.N_i$ permet d'élargir l'espace de recherche des mots du code les plus proches de r, ce qui permet d'améliorer les performances du DAG. La figure 1 montre l'effet des paramètres N_i et N_g sur DAG pour le code $RQ(103, 52, 19)$. Le gain de la courbe associée à $N_g.N_i = 30000$ par rapport à celle utilisant 1000 individus est de $1.4dB$ à un $BER = 10^{-4}$.

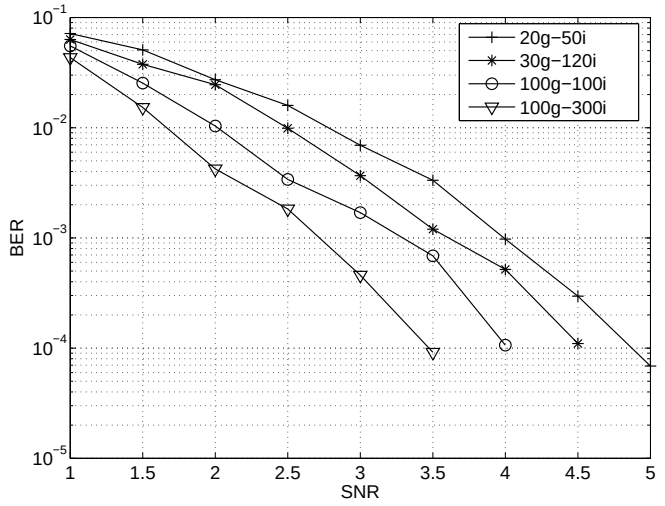

FIG. 1 – Effet du nombre d'individus et de générations sur DAG

4.2 Effet de la dimension du code

La figure 2 compare les performances de DAG pour cinq codes de taux $\simeq \frac{1}{2}$. Excepté le petit code, tous les autres codes ont des performances presque égales ; ceci est dû éventuellement aux paramètres de DAG, en particulier le nombre d'individus, choisit fixe pour tous les codes. En effet, quand la dimension du code k augmente, ce nombre couvre un petit pourcentage de l'espace de recherche. Ce qui peut diminuer les chances de trouver le mot du code le plus proche, en particulier pour les petits SNR.

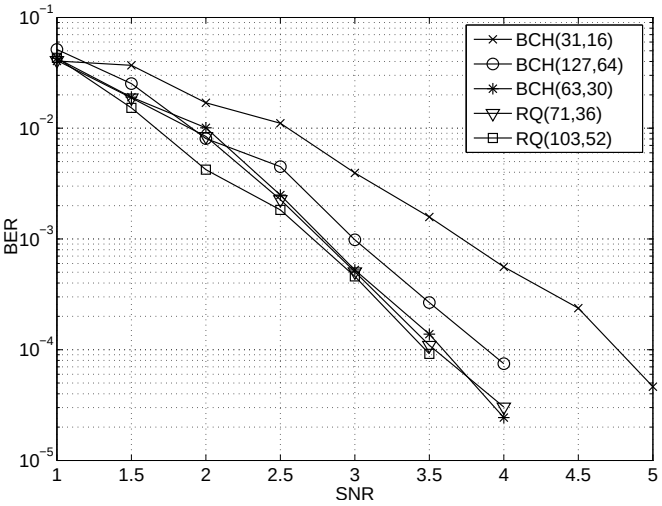

FIG. 2 – Effet de la dimension du code sur DAG

4.3 Performances du code BCH(127,64,21)

La figure 3 compare les trois algorithmes de décodage appliqués au code $BCH(127, 64, 21)$. ; elle montre que le DAG est plus performant que Chase-2 et l'$OSD(1)$.
Malgré le grand nombre de séquences de test utilisées par Chase-2, l'algorithme DAG le dépasse au niveau performance en étant moins complexe. Une augmentation de l'ordre de l'OSD permet de dépasser les performances de DAG sur tout l'intervalle du rapport SNR. Le gain du décodeur basé sur DAG par rapport à l'$OSD(1)$ au $BER = 2.10^{-5}$ est de $1dB$.

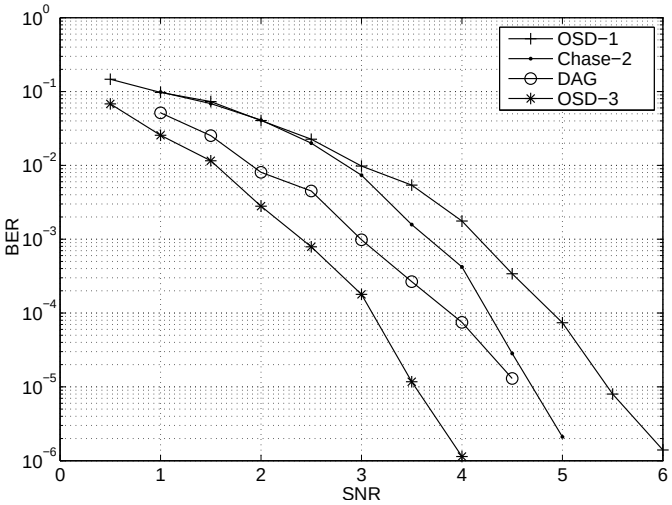

FIG. 3 – Performances de DAG, Chase-2 et OSD pour le $BCH(127, 64, 21)$

4.4 Performance du code RQ(71,36,11)

La figure 4 montre que DAG atteint l'OSD-3 pour le code $RQ(71, 36, 11)$. Une augmentation de nombres d'individus $N_g.N_i$ permet de dépasser l'OSD-3.

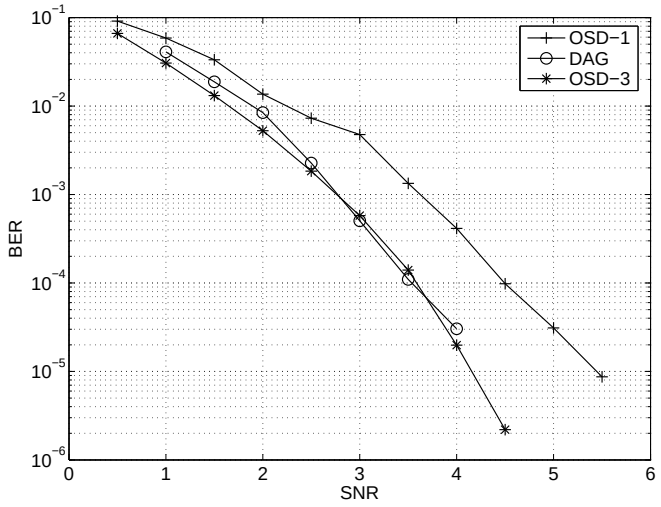

FIG. 4 – Performances de DAG et OSD pour le $RQ(71, 36, 11)$

4.5 Complexité

La figure 5 montre à la fois la complexité temporelle et théorique des trois décodeurs. Les complexités temporelles (courbes en trait continu) sont issues du calcul de la durée d'exécution de 1000 blocs, et celles théoriques sont calculées en multipliant par une constante ($K = 3.10^{-9}$) les fonctions de la table 1, et ceci pour montrer la corrélation entre les deux complexités. Cette figure montre que la complexité est polynomiale pour DAG et exponentielle pour Chase-2 et l'OSD.
Les courbes théoriques convergent vers les courbes de simulation pour les n assez grands. Les algorithmes de Chase-2 et l'OSD-3 sont moins complexes que DAG pour les petits n. Leurs aspects exponentiels les rendent moins intéressants pour les n assez grands.

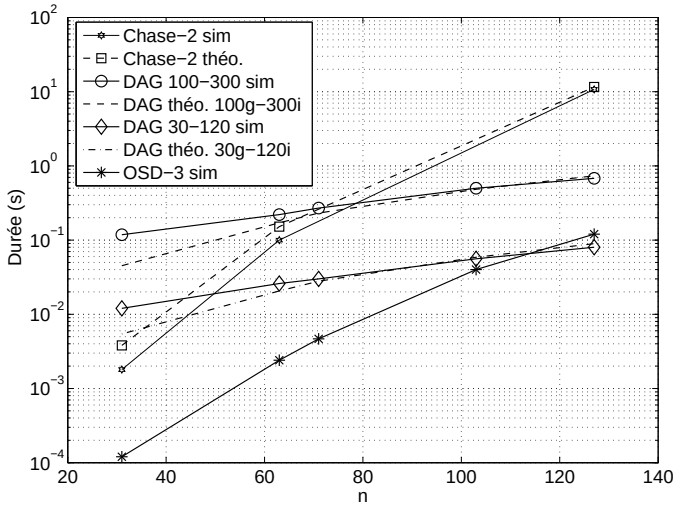

FIG. 5 – Complexité des trois algorithmes

5 Conclusion

Dans ce travail, nous avons comparé trois algorithmes de décodage à décision pondérée (Chase-2, OSD et DAG). Le décodeur à base des AG est meilleur que l'OSD-1 et Chase-2, par contre, il est dépassé par l'OSD d'ordre supérieur à quatre. Les effets des différents paramètres sur les performances de DAG nous ont amené à étudier d'autres méthodes de sélection, de croisement et de mutation, ainsi que les critères d'arrêt.

Références

[1] D. Chase, "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information", IEEE Trans. Inform. Theory, vol, 18, pp. 170–181, January 1972

[2] G. C. Clarck, J.B. Cain , "Error-Correction Coding for Digital Communications", New York Plenum, 1981

[3] G.D. Forney, Jr., "Generalized Minimum Distance Decoding", IEEE Transactions on Information Theory, vol. IT-12, pp. 125-131, April 1966

[4] D. E. Goldberg," Genetic Algorithms in search, Optimization, and machine learning", Addison-Wesly, Reading M.A, 1989.

[5] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient maximum-likelihood soft-decision decoding of linear block codes using algorithm A*", Technical Report SU-CIS-91-42, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244, December 1991

[6] H.S. Maini, K. G. Mehrotra,C. Mohan, S. Ranka, "Genetic Algorithms for Soft Decision Decoding of Linear Block Codes", Journal of Evolutionary Computation, Vol.2, No.2, pp.145-164, Nov.1994

[7] Ja-Ling Wu, Yuen-Hsien Tseng, and Yuh-Ming Huang, "Neural Networks Decoders for Linear Block Codes", International Journal of Computational Engineering Science, vol.3, No.3, pp.235-255, 2002

[8] M.P.C. Fossorier and S. lin, "Soft decision decoding of linear block codes based on ordered statistics", IEEE Trans. information theory Vol. 41, pp. 1379-1396, sep. 1995.

[9] H. Morelos-Zaragoza, "The Art of Error Correcting Coding", Second Edition Robert , John Wiley & Sons, Ltd. ISBN : 0-470-01558-6, 2006