D'un algorithme ensembliste vers un algorithme génétique pour la synchronisation d'un décodeur chaotique en présence de bruit.

Jean-Philippe Mangeot*, Frédéric Launay *, Laurent Signac * et Patrick Coirault*
*Laboratoire d'Automatique et d'Informatique Industrielle, 40, avenue du recteur Pineau, 86000 Poitiers. Site web: http://laii.univ-poitiers

***Abstract*—L'utilisation de codes chaotiques présente de nombreux avantages pour la transmission d'informations. Notamment en terme de sécurité où l'information peut être soit cryptée, soit codée et masquée dans le bruit (Technique DS-CDMA). Toutefois, les techniques de synchronisation, c'est à dire de connaissance de l'état du codeur se révèlent difficiles notamment en présence de bruit.**

Nous présentons dans un premier temps les algorithmes ensemblistes : ceux-ci sont adaptés pour synchroniser des générateurs chaotiques discrets en présence de bruit et en temps réel mais ils nécessitent une ressource mémoire importante. Dans ce papier, nous proposons l'évolution de ces algorithmes pour une implémentation matérielle sur composant programmable. Une réduction des ressources est proposée au travers un algorithme génétique, qui permet de travailler sur un nombre de ressource fixe.

I. Introduction

Un signal chaotique est un signal d'apparence aléatoire mais qui est issu d'un système déterministe. Par conséquent, l'évolution du signal est entièrement définie par les conditions initiales. Cette particularité rend donc cette méthode très attractive dans le secteur des télécommunications pour le multiplexage d'accès et la sécurisation des données. Ainsi, pour permettre la synchronisation, il est nécessaire d'implémenter au niveau du récepteur le même système déterministe que celui utilisé par l'émetteur.

Les premières expérimentations de communications basées sur le chaos ont été réalisées sur des circuits électroniques analogiques [1], mais la sensibilité des systèmes à la déviation des composants a longtemps posé problème dans les systèmes de communication. En effet, les systèmes chaotiques sont très sensibles aux perturbations et le signal issu de deux systèmes très proches divergent très rapidement (coefficient de Lyapunov).

En revanche, les systèmes numériques programmables (DSP, FPGA, microcontrôleurs) permettent de générer aisément et de manière reproductible des signaux issus de la discrétisation d'équations chaotiques [2]. Les séquences générées deviennent quasi-chaotiques puisqu'elles sont représentées sur un nombre fini de bits et perdent de ce fait quelques particularités mathématiques (unicité, aperiodicité). Néanmoins, les propriétés recherchées comme la largeur de spectre et l'aperiodicité (étalement du signal), l'orthogonalité des séquences (pour éviter les intérférences entre codes pseudo-chaotiques dans un canal partagé) peuvent être approchées en choisissant des codes de longueurs suffisantes [3], [4]. En présence de bruit dans le canal de communication, la synchronisation du chaos devient plus difficile. De nombreuses méthodes de réduction de bruit peuvent être trouvées dans la littérature. Elles sont souvent réalisées par un algorithme adaptatif s'appuyant sur une minimisation d'une fonction de coût [5],[6],[7]. Les algorithmes ensemblistes, plus gourmands en ressources permettent néammoins une parallélisation des calculs [8].

Le synoptique du générateur chaotique utilisé dans ce papier est présenté au chapitre suivant. Dans le chapitre III, nous présenterons l'architecture d'un décodeur utilisant un algorithme ensembliste et nous mettrons en avant les limites vis-à-vis de la consommation de ressources mémoires. Enfin dans le chapitre IV, nous formaliserons une évolution de cet algorithme vers un algorithme génétique afin d'optimiser les ressources matérielles, principal défaut de l'algorithme ensembliste.

II. Conception du système

Considérons une système dynamique discret, de structure AR comme décrit sur la figure 1.

Soit $\underline{\mathbf{z}}(k) = [y(k), y(k-1), \ldots, y(k-n+1)]^T$ le vecteur d'état du système. L'entier k représente le temps discret. L'évolution temporelle du système est décrit par les équations suivantes

$$\begin{cases} z_1(k+1) = \sum_{i=1}^{N} G_i z_i(k) + e(k) + f_{NL}(\underline{\mathbf{z}}(k)) \\ z_2(k+1) = z_1(k) \\ \vdots \\ z_N(k+1) = z_{N-1}(k) \\ y(k) = z_1(k+1) \end{cases} \quad (1)$$

Sous forme matricielle, le système peut s'écrire:

$$\begin{cases} \underline{\mathbf{z}}(k+1) = A\underline{\mathbf{z}}(k) + Be(k) + f_{NL}(\underline{\mathbf{z}}(k)) \\ y(k) = C\underline{\mathbf{z}}(k) \end{cases} \quad (2)$$

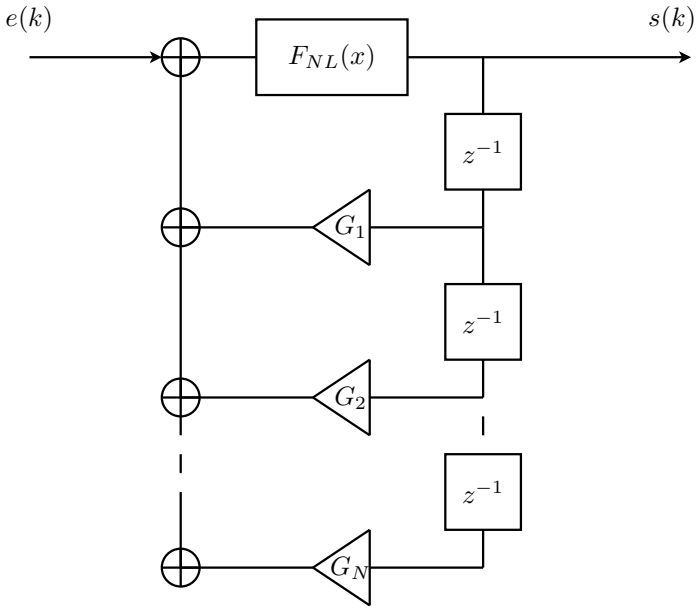

Fig. 1. Structure du codeur chaotique

$$\begin{cases} \underline{\mathbf{z}}(k+1) = \begin{bmatrix} G_1 & G_2 & \dots & G_{N-1} & G_N \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \underline{\mathbf{z}}(k) + f_{NL}(\underline{\mathbf{z}}(k)) \\ y(k) = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \end{pmatrix} \underline{\mathbf{z}}(k) \end{cases} \tag{3}$$

Dans le cas où la matrice A possède n valeurs propres distinctes γ_j avec $j = 1, \dots, n.$, celles-ci sont déterminées par les racines du polynôme caractéristique.

$$C(z) = z^n - \sum_{j=1}^{N} G_j z^{N-j} \tag{4}$$

Dans [9], Kelber a montré que le filtre était ergodique et préservait une distribution uniforme n-dimensionnelle si :

i le système n'est pas décomposable et les coefficients $G_n \in Z, |G_n| > 1$;
ii $f_{NL}(.)$ est une fonction qui préserve la distribution uniforme ;
iii $|\gamma_j| \neq 1, \forall j \in [1 \dots n]$.

En [10], une structure de systèmes chaotiques discrétisés générant des séquences à distribution uniforme a été proposée. Grâce à l'utilisation des fonctions modulo et des fonctions à rotation circulaire, cette structure s'implémente facilement sur un composant programmable comme le FPGA :

$$y(k) = mod(e(k) + \sum_{j=1}^{N} G_j y(k-j)) \tag{5}$$

et

$$e^*(k) = mod(r(k) - \sum_{j=1}^{N} G_j r(k-j)) \tag{6}$$

Dans ces équations, $r(k)$ est le signal reçu par le décodeur et $e^*(k)$ est le signal estimé. En l'absence de bruit et pour $k > N$, il vient $e(k) = e^*(k)$. Cette architecture dite autosynchronisante est souvent utilisée pour du chiffrement, en revanche, la présence d'une perturbation entre le codeur et le décodeur a pour effet de désynchroniser le récepteur sur une durée de N symboles. En appelant $n(k)$ un bruit gaussien additif, l'équation du signal reçu devient

$$r(k) = y(k) + n(k) \tag{7}$$

En remplaçant eq.5 dans eq.7 puis eq.7 dans eq.6, il vient:

$$e^*(k) = mod\left(e(k) + mod\left(n(k) - \sum_{j=1}^{N} G_j n(k-j)\right)\right) \tag{8}$$

Compte tenu du caractère aléatoire du bruit, nous pouvons écrire:

$$e^*(k) \approx mod(e(k) + n(k)) \tag{9}$$

En d'autres termes le bruit ajouté se retrouve directement sur la séquence estimée au modulo près. Même si la méthode de l'inversion du générateur ne présente pas en numérique les défauts liés à la dérive et à la précision des composants, elle présente une sensibilité au bruit. La présence d'un outil de contrôle est donc nécessaire à la synchronisation du récepteur.

III. Synchronisation en présence de bruit

A. Contexte

Nous allons considérer le récepteur sous la forme présentée par la figure III-A

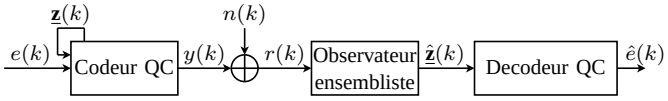

Fig. 2. Schéma du codage et decodage chaotique

Pour synchroniser le récepteur en présence de bruit, un estimateur ensembliste idéal va être défini. Considérons la forme réduite du système décrit en (2) lorsque celui-ci est en mode autonome ($e(k) = 0$) :

$$\begin{aligned} \underline{\mathbf{z}}(k+1) &= \varphi(\underline{\mathbf{z}}(k)) \\ y(k) &= h(\underline{\mathbf{z}}(k)) \end{aligned} \tag{10}$$

Nous appellons $\underline{\mathbf{z}}(k)$ la variable d'état. La fonction $\varphi(.)$ est une fonction linéaire par morceaux de $\mathbb{R}^N$ dans $\mathbb{R}^N$ assurant les propriétés chaotiques du codeur $h(.)$ est la fonction de sortie du système. La variable de sortie est appelée $y(k)$. Le signal de sortie généré dépend à la fois des équations d'état

mais aussi des conditions initiales. Ce signal de sortie peut être paramétré en fonction des conditions initiales:

$$y(\underline{\mathbf{z}}_0) = (h \circ \underline{\mathbf{z}}_0, h \circ \varphi(\underline{\mathbf{z}}_0), h \circ \varphi^2(\underline{\mathbf{z}}_0), \ldots, h \circ \varphi^k(\underline{\mathbf{z}}_0), \ldots) \quad (11)$$

où $\underline{\mathbf{z}}_0$ est l'état initial de la variable d'état. Ce signal de sortie est perturbé par un bruit $n(k)$, supposé borné par un réel α positif :

$$\forall k, \|n(k)\| < \alpha \quad (12)$$

Le signal reçu est appelé $r(k) = y(k) + n(k)$

B. Des ensembles contenant tous les vecteurs possibles

Rappelons que dans un cadre ensembliste, les incertitudes sur les grandeurs manipulées sont prises en compte en associant aux vecteurs $\underline{\mathbf{z}}(k)$ et $y(k)$ les domaines $\mathbb{Z}(k)$ et $\mathbb{Y}(k)$, censés les contenir.

A chaque instant le sous-ensemble $\mathbb{Y}(k)$ contient la vraie mesure de $y(k)$, c'est à dire que $\mathbb{Y}(k)$ est défini comme suit:

$$\mathbb{Y}(k) = \{y(k) \in \mathbb{Y}(k) | \, \|y(k) - r(k)\|_2 < \alpha\} \quad (13)$$

$\mathbb{Y}(k)$ est l'ensemble des sorties possibles ayant été émises à l'instant k. Cet ensemble ne tient compte que de la contrainte due à l'incertitude sur le signal reçu (présence de bruit).

A présent définissons l'ensemble de vecteurs qui vérifient les conditions de transition d'un instant au suivant, à savoir les deux équations : équations d'états et équation de sortie. Nous appellerons cette transition $h \circ \varphi$ allant de $\mathbb{R}^N$ dans $\mathbb{R}$. Nous définissons alors l'image directe de $\mathbb{Z}$ par $h \circ \varphi$:

$$h \circ \varphi(\mathbb{Z}) = \{y \in \mathbb{R} | \exists \underline{\mathbf{z}} \in \mathbb{Z}, h \circ \varphi(z) = y\}, \quad (14)$$

De la même manière, nous introduisons la notion d'image inverse de $\mathbb{Y}$ par $h \circ \varphi$:

$$(h \circ \varphi)^{-1}(\mathbb{Y}) = \left\{\underline{\mathbf{z}} \in \mathbb{R}^N | \exists y \in \mathbb{Y}, h \circ \varphi(z) = y\right\}. \quad (15)$$

Nous cherchons dans cette partie à réaliser un observateur qui tente d'estimer les états $\underline{\mathbf{z}}(k)$ en fonction des mesures bruitées des vecteurs de sortie $y(0), \ldots, y(k), y(k+1)$.

IV. L'OBSERVATEUR ENSEMBLISTE CAUSAL

Considérons la suite d'ensembles $\mathbb{Z}(k)$ obtenus par l'algorithme ensembliste suivant:

Entrée :	$h \circ \varphi, \mathbb{Y}(1), \mathbb{Y}(2), \ldots,$
Initialisation :	$k := 0;$
	$\mathbb{Z}(0) = (h \circ \varphi)^{-1}(\mathbb{Y}(1))$
Répéter:	$k := k + 1;$
	$\mathbb{Y}(k) := h \circ \varphi(\mathbb{Z}(k-1)) \cap \mathbb{Y}(k)$
	$\mathbb{Z}(k-1) := (h \circ \varphi)^{-1}(\mathbb{Y}(k))$
	$\mathbb{Z}(k) := \varphi(\mathbb{Z}(k-1)))$
Tant que	$card(\mathbb{Z}(k)) > 1$
Sortie:	$\mathbb{Z}(k)$ est reduit au singleton $\underline{\mathbf{z}}(k)$

Cet algorithme recense l'ensemble des conditions initiales qui vérifient à la fois les conditions d'évolutions des trajectoires et à la fois les contraintes liées au bruit. Au fur et à mesure des itérations, le nombre de candidats qui respectent ces deux contraintes diminue jusqu'à l'obtention d'un seul candidat.

Les simulations effectuées montrent qu'en mode autonome ($e(k) = 0$) le décodeur peut se synchroniser grâce à cet algorithme pour des SNRs inférieurs à 3dB. Les simulations ont été réalisées avec le codeur de Frey [11] c'est à dire un système d'ordre $N = 2$, avec $G_1 = 2$; $G_2 = 2$ et f_{NL} la fonction rotation circulaire à gauche. Les états ont été codés sur 8 bits. L'amplitude du signal de sortie est donc de 256 niveaux.
Le bruit $n(k)$ est un bruit aléatoire avec une densité de probabilité uniformément repartie sur $[-\alpha, \alpha]$ avec $\alpha = 100$, soit un rapport signal à bruit de 2.45 dB. Sur la figure 3, nous présentons la convergence de l'algorithme en fonction de l'évolution du signal. En pointillée, nous visualisons la séquence émise. Pour des raisons de clarté et de simplification, nous avons initialisé le récepteur sur l'amplitude du premier échantillon émis. Ainsi, en trait plein démarrant à la valeur de 100 (vert), nous présentons l'évolution du signal estimé. Le deuxième signal en trait plein (rouge) représente le signal reçu, c'est à dire le signal émis avec un bruit gaussien. Le signal estimé s'appuye sur le signal reçu, mais l'évolution du signal est déterministe. Au bout d'une dizaine d'itérations, l'erreur entre le signal émis et le signal estimé est très faible comme le montre la figure (3-b)

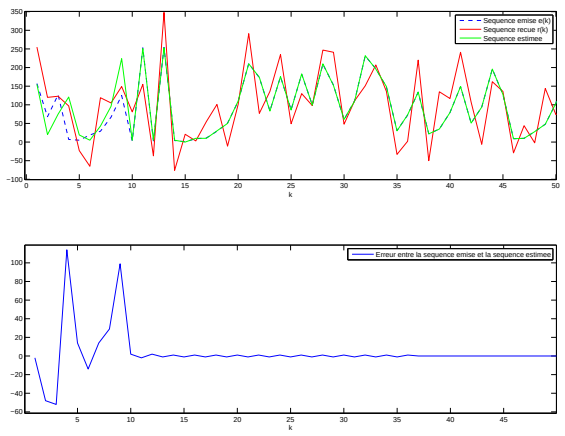

Fig. 3. Erreur de synchronisation en fonction du nombre d'échantillons reçus. Il a été nécessaire de réaliser 37 itérations pour retrouver les conditions initiales . $\alpha = 100$ ce qui correspond à un SNR de 2.45 dB

Sur la figure 4, nous généralisons la procédure de synchronisation en prenant en compte l'ensemble des erreurs possibles, c'est à dire bornée avec une erreur de +/- 100 au niveau de chaque échantillon reçu. Nous nous s'aperçevons que le nombre de candidats potentiels devient très limité au bout de 17 itérations. La simulation nécessite néanmoins de choisir deux échantillons bruités, d'estimer le troisième échantillon et de le conserver si l'erreur entre l'échantillon estimé et l'échantillon reçu est dans la borne d'incertitude. Nous devons ainsi estimer l'évolution des 200*200 premiers candidats.

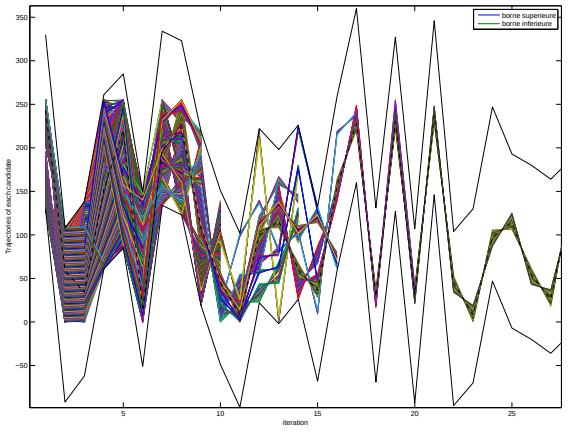

Fig. 4. Pour chaque candidat, la trajectoire est affichée. Si la trajectoire sort des bornes, les candidats sont éliminés.

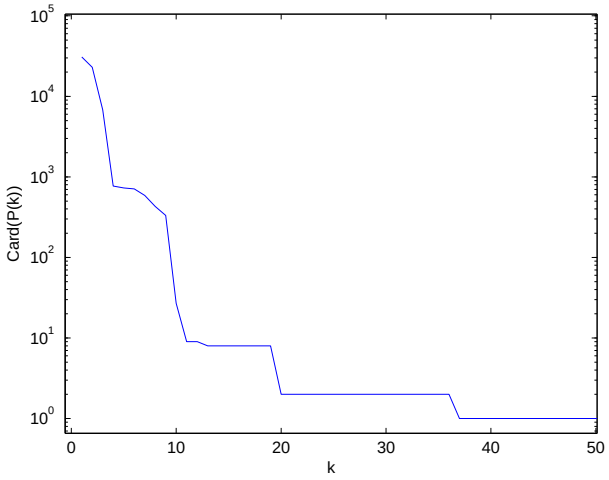

Fig. 5. Evolution de$card(\mathbb{Z}(k))$ en fonction des échantillons reçus. Le bruit $n(k)$ est un bruit aléatoire avec une densité de probabilité uniformément repartie sur $[-100, 100]$ (amplitude 200), et le signal émis est d'amplitude $2^{N_b} = 2^8 = 256$

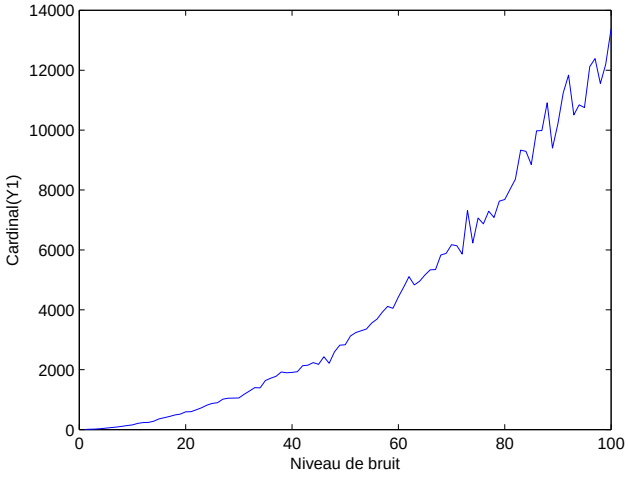

Fig. 6. Evolution de $card(\mathbb{Z}(0))$ en fonction de l'intervalle de bruit α. Cet essai est une moyenne de $card(\mathbb{Z}(0))$ effectuée sur un tirage de 100 états initiaux aléatoires pour chaque niveau de bruit.

La figure 5 montre le nombre de candidats possibles respectant les équations de transitions chaotiques ainsi que les équations de bruit en fonction du nombre d'itérations. Les résultats montrent que pour des rapports signal à bruit inférieurs à 3dB, le cardinal de la population initiale des cas possibles est supérieur à 10^5 ce qui rend difficile l'implantation de cet algorithme ensembliste sur un composant programmable avec un nombre fini de ressources.

V. Vers un algorithme génétique pour minimiser le nombre de candidats à tester.

L'utilisation d'algorithmes génétiques dans la résolution de problèmes est à l'origine le fruit de recherche assez anciennes

[12] . Récemment, de tels algorithmes ont été implantés sur FPGAs en [13] [14]. L'avantage d'un algorithme génétique dans une application telle que la synchronisation d'une séquence chaotique en présence de bruit est qu'il permet de travailler avec une population beaucoup plus faible que dans le cas d'un algorithme ensembliste.

Dans la simulation précédente, les symboles chaotiques codés sur Nb bits étaient transmis en bande de base. Dans un contexte plus réaliste de communications numériques, ces symboles sont généralement sérialisés et transmis via une modulation de phase à plusieurs états (BPSK, QPSK...). Dans ce cas, l'effet du bruit sur le canal ne se modélise plus comme une variation d'amplitude du signal reçu, mais comme une altération du message reçu en termes de taux d'erreurs binaire. Nous proposons donc de mesurer les performances de synchronisation de l'algorithme directement en fonction du taux d'erreur binaire, et ceux, indépendamment du type de modulation ou du rapport signal sur bruit.

Soit $r(k)$ le symbole reçu sur Nb bits. Nous définissons le sous-ensemble $\mathbb{Y}(k)$ comme l'ensemble des symboles ayant une distance de Hamming avec le signal reçu inférieure à une constante β :

$$\mathbb{Y}(k) = \{y(k) \in \mathbb{Y}(k) | d(r(k)), y(k)) \leq \beta\} \tag{16}$$

Dans cette définition, $d(.,.)$ représente la distance de Hamming entre deux symboles binaires et β correspond au nombre maximum de bits erronés par symbole. Par conséquent le nombre de candidats dans cette population est :

$$card(\mathbb{Y}(k)) = \sum_{k=0}^{\beta} C_{Nb}^{k} \tag{17}$$

Par exemple, cela correspond sur 8 bits et avec $\beta = 2$ à une population de 137 gènes.

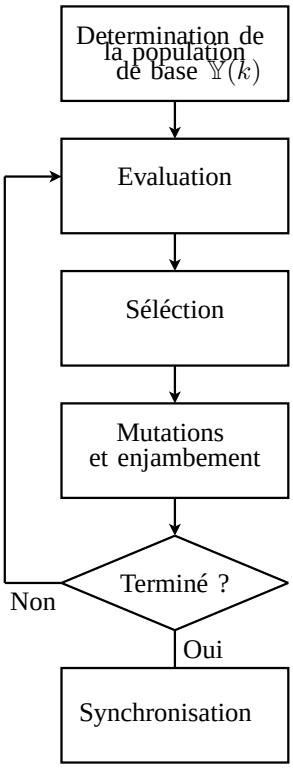

Fig. 7. Principe de l'algorithme génétique

A. Détermination de la population de base

A l'initialisation, l'algorithme construit sa population de base $\mathbb{Z}(0)$ à partir de $r(0)$ et l'équation 16

B. Evaluation

L'image directe de la population courante $\mathbb{Y}(k)$ est calculée par la fonction $h \circ \varphi$. Une note correspondant à l'adaptation au problème est attribuée à chaque gène de $h \circ \varphi(\mathbb{Y}(k))$.
Cette note prend en compte la distance de Hamming entre le gène considéré et le symbole reçu. Cette distance est cumulée au fil des itérations et est moyennée de la façon suivante:

$$J_{y^*(k)} = \frac{1}{p^{\gamma}} \sum_{i=k-p+1}^{k} d(y^*(i), r(i)) \tag{18}$$

$y^*(k) \in \mathbb{Y}(k)$ est un gène quelconque de la population courante. La variable p correspond à la durée de vie du gène, c'est à dire le nombre d'itérations où la note verifie le critère de sélection. Le réel γ permet d' ajuster l'importance de la durée de vie par rapport au nombre d'erreurs cumulées.
Une valeur de $\gamma > 1$ favorise la survie des gènes "agés" face à des "gènes" jeunes ayant un nombre d'erreurs cumulées faible.

C. Sélection

La sélection est effectuée en gardant la moitié des gènes de la population. Ceux qui présentent la note la plus élevée disparaissent.

D. Enjambements et mutations

Contrairement à un algorithme génétique classique, où l'on recherche un vecteur qui minimise une fonction multivariables, nous recherchons un vecteur qui est celui de l'état du codeur chaotique. Celui-ci change à chaque iteration, c'est à dire à chaque symbole r(k) reçu.

Lors de la sélection, une moitié de la population a été gardée, une autre écartée. La moitié de la population gardée est simplement mise à jour en incluant dans le vecteur le nouveau symbole r(k) reçu.
Pour la moitiée "gardée", chaque gène est réactualisé selon la formule:

$$\begin{aligned} \underline{\mathbf{z}}(k+1) &= \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \underline{\mathbf{z}}(k) + r(k) \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \\ y(k) &= \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \end{pmatrix} \underline{\mathbf{z}}(k) \end{aligned} \tag{19}$$

Le repeuplement de la moitié vide s'effectue par une copie de la moitié mise à jour à l'exception d'un terme qui subit des mutations. Le terme correspondant au symbole binaire reçu $r(k)$ subit des mutations aléatoires. Le nombre de bits affectés par ces mutations est inférieur à β.

E. Critère d'arrêt et synchronisation

Le gène qui a la meilleure note après un nombre d'itérations fixé est considéré comme le vecteur d'état actuel du codeur chaotique

Dans les simulations proposées, le codeur utilisé est celui décrit en [11]. Les 2 états internes ont été codés sur 8 bits chacun. L'état du codeur à estimer est donc codé sur 16 bits. En choisissant $\beta = 2$, cela conduit à une population de 137 gènes. La figure suivante montre la probabilité de synchronisation en fonction du taux d'erreur binaire. A chaque nouveau symbole reçu les 68 meilleurs gènes en terme de distance de hamming moyennée sont conservés et 69 nouveaux gènes sont créés par enjambement et mutations.

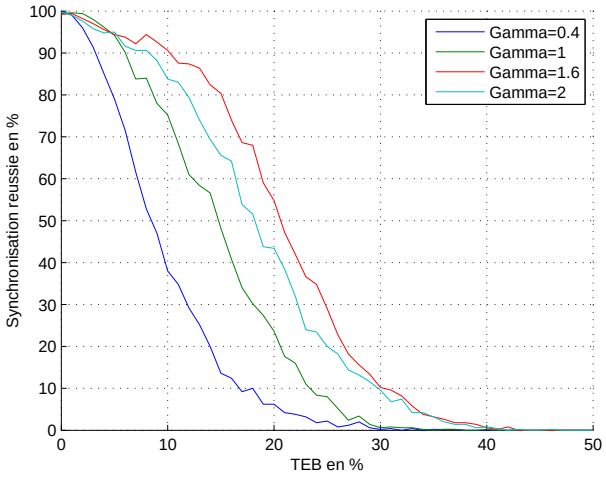

Fig. 8. Evolution de la probabibilté de synchronisation en fonction du taux d'erreurs binaires lors de la transmission. Les simulations ont été réalisées pour differentes valeurs de gamma. Le nombre d'iterations est limité à 20. Les résultats présentés de pourcentage de synchronisation ont été réalisés sur 1000 essais avec des conditions initiales aléatoires.

La figure 8 montre que l'algorithme génétique permet d'estimer l'état courant du codeur aprés 20 itérations. Nous avons mené le calcul sur 20 itérations pour comparer les résultats avec l'algorithme ensembliste. Nous constatons que le facteur d'optimisation γ permet d'accroitre le nombre de synchronisations réussies sur 20 itérations en présence de bruit plus conséquent.

VI. Conclusion

Dans ce papier un moyen de synchronisation en temps réel et en présence de bruit d'un codeur chaotique a été proposé par l'approche ensembliste. Comme nous l'avons démontré expérimentalement, l'approche ensembliste permet, au bout d'une vingtaine d'itérations, d'obtenir un ensemble de candidats potentiels (estimation conditions initiales) assez restreints. Néanmoins, pour une incertitude de 100, le nombre de trajectoires à estimer est de 200*200, ce qui représente une ressource mémoire importante. Une évolution de cet algorithme est présentée sous la forme d'algorithme génétique. Les résultats obtenus en temps de synchronisation sont similaires, mais avec comme avantage une réduction importante de la ressource mémoire (137 gènes). De plus, cette méthode permet la synchronisation, même dans le cas d'un bruit non borné.

References

[1] L. M. Pecora and T. L. Caroll, "Synchronization in chaotic systems," *Phys. Rev. Lett.*, vol. 64, pp. 973–977, 1992.

[2] L.Cong and W. Xiaofu, "Design and realization of an fpga- based generator for chaotic frequency hopping sequences," *IEEE Transactions on circuits and systems-I Fundamental theory and applications*, vol. 48, pp. 521–532, 2001.

[3] G. Mazzini, G. Setti, and R. Rovatti, "Chaotic complex spreading sequences for asynchronous ds-cdma-part i: system modelling and results," *IEEE Trans. Circuits Systems-I: Fundam. Theory Appl.*, vol. 10, pp. 937–947, 1997.

[4] ——, "Chaotic complex spreading sequences for asynchronous ds-cdma-part ii: some theoretical performance bounds," *IEEE Trans. Circuits Systems-I: Fundam. Theory Appl.*, vol. 4, pp. 496–506, 1998.

[5] H. Dedieu and J. Schweizer, “Chaos communication from a maximum likelihood perspective,” in *Proceedings of the 6th Int.Workshop on Nonlinear Dynamics of Electronic Systems , NDES’98, pp. 53-62*, 98.

[6] J. Mangeot, F. Launay, and P. Coirault, “A multi-user transmission using chaotic carriers,” *Commun Nonlinear Sci Numer Simulat*, 2008.

[7] E. Kostelich and J. Yorke, “Noise reduction in dynamical systems,” *Physical Rev.A*, vol. 38, No.3, pp. 1649–1652, 1988.

[8] T. Alamo, J. Bravo, M. Redondo, and E. Camacho, “A set-membership state estimation algorithm based on dc programming,” *Automatica*, vol. 44, pp. 216–224, 2008.

[9] K. Kelber, “N-dimensional uniform probability distribution in nonlinear auto-regressive filter structures,” *IEE Trans.Circuits Syst. I*, vol. 47, pp. 143–1417, 2000.

[10] M. Gotz, K.Kelber, and W. Schwartz, “Discrete-time chaotic encryption systems- part i: statistical design approach,” *IEEE Trans. Circuits Sys. I*, vol. 44, pp. 963–970, 1997.

[11] D. R. Frey, “Chaotic digital encoding : An approach to secure communication,” *IEEE TRans. Circuit & Systs. II :Analog and Digital Signal Processing*, vol. 40, Nř10, pp. 660–666, oct 1993.

[12] J.Holland, *Adaptation in Natural and Artificial Systems*. MIT Press Cambridge, MA, USA, 1975.

[13] M. Y. K. Glette, J. Torresen and Y. Yamaguchi, “On-chip evolution using a soft processor core applied to image recognition,” *proc. of 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS’2006). IEEE Computer Society. , Istanbul, Turkey.*, vol. ISBN 0-7695-2614-4„ pp. pp. 373 – 380., June 2006.

[14] K. Glette and J. Torresen, “A flexible on-chip evolution system implemented on a xilinx virtex-ii pro device,” in *Proc. of Sixth International Conference on Evolvable Hardware (ICES05) , Springer LNCS 3637, pp. 66-75*, September 2005, Sitges, Spain.