SETIT 2007
4th International Conference: **S**ciences of **E**lectronic,
Technologies of **I**nformation and **T**elecommunications
March 25-29, 2007 – TUNISIA

Décodage MVP des Codes Arithmétiques

ZRIBI Amin*, ZAIBI Sonia* et BOUALLEGUE Ammar*

*Laboratoire SYS'COM, ENIT
amin.zribi@gmail.com
sonia.zaibi@enit.rnu.tn
ammar.bouallegue@enit.rnu.tn

Résumé: Dans cet article, un algorithme de décodage conjoint source canal, selon le critère MVP (Maximum de Vraisemblance *a Posteriori*), d'un code arithmétique intégrant la détection d'erreurs est présenté. Ce décodage conjoint est appliqué à un système de transmission d'images fixes sur un canal bruité. L'algorithme sous optimal de recherche séquentielle SA (Stack Algorithm) est utilisé pour réduire l'espace de recherche et déterminer, dans cet espace, la séquence la plus vraisemblable *a posteriori*, qui ne présente pas d'erreur détectée par le décodeur arithmétique. Cet algorithme est appliqué à un système de transmission progressive d'images utilisant le codeur SPIHT. Dans un premier temps, les effets des paramètres internes de l'algorithme présenté sont étudiés. Ensuite, les performances du système étudié sont comparées à celles d'un système de codage classique utilisant le code CRC pour la détection d'erreurs et le code RCPC pour leur correction. En terme de TEP, et dans le cas de décodage pondéré, le système utilisant le décodage conjoint est plus performant que le système classique pour les rapports signal à bruit relativement élevés.
Mots clés: Algorithme SA, Codage arithmétique, Codage Conjoint Source Canal, Codage SPIHT.

INTRODUCTION

La migration au canal sans fil des services basés sur l'Internet, tels que la vidéo conférence et le partage des images et de la musique numérique, entre en collision avec les limitations imposées par l'environnement mobile sur la bande passante et la consommation de la puissance. Par conséquent, la communauté de recherche s'oriente vers des approches interdisciplinaires regroupant la communication numérique et l'expertise de l'outil multimédia pour aboutir à un système présentant une meilleure qualité de service.

Dans ce contexte, les techniques de codage conjoint source canal (CCSC) présentent une intégration naturelle du monde multimédia avec celui de la communication numérique. En effet, dans certains cas, le codeur de source est incapable de décorréler totalement la séquence d'entrée; le flux de données compressé contient encore un résidu de redondance qui peut être exploité pour le contrôle des erreurs par le codeur de canal. Ainsi, on pourra améliorer les performances du décodeur en considérant le codage de source et de canal conjointement.

Dans le domaine du CCSC, une grande attention a été donnée aux codes à longueurs variables (CLV) en raison de leur forte sensibilité aux erreurs. Les standards de codage tels que JPEG2000 et JBIG2 pour les images fixes et H.264 pour les séquences vidéo utilisent le codeur arithmétique (CA) comme codeur entropique. Le CA alloue des nombres fractionnels à la séquence d'entrée, ce qui permet un bon taux de compression [9]. En plus, le codage arithmétique permet des solutions de codage adaptatives efficaces. Mais, le problème est que le CA est très vulnérable aux erreurs de transmission. Dans [4], Boyd et *al.* ont présenté une méthode permettant au codage arithmétique de détecter les erreurs, et ce, par l'introduction d'une région dite « interdite » dans l'espace de codage.

A travers cet article nous allons étudier et implémenter un algorithme de décodage MVP d'un code arithmétique intégrant la détection d'erreurs ; cet algorithme sera appliqué à un système de transmission d'images.

L'implémentation des codes arithmétiques est présentée dans la première section. On mettra l'accent sur la technique d'intégration de la détection d'erreurs dans le processus de codage/décodage arithmétique. Le principe de décodage MVP d'un code arithmétique intégrant la détection d'erreurs, est présenté dans la seconde section. En particulier, les métriques

correspondantes au décodage MVP dans le cas ferme et pondéré sont évaluées. Dans la section 3, une description de l'algorithme SA (Stack Algorithm) pour la recherche séquentielle est donnée, avant de présenter l'algorithme de décodage MVP étudié. Dans la section 4, ce dernier est appliqué à un système de transmission d'images utilisant le codeur SPIHT. Les performances du système, ainsi obtenu, sont évaluées et comparées à celles d'un système de codage classique utilisant un code CRC (Cyclic Redundancy Check) pour la détection des erreurs et un code convolutif poinçonné (RCPC) [10] pour leur correction.

1. Détection d'erreurs par le codage / décodage arithmétique

1.1. Principe du codage arithmétique

L'encodage arithmétique [9] traite l'ensemble d'un message comme une seule entité. Il se base sur la représentation d'un message par un intervalle $I = [Inf, Sup)$ de nombres réels compris entre 0 et 1. A mesure que le message s'allonge, l'intervalle requis pour le représenter diminue, et le nombre de bits qui servent à préciser la largeur de cet intervalle s'accroît. Ainsi, on partitionne l'intervalle [0,1) selon les probabilités des symboles. En itérant ce procédé pour chaque symbole du message, on raffine l'intervalle à un résultat unique représentant le message. Chaque nombre existant dans l'intervalle final sera un code valide. Dans notre étude, nous allons utiliser un modèle statique, et par suite, les probabilités sont calculées après la lecture du message et resteront constantes durant le codage / décodage.

Supposons, que pour un alphabet de taille *N*, notre modèle *M* donne une probabilité $P_M(x_i)$ et une probabilité cumulative $K_M(x_i)$, à chaque symbole x_i existant dans le message, définie par.

$$K_M(x_1) = 0$$
$$K_M(x_k) = \sum_{i=1}^{k-1} P_M(x_i) \text{ pour } 2 \le k \le N \qquad (1)$$
$$K_M(x_{N+1}) = 1$$

Le codage se base alors sur une mise à jour continuelle de l'intervalle *I*, soit des limites *Inf* et *Sup*. Ces limites sont initialisées par *Inf*=0 et *Sup*=1. Puis, pour chaque symbole x_j, nous mettons à jour les bornes de *I* par :

$$Inf = Inf + K_M(x_j)(Sup - Inf)$$
$$Sup = Inf + K_M(x_{j+1})(Sup - Inf) \qquad (2)$$

Pour décoder une séquence *S* de longueur *L* connue, nous appliquons les opérations inverses du codeur. La valeur reçue *V* est comparée avec les limites des intervalles $I_k = [K_M(x_k), K_M(x_{k+1}))$ jusqu'à ce que nous déterminions celui le contenant. Si $K_M(x_i) \le V < K_M(x_{i+1})$, le premier symbole décodé est $\hat{s}_1 = x_i$. Ensuite, la valeur de *V* sera mise à jour, pour le décodage du symbole suivant, par :

$$V = \frac{V - K_M(x_i)}{K_M(x_{i+1}) - K_M(x_i)} \qquad (3)$$

Après *L* itérations de décodage, et en absence d'erreurs de transmission, la séquence *S* sera reconstituée. Dans la figure (1), nous présentons un exemple de codage arithmétique de la séquence binaire 101 dont le modèle est défini par P_0=0.75 et P_1=0.25.

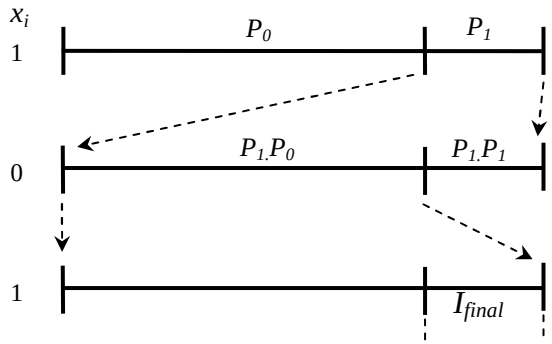

Figure 1. *Codage arithmétique de la séquence 101.*

Le problème de cette méthode est qu'elle manipule des valeurs réelles à grands nombres de fractions ce qui est irréalisable pour la plupart des machines existantes. En 1987, Witten et *al.* [7] ont présenté une implémentation pratique pour les codes arithmétiques, qui se base sur la manipulation de valeurs entières, de registres de longueurs constantes et de techniques de mises en échelles. Cette implémentation a été utilisée pour nos simulations.

1.2. Intégration de la détection d'erreurs dans le codage arithmétique

Il a été démontré que, généralement, les codes arithmétiques offrent la meilleure efficacité de compression. Malheureusement, le prix à payer pour cette compression quasi-optimale est la vulnérabilité aux erreurs de transmission affectant la trame binaire compressée. En effet, une erreur unique dans la trame compressée peut causer la perte de l'ensemble des données situées après l'erreur. Pour surpasser ce problème, C.Boyd et *al.* [4] ont proposé une méthode

permettant d'intégrer un mécanisme de détection d'erreurs dans le processus de codage/décodage arithmétique.

Nous avons vu que le codage arithmétique partitionne l'espace de codage selon la probabilité du symbole à coder, déterminée par le modèle de la source. La redondance est introduite par l'ajustement de l'espace de codage de telle sorte que des parties de cet espace ne soient jamais utilisées par le codeur. Ces parties définissent la région dite ‘‘interdite’’. Durant le décodage, si le nombre défini par la séquence codée entre dans la région ‘‘interdite’’, on est sûr qu'il y a une erreur dans la trame compressée.

Pour ajuster l'espace de codage, la procédure de codage réduit, dans des points particuliers, l'intervalle courant par un certain facteur de réduction noté f_r. Le décodeur applique de son côté la même réduction. La redondance est contrôlée en faisant varier le facteur de réduction. La figure (2) présente l’exemple du partitionnement de l’intervalle [0,1) pour le codage avec considération de la détection des erreurs. On considère les mêmes conditions que dans l’exemple de la figure (1).

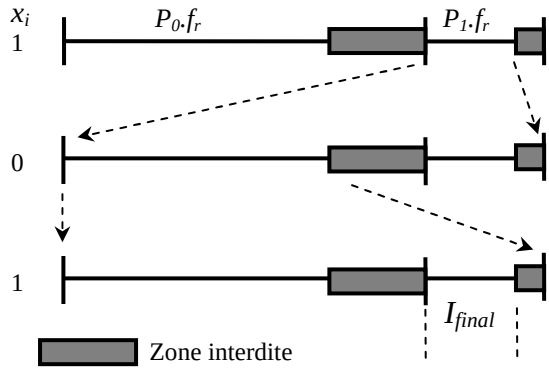

Figure 2. *Codage arithmétique de la séquence 101 avec considération de la détection d’erreurs avec un facteur de réduction f_r=0.25.*

Il est à noter que la détection des erreurs intégrée dans le processus de codage/décodage arithmétique n’est pas instantanée. En effet, le facteur de réduction f_r détermine la rapidité moyenne de détection des erreurs et la quantité de redondance intégrée dans le mot de sortie.

Dans son article [4], Boyd a montré que pour ajouter x% de redondance, le facteur de réduction f_r est donné par:

$$f_r = \frac{1}{1+0.00693x} \tag{4}$$

A titre d'exemple, une redondance x=1% correspond à un facteur de réduction de 0.9931 et une redondance x=2% correspond à f_r=0,9863. Il a été également démontré que le nombre moyen de bits séparant la position d'une erreur de celle de sa détection est égal à $1/(1-f_r)$, soit 145 bits pour 1% de redondance et sa moitié pour 2% de redondance.

Il est à mentionner que la détection des erreurs intégrée dans le codage/décodage arithmétique présente plusieurs avantages. En effet, contrairement aux techniques classiques de détection des erreurs, la quantité de redondance introduite suite à l’intégration de la détection d’erreurs est souplement manipulée par la variation d’un paramètre unique à savoir le facteur de réduction. D’un autre côté, l’introduction de la zone de détection permet la détection des erreurs au fur et à mesure du décodage, alors que les mécanismes classiques de détection d’erreurs tels que le CRC ne permettent la détection qu’après traitement du bloc entier de données.

2. Décodage MVP d’un code arithmétique

Dans la suite, nous considérons la transmission d'un mot de code de longueur variable N_i: $y_i = \{y_{i,0}, y_{i,1}, ..., y_{i,N_i-1}\}$, obtenu par codage arithmétique d'une séquence de données $x_i = \{x_{i,0}, x_{i,1}, ..., x_{i,L-1}\}$, à travers un canal de probabilité de transition $P(r|y_i)$ où $r = \{r_0, r_1, ..., r_{N_i-1}\}$ est la séquence des observations. Notons que le canal inclut la modulation, le milieu de transmission et la démodulation.

L'objectif du décodeur à maximum de vraisemblance *a posteriori* (MVP) est de déterminer la séquence la plus probable x_l vérifiant:

$$P(x_l|r) \geq P(x_k|r), \forall k \neq l \tag{5}$$

Par conséquent, l'estimation se repose sur la maximisation de la métrique suivante:

$$P(x_k|r) = \frac{P(r|x_k)P(x_k)}{P(r)} = \frac{P(r|y_k)P(x_k)}{P(r)} \tag{6}$$

En principe, la tâche de décodage se base sur l'évaluation de la métrique (6) pour toutes les paires (x_k, y_k) pour lesquelles la longueur de la séquence codée y_k est N_i. Dans la suite, nous allons noter par B_{Ni} l'ensemble des mots de code de longueurs N_i.

La métrique (6) fait intervenir la probabilité de

transition du canal, la probabilité *a priori* de la source et le terme $P(r)$. Les deux premiers termes peuvent être évalués à travers le modèle du canal et celui de la source *a priori*. Le dernier terme peut s'écrire sous la forme:

$$P(r) = \sum_{k \in B_{N_i}} P(r|y_k) P(x_k) \quad (7)$$

Il est clair que l'évaluation de ce terme est relativement complexe. En effet, son calcul demande la connaissance de tout l'ensemble B_{Ni}, ce qui est impossible pour des valeurs de N_i utilisées dans la pratique. Pour cela, nous aurons besoin de quelques approximations pour le calcul de $P(r)$.

Dans le cas du canal sans mémoire, il est plus intéressant d'exprimer la métrique de décodage sous une forme additive:

$$m_k = \log[P(x_k|r)] = \sum_{j=0}^{N_i-1} \underbrace{\log[P(r_j|y_{k,j})] + \log[P(x_{k,j})] - \log[P(r_j)]}_{m_{k,j}} \quad (8)$$

Le terme $P(x_{k,j})$ représente la probabilité *a priori* des symboles sources à la sortie du décodeur arithmétique associés au bit *j* du mot code y_k. Il est à noter que le nombre de symboles sources décodés est variable et dépend à la fois du mot de code y_k et de la position *j* du bit (il est possible de n'avoir aucun symbole décodé).

La valeur de la métrique additive $m_{k,j}$ peut être employée comme étant la métrique d'une branche utilisée par une technique de recherche séquentielle permettant de déterminer la meilleure estimation. Dans la suite, nous allons évaluer la métrique par branche, pour les deux types de décodage ferme et pondéré, dans le cas d'un canal gaussien.

2.1. Cas d'un décodage ferme

Pour une transmission sur un canal gaussien à bruit additif, et avec une modulation BPSK (Binary Phase Shift Keying) et un rapport signal à bruit E_s / N_0, la probabilité de transition du canal dans le cas ferme est:

$$P(r_j|y_{k,j}) = 1 - p, \text{ si } y_{k,j} = r_j$$
$$P(r_j|y_{k,j}) = p \quad , \text{ si } y_{k,j} \neq r_j \quad (9)$$

avec $p = 0.5 erfc(\sqrt{E_s / N_0})$.

Pour l'évaluation de la valeur de $P(r)$, on adopte l'approximation suivante: $P(r) = 2^{-N_i}$. Cette approximation suppose que les 2^{N_i} séquences possibles r de longueur N_i sont équiprobables. Cette hypothèse n'est généralement pas vérifiée à cause du caractère ''longueur variable'' des codes arithmétiques. En effet, toutes les séquences possibles de longueurs N_i ne correspondent pas forcément à des mots codes valides pour un codeur arithmétique. Pourtant, cette approximation a donné des résultats satisfaisants pour le décodage MVP des codes à longueurs variables [12]. En conclusion, et sous cette approximation, la métrique de la branche devient:

$$m_{k,j} = \log[P(r_j|y_{k,j})] + \log[P(x_{k,j})] + \log[2] \quad (10)$$

2.2. Cas d'un décodage pondéré

Pour une modulation BPSK avec un décodage pondéré, l'estimateur au sens du critère MVP observe à la sortie du canal les valeurs $r_j = h_{k,j} + n_j$, $j = 0...(N_i - 1)$, où $h_{k,j} = \sqrt{E_s}(2y_{k,j} - 1)$ est le symbole associé par le modulateur au bit $y_{k,j}$ et n_j est un échantillon du bruit gaussien de moyenne nulle et de variance σ^2. Ce modèle de canal donne une probabilité de transition égale à:

$$\log[P(r_j|y_{k,j})] = -\frac{1}{2}\log[2\pi\sigma^2] - \frac{(r_j - h_{k,j})^2}{2\sigma^2} \quad (11)$$

En intégrant les probabilités conditionnelles, on peut écrire $P(r_j) = \sum_{k=0}^{1} P(r_j|y_j = k) P(y_j = k)$, où y_j est le $j^{\text{ème}}$ bit du mot code transmis. Il est supposé prendre les valeurs 0 ou 1 avec la même probabilité. Par conséquent, $P(y_j = 0) = P(y_j = 1) = 0.5$.

D'autre part, la formule $r_j = \sqrt{E_s}(2y_{k,j} - 1) + n_j$ postule que la variable aléatoire $(r_j|y_j = 0)$ suit une loi gaussienne $\mathcal{N}(-\sqrt{E_s}, \sigma^2)$ et $(r_j|y_j = 1)$ suit une loi gaussienne $\mathcal{N}(\sqrt{E_s}, \sigma^2)$, et par suite:

$$\log[P(r_j)] = -\log[2] - \frac{1}{2}\log[2\pi\sigma^2] - \frac{(r_j + \sqrt{E_s})^2}{2\sigma^2} + \log[1 + \exp(2r_j \frac{\sqrt{E_s}}{\sigma^2})] \quad (12)$$

Ainsi, et dans le cas de décodage pondéré, la métrique par branche est donnée par:

$$m_{k,j} = -\frac{(r_j - h_{k,j})^2}{2\sigma^2} + \log[P(x_{k,j})] + \log[2] + \frac{(r_j + \sqrt{E_s})^2}{2\sigma^2} - \log[1 + \exp(2r_j \frac{\sqrt{E_s}}{\sigma^2})] \quad (13)$$

3. Algorithme de décodage MVP des codes arithmétiques

Après la définition des métriques par branche dans le cas de décodage ferme (10) et pondéré (13), nous allons voir comment les intégrer dans le processus de décodage. Il est clair qu'il n'est pas évident de calculer les métriques de tous les chemins appartenant à l'ensemble B_{Ni} des mots de code de longueurs N_i. Le premier problème est qu'il n'est pas logique de stocker la totalité des 2^L mots codes y_k et d'en dégager ceux appartenant à l'ensemble B_{Ni}. D'autre part, la large cardinalité de l'espace B_{Ni} rend la tâche du calcul des métriques MVP très difficile. Par conséquent, on a recours à des techniques de recherche sous optimales permettant de parcourir B_{Ni} et d'en dégager la séquence la plus probable *a posteriori*. Le premier problème peut être surmonté grâce au pouvoir de détection d'erreurs intégré dans le code arithmétique. En effet, ce critère nous permet de raffiner l'espace de recherche de B_{Ni} à $X_{N_i} \subset B_{N_i}$ en éliminant tous les mots de code déclarés erronés suite à une détection d'erreur par le décodeur arithmétique.

Le problème de la large cardinalité de l'espace de recherche peut être surmonté par l'utilisation d'algorithmes de recherche séquentielle sous optimaux. Dans notre étude, nous avons opté pour l'utilisation de l'algorithme ''Stack Algorithm'' (SA) [8] pour la recherche du mot le plus vraisemblable dans l'arbre binaire.

Dans notre implémentation de l'algorithme SA, nous utilisons comme critère de contrôle la détection des erreurs de transmission intégrée dans le codage arithmétique. Le critère de terminaison est le suivant: le meilleur chemin a une longueur égale à celle du mot de code transmis N_i (supposée connue par le décodeur) et se décode arithmétiquement sans erreur détectée. Dans la figure (3), nous présentons le déroulement de cet algorithme de décodage présenté par M.Grangetto [13] dans le cas binaire et désigné par décodeur « CA-MVP ».

L'algorithme présenté se base sur la mise à jour d'une liste contenant au maximum M chemins ordonnés. Chaque chemin est défini par sa longueur courante ($\leq N_i$), sa métrique cumulée et le mot de code qu'il définit. A chaque itération, le meilleur chemin est étendu d'une branche puis supprimé de la liste. Dans le cas binaire, nous avons deux nouveaux chemins correspondants respectivement aux branches 0 et 1. Pour chaque chemin, nous appliquons un décodage arithmétique avec détection d'erreurs de facteur de réduction f_r. Si une erreur est détectée, le chemin est ignoré. Dans le cas contraire, nous calculons la métrique cumulée correspondante, et le chemin est inséré dans la liste en respectant l'ordre des métriques.

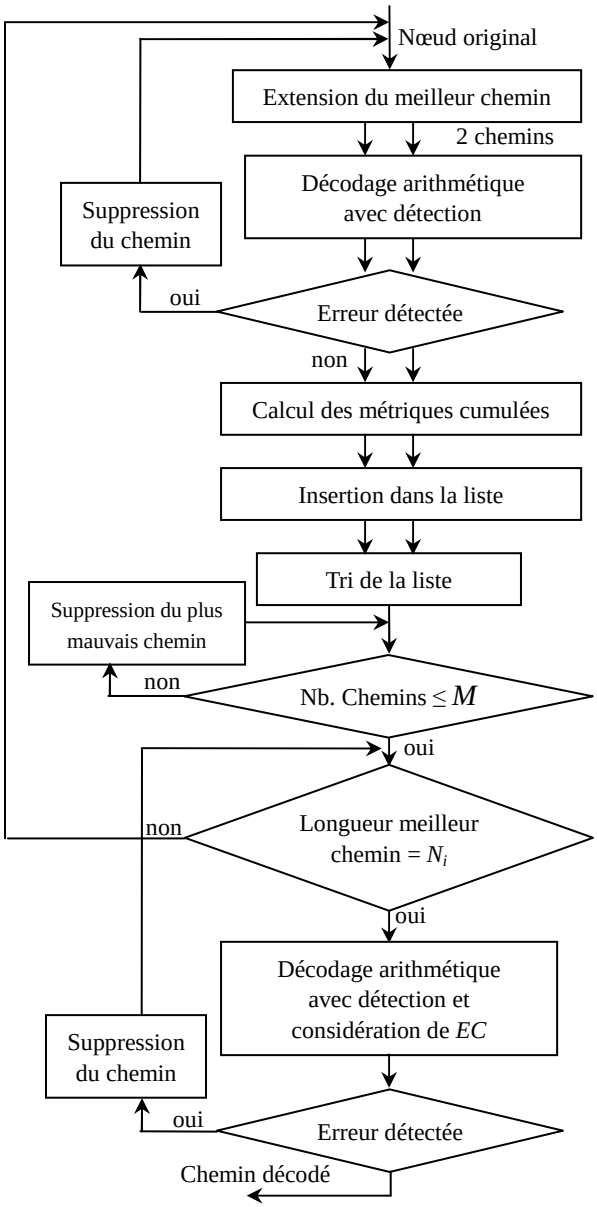

Figure 3. *Algorithme de décodage MVP des codes arithmétiques intégrant la détection d'erreurs : Décodeur « CA-MVP »*

Le critère de terminaison est vérifié à la fin de chaque itération. En effet, si le meilleur chemin de la liste correspond à une longueur N_i et se décode correctement, on quitte l'algorithme avec la séquence décodée comme sortie. Dans le cas contraire, le chemin est supprimé de la liste et le même contrôle est appliqué au chemin suivant. On note ici que, lors de la vérification du critère de terminaison, le décodage arithmétique prend en compte un symbole spécial « *EC* » (*Error Check*) de probabilité ω. En effet, ce symbole est codé à la fin (après le dernier symbole de la séquence) avec un modèle modifié de probabilités ω pour le symbole *EC* et (1-ω) pour (!*EC*) (la fonction !x désigne le complémentaire de x). Le décodeur arithmétique utilise ce modèle lorsque la longueur de la séquence atteint $(N_i\text{-}1)$. Si le symbole (!*EC*) est décodé, le chemin considéré est erroné et doit être supprimé de la liste [4].

A la fin de chaque itération, nous devons également contrôler le nombre de chemins stockés dans la liste; s'il dépasse M, le plus mauvais chemin en terme de métrique cumulée est supprimé. Une erreur de décodage est déclarée lorsque la liste ne contient plus de chemins (tous les chemins ont été supprimés suite à une détection d'erreur par décodage arithmétique).

4. Application à la transmission d'images fixes

Pour mettre en évidence l'efficacité de l'algorithme étudié, nous allons considérer le contexte d'une transmission d'images fixes. Nous allons, pour cela, utiliser un système de transmission d'images se basant sur l'algorithme SPIHT [11] pour la compression des images. Un codeur arithmétique intégrant la détection d'erreurs sera appliqué pour la protection des données compressées. Du côté du récepteur, nous allons appliquer le schéma de décodage MVP des codes arithmétiques décrit dans la section 3.

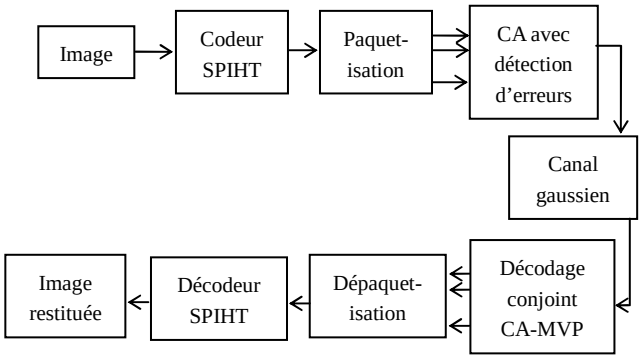

Figure 4. *Système de transmission d'images intégrant le décodeur CA-MVP*

Le diagramme en blocs du système de transmission d'images proposé est donné par la figure (4). L'image à transmettre est compressée par l'algorithme SPIHT. Notons D_{SPIHT}(bpp) le débit à la sortie du codeur SPIHT. Ce dernier fournit une séquence binaire qui présente un modèle de probabilité presque équiprobable ($P_0 \approx P_1$). Par conséquent, le codeur arithmétique, appliqué ensuite, n'apporte pas une amélioration significative de la compression. Il est plutôt utilisé pour la détection et la correction des erreurs de transmission. Par ailleurs, le terme relatif aux probabilités *a priori* de la source dans les expressions des métriques (10) et (13) est ignoré en raison de la quasi-équiprobabilité des symboles 0 et 1 à l'entrée du codeur arithmétique.

Le train binaire de longueur $N_{image}.N_{image}.D_{SPIHT}$, obtenu par la compression SPIHT d'une image de taille $N_{image}\times N_{image}$, est partitionné en paquets de longueur L bits chacun. Chaque paquet est codé arithmétiquement, en utilisant un modèle statique d'ordre 0 et un facteur de réduction f_r pour la détection des erreurs de transmission. Après le codage du dernier symbole de la séquence, le symbole *EC* est codé selon la méthode décrite dans la section (3). Les mots de code obtenus auront des longueurs variables N_i. Le rendement du code arithmétique R_{CA} sera alors donné par le rapport de L par la moyenne des N_i. Un tel système génère un débit total de D_{SPIHT}/R_{CA} (bpp) à l'entrée du canal supposé gaussien. Au niveau du récepteur, on applique l'algorithme de décodage conjoint « CA-MVP » utilisant le critère MVP et la détection des erreurs intégrée dans le décodage arithmétique. On arrête le décodage au premier paquet déclaré erroné puisque les paquets suivants sont généralement incapables d'améliorer la qualité de l'image décodée [14]. Cette dernière est alors obtenue par décodage SPIHT des paquets qui ont été correctement décodés (selon le décodeur arithmétique). Dans toutes nos simulations, l'image Lenna de taille 512×512 pixels (8 bpp), est considérée comme image de test.

4.1. Effets de *M* et f_r sur les performances

4.1.1. *Effet de la longueur de la liste M*

Dans cette partie, nous fixons un rapport signal à bruit $E_s / N_0 = 5$ dB. Le débit à la sortie du codeur SPIHT est tel que D_{SPIHT}=0.5 bpp et par suite la trame compressé contient 131072 bits divisés en 256 paquets de longueur 512 bits chacun. Pour la détection des erreurs dans le codage arithmétique, nous avons considéré un facteur de réduction f_r=0.95, résultant en un rendement R_{CA}=0.91. Le symbole *EC* présente une probabilité ω=6.10^{-5}. Avec un tel système, le débit

total est égal à 0,54bpp. Notons que nous considérons le cas de décodage pondéré ; les données à la sortie du canal gaussien ne sont pas seuillées et la métrique utilisée par branche est celle donnée par la relation (13). Dans la figure (5), nous présentons les taux d'erreur par paquet TEP obtenus pour différentes valeurs de *M*.

Nous pouvons constater que plus *M* est important, plus le TEP est faible. Ceci est tout à fait prévisible, en effet, l'algorithme SA compare des chemins de différentes longueurs, et par suite, favorise les chemins de longueurs faibles (métriques cumulées plus grandes). Plus *M* est petit, plus la probabilité de perdre le chemin correct est importante et par suite plus le nombre de paquets erronés est grand. Notons également qu'au delas de *M*=3072, l'amélioration induite par l'augmentation de *M* se réduit remarquablement et le TEP se stabilise.

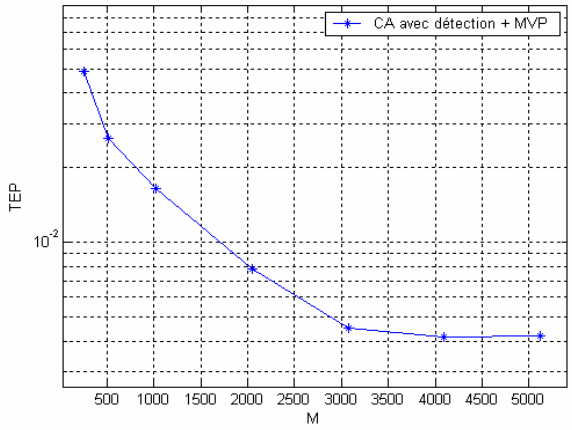

Figure 5. *Variation du TEP en fonction de la taille de la liste M*

4.1.2. *Effet du facteur de réduction f_r*

Comme nous l'avons déjà mentionné, le facteur de réduction f_r influe sur le délai de la détection intégrée dans le codeur arithmétique. Donc, plus f_r est important, plus l'erreur est rapidement détectée et plus on gagne de la place dans la liste. Nous avons déterminé le TEP induit par différentes valeurs de f_r. Pour cela, nous avons considéré les mêmes conditions de la simulation précédente en fixant *M*=4096. La courbe obtenue est représentée dans la figure (6).

On remarque que la diminution du facteur de réduction améliore les performances du décodeur. Mais, le gain obtenu en TEP diminue lui aussi lorsque f_r décroît. En effet, le délai de détection des erreurs diminue quand f_r augmente, mais à un certain moment ce délai se stabilise et par suite les performances ne s'améliorent plus.

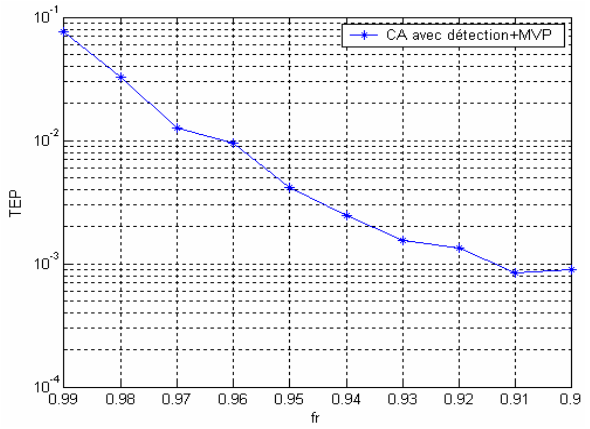

Figure 6. *Variation du TEP en fonction du facteur de réduction f_r*

4.2. Comparaison à un système de codage classique "CRC-RCPC"

Dans la suite, nous allons comparer les performances du système étudié à celles d'un système classique utilisant un code CRC et un code RCPC. Le code CRC utilise 16 bits de redondance pour la détection des erreurs de transmission. Le RCPC est extrait du code convolutif (133,171,145) de rendement 1/3 et de longueur de contrainte ν=7. Le rendement du code RCPC est pris égal à R_{RCPC}=8/9. La matrice de poinçonnage correspondante est présentée dans l'article de Hagenauer [10]. Le débit à la sortie du codeur SPIHT est fixé à 0.5bpp. Il en résulte que le débit total à l'entrée du canal est égal à 0.58bpp. Le système ainsi obtenu sera désigné par ''SPIHT-CRC-RCPC''.

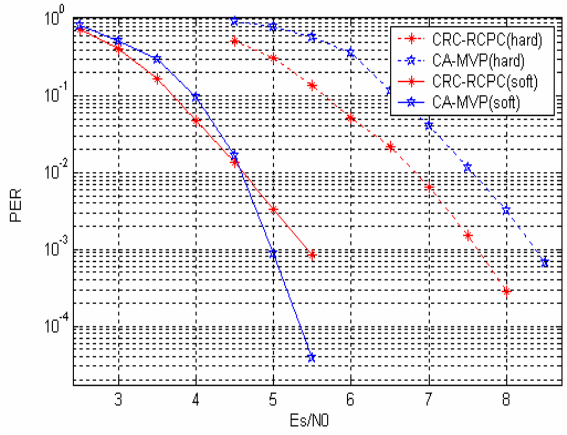

Figure 7. *Comparaison des performances en terme de PER des systèmes "SPIHT-CA-MVP" et "SPIHT-CRC-RCPC"*

Par ailleurs, nous désignerons par ''SPIHT-CA-MVP'' le système qui fait l'objet de notre étude (figure (4)). Nous traiterons le cas de décodage ferme et pondéré. Un atout du système ''SPIHT-CA-MVP'' réside dans la souplesse dans le réglage du rendement du code.

Pour aboutir à un débit total de 0.58bpp, nous choisissons f_r=0.905 qui engendre un rendement R_{CA}=0.85. Sous ces conditions, la simulation de la chaîne ‘‘SPIHT-CA-MVP’’ pour M=4096, dans les cas de décodage ferme et pondéré donne les courbes de la figure (7). Dans ces courbes, nous avons présenté les performances en termes de TEP en fonction de E_s/N_0.

Nous remarquons que dans le cas de décodage ferme, le système de référence ‘‘SPIHT-CRC-RCPC’’ est plus performant que le système étudié (écart de 0.75 dB en E_s/N_0 pour un TEP de l'ordre de 10^{-3}). Donc, il n'est pas recommandé d'utiliser le système ‘‘SPIHT-CA-MVP’’ dans le cas de décodage ferme. Dans le cas de décodage pondéré, à partir d'un rapport signal à bruit égal à 4.5 dB notre système ‘‘SPIHT-CA-MVP’’ devient plus performant que le système de référence ‘‘SPIHT-CRC-RCPC’’. Le gain atteint 0.5dB pour un taux d'erreur par paquet de l'ordre de 10^{-3}.

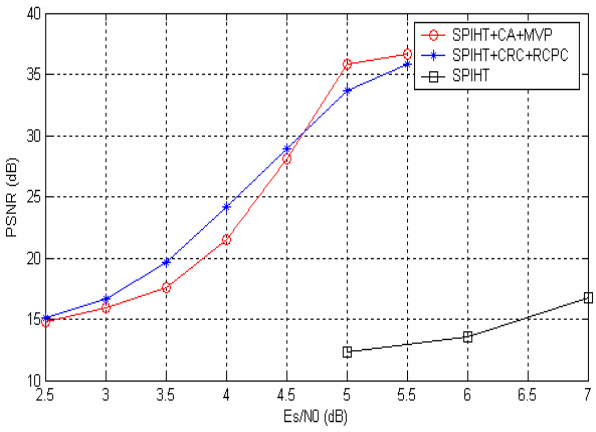

Figure 8. *Comparaison des performances en terme de PSNR moyen des systèmes “SPIHT-CA-MVP” et “SPIHT-CRC-RCPC”*

Nous avons, par ailleurs, dégagé les performances de ces systèmes en terme de PSNR moyen. Nous avons également représenté, dans la figure (8), les résultats obtenus pour un système de compression d'images SPIHT sans codage de canal. Notons que nous nous sommes toujours fixés le même débit total sur le canal, à savoir 0.58bpp. Il est clair qu'un tel système (SPIHT sans codage de canal) est très vulnérable aux erreurs. Nous constatons également que le système ‘‘SPIHT-CRC-RCPC’’ est meilleur que le système ‘‘SPIHT-CA-MVP’’ pour des PSNR moyens inférieurs à 30dB. Or, nous savons que de nombreuses applications de transmission d'images ne tolèrent pas une telle dégradation de l'image décodée. Pour ces applications, il est plus judicieux de considérer le système ‘‘SPIHT-CA-MVP’’, puisqu'il offre un gain significatif dans la région des PSNR moyens supérieurs à 30dB (bonne qualité moyenne de l'image décodé). En effet, pour un rapport signal à bruit de 5dB, le système ‘‘SPIHT-CA-MVP’’ conduit à un PSNR moyen de 35.5dB, alors que le système ‘‘SPIHT-CRC-RCPC’’ donne un PSNR moyen de 33dB, soit un gain en PSNR moyen de 2.5dB.

5. Conclusions

A travers cet article, nous avons étudié un système de codage conjoint source canal se basant sur un décodage MVP fonctionnant conjointement avec un décodeur arithmétique intégrant le détection d'erreurs. L'objectif de cette étude est d'améliorer la qualité globale d'une chaîne de transmission d'images et de minimiser l'impact des perturbations liées à la transmission sur la qualité des images reconstruites.

Le système de transmission d'images proposé utilise l’algorithme SPIHT pour la compression d’images et un code arithmétique intégrant la détection d’erreurs pour la protection des données compressées. Ce dernier est décodé à l’aide d’un algorithme sous optimal de recherche séquentielle fonctionnant selon le critère MVP. Les résultats des simulations ont montré que dans le cas de décodage pondéré et pour des rapports signal à bruit relativement élevés, les performances du système proposé sont meilleures que celles d’un système de codage classique, utilisant un code CRC pour la détection des erreurs et un code RCPC pour la correction.

Comme perspectives, nous nous proposons le développement d’un algorithme de décodage itératif pour les codes arithmétiques. En effet, pour cela on pourra utiliser l’algorithme SOVA (Soft Output Viterbi Algorithm) permettant au décodeur MVP de générer des sorties pondérées.

REFERENCES

[1] A.J.Viterbi, ‘‘Error bounds for convolutional codes and an asymptotically optimum decoding algorithm’’. IEEE trans. Inform. Theory, April 1967.

[2] A.Said, and W.A.Pearlman ‘‘A new fast and efficient image codec based on set partitioning in hierarchical trees’’. IEEE Trans. on Circuits and Systems for Video technology, June 1996.

[3] B.D.Pettijohn, M.W.Hoffman, and K.Sayood ‘‘Joint source channel coding using arithmetic codes’’. IEEE Trans. Commun., May 2001 .

[4] C.Boyd, J.Cleary, S.Irvine, I.Rinsma-Melchert, and I.Witten, ‘‘Integrating error detection into arithmetic coding’’. IEEE Trans. Commun., Jan. 1997.

[5] C.E. Shannon, ‘‘A mathematical theory of communication’’. *The Bell system technical journal*, vol. 27,

Juillet-Octobre 1948.

[6] E.Bodden, M.Clasen and J.Kneis ‘‘Arithmetic coding revealed: A guided tour from theory to practice’’. Seminar Data Compresion, May 2004.

[7] I.H.Witten, J.G.Cleary, and R.Neal, ‘‘Arithmetic coding for data compression’’. Commun. ACM, 30(6):520-540, June 1987.

[8] J.B.Anderson, and S.Mohan ‘‘Sequential coding algorithms: a survey and cost analysis’’. IEEE Trans. Commun., February 1984.

[9] J.J.Rissanen and G.G.Langdon, ‘‘*Arithmetic coding*’’. IBM J. Res. & Dev, 1979.

[10] J.Hagenauer ‘‘Rate-compatible punctured convolutional codes (RCPC codes) and their applications’’. IEEE Trans. Commun., April 1988.

[11] J.M.Shapiro, ‘‘Embedded image coding using zerotrees of wavelet coefficients’’. IEEE Trans. on signal processing, December 1993 .

[12] K.Sayood, H.Otu, and N.Demir ‘‘Joint source channel coding for variable length codes’’. IEEE Trans. Commun. , January 2000.

[13] M.Grangetto, P.Cosman, and G.Olmo ‘‘Joint source channel coding and MAP decoding of arithmetic codes’’. IEEE Trans. Commun., June 2005.

[14] S.Zaibi ‘‘Optimisation conjointe du codage (décodage) source canal pour la transmission d'images’’. Thèse de doctorat à l'ENST Bretagne, Fevrier 2004.