Méthode de décodage robuste pour la transmission de flux html comprimé via un lien mobile bruité

Zied Jaoua[+*], Anissa Mokraoui-Zergaïnoh[+], Pierre Duhamel[*]

[*]LSS/CNRS, SUPELEC,
Plateau de Moulon, 91 192 Gif sur Yvette, France

[+]L2TI, Institut Galilée, Université Paris 13,
99, Avenue Jean Baptiste Clément, 93 430 Villetaneuse, France

{jaoua, duhamel} @lss.supelec.fr ; {anissa.mokraoui@galilee.univ-paris13.fr}

Résumé

Cet article s'intéresse au problème de la correction des erreurs survenues lors de la transmission de pages html comprimées via un lien mobile bruité. La méthode proposée est basée sur une approche de décodage conjoint source canal. Le récepteur développé est construit sur le principe turbo de décodage itératif de codes concaténés en série. Le décodeur extérieur s'appuie sur l'algorithme classique de décodage séquentiel à M-chemins. Ce dernier, pour améliorer ses performances a été modifié de façon à ce qu'il exploite les règles grammaticales et syntaxiques liées (i) aux codes deflates spécifiés par le dernier protocole d'échanges de fichiers, et (ii) au langage html. Les résultats de simulation montrent que notre récepteur améliore significativement la qualité de la protection des pages html transmises comparée à un décodage canal classique.

Mots clefs

Lempel-Ziv, Huffman, deflate, html, décodage séquentiel à M-chemins, Turbo-codes.

1 Introduction

Dans la nouvelle version du protocole d'échanges de fichiers html (http1.1) entre un client (mobile) et un serveur (dans le réseau fixe), une nouvelle fonctionnalité y est proposée ([1]). Celle-ci permet de comprimer les fichiers html selon l'algorithme de codage entropique deflate ([2]). Le flux deflate est ensuite encapsulé selon le format Gzip ([3]) ou Zip ([4]) avant d'être transmis ou téléchargé.

Cet article s'intéresse au décodage robuste de telles pages html (transmises ou téléchargées via un lien mobile bruité). Pour lutter contre la propagation d'erreurs au niveau du décodeur, généralement les méthodes classiques de communications ajoutent de la redondance dans le train binaire à transmettre. Ce procédé se traduit malheureusement par une forte réduction du débit de transmission. Afin de contourner ce problème de réduction de débit, nous proposons plutôt d'orienter notre travail vers une méthode de décodage conjoint source canal. Dans cette thématique de décodage conjoint source canal, un seul travail a été développé (à notre connaissance) pour robustifier les codes Lempel-Ziv-77 (LZ-77) ([5]). Les auteurs de cette référence proposent une structuration des données LZ-77 combinée à des codes Reed Solomon.

Le problème de robustification des codes deflates s'avère complexe comparé au problème que nous avions déjà résolu, où nous ne tenions compte que des codes LZ-77 ([6], [7]). En effet, les codes deflates correspondent à un empilement de deux codes entropiques : Lempel-Ziv et Huffman. Une recherche poussée sur le décodage conjoint source canal nous a amené à nous orienter vers des techniques de décodage souple adaptées aux codes deflates. Des investigations ont porté sur la question de la redondance de ces codes imbriqués de façon à pouvoir l'exploiter efficacement au niveau du décodeur. Nous avons proposé un schéma robuste basé sur l'algorithme de décodage séquentiel à *M*-chemins que nous avons adapté à notre problématique. Celui-ci est construit de façon à tenir compte des règles grammaticales et syntaxiques (i) des codes Lempel Ziv ; (ii) des codes d'Huffman ; et (iii) du langage html.

Cet article est organisé comme suit. La section 2 introduit les codes entropiques de type deflate. La section 3 présente les différentes modifications que nous avons apportées à l'algorithme générique de décodage séquentiel à M-chemins. La section 4 présente le récepteur itératif développé. Les résultats de simulation sont illustrés en section 5. La dernière section conclut notre travail.

2 Présentation des codes deflates

Les fichiers html à transmettre ou à télécharger sont comprimés selon les spécifications de la norme deflate ([2]). Celle-ci combine deux algorithmes de codage entropique : (i) une variante de l'algorithme Lempel-Ziv; et (ii) l'algorithme d'Huffman. Deflate propose trois modes de fonctionnement. Le premier mode correspond uniquement à la segmentation d'un fichier de taille importante sur différent support de stockage. Le deuxième mode spécifie que le codage de Huffman, appliqué aux

codes LZ, est réalisé par le biais de tables statiques prédéfinies dans deflate. Ce mode permet d'obtenir une compression et décompression rapide. De plus les tables ne sont pas transmises dans le flux comprimé. Tandis que le troisième mode comprime le train LZ par le biais de tables d'Huffman construites de manière dynamique. Ces tables doivent être à leur tour transmises dans le flux compressé. Le deuxième mode est plus rapide et moins complexe que le troisième mode. Nous nous intéresserons dans cet article au deuxième mode.

Le texte html à comprimer, selon l'algorithme LZ, est analysé séquentiellement de gauche à droite. L'algorithme est basé sur le principe de la construction adaptative d'un dictionnaire de taille initialement fixée par le standard deflate à 32 K-octets ([2]). Introduisons tout d'abord quelques notations.

Notons par T le texte html composé de n caractères consécutifs. Le i-ème caractère dans T est noté $T[i]$. $T[i,j]$ représente la phrase composée par l'ensemble des caractères $T[i]T[i+1]T[i+2]..T[j]$. Supposons que les $i-1$ caractères ont déjà été analysés pour construire $h-1$ phrases. Ces phrases constituent le dictionnaire noté $T[1,i-1]=s_1 s_2 ... s_{h-1}$, où s_k représente la k-ème phrase.

A une étape donnée i, l'algorithme LZ cherche dans le dictionnaire (en l'occurrence dans $T[1,i-1]$) la plus longue h-ème chaîne de caractères qui serait identique à celle disponible dans $T[i,i+l_h-1]$ avec $l_h \le L$ où L représente la taille de la fenêtre de recherche fixée à 256 octets. La chaîne retenue est codée par la paire LZ (ou symbole LZ) notée $\langle l_i, p_i \rangle$, où l_i représente la longueur de la nouvelle chaîne à inclure dans le dictionnaire et p_i le pointeur vers le dictionnaire indiquant le début de la chaîne à coder. C'est cette sous chaîne qui vient mettre à jour le dictionnaire.

Dans le cas où aucune mise en correspondance de chaîne n'a été trouvée, l'algorithme transmet le caractère c_i correspondant à $T[i]$. Celui-ci est ensuite introduit dans le dictionnaire.

Le texte ainsi codé par l'algorithme LZ est représenté par une suite de symboles LZ : $\langle c_0 \rangle, \langle c_1 \rangle, \langle l_0, p_0 \rangle, ..., \langle l_i, p_i \rangle, ...$ Les mots p_i, l_i et c_i sont alors respectivement codés sur 15 bits ($p_i = p_i^{14} p_i^{13} ... p_i^0$), 8 bits ($l_i = l_i^7 l_i^6 ... l_i^0$) et 8 bits ($c_i = c_i^7 c_i^6 ... c_i^0$). Il a été montré, pour ce choix particulier de paramètres (i. e. taille du dictionnaire, taille de la fenêtre de recherche), que la longueur moyenne de ce code est aussi proche du seuil de l'entropie de la source. Les codes LZ sont à leur tour codés par l'algorithme d'Huffman en s'appuyant sur les trois tables fournies par la norme defltate (deuxième mode, [2]).

3 Décodage séquentiel des codes deflates

Le récepteur proposé est construit sur le principe turbo de décodage itératif de code concaténé en série. Avant de présenter notre récepteur, rappelons tout d'abord le principe de l'algorithme classique de décodage séquentiel à M-chemins sur lequel s'appuie notre méthode robuste de décodage.

3.1 Algorithme générique de décodage séquentiel à M-chemins

L'algorithme de décodage séquentiel à M-chemins, utilise une structure en arbre où il ne mémorise que les M meilleurs chemins, selon une métrique, à chaque profondeur de l'arbre. L'algorithme étend les M-chemins mémorisés, en les prolongeant à l'aide de l'arbre décrivant le problème, met à jour les valeurs de vraisemblance (métriques) correspondant à chacune de ces extensions, et mémorise à nouveau les M meilleurs chemins retenus.

L'algorithme cherche à maximiser la vraisemblance entre les symboles reçus ($\boldsymbol{y} = (y_1,...,y_t.,..y_N)$), et ceux émis ($\boldsymbol{x} = (x_1,...,x_t.,..x_N)$) à l'entrée du canal. Le message estimé au sens du maximum de vraisemblance est alors donné par :

$$\hat{\boldsymbol{x}} = \arg\max_x \log(P(y/x)) \qquad (1)$$

où $P(y/x)$ est la vraisemblance définie comme suit pour un canal Gaussien :

$$P(y/x) = \prod_{k=1}^{N} P(y_k/x_k) = \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_k - x_k)^2}{2\sigma^2} \right) \qquad (2)$$

où σ^2 représente la variance du bruit blanc gaussien centré.

Le message estimé est alors donné par l'équation ci-dessous:

$$\hat{\boldsymbol{x}} = \arg\min_x \sum_{k=1}^{N} (y_k - x_k)^2 \qquad (3)$$

où $\mu(N) = \sum_{k=1}^{N} (y_k - x_k)^2$ correspond à la métrique utilisée pour déterminer le chemin le plus probable.

L'algorithme de décodage séquentiel adopte une recherche en largeur dans l'arbre (Breadth first). Les principales étapes suivies par cet algorithme classique sont listées ci-dessous :

1. Lire une information reçue y_k ;
2. Prolonger tous les M noeuds courants associés à cette information y_k ;
3. Calculer les métriques des branches prolongées ;
4. Trier les $2M$ nouveaux noeuds selon les métriques cumulées calculées en 3 ;
5. Choisir parmi les $2M$ chemins, les M meilleurs chemins selon les métriques cumulées des branches ;
6. Recommencer à partir de 1 jusqu'à atteindre la profondeur finale de l'arbre (lecture de la dernière information y_N ;

3.2 Algorithme de décodage séquentiel à M-chemins modifié

La difficulté du problème réside dans le fait que nous avons à traiter un empilement de deux codes entropiques utilisés pour comprimer un texte html. Nous sommes confrontés à un problème de robustification de codes à longueur variable. Afin d'améliorer les performances de l'algorithme générique de décodage à M chemins et de réduire sa complexité de calcul, nous imposons à cet algorithme un ensemble de règles à satisfaire lors du

décodage de la séquence transmise. Ces règles exploitent les règles grammaticales et syntaxiques (i) des codes LZ ; (ii) des codes d'Huffman appliqués aux codes LZ et (iii) du langage html.
Parmi les chemins retenus selon le critère des métriques, l'algorithme développé, complète la vérification de la validité de ces chemins en s'appuyant sur l'ensemble des règles que nous avons prédéfinies. La décision ne peut être prise qu'après avoir lu tous les éléments d'un même mot. Si ces règles ne sont pas satisfaites par le décodage de certaines branches, dans ce cas l'algorithme élague les branches correspondantes de l'arbre.

3.2.1 Validités des règles grammaticales et syntaxiques des codes deflates et de la syntaxe html

Les règles syntaxiques et grammaticales à vérifier lors du décodage séquentiel sont listées ci-dessous. Nous commençons tout d'abord par les règles associées aux codes d'Huffman puis aux codes LZ et nous terminerons enfin par celles associées au langage html.

• Validité des codes de Huffman :

Rappelons que les codes deflates ainsi construits sont des codes à longueur variable. Dans un premier temps, l'analyse des différentes tables de codage décrites dans [2] nous a amené à conclure sur les longueurs des codes deflates associés aux codes LZ (voir Tableau 1). Cette analyse est utile pour le décodage conjoint source canal si nous voulons améliorer les performances de l'algorithme générique de décodage séquentiel.
En effet, le code deflate associé au codage d'un pointeur LZ est obtenu à partir d'un préfixe sur 5 bits et d'un suffixe de longueur pouvant aller de 0 à 13 bits. Pour le suffixe, nous pouvons remarquer que le code est incomplet (voir [2]). Nous utiliserons ce fait pour vérifier la validité des codes deflates.

Code Caractère		Code Pointeur	Préfixe	Suffixe	Code Longueur	Préfixe	Suffixe
8-9 bits			5 bits	0-13 bits		7-8 bits	0-5 bits

Tableau 1 – *Taille minimale et maximale (en bits) des codes de Huffman appliqués aux codes LZ*

• Validité des codes LZ :

Rappelons qu'un code LZ est représenté par la paire $\langle l_i, p_i \rangle$ ou le singleton $\langle c_i \rangle$. Un code LZ est valide s'il vérifie les propriétés énoncées ci-dessous :
- Le singleton caractère décodé $\langle c_i \rangle$ doit appartenir à la plage des codes ASCII.
- Le symbole pointeur p_i décodé ne doit pas excéder le nombre de caractères déjà lus et insérés dans le dictionnaire.
- Si le symbole pointeur décodé p_i est nul alors le symbole longueur l_i doit être aussi nul.
- Dans le cas où le décodage d'Huffman d'un symbole deflate correspond à une longueur LZ, le symbole LZ décodé qui suit ne peut correspondre qu'à un pointeur LZ.
- Dans le cas où le décodage d'Huffman d'un symbole deflate correspond à un pointeur LZ, le symbole LZ décodé qui suit ne peut correspondre qu'à une longueur LZ ou à un caractère LZ.

• Conformité du langage html :

Rappelons qu'un fichier html est un document html écrit selon les règles syntaxiques du langage html ([8]). Ce dernier est un langage de balisage. Un document html est représenté par un ensemble d'éléments de texte codés afin de mettre en évidence la structure et le format du document qui sera interprété par le navigateur afin d'afficher des pages Web. Si les éléments html du document sont organisés de façon conforme à la définition formelle du langage html, le document est dit valide. Il existe plus d'une centaine d'éléments html utilisées par le langage html.

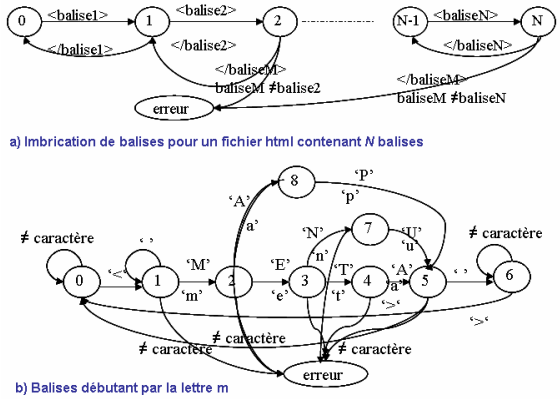

Figure 1 – *Analyseur syntaxique d'un document html*

Nous avons construit un analyseur syntaxique qui vérifie les propriétés énoncées dans [8] et [9]. Par manque de place, nous en listons ci-dessous uniquement quelques unes :
- Les navigateurs ne font pas la différence entre les majuscules et les minuscules pour l'écriture des éléments html.
- Les éléments html sont constitués de caractères alphanumériques encadrés par les signes < et >.
- La plupart des éléments sont constitués d'une paire de balises c'est-à-dire une balise de début et une balise de fin.
- La balise de début concerne le symbole de l'élément, placé entre les caractères < et >.
- La balise de fin est semblable à la balise de début sauf qu'elle est précédée par le caractère /. Les caractéristiques de l'élément sont appliquées au texte qui se trouve entre la balise de début et la balise de fin correspondante.
- Les éléments html peuvent être imbriqués mais jamais croisés. Il est assez fréquent de trouver des éléments html qui en contiennent d'autres. C'est-à-dire que si un élément html commence à l'intérieur d'un autre, il doit nécessairement se terminer à l'intérieur de celui-ci.

L'analyseur syntaxique donné par le graphe d'état de la Figure 1.a permet de vérifier la validité de l'imbrication

des balises. Tandis que le deuxième graphe d'état donné par la Figure 1.b vérifie certaines propriétés énoncées ci-dessous lorsque la balise commence par exemple par la lettre 'm'.

3.2.2 Algorithme de décodage séquentiel modifié

L'algorithme de décodage séquentiel à M chemins adapté aux règles grammaticales et syntaxiques des codes deflates et du langage html (énoncées en section 3.2.1) est illustré par la Figure 2.

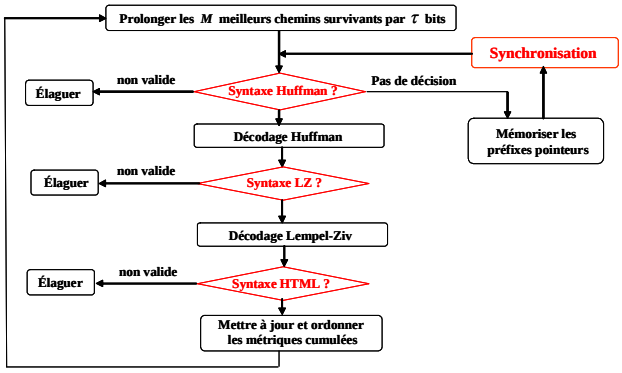

Figure 2 –*Décodage séquentiel adapté aux codes deflates et à la syntaxe du langage html*

L'algorithme se déroule comme suit :

1. Lire τ informations reçues $y_k,...,y_{k+\tau}$ où τ est un paramètre dont la valeur est choisie selon le compromis performance complexité ;
2. Pour chaque information lue, prolonger les M branches courantes de l'arbre jusqu'à construire $2^{\tau}M$ branches ;
3. Parmi les $2^{\tau}M$ banches retenues, élaguer les banches qui ne satisfont pas les règles d'Huffman prédéfinies par le standard deflate des codes de Huffman;
4. Pour chaque branche retenue, analyser le flux Huffman décodé comme suit :
   a) Si le nombre d'éléments Huffman décodés n'est pas suffisant pour prendre une décision, et que le code semble correspondre à un code préfixe pointeur LZ, dans ce cas mémoriser cette branche et aller à l'étape 9; autrement continuer ;
   b) Si le nombre d'éléments Huffman décodés est suffisant pour considérer que le mot de code peut être traité comme un code LZ, vérifier alors les règles grammaticales des codes LZ et prendre une décision. Si ce code LZ n'est pas valide, élaguer dans ce cas la branche correspondante et aller à l'étape 5.
   c) Si le nombre d'éléments Huffman décodés n'est pas suffisant pour considérer que le mot de code est un code LZ, continuer le prolongement de la branche correspondante en reprenant l'étape 1.
5. Pour chaque branche retenue, vérifier la syntaxe html du flux LZ décodé ;
6. Calculer les métriques cumulées des branches retenues (i.e. celles qui satisfont les règles grammaticales des codes deflates) ;
7. Trier les chemins selon les métriques calculées en 6 ;
8. Sectionner au plus M meilleurs chemins, puis recommencer à partir de l'étape 1.
9. Parmi les différents préfixes LZ localisés et mémorisés, étendre chaque branche mémorisée à partir de la dernière occurrence préfixe pointeur en utilisant une extension de $\alpha = 5$ bits (nombre de bits requis pour représenter un préfixe pointeur (i.e. synchronisation), voir Tableau 1) ;
10. Calculer les métriques des branches retenues ;
11. Trier les branches selon leurs métriques ;
12. Sélectionner au plus M meilleurs chemins selon leurs métriques puis recommencer à partir de l'étape 1 jusqu'à la lecture complète de toutes les informations reçues.

4 Décodage conjoint source canal : Récepteur itératif

Avant de présenter le récepteur itératif proposé, présentons tout d'abord le modèle de transmission adopté. Notons $\boldsymbol{d} = (d_1,...,d_k,...,d_N)$ la séquence binaire générée par l'algorithme de compression deflate où N représente la longueur de la séquence et d_k correspond au bit d'information à l'instant k. Les $\{d_k\}$ sont supposés équiprobables. Cette séquence binaire deflate est entrelacée puis envoyée à l'entrée d'un codeur convolutif de rendement $1/n$. Les informations récupérées à la sortie du codeur convolutif sont ensuite modulées à deux états (MDP2). Notons par $\boldsymbol{x}_1^N = (\boldsymbol{x}_1,...,\boldsymbol{x}_k,...,\boldsymbol{x}_N)$ cette séquence modulée où $\boldsymbol{x}_k = (x_{k,0}, x_{k,1},..., x_{k,n-1})$.
Ce train d'information est ensuite transmis via un canal sans mémoire supposé additif gaussien et blanc. Notons par $\boldsymbol{y}_1^N = (\boldsymbol{y}_1,...,\boldsymbol{y}_k,...,\boldsymbol{y}_N)$ le train d'information obtenu à la sortie de ce canal où $\boldsymbol{y}_k = (y_{k,0}, y_{k,1},..., y_{k,n-1})$.
Définissons deux ensembles notés $\boldsymbol{R}$ et $\boldsymbol{D}$. Le premier ensemble $\boldsymbol{R} = \{[d_1, d_2,..., d_N] \in \{0,1\}^N\}$ contient toutes les séquences binaires deflate possibles $\boldsymbol{d}$. Tandis que le deuxième ensemble $\boldsymbol{D} = \{[d_1, d_2,..., d_N] \in \{0,1\}^N\}$ ne contient que les séquences binaires deflate qui satisfont la structure du code deflate et la syntaxe du langage html.
Nous proposons de résoudre le problème de l'estimation des données bruitées par un canal de transmission par une approche itérative basée sur le principe des turbocodes. L'approche proposée s'appuie sur les travaux de décodage des codes concaténés en série développés dans la référence [10]. La dépendance entre les bits codés est alors prise en compte tout en réduisant la complexité de calcul prohibitive.
Le récepteur itératif proposé estime le message transmis $\boldsymbol{b}$ au sens du maximum a posteriori (MAP) à partir des données reçues $\boldsymbol{y}$ à la sortie du canal. Le décodeur itératif est constitué de deux blocs SISO (soft-in\soft-out) consécutifs (Figure 3). Le premier bloc correspond au décodeur canal (BCJR) et le second au décodeur source

basé sur l'algorithme de décodage séquentiel modifié de codes deflate décrit ci-dessus (DS-deflate).
L'algorithme itératif met en œuvre des échanges d'informations souples (probabilités extrinsèques associées à chaque bit codé) entre les deux décodeurs en série et les contraints à converger au fil de ces échanges.
A une itération quelconque I de l'algorithme, le récepteur itératif s'appuie sur trois traitements principaux que nous décrirons ci-dessous.

• Le décodeur intérieur : le BCJR

Le décodeur canal BCJR, de type SISO (soft-in\soft-out) basé sur l'algorithme MAP, calcule à la première étape, les probabilités a posteriori pour toutes les séquences possibles $P_R(\boldsymbol{d}/\boldsymbol{y})$ avec $\boldsymbol{d} \in \boldsymbol{R}$. Puisque le canal de transmission est sans mémoire et que les $\{d_k\}$ sont indépendants, le calcul des probabilités a posteriori est équivalent aux produits des probabilités marginales a posteriori (APP) comme suit $\prod_{k=1}^{N} P^I_{BCJR}(d_k/y_k)$. A la sortie du décodeur BCJR, la probabilité extrinsèque associée au bit d'information codé par la source, à une itération quelconque I, est donnée par l'équation ci-dessous :

$$E^I_{BCJR}(d_k) = K^I_{BCJR} \frac{P^I_{BCJR}(d_k/y_k)}{P^I_{BCJR}(d_k)} \qquad (4)$$

où K^I_{BCJR} représente le facteur de normalisation qui se déduit à partir de $E^I_{BCJR}(d_k=0)+E^I_{BCJR}(d_k=1)=1$ et $P^I_{BCJR}(d_k)$ représente la probabilité a priori associé au bit codé. Cette probabilité correspond à l'information extrinsèque délivrée par le bloc extérieur DS-defltate à l'itération précédente ($E^{I-1}_{DS-deflate}(d_k)$). L'information extrinsèque $E^I_{BCJR}(d_k)$ désentrelacée est envoyée à l'entrée du décodeur DS-deflate.

• Le décodeur extérieur : le DS-deflate

Le décodeur source DS-deflate, de type SISO, réalise la projection des distributions des APP évaluées sur $\boldsymbol{R}$, sur l'ensemble des distributions des APP compatibles avec la structure du code deflate et la syntaxe du langage html.
La distribution des APP recherchée, notée $P^+_{DS-deflate}(\boldsymbol{d})$ est celle qui minimise la distance de Kullback-Leibler donnée par ([11]):

$$P^+_{DS-deflate}(\boldsymbol{d}) = \arg\min_{P_D(d)} dist(P_D(\boldsymbol{d}), P_R(\boldsymbol{d}/\boldsymbol{y})) \qquad (5)$$

Il a été montré, dans la référence [12], que la distribution des APP est donnée par la relation suivante :

$$P^+_{DS-deflate}(\boldsymbol{d}) = \begin{cases} \dfrac{P_R(\boldsymbol{d}/\boldsymbol{y})}{\sum_{\boldsymbol{d}\in\boldsymbol{D}} P_R(\boldsymbol{d}/\boldsymbol{y})} & \text{si } \boldsymbol{d} \in \boldsymbol{D} \\ 0 & \text{si } \boldsymbol{d} \in \boldsymbol{R}/\boldsymbol{D} \end{cases} \qquad (6)$$

A une itération quelconque I de l'algorithme, la sortie souple du bloc DS-deflate est donnée par :

$$E^I_{DS-deflate}(d_k) = K^I_{DS-deflate} \frac{P^I_{DS-deflate}(d_k/y_k)}{P^I_{DS-deflate}(d_k)} \qquad (7)$$

où $K^I_{DS-deflate}$ représente le facteur de normalisation, déduit à partir de la propriété suivante $E^I_{DS-deflate}(d_k=0)+E^I_{DS-deflate}(d_k=1)=1$ et $P^I_{DS-deflate}(d_k)$ représente la probabilité a priori associée au bit codé correspondant à l'information extrinsèque délivrée par le décodeur intérieur BCJR à l'itération précédente ($E^{I-1}_{BCJR}(d_k)$).
Le calcul de l'information intrinsèque $E^I_{DS-deflate}(d_k)$ requière la connaissance de l'APP marginale DS-deflate déduite de l'équation (6) comme suit :

$$P^I_{DS-deflate}(d_k/y_k) = \sum_{d_k:0,1} P^+_{DS-deflate}(\boldsymbol{d}) \qquad (8)$$

L'information extrinsèque $E^I_{DS-deflate}(d_k)$ désentrelacée est ensuite envoyée à l'entrée du bloc BCJR. Pour estimer le bit d'information codé d_k, au sens du MAP, le décodeur DS-deflate cherche la distribution des APP qui maximise $P^+_{DS-deflate}(\boldsymbol{d})$:

$$\hat{d}_k = \arg\max_{d_k:0,1} \log\Big(\sum_{\boldsymbol{d}\in\boldsymbol{D}} P^+_{DS-deflate}(\boldsymbol{d})\Big) \qquad (9)$$

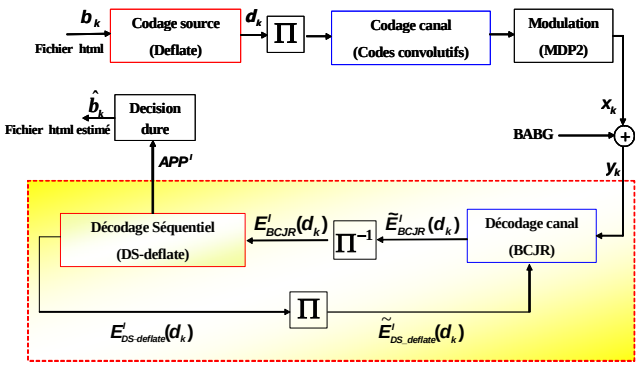

Figure 3 – *Récepteur itératif*

5 Performances du récepteur itératif

Nous avons simulé la chaîne de transmission présentée par la Figure 3. Nous avons choisi de travailler selon le premier mode spécifié par la norme IEEE 802.11.a ([16]) En effet ce mode est le plus difficile à traiter puisque c'est le moins robuste. Pour ce mode de fonctionnement, les données sont modulées selon la modulation à deux états MDP2. Le codeur convolutifs spécifié a les caractéristiques suivantes : longueur de contrainte 7, polynôme générateur donné par la représentation octale [171,133] et un rendement de ½. L'entrelaceur utilisé est un entrelaceur aléatoire, appliqué sur toute la longueur des données à traiter. Les simulations présentées sont réalisées sur deux fichiers html de test. Le premier fichier html ([14]), de taille 413 octets, contient 60.5% de balises par rapport au nombre total de caractères dans le fichier html. Ce fichier est un exemple type de fichier wap. Tandis que le deuxième fichier html ([15]), de taille 3031 octets, contient 11.5 % de balises par rapport au nombre total de caractères dans le fichier html. La qualité de la correction des erreurs de l'algorithme de récepteur itératif est mesurée par le taux d'erreur symbole (TES) en fonction du rapport signal à bruit (E_b/N_0). Les courbes, présentées par les figures (Figure 4, Figure 5) illustrent les résultats

obtenus avec les paramètres suivants : $M = 10$, $\tau = 5$ et un nombre d'itérations fixé à trois. Les résultats sont comparés avec un décodage canal classique (BCJR et décision dure).

Pour le premier fichier html de test (Figure 4), nous remarquons pour un $E_b/N_0 = 7dB$ (i.e. dans la zone de fonctionnement), que le récepteur itératif à la troisième itération réduit le TES d'un facteur 9 comparé aux résultats fournis par un décodage de canal classique (BCJR). Pour cet exemple particulier, augmenter le nombre de chemins de l'algorithme DS-deflate à 20 ($M = 20$) n'améliore pas les performances de l'algorithme de décodage robuste. En revanche le temps de calcul est multiplié par 2.

Pour le deuxième fichier html de test (Figure 5), nous constatons pour un $E_b/N_0 = 7dB$ (i.e. dans la zone de fonctionnement), que le récepteur itératif à la troisième itération réduit le TES d'un décodage canal classique (BCJR) par un facteur 5. Notons que la correction des erreurs est d'autant plus importante que le nombre de balises contenues dans un fichier html est élevé.

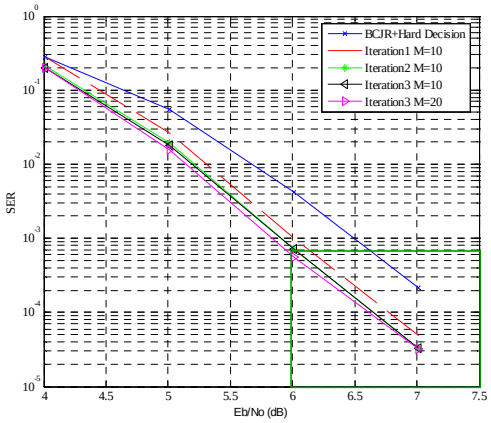

Figure 4 – *Performances du récepteur itératif appliqué au fichier html 1*

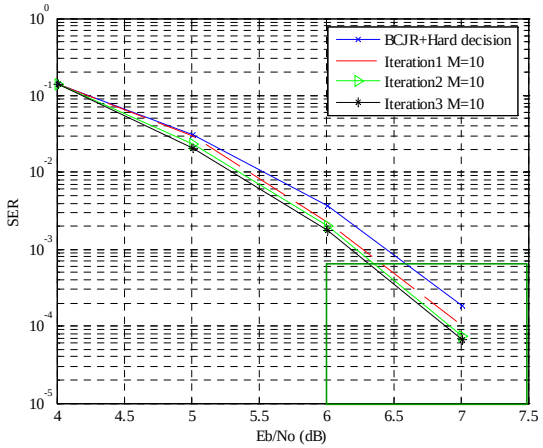

Figure 5 – *Performances du récepteur itératif appliqué au fichier html 2*

6 Conclusion

Nous avons proposé une approche de décodage source canal conjoint pour la correction des erreurs survenues lors du téléchargement ou de la transmission de fichier html comprimé via un lien mobile bruité. Nous avons développé un algorithme de décodage itératif basé sur l'algorithme de décodage séquentiel à *M*-chemins qui exploite les règles grammaticales et syntaxiques des codes deflates et du langage html. Les résultats de simulation ont montré que notre récepteur itératif améliore significativement la qualité de la protection des codes deflates comparée à un décodage canal classique.

Remerciements : Ce travail est financé par l'ANR dans le cadre du projet DITEMOI

Références

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T.Berners-Lee, \Hypertext transfer protocol, http1.1," June 1999.

[2] L.P. Deutsch, "DEFLATE Compressed data format specification," in *rfc1951*, May 19.

[3] L.P. Deutsch, "GZIP Compressed data format specification," in *rfc1952*, May 1996.

[4] L.P. Deutsch, "ZLIB Compressed data format specification," in *rfc1950*, May 1996.

[5] S. Lonardi and W. Szpankowski, \Joint source-channel LZ'77 coding," in *proc.of DCC*, 2003.

[6] Z. Jaoua, A. Zergaïnoh, P. Duhamel, *Algorithme de décodage séquentiel de pages html comprimées par Lempel-Ziv-77*, GRETSI 2007, 11-14 Septembre 2007, Troyes, France, pp. 497-500.

[7] Z. Jaoua, A. Zergaïnoh-Mokraoui, and P. Duhamel, "Robust transmission of html files: Iterative joint source-channel decoding of lempel ziv-77 codes," in *IEEE ICASSP*, Las Vegas, Nevada, USA, 2008.

[8] "Html specification, w3c recommendation," December 1999.

[9] http://www.w3.org/markup/sgml/.

[10] S. Bendetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes : performance analysis, design, and iterative decoding," in *IEEE Trans. on Inforamtion Theory, vol.44, no. 3*, May. 1998, pp. 909-926.

[11] G. Battail, "Le décodage pondéré en tant que procédé de réévaluation d'une distribution de probabilité," in *Les Annales des télécommunications, vol. 42, no. 9-10*, Sept 1987, pp. 499-509.

[12] P. Magniez, B. Muquet, P. Duhamel, V. Buzenac, and M. de Courville, "Optimal decoding of bit-interleaved modulations: Theoretical aspects and practical algorithms," in *2nd Intl. Symposium on Turbo Codes and Related Topics*, Sept. 2000, pp. 284-287.

[13] Z. Jaoua, A. Mokraoui-Zergaïnoh, and P. Duhamel, "Robust transmission of html files: Iterative joint source-channel decoding of deflate codes," *in EUSIPCO 2008*, Lausanne, Switzerland, 2008.

[14] base d'exemples de sdk www.forum.nokia.com/tools worldcup.html.

[15] ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc- index. html.

[16] http://www.ieee802.org/11/.