Algorithme de décodage séquentiel de pages HTML comprimées par Lempel-Ziv-77

Zied JAOUA[1] [2], Anissa ZERGAÏNOH[1] [2], Pierre DUHAMEL[1]

[1]LSS/CNRS, Supélec
Plateau de Moulon, 91 192 Gif sur Yvette, France

[2]L2TI, Institut Galilée, Université Paris 13
99, Avenue Jean Baptiste Clément, 93 430 Villetaneuse, France
jaoua@lss.supelec.fr, zergainoh@lss.supelec.fr
duhamel@lss.supelec.fr

Résumé – Cet article traite le problème de la correction des erreurs survenues lors de la transmission de pages HTML comprimées via un lien mobile bruité par une approche de décodage itérative. Le récepteur proposé s'appuie sur l'algorithme classique de décodage séquentiel à M-chemins. Pour améliorer les performances du récepteur, nous apportons des modifications au M-algorithme de façon à ce qu'il exploite les règles spécifiques liées (i) à l'algorithme de codage entropique Lempel-Ziv-77 adopté par le dernier protocole d'échanges de fichiers, et (ii) à la syntaxe du langage HTML. Les résultats de simulation montrent que la méthode proposée améliore la qualité de la protection des pages HTML transmises comparée à un décodage canal classique.

Abstract – This paper concerns the correction of errors occurring in the compressed HTML pages during their transmission via a noisy mobile channel using an iterative decoding approach. The proposed receiver is based on the conventional sequential decoding M-algorithm. To improve the performances of the receiver, we propose to modify the M-algorithm in order to exploit the specific rules related to (i) the entropic Lempel-Ziv-77 coding algorithm adopted by the recent file exchange protocol, and (ii) the syntax of the HTML language. The simulation results show that the proposed method improves the quality of the protection of the transmitted HTML pages compared to a conventional channel decoding.

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 le codage entropique Lempel-Ziv-77 (LZ). Les fichiers HTML sont alors transmis ou téléchargés sous le format Gzip ([2]) ou Zip ([3]). Cet article s'intéresse au décodage robuste de telles pages HTML transmises ou téléchargées via un lien mobile bruité. Afin de ne pas dégrader le débit de transmission initialement fixé par le mode de fonctionnement du système de communication, le décodeur proposé corrige les erreurs survenues dans le message transmis uniquement à partir des informations résiduelles contenues dans le train d'information codé. L'objectif de notre travail est de pouvoir réaliser un décodage conjoint source canal. Dans cette thématique de décodage conjoint source canal (appliqué à Lempel-Ziv), un seul travail a été développé (à notre connaissance) dans la référence [4]. Les auteurs de cette référence proposent une structuration des données Lempel-Ziv-77 combinée à des codes Reed Solomon.

Le récepteur proposé est construit sur le principe turbo de décodage itératif de code concaténé en série. Dans cet article, l'algorithme de décodage itératif est basé sur l'algorithme classique de décodage séquentiel à M-chemins (décodage souple) auquel nous avons apporté des modifications afin d'améliorer les performances du récepteur. L'algorithme de décodage séquentiel proposé procède à l'élimination de certains chemins (parmi les M-chemins) dans l'arbre en exploitant les informations sur les codes Lempel-Ziv-77 ainsi que les règles de syntaxe du langage HTML.

L'article est organisé comme suit. La section 2 introduit l'algorithme de codage entropique Lempel-Ziv-77. L'algorithme de décodage séquentiel à M-chemins est ensuite décrit en section 3. La section 4 présente l'algorithme de décodage séquentiel adapté aux codes Lempel-Ziv-77 combinés aux règles syntaxiques du langage HTML. La section 5 présente le décodeur itératif proposé. Les résultats de simulation sont illustrés en section 6. La dernière section conclut notre travail.

2 Algorithme de codage entropique Lempel-Ziv-77

Parmi les différentes variantes d'algorithmes de codage Lempel-Ziv, nous nous intéressons dans cet article plus particulièrement à Lempel-Ziv-77 (LZ). En effet c'est cette version d'algorithme qui est implantée dans le protocole HTTP1.1 ([1], [5]). Lempel-Ziv-77 est une méthode de codage entropique développée initialement pour la compression de texte ([6]). Les données du fichier à comprimer sont analysées séquentiellement de gauche à droite. Notons T le texte 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. La méthode de codage est basée sur la construction adaptative d'un dictionnaire. A cette étape, l'algorithme de codage cherche dans le dictionnaire (en l'occurrence dans $T[1,i-1]$) la plus longue h-ème chaîne de ca-

ractères qui serait identique à celle disponible dans $T[i, i+l_h-1]$ avec $l_h \leq L$ où L représente la taille de la fenêtre de recherche prédéfinie. La chaîne retenue est codée par un triplet (ou symbole) noté $< p_i, l_i, c_i >$, où p_i représente le *pointeur* vers le dictionnaire indiquant le début de la chaîne à coder, l_i la *longueur* de la nouvelle chaîne à inclure dans le dictionnaire, et c_i le *caractère* suivant $T[i + l_i]$ à inclure dans le dictionnaire. Ainsi, le texte codé par l'algorithme Lempel-Ziv-77 est représenté par une suite de triplets (ou symboles) : $< p_0, l_0, c_0 >, < p_1, l_1, c_1 >, ..., < p_i, l_i, c_i >, ...$

Dans la référence de normalisation de Deflate ([5]), la taille du dictionnaire ainsi que celle de la fenêtre de recherche sont respectivement fixées à 256 octets et 32 K-octets. Les mots p_i, l_i et c_i sont alors respectivement codés sur 15 bits ($p_i = p_i^{14}...p_i^j...p_i^0$), 8 bits ($l_i = l_i^7...l_i^j...l_i^0$) et 8 bits ($c_i = c_i^7...c_i^j...c_i^0$). Il faut noter que dans certains cas, un codage de Huffman est appliqué aux triplets ainsi considérés. Nous ne considèrerons pas cette étape dans notre travail. En effet, son utilité est modérée. A titre d'exemple, pour le fichier de 47 K-octets de la page d'accueil de notre laboratoire, elle apporte une amélioration de 30 % de la taille du fichier, à comparer au facteur 7 apporté par la première partie du travail. De toute façon, asymptotiquement, les propriétés de LZ sont obtenues sans application de ce codage de Huffman : l'amélioration apportée par Huffman doit logiquement décroitre avec la taille du fichier.

3 Décodage séquentiel à M-chemins

Introduisons le principe de l'algorithme classique de décodage séquentiel à M-chemins sur lequel s'appuie le récepteur proposé. L'algorithme de décodage séquentiel à M-chemins, appelé DS, utilise une structure en arbre avec un nombre M fixe de chemins mémorisés ([7]). Le principe de base consiste à étendre les M chemins mémorisés, en les prolongeant à l'aide de l'arbre décrivant le problème, de calculer les valeurs de vraisemblance (métriques) correspondant à chacune de ces extensions, et de mémoriser à nouveau les M meilleures séquences parmi toutes celles qui ont été obtenues. On cherche à maximiser la vraisemblance entre les symboles reçus ($\mathbf{y} = (y_1, ..., y_t, ..., y_N)$) et ceux émis ($\mathbf{x} = (x_1, ..., x_t, ..., x_N)$) à l'entrée du canal (voir Fig.1).

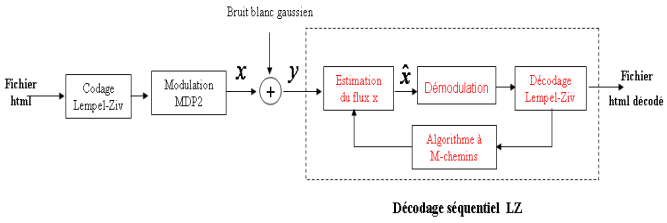

FIG. 1 – Décodeur séquentiel adapté aux codes LZ (DSLZ)

Le message estimé au sens du maximum de vraisemblance est alors donné par :

$$\widehat{\mathbf{x}} = \arg\max_x \log(P(\mathbf{y}/\mathbf{x})) \quad (1)$$

où $P(\mathbf{y}/\mathbf{x})$ est la vraisemblance définie comme suit pour un canal gaussien :

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

σ^2 représente la variance du bruit blanc gaussien centré. Nous déduisons que le message estimé est donné par : $\widehat{\mathbf{x}} = \arg\min_x \sum_{k=1}^{N} (x_k - y_k)^2$ où $\mu(N) = \sum_{k=1}^{N} (x_k - y_k)^2$ 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). Listons les principales étapes suivies par cet algorithme classique :

1. Lire une information reçue y_k,
2. Prolonger tous les M nœuds courants associés à cette information y_k,
3. Calculer les métriques des branches prolongées,
4. Trier les $2M$ nouveaux nœuds 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).

4 Décodage séquentiel modifié

Pour améliorer les performances de l'algorithme classique de décodage à M-chemins et réduire sa complexité de calcul, nous imposons un ensemble de règles à satisfaire lors du décodage de la séquence transmise. Ces règles exploitent les caractéristiques des codes LZ combinées aux règles syntaxiques du langage HTML. Parmi les chemins retenus selon le critère des métriques, l'algorithme proposé, appelé DSLZ, complète la vérification de la validité de ces M-chemins en s'appuyant sur des règles que nous avons prédéfinies. La décision n'est 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. Les règles à vérifier lors du décodage sont les règles de grammaire du codeur LZ, combinées avec celles du langage HTML. Elles sont résumées ci-dessous :

- Le symbole pointeur décodé ne doit pas excéder le nombre de caractères déjà lus et insérés dans le dictionnaire.
- Si le symbole pointeur décodé est nul alors le symbole longueur est aussi nul.
- Le symbole caractère décodé doit appartenir à la plage des codes ASCII.
- A chaque balise de début doit correspondre une balise de fin portant le même nom ([8]).

L'algorithme de décodage séquentiel à M-chemins adapté aux codes LZ et à la syntaxe du langage HTML se déroule comme suit :

1. Lire une information reçue y_k,
2. Pour chaque information lue (prolonger les M branches courantes de l'arbre jusqu'à construire $2^\alpha M$ branches (avec $\alpha = 15$ pour le mot pointeur, $\alpha = 8$ pour le mot caractère ou longueur),
3. Calculer les métriques de ces $2^\alpha M$ branches prolongées,
4. Trier les $2^\alpha M$ branches selon les métriques cumulées,
5. Parmi les $2^\alpha M$ chemins, élaguer les branches qui ne satisfont pas les règles prédéfinies. Le nombre de branches retenues est noté M',
6. Parmi les M' chemins retenus en 5, choisir au plus M meilleurs chemins,
7. Recommencer à partir de 1 jusqu'à atteindre la profondeur finale de l'arbre.

5 Récepteur adapté aux codes LZ et à la syntaxe HTML

A l'heure d'aujourd'hui, il n'est pas concevable de considérer un système de communication sans codage canal. La chaîne

de transmission considérée dans cet article est présentée en Fig.2. Notons $\mathbf{d} = (d_1, ..., d_k, ..., d_N)$ la séquence binaire générée par LZ ($< p_i, l_i, c_i >$) 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 LZ est entrelacée, envoyée à l'entrée d'un codeur convolutif de rendement $\frac{1}{n}$ puis modulée à deux états (MDP2). Notons la par $\mathbf{x}_1^N = (\mathbf{x}_1, ..., \mathbf{x}_k, ..., \mathbf{x}_N)$ où $\mathbf{x}_k = (x_{k,0}, x_{k,1}, ..., x_{k,n-1})$. Ce train d'information est ensuite transmis via un canal sans mémoire supposé additif gaussien et blanc. Le train d'information obtenu à la sortie de ce canal est noté $\mathbf{y} = \mathbf{y}_1^N = (\mathbf{y}_1, ..., \mathbf{y}_k, ..., \mathbf{y}_N)$ où $\mathbf{y}_k = (y_{k,0}, y_{k,1}, ..., y_{k,n-1})$.
Définissons deux ensembles $\boldsymbol{R}$ et $\boldsymbol{D}$ de séquences binaires $\mathbf{d}$ de longueur N. Le premier ensemble $\boldsymbol{R} = \{[d_1, d_2, ..., d_N] \in \{0,1\}^N\}$ contient toutes les séquences possibles. Tandis que le deuxième ensemble $\boldsymbol{D} = \{[d_1, d_2, ..., d_N] \in \{0,1\}^N\}$ ne contient que les séquences respectant la structure du code LZ ainsi que la syntaxe du langage HTML. Le problème de l'estimation des

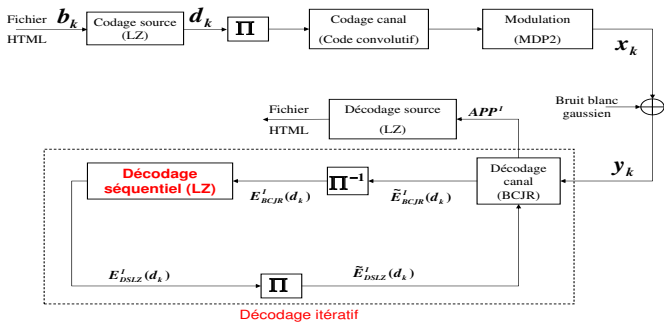

FIG. 2 – Décodeur itératif basé sur le DSLZ

données bruitées par un canal de transmission est résolu par une approche itérative basée sur le principe turbo. La dépendance entre les bits codés est alors prise en compte tout en réduisant la complexité de calcul prohibitive. L'approche proposée s'appuie sur les travaux le décodage des codes concaténés en série développés dans les références ([9],[10]). Le récepteur itératif estime le message transmis $\mathbf{b}$ au sens du Maximum A Posteriori (MAP) à partir des données reçues $\mathbf{y}$ à la sortie du canal :

$$\widehat{\mathbf{b}} = \arg\max_{b} \log(P(\mathbf{b}/\mathbf{y})) \tag{3}$$

Le décodeur itératif proposé est constitué de deux blocs consécutifs (voir Fig.2). Le premier bloc correspond au décodeur canal (BCJR [11]) et le second au décodeur séquentiel LZ (DSLZ) précédemment décrit. Chaque bloc utilise les informations transmises par le bloc précédent dans la chaîne de décodage itérative. A une itération I donnée, l'algorithme de décodage procède en trois étapes principales. A la première étape, le décodeur BCJR maximise la probabilité a posteriori $P(\mathbf{d}/\mathbf{y})$

$$\widehat{\mathbf{d}} = \arg\max_{d \in \boldsymbol{R}} \log(P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})) \tag{4}$$

qui est équivalente à la maximisation des probabilités a posteriori (APP) marginales $\prod_{k=1}^{N} P^I_{BCJR}(d_k/y_k)$ puisqu'on suppose que le canal de transmission est sans mémoire et que les d_k sont indépendants. Ces probabilités marginales sont calculées pour toutes les séquences possibles $\mathbf{d} \in \boldsymbol{R}$. Rappelons que la probabilité extrinsèque marginale associée au bit d'information codé, à l'itération I, est donnée par :

$$E^I_{BCJR}(d_k) = K \frac{P^I_{BCJR}(d_k/y_k)}{P^I_{BCJR}(y_k)} \tag{5}$$

où K est le facteur de normalisation tel que $E^I_{BCJR}(d_k = 0) + E^I_{BCJR}(d_k = 1) = 1$; $P^I_{BCJR}(d_k)$ est la probabilité a priori associé au bit d'information codé correspondant à l'information extrinsèque entralacée ($E^{I-1}_{DSLZ}(d_k)$) calculée par le DSLZ à l'itération $I-1$. L'information extrinsèque désentrelacée $E^I_{BCJR}(d_k)$ est ensuite envoyée à l'entrée du décodeur DSLZ. La deuxième étape consiste à projeter les distribution des APP évaluées sur $\boldsymbol{R}$, sur l'ensemble des distributions des APP compatibles avec la structure du code LZ et la syntaxe du langage HTML. La distribution des APP recherchée, notée $P^+(\mathbf{d})$, est celle qui minimise la distance de Kullback-Leibler donnée par :

$$P^+(\mathbf{d}) = \arg\min_{P_{\boldsymbol{D}}(\mathbf{d})} dist(P_{\boldsymbol{D}}(\mathbf{d}), P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})) \tag{6}$$

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

$$P^+(\mathbf{d}) = \begin{cases} \frac{P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})}{\sum_{d \in \mathbf{D}} P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})} & \text{si } d \in \mathbf{D} \\ 0 & \text{si } d \in \mathbf{R} \backslash \mathbf{D} \end{cases} \tag{7}$$

La troisième étape consiste à maximiser $P^+(\mathbf{d})$. Puisque nous supposons que les y_k sont équiprobables, dans ce cas :

$$\prod_{k=1}^{N} P^I_{DSLZ}(y_k/d_k) \propto \prod_{k=1}^{N} P^I_{BCJR}(d_k/y_k) \tag{8}$$

Le DSLZ recherche la distribution des APP qui maximise $P^+(\mathbf{d})$:

$$\prod_{k=1}^{N} P^I_{DSLZ}(d_k/y_k) = \prod_{k=1}^{N} E^I_{BCJR}(d_k) E^{I-1}_{DSLZ}(d_k) \tag{9}$$

Ces APP marginales correspondent aux métriques de branche calculées à l'instant k par l'algorithme DSLZ. Les informations extrinsèques marginales sont ainsi mises à jour par le DSLZ puis transmises au BCJR. Ces informations sont considérées comme informations a priori par le décodeur BCJR à l'itération suivante.

6 Résultats de simulation

Les tests présentés visent à montrer l'intérêt de la méthode de décodage conjoint source canal proposée. Nous avons simulé la chaîne de transmission présentée en Fig.2 où le codeur convolutif choisi est celui spécifié par le premier mode de fonctionnement dans la norme IEEE 802.11.a ([13]). Ses caractéristiques sont les suivantes : longueur de contrainte 7, polynôme générateur donné par la représentation octale [171,133] et un rendement de 1/2. 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 caractère dans le fichier HTML. Tandis que le deuxième fichier HTML ([15]), de taille 3031 octets, contient 11.5 % de balises par rapport au nombre total de caractère dans le fichier HTML. La qualité de la correction des erreurs de l'algorithme de décodage séquentiel modifié est mesurée par le taux d'erreur symbole (TES) en fonction du rapport signal à bruit (SNR).

Les résultats de décodage séquentiel sont tout d'abord donnés pour une transmission sans codage canal en Fig.3. Les résultats sont comparés par rapport (i) au décodage avec décision ferme, (ii) au décodage séquentiel adapté uniquement aux règles des codes LZ pour différentes valeurs de M, (iii) au décodage séquentiel adapté aux codes LZ et à la syntaxe du langage HTML pour différentes valeurs de M. Nous constatons que le décodage séquentiel, combinant les règles liées aux codes LZ et à la syntaxe du langage HTML, livre au décodeur une quantité supérieure d'information améliorant ainsi la qualité de la

protection du message. Notons que la correction est d'autant plus importante que le nombre de balises contenues dans un fichier est élevé. En effet pour un TES équivalent, le décodage séquentiel (avec $M = 50$) appliqué au premier fichier HTML de test offre dans le meilleur des cas un gain de $1dB$ par rapport à une décision ferme. Tandis que le décodage du deuxième fichier de test, pour un TES équivalent, dans le meilleur des cas l'algorithme de décodage (avec $M = 80$) permet de gagner $0.5dB$.

Examinons les résultats de simulation obtenus par le récepteur itératif proposé. Les courbes, présentées par Fig.4, illustrent les résultats obtenus sur le premier fichier HTML de test avec $M = 10$ et un nombre d'itérations fixé à deux. Nous constatons, tout d'abord, que le récepteur proposé fonctionne dans une zone où la puissance du bruit est plus élevé comparé aux résultats présentés par Fig.3. Ce qui est du essentiellement à la réduction du nombre de pile vide (lié à l'élagage) par rapport aux résultats sans codage canal puisque la dépendance entre les bits codés est prise en compte dans le système itératif. Pour un rapport signal à bruit de $3dB$, à la deuxième itération, le récepteur proposé réduit le TES d'un facteur 10 par rapport aux résultats obtenus à partir d'une décision dure prise à la sortie du BCJR.

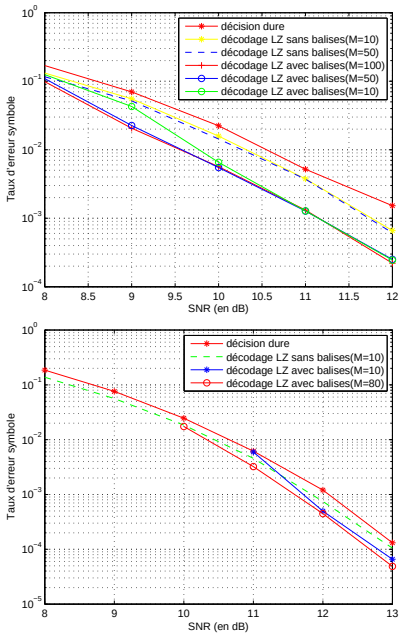

FIG. 3 – Performances de l'algorithme DSLZ appliqué aux fichiers HTML 1 (haut) et HTML 2 (bas)

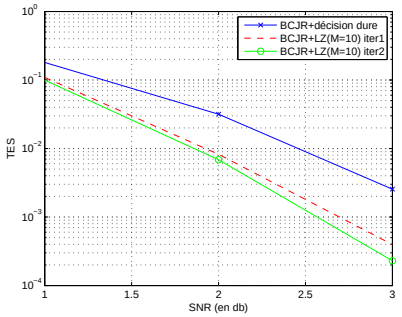

FIG. 4 – Performances du récepteur itératif proposé appliqué au premier fichier HTML

7 Conclusion

Dans cet article, nous avons proposé une approche de décodage source canal conjoint pour la correction des erreurs survenues lors du téléchargement de fichier HTML comprimé par Lemplel-Ziv-77 via un lien mobile. Nous avons proposé un algorithme de décodage itératif basé sur l'algorithme de décodage séquentiel à M-chemins qui exploite les règles associées aux codes LZ ainsi que les règles grammaticales du langage HTML. Ces premiers résultats sont encourageants. Nous envisageons de poursuivre ces travaux afin de tenir compte également du codage entropique d'Huffman combiné à LZ.

Références

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

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

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

[4] S. Lonardi and W. Szpankowski, "Joint source-channel lź77 coding," in *proceeding of data compression DCC*, 2003.

[5] L.P. Deutsch, "DEFLATE Compressed data format specification"," in *rfc1951*, May 1996.

[6] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," in *IEEE Transactions on Information Theory, 23(3)*, May 1977, pp. 337–343.

[7] J.B. Anderson and S. Mohan, "Source and channel coding : an algorithmic approach," in *Kluwer Academic Publishers, Norwell*, MA 1991.

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

[9] 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.

[10] P. Magniez, B. Muquet, P. Duhamel, V. Buzenac, and M. deCourville, "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.

[11] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," in *IEEE Transactions on Information Theory, 20(2)*, March 1974, pp. 284–287.

[12] 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.

[13] "http ://www.ieee802.org/11/," .

[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," .